

COMPUTER-AIDED TIMING TRAINING SYSTEM FOR MUSICIANS

David Manchip

Thesis presented in partial fulfilment of the requirements for the degree of
Master of Philosophy (Information Technology)
In the Department of Computer Science, Faculty of Science
University of Cape Town

Supervised by Dr. Audrey Mbogho

November 2011

I know the meaning of plagiarism and declare that all of the work in the thesis, save for that which is properly acknowledged, is my own.

A handwritten signature in black ink, appearing to be the initials 'DJ' or similar, written in a cursive style.

Abstract

Traditionally, musicians make use of a metronome for timing training. A typical metronome, whether hardware or software emulation, will provide the musician with a regular, metrical click to use as a temporal guide. The musician will synchronise his or her actions to the metronome click, thereby producing music that is in time. With regular usage, a musician's sense of time will gradually improve. To investigate potential benefits offered by computer-assisted instruction, an Alternate Timing Training System was designed and a prototype software implementation developed. The system employed alternative training methods and exercises beyond those offered by a standard metronome. An experiment was conducted with a sample of musicians that attempted to measure and compare improvements in timing accuracy using a standard metronome and the Alternate Timing Training System. The software was also made available for public download and evaluated by a number of musicians who subsequently completed an online survey. A number of limitations were identified in the experiment, including too short a training period, too small a sample size and subjects that already had a highly developed sense of time. Whilst the results of the experiment were inconclusive, analysis of survey results indicated a significant preference for the Alternate Timing Training System over a standard metronome as an effective means of timing training.

Acknowledgements

This work would not have been possible were it not for the help of the following people:

Wynand van der Walt, Geoffrey Smith, Kerry Hiles, Marcio de Brito, Avzal Ismail, Brett Collings, Travis Marc and Neil Kuny – thank you for committing to such a stringent evaluation program, and for giving such valuable feedback. Thank you to Lorna Morris for proof-reading and suggestions.

Dr. Audrey Mbogho, thank you for your patience and for agreeing to supervise such a variety of disparate subject matter.

Contents

1	Introduction	4
2	Background and Related Work	6
2.1	Music in Time: Pulse, Tempo and Subdivision	6
2.1.1	Maelzel’s Metronome	8
2.1.2	Software Metronomes	9
2.2	Cognitive Psychology: Perception and Action	18
2.2.1	Sensorimotor Synchronisation	18
2.2.2	Skills Acquisition	20
2.2.3	Related Work	21
2.3	Computer-assisted Music Instruction	22
2.4	Summary	23
3	Methodology	24
3.1	Overview	24
3.2	Experiment	24
3.2.1	Assessment	25
3.2.2	Usage Control	29
3.2.3	Subject Requirements	31
3.2.4	Limiting Factors	32
3.2.5	Data Extraction	33
3.2.6	Summary	37
3.3	Survey	37
3.3.1	Survey Design	38
3.3.2	Competency Assessment – 5 Questions	38
3.3.3	Application Usage Assessment – 3 Questions	39
3.3.4	Opinion – 8 Questions	39
3.3.5	Summary	39

4	Implementation	40
4.1	Design	40
4.2	Principles and Concepts	41
4.2.1	Reference Points	41
4.2.2	Methods of Pattern Generation	42
4.2.3	Summary	48
4.3	Algorithmic Pattern Generation	49
4.4	Implementation	51
4.4.1	Requirements	52
4.4.2	User Interface	53
4.4.3	Usage	56
5	Results and Discussion	57
5.1	Experiment Results Analysis	57
5.1.1	Full Dataset	61
5.1.2	Faster Tempi	63
5.1.3	Slower Tempi	64
5.1.4	First 10 Beats per Tempo Removed	65
5.1.5	Discussion	67
5.2	Survey Response Analysis	68
5.2.1	Competency Assessment Questions	69
5.2.2	Application Usage Assessment Questions	75
5.2.3	Opinion Questions	77
5.2.4	Discussion	91
5.2.5	Supplementary Discussion	92
5.3	Summary	93
6	Conclusions and Future Work	94
6.1	Conclusions	94
6.2	Future Work	96
	Bibliography	98
	Appendix A Source Code	103
A.1	Alternate Timing Training System Implementation	103
A.2	Log File Parser	143
A.3	SQL Queries	153
	Appendix B Class Diagram	156

Appendix C Survey Questions	158
Appendix D Survey – Supplementary Discussion	165
D.1 Email and Online Forum Discussions	165
D.2 Survey Comments	173
Appendix E Survey Website	177

Chapter 1

Introduction

A good sense of time is an essential skill for the professional or semi-professional musician. Modern performing and recording situations require a highly developed sense of tempo (musical pace) and temporal interval subdivision (the ability to mentally subdivide the beat into smaller intervals) [Voros, 1995, p16 - 20]. Timing training is often an integral part of a musician's daily practice routine, along with many other types of musical training such as aural training and sight reading. Through years of practicing and performing, a musician's temporal perception improves well beyond that of non-musicians [Franek et al., 1991].

Traditionally a metronome is used during musical practice, both as a means to improve technical facility and fine motor skills, and as a temporal guide. The speed of a technically challenging exercise or musical passage can be perfected at a slower tempo and then gradually increased in a controlled manner by using a metronome. The metrical clicks of a metronome provide the musician with perfect temporal 'check points' against which he continuously error-corrects the timing of his playing. Through synchronising his playing to the temporal guide provided by the metronome, a musician's sense of time gradually improves.

Modern technology has seen a multitude of software metronome emulations become available, many of them free for download from the Internet. These software devices, although often offering extra features, serve the same function as a traditional metronome by providing a regular, metrical click. Extra features might include the ability to choose different click sounds or add subdivisions, but the essential functionality is the same. It seems then, that the options and possibilities offered by Information Technology have not really been explored in terms of alternate timing training methods and systems.

Research into the cognitive functioning of temporal perception and action in musicians and non-musicians is extensive (see Chapter 2), but little research appears to exist into the actual development and training of a person's sense of time. Meegan et al. [2000] have indirectly demonstrated that training and improvement is possible, but other research findings indicate that it is only measurable over the long term [Drake and Palmer, 2000].

The intention of this research is to investigate the efficacy of timing training systems that make use of computers and that go beyond the use of a standard metronome. To achieve this an Alternate Timing Training System was designed and a prototype software implementation was built (Chapter 4). To assess the efficacy of the system an experiment was conducted over a number of weeks using a selection of volunteers. The experiment is described in detail in Chapter 3, with a discussion of the results in Chapter 5. It must be emphasised that it is not the software prototype that is being evaluated, but rather the efficacy of the underlying training system itself, as implemented by the prototype.

Due to the long-term nature of timing training [Drake and Palmer, 2000] and the limitations of the conducted experiment, a public survey was conducted in order to assess perceived benefits of the Alternate Timing Training System by gathering opinion from domain experts. The software implementation was made available on a public website for download. Along with the website an online survey was designed consisting of 16 questions relating to the Alternate Timing Training System implemented by the software. A number of public online music forums were canvassed for potential survey candidates who were asked to evaluate the prototype software for a period and then complete the survey. Survey results are presented and analysed in Chapter 5.

An attempt was made to answer the following research question:

- Can the timing training function of the standard metronome be improved upon using information technology and computer-assisted training?

More specifically:

- Can a musician's sensitivity to tempo and temporal subdivision, and therefore timing accuracy, be improved more effectively using computer-assisted training than using a standard metronome?

Other benefits of computer-assisted timing training will also be assessed, such as potential use as a teaching tool, and as a means to add variety and interest to what is essentially repetitive drill-type training.

Chapter 2

Background and Related Work

Investigation appears to reveal little research into effective training methods for the enhancement and development of temporal perception. This section will examine some existing timing training software (metronome emulations) in an attempt to understand the underlying training systems implemented. Also discussed is some of the work that has been done in the field of cognitive psychology, specifically in the areas of sensorimotor synchronisation and the processes controlling the sensory and motor systems in terms of perception and action. Applicable research into computer-assisted instruction within music education will also be reviewed.

2.1 Music in Time: Pulse, Tempo and Subdivision

Most western music has a primary pulse or beat, more obvious in some musical genres than in others. The primary pulse can be loosely defined as a temporal interval that functions as a common denominator within a piece of music. All musical events that take place, whether notes or durations of silence, are related to this common denominator by some simple ratio [Duke et al., 1991], often a multiple of 2 or 3. It is this pulse around which the music takes place, and is the temporal framework that a musician uses to time the actions that produce sound on a given musical instrument. A musician with a poor sense of time will be unable to create the notes at precisely the right time, resulting in a poor

quality performance.

There are two closely related temporal skills that need to be developed by a musician, namely sensitivity to tempo changes (tempo modulation) and the ability to subdivide temporal intervals.

Different styles of music require different approaches to tempo modulation (speed increase or decrease). Many genres, such as classical music, make use of tempo modulation as an expressive device, or as means of outlining musical phrasing within a piece of music. Speeding up at appropriate points in a performance can add to the sense of excitement or build tension for the listener. Western popular music, on the other hand, is generally recorded to a strict tempo that does not vary even slightly for the entire duration of the composition. A trained musician therefore needs to have a highly developed sense of tempo, especially when performing for a recording. Sensitivity to tempo changes implies the ability to accurately maintain a consistent tempo throughout a musical performance, without speeding up or slowing down.

Although most music has a fairly obvious primary pulse, a large proportion of musical events (notes) do not coincide with the pulse. The temporal distance, or inter-onset interval (IOI) between pulse points can be evenly divided into smaller temporal intervals to provide further points for temporal note placement. How the primary pulse is divided depends on various factors, such as the time signature, style and tempo of the music being played (see Figure 2.1). Sensitivity to subdivisions implies that a musician needs to be able to accurately subdivide the primary pulse to enable the precise placement of notes that do not coincide with the primary pulse but with a subdivision point.

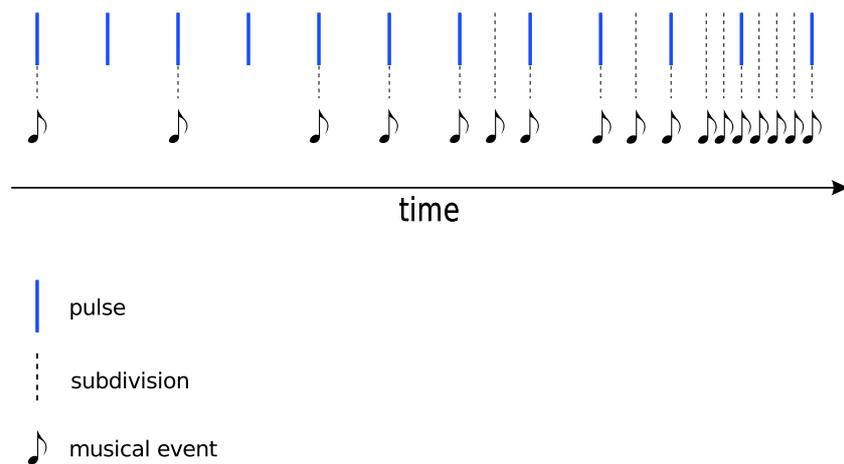


Figure 2.1: Musical Events

The two skills are closely related, and a musician performing music that has a very slow tempo will tend to subdivide the primary pulse itself to enable better timekeeping. For tempi below 60 bpm (beats per minute), musicians will subdivide the beat into smaller temporal intervals, whereas non-musicians will hear and reproduce only the slower beat [Duke et al., 1991].

2.1.1 Maelzel's Metronome

Maelzel's Metronome has been the de facto standard for musical timing training for nearly 200 years. Invented by Dietrich Nikolaus Winkelt in 1812 and patented by Johann Nepomuk Maelzel in 1916, it was originally a mechanical device that used a pendulum to produce an isochronous¹ click. It had a sliding weight on the pendulum arm that was used to set the desired tempo. Initially used to determine the tempo for a piece of music up front and then stopped and set aside, modern usage has the musician synchronising his practice or performance to a continuously clicking metronome.

Although the traditional mechanical metronome is still in use, modern metronomes tend to be either electronic devices or software emulations, often with a variety of features and enhancements. Their usage is still the same, with the musician selecting the desired tempo, starting the metronome and attempting to synchronise whatever he is playing with the click.

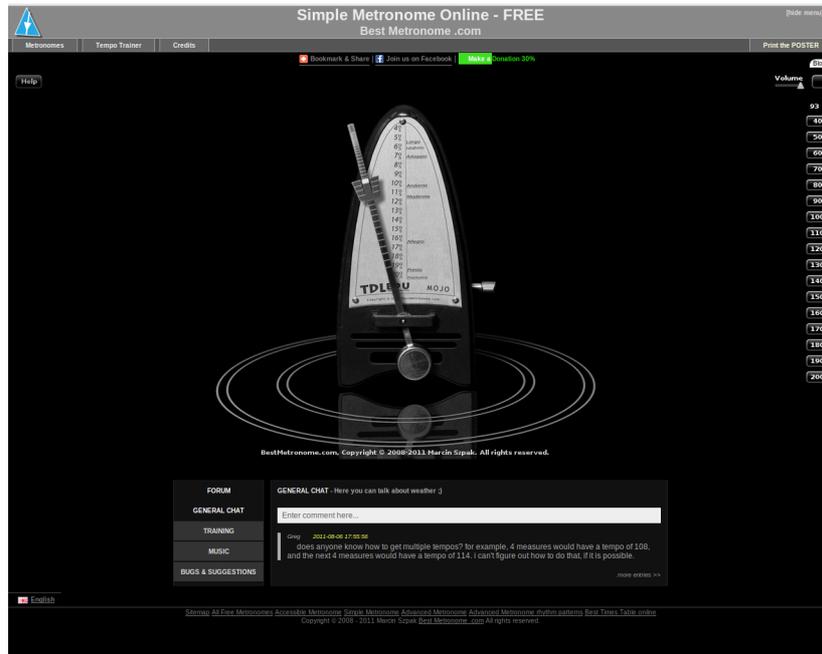
¹Occurring at equal time intervals; having a uniform period of vibration or oscillation [Crozier, 2005].

2.1.2 Software Metronomes

The author was unable to source existing research into the efficacy of timing training with a metronome. An evaluation of a selection of software metronomes and their features follows, as well as an attempt to determine some commonality in terms of underlying training methodology provided.

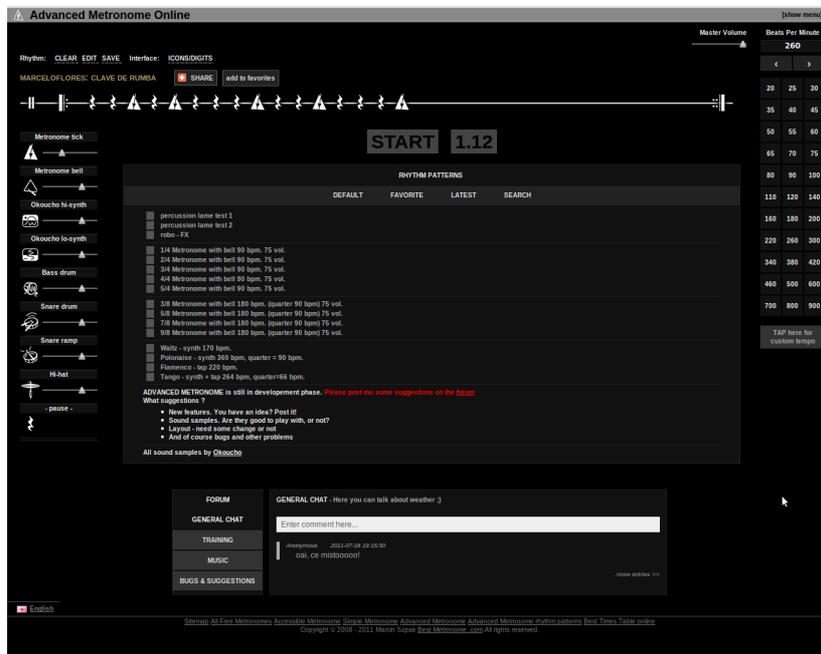
Eight software metronomes were arbitrarily selected for evaluation, with an attempt being made to cover more than just the Windows platform. All of the metronomes have certain basic features: at a bare minimum they all have a means of choosing a tempo and a method of starting and stopping the click. Other fairly common features include a volume control, a facility for tapping to set the desired tempo, the ability to customise the click sounds, and a visual ‘click’ of sorts. Volume control is usually an overall effect, but on some metronomes it is possible to control the volume level of individual click sounds. The tap tempo feature, where offered, is either applied via the keyboard or via clicking an on-screen button with the mouse. Metronomes with customisable click sounds usually offer a selection of built-in sounds, with some having the ability to load any sound in the correct audio format. Besides an audible click, many metronomes provide visual feedback via a flashing icon or visual bouncing ball or some other visual indicator of the beat. Many metronomes also offer some kind of preset management for the saving and loading of settings. The available tempo ranges on the chosen metronomes vary considerably, with some offering traditional ranges (40-208 bpm) and others having much wider ranges (20-6000 bpm).

Best Metronome (Simple)



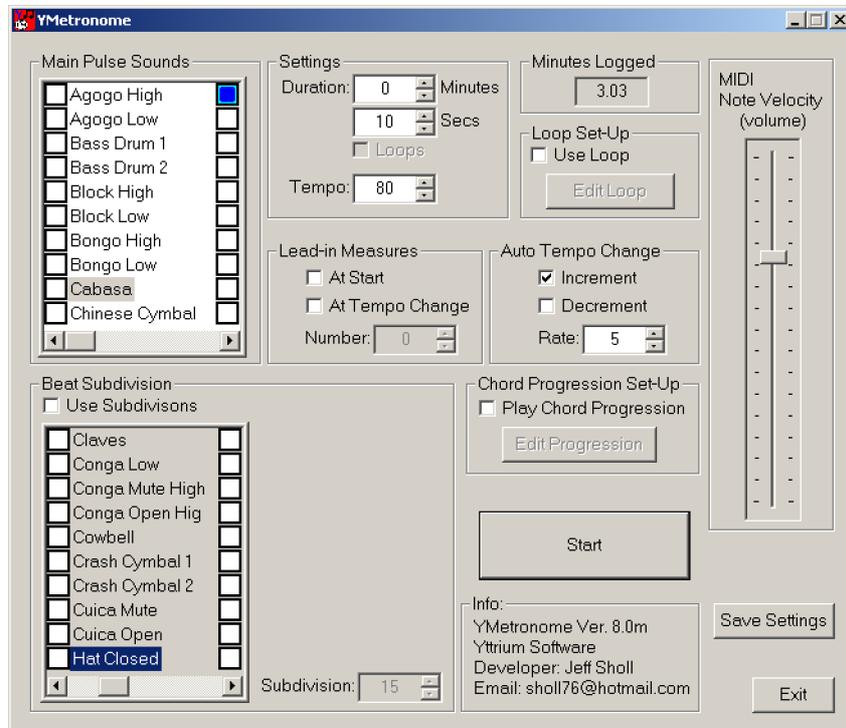
Web	http://simple.bestmetronome.com
Platform	cross-platform (online, flash)
Tempo Range	40 – 208 bpm
Features Summary	Single sound, isochronous click, volume control
Description	An online software emulation of a typical mechanical metronome where the user selects the desired tempo using either the slider or the tempo buttons. The user can start and stop the click either using the mouse and the on screen button or the spacebar.

Best Metronome (Advanced)



Web Platform	http://advanced.bestmetronome.com
Tempo Range	20 – 6000 bpm
Features Summary	Multiple sounds, pattern sequencer, shared pattern library, volume control (both overall and individual sounds), tap tempo, preset management
Description	As the name suggests, this is a far more advanced online metronome with a number of features. The user has a number of different click sounds to choose from which can be arranged in any order. In addition, the user has the option of creating patterns that include rests (periods of silence), thereby facilitating the creation of complex, non-isochronous sequences. Patterns can be saved and shared with others via an online library.

YMetronome 8.0m

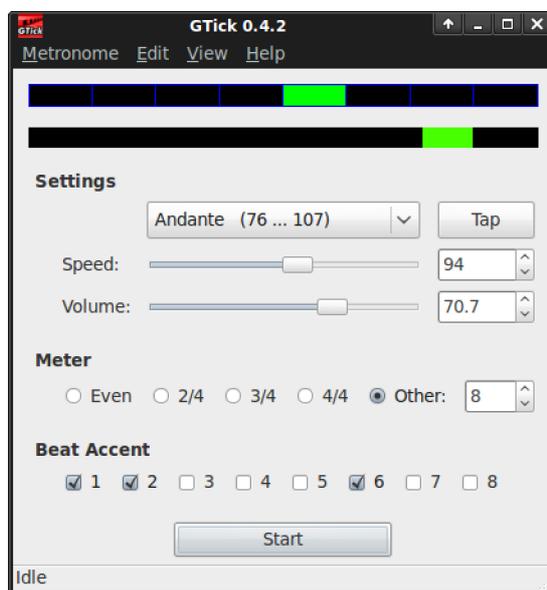


Web	http://www.raleighmusicacademy.com/links.html
Platform	Windows
Tempo Range	10 – 300 bpm
Features Summary	User-selectable sounds for beat and subdivisions (MIDI), auto tempo change after user-defined period, complex metres using the loop functionality, lead in measures, customizable subdivisions, volume control

YMetronome 8.0m cont...

Description This desktop metronome offers some features not found in other, simpler metronomes. It provides user-selectable subdivisions (up to 15 per beat) and looping functionality that enables the creation of complex metres. Its distinguishing feature is the ability to define an automatic tempo change after a specified period of time. For example, the YMetronome can be set up to increase the tempo by 5 bpm every minute. This is a step-wise increase, not a gradual increase. This feature could be used to train motor skills for speed and stamina by gradually pushing the tempo faster and faster. However, all possibilities are isochronous and no custom, non-isochronous patterns are possible.

GTick 0.4.2



Web <http://www.antcom.de/gtick/>
 Platform Linux
 Tempo Range 10 – 1000 bpm, customisable

GTick 0.4.2 cont...

Features Summary	Different metres (Even, 2/4, 3/4, 4/4, more), customisable sounds for beat and accents, customisable accent table - any beat within a chosen metre can be accented, tap tempo, volume control, visual tick, preset management
Description	A linux desktop metronome, GTick has a number of extra features, most notably an accent table. This allows the user to accent any beats within the current metre. This gives a semblance of custom pattern creation, but the metronome is still isochronous. Other features include user-selectable sounds for both beats and accents, preset management for the current settings and irregular metres.

Metronome 4.0



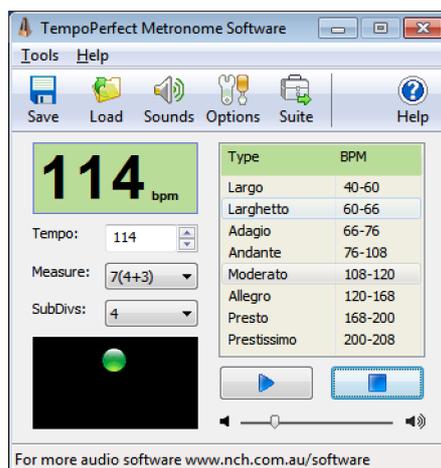
Web	http://www.earpower.com/metronome.php
Platform	Windows
Tempo Range	1 – 360 bpm
Features Summary	Custom metres, MIDI or built in wave sounds, independent volume control per sound, pan per sound, visual tick and beat counter, various double and half time buttons

Metronome 4.0 cont...

Description

This desktop metronome offers features that are common in other similar metronomes. Custom metres are possible through the use of different click sounds, both built-in wave sounds and platform-provided MIDI sounds. Features that appear unique to this software are the ability to double or halve the current tempo using a single button click, as well as the ability to pan individual sounds left and right within the stereo field.

TempoPerfect Metronome Software 2.02



Web

<http://www.nch.com.au/metronome>

Platform

Windows

Tempo Range

40 – 280 bpm

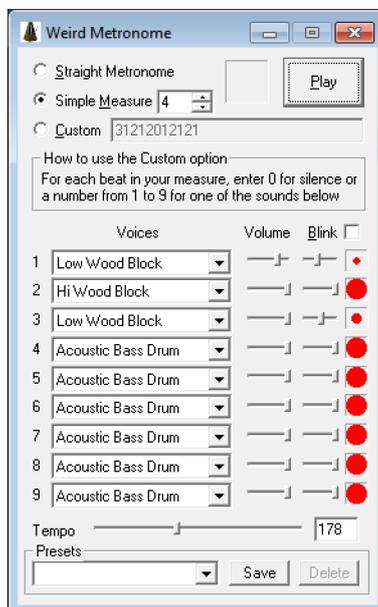
Features Summary

Configurable metres (none, 2 - 8), with accented groupings for odd metres, e.g. 7 (3+4), custom sounds for accents and beats, subdivisions, preset management, visual bouncing ball, volume control

Description

Another free desktop metronome for the Windows platform, this one also features irregular metres with accented groupings, should the user wish to work with odd time signatures. Beats can also be subdivided into smaller portions. Presets can be easily saved and loaded.

Weird Metronome 1.4



Web	http://www.pinkandaint.com/weirdmet.shtml
Platform	Windows
Tempo Range	4 – 1000 bpm
Features Summary	3 Modes of operation: straight (simple isochronous click), simple measure (accented measure), custom (user-defined pattern). Multiple sounds, independent volumes, visual ‘blip’, preset management
Description	Weird Metronome is slightly unusual in its usage in that it has three different modes of operation. The first mode, <i>Straight Metronome</i> is a single sound, isochronous click with no accents or subdivisions. The second mode, <i>Simple Measure</i> uses two different click sounds to provide an accented measure of user-definable length. This is useful for odd time signatures. The third mode, <i>Custom</i> allows the user to create patterns of different sounds. Patterns can be of arbitrary length, and can included silent beats (rests) with non-isochronous results. This functionality is similar to that offered by the advanced version of the online metronome <i>bestmetronome.com</i> .

Fine Metronome 3.5.0



Web	http://www.finemetronome.com
Platform	Windows
Tempo Range	1 – 999 bpm, configurable
Features Summary	Configurable metres, subdivisions, editable patterns, volume control (overall, individual sounds), tap tempo, custom sounds (.wav) many built in, export to .wav/.mp3, preset management, optional visual click
Description	Fine Metronome is a commercial product offering a variety of flexible features. Along with fairly common features like tap tempo and configurable metres it also offers user-editable patterns that can be created within a pre-selected ‘metre template’. These non-isochronous patterns are displayed on screen using the groupings of the chosen metre. The user also has the ability to load custom sounds into the metronome. A unique feature amongst the metronomes evaluated here is the ability to export audio to either .mp3 or .wav format.

Although the metronomes evaluated all have different user interfaces and all offer slightly different features, the underlying fundamentals are essentially the same: provide a temporal grid-like framework within which the musician is supposed to perform (see again Figure 2.1). In most cases (5 out of 8 metronomes) this temporal grid is isochronous, meaning the same cyclical pattern is repeated indefinitely until the metronome is stopped. Two of the eight metronomes have the ability to sequence much longer, non-isochronous patterns of sounds. This functionality is closer to that of audio sequencing software².

²Software used for sequencing audio events along a timeline such as music recording and mixing applications.

In the subsequent section on cognitive psychology the notion of an internal clock is discussed. This is essentially the development of an internal temporal grid-like framework within the musician. The musician, through training, improves his own sense of time so as to provide an inner metronome, or clock. This means that in an environment where there is no metronome, a musician can still perform with an accurate sense of time. All the metronomes evaluated here provide the musician with the external temporal framework, and while improvement of his inner clock may take place, none of the metronomes require the musician to rely on this inner clock at any point during training. This research is an attempt at developing a possible alternate timing training method which more pro-actively targets the improvement of the musician's sense of time by gradually increasing reliance on the inner clock as part of the method.

2.2 Cognitive Psychology: Perception and Action

There is a substantial body of research into the way in which humans perceive and perform music. Various theories have been proposed on what controls the temporal aspects of music performance, including various inner clock and timekeeper models. The two primary cognitive systems involved in music performance are the sensory system (perception) and the motor system (action).

A few applicable research areas are briefly covered below. Typical equipment used when conducting experiments in this area is a set of headphones for delivery of aural pacing stimuli and a key that is tapped by the subject in different ways, depending on the requirements of the specific research.

2.2.1 Sensorimotor Synchronisation

Sensorimotor synchronisation is the synchronisation of movement to an external stimulus, such as tapping in time to a metronome click. Various aspects of sensorimotor synchronisation have been studied, such as discrimination of tempo modulation, error correction in response to changes in stimulus and various thresholds of tempo and just noticeable differences, amongst many others [Repp, 2005b].

A well known but not very well understood phenomenon that has been observed in sensorimotor synchronisation studies is a synchronisation error referred to as *negative asynchrony*. When tapping in time to an isochronous click, the vast majority of test subjects demonstrate an anticipation error of between 20

ms and 80 ms; they consistently tap slightly early. This is even true of professional musicians, although the mean asynchrony is substantially smaller for musicians (-14 ms). This can be interpreted to mean that musicians have a more highly developed sense of time than non-musicians, further evidence that sense of time can be improved through training, whether directly or indirectly. It has further been demonstrated that replacing the pacing click with a click generated by the tap itself partway through the experiment results in the subject speeding up, possibly in a subconscious attempt to create the anticipation error [Aschersleben, 2002]. A typical negative mean asynchrony of between 20 ms and 80 ms is fairly large when considering that the threshold of perception in determining two distinct sounds occurring close together is around 2 ms. The threshold for ascertaining the order of two sounds played close together is slightly larger at around 20 ms, but still significantly smaller than the typical negative asynchrony [Hirsh, 1959, Friberg and Sundberg, 1993].

Kuhn [1974] demonstrated a possibly related phenomenon by showing that professional musicians were significantly quicker at detecting a decrease in tempo than they were at detecting an increase. A study was conducted in which subjects listened to recordings of a metronome that either increased in tempo, decreased or remained the same. One of the recorded variables was the time taken to identify a change, with the mean detection time for decreases approximately 2 seconds less than for increases. Kuhn proposes that based on these findings one would expect tempo related performance errors would tend towards increases rather than decreases. This is consistent with the phenomenon of negative asynchrony.

A number of different theories have been put forward on the relationship between musical rhythm and associated movement. Palmer [1997] discusses some of these models, identifying two primary views: movement causes musical rhythm and musical rhythm causes movement. In either case, some form of internal timekeeper needs to control the placement of musical events in time.

Internal Clock

In order for a musician to consistently reproduce a piece of music an internal clock is needed. This is used by the motor system and motor program to execute the necessary physical movements needed to play the piece of music. The internal clock provides a continuous timing reference against which musical events are set by the player's motor system. For a musician to be able to perform the same piece of music at different tempi, the internal clock rate must be adjustable, meaning that at some level intervals between temporal events have a

relative relationship [Shaffer, 1981].

Evidence to suggest that the internal clock, or timer, controls both the sensory and the motor system has been demonstrated in studies that showed improved motor timing after subjects underwent sensory-only timing training. During training, subjects listened to specifically timed aural stimuli and were asked to repeatedly identify the longer of two temporal intervals. Motor tests that were administered before and after the 5 training sessions showed significant improvement in ability to match the trained temporal intervals, supporting the idea that both the sensory system and the motor system are governed by the same internal timing mechanism [Meegan et al., 2000]. The findings of Meegan et al. are significant in the context of this research in that they demonstrate that whatever the psychological or neurological basis for an internal timekeeper, the fact is that *it can be trained*.

The temporal structure of music is hierarchical, consisting of a variety of temporal structures. Starting with the overall length of a particular performance or piece of music it can be further divided into movements, sections, verses, phrases, bars, beats and beat subdivisions. Timing accuracy in music performance can be attributed to a number of related musical skills. A musician needs to be able to perform without stopping, even when errors are encountered. Although related to the internal clock, this skill is more about the reproduction of musical notes, whether from an internal representation of the music or from external sheet music. Accuracy of note placement with respect to the beat is another skill that needs to be acquired by the serious musician. Drake and Palmer [2000] have shown that improvement in beat-accuracy skill is an aspect of long term learning, and does not take place over the short term.

2.2.2 Skills Acquisition

Music perception and performance involves a complex set of skills that can include technical facility (motor skills), aural perception, the reading of music notation and improvisation skills, amongst others. Auditory imagery, or *auralisation* as it is sometimes called, is the aural equivalent of visualisation. It is the ability of a musician to hear internally in his ‘mind’s ear’. As discussed by Covington [2005], this skill is rarely addressed directly as something to be developed, but is expected to occur almost as a by-product of other musical training such as ear-training and sight-singing. She states that auditory imagery “is strongly correlated with aural perception, in that similar and often the same brain areas are activated. The implications for training the skill of auditory imagery are that similar techniques may be used, but there is the need

to find additional ways of training and refining the skill.”

Timing training with a standard metronome is usually coupled with other musical skills development, for example the metronome will be used to improve motor skills by gradually increasing the tempo of a particular exercise. As such the development of a musician’s temporal sense is not something typically isolated and focused on directly. Similar to the skill of auralisation, new and improved training methods could benefit the musician, rather than simply expecting that through normal practice and performance, and through listening to music, the musician’s inner clock will improve over time. A standard metronome is sometimes used as part of a practice routine, or during tuition, but is generally used as a temporal guide rather than as tool for fine-tuning the inner clock.

In the context of musical improvisation, Moses [1989] proposes a slightly different view of internal hearing. He believes that the performance (external) is not necessarily the same as the internal representation, but is based on a simple internal musical structure such as a melody. This internally heard structure is used as inspiration for the external improvisation, something to “play off of”.

Modern neurological mapping tools such as transcranial magnetic stimulation (TMS) have enabled research into the physical changes that occur in the brain during motor skills acquisition [Pascual-Leone, 2001]. Both the motor system and the sensory system are involved in and are changed by the training that takes place when learning a musical instrument. New connections are made in the brain, and certain sensorimotor connections that already exist are unmasked [Pascual-Leone, 2001, Schlaug et al., 1995]. Due to the long term nature of learning a musical instrument, especially considering the number of years required to attain a high level of proficiency, very few long term studies have been conducted.

2.2.3 Related Work

Libkuman et al. [2002] conducted an interesting experiment into whether timing training using a metronome would improve a person’s golf swing. They found fairly strong evidence in support of improvement as a consequence of the training, although they note that future research is needed to rule out several other possible factors. Their findings support those of Meegan et al. [2000] who found significant improvement in motor timing skills subsequent to sensory-only timing training.

Other related research into the relationship between timing training using a metronome and improved motor skills has been conducted in various areas

of medicine and therapy: gait training for Parkinson's suffers using recordings of a metronome showed significant improvement in stride and velocity [Thaut et al., 1996], gait training for stroke rehabilitation [Thaut et al., 1997] and metronome-paced speech therapy for stutterers [Brady, 1969, Hutchinson and Navarre, 1977]. Metronomes have also been used in training for children with Attention Deficit Hyperactivity Disorder (ADHD) [Shaffer et al., 2001, Cospers et al., 2009] and blood pressure reduction through metronome-conditioned relaxation [Brady, 1973, Brady et al., 1974].

2.3 Computer-assisted Music Instruction

A significant amount of time is needed to develop the skills needed to master a musical instrument. A musician must spend a number of hours every day, usually for many years, to reach a high level of proficiency. Students involved in secondary education music curricula or tertiary music programs have little time to spare, and the typical one-on-one teaching requirements of traditional music instruction make skills acquisition time consuming for both student and teacher [Chan et al., 2006, Kuhn, 1974b].

The acquisition of skills such as the motor skills required for music performance, relative pitch perception and music notation skills mostly require drill-type practice for improvement to take place. Any instrument requires fine motor skills for the coordinated execution of fingerings, breathing and other movement to produce music. Relative pitch perception, the ability to discern the musical distance between notes, is also developed through repetitive drilling. The reading of music notation, once understood at a semantic level, needs to be practiced to the point where the translation of notation into music takes place in real time (sight reading).

The repetitive nature of all of these tasks lends itself to the use of computers as a practice aid. Typical of music software such as *Band-in-a-box* created by PG Music Inc. is the ability to loop a section of music, repeating it indefinitely [Nardo and Choe, 2010]. Although not a metronome, this functionality demonstrates one of the strengths of using a computer as a training aide: drill-type exercises and tasks can be repeated indefinitely. Computers and software can also be used both to present guided exercises to the user and to monitor progress.

In related studies [Dalby, 1992, Willett and Netusil, 1989, Kuhn, 1974b, Deihl and Zeigler, 1973], the effectiveness of Computer Based Instruction (CBI) has shown positive results in favour of CBI. Arguments for using CBI include

the self-paced nature of CBI and the fact that the presence of an instructor is not required, for example CBI “makes it possible to provide intonation training without impinging on precious classroom time.” [Dalby, 1992]. Dalby conducted a study using CBI for the training of harmonic intonation perception (the ability to assess whether something is out of tune in an ensemble situation), something difficult to achieve without computers due to the logistical requirements of providing a live ensemble as a training aid. Chan [2006] has shown that the use of commercially available software (*Teach Me Piano Deluxe*) significantly increased the note reading skills of students at the secondary school level.

Some of the negative aspects of computer-assisted music instruction as expressed by tertiary music students surveyed by Ho [2007] are that software cannot offer specific solutions to any problems that a student may be experiencing, whereas a music teacher can provide immediate and appropriate feedback and solutions. Although the students surveyed had full access to various technology and multimedia learning aids, they still perceived their primary source of knowledge to be their instrumental coaches.

2.4 Summary

The use of computers as an aid to the training of a musician’s cognitive inner clock has many potential benefits that are typically associated with computers and software as a teaching aid. Students do not require the presence of their instructors, and they can work at their own pace. It is possible that exercises could be computer-generated, meaning that there is a practically endless source of training material. Software could be structured to provide guided and graded modules in a lesson format, and could be used to monitor and ultimately assess a student’s progress, all without requiring direct input from an instructor and without taking up classroom time. The repetitive drill-type nature of timing training is ideally suited to the use of computers as an aid.

Chapter 3

Methodology

3.1 Overview

Evaluation of the Alternate Timing Training System (described in detail in Chapter 4) was approached in two ways: an experiment was conducted using a number of volunteers in an attempt to measure changes in timing accuracy, and a survey was conducted in order to assess perceived benefits of the system. This chapter will describe the methodologies used to run the experiment, as well as the methodologies used to design and conduct the survey.

3.2 Experiment

An experiment was designed with the intention of determining the efficacy of the Alternate Timing Training System in direct relation to the training implemented by a standard metronome¹.

Members of a group of volunteers were individually assessed for timing accuracy, and were then asked to use a standard software metronome for 20 minutes a day for a period of two weeks. They were then re-assessed and asked to use the Alternate Timing Training System for 20 minutes a day for two weeks. Subjects were then re-assessed for a third and final time. The same subjects were used for both treatments in the experiment, a *within-subjects* design.

A *within-subjects* design was used for the experiment for two reasons:

¹In the context of this thesis, ‘standard metronome’ refers to a simple software metronome developed for the experiment that generates an isochronous audio click. The user is able to set the tempo and choose the sound of the click.

1. The individual subjects were musicians at varying levels of development. To minimise differences the same subjects were used in both treatments.
2. The number of suitable subjects available was limited. Using the same subjects for both treatments effectively doubles the sample size available for analysis.

A usage log was necessary in order to track how much subjects made use of the two training systems. This required a custom implementation of a standard software metronome that included usage logging functionality. Usage logging was also built into the Alternate Timing Training System implementation. Logging data for each subject were parsed and written to a SQL database. Observation data were also subsequently extracted from the timing assessments, transformed and written to the database for later query and analysis.

The goal of the experiment was to compare the rate of improvement using a standard metronome to the rate of improvement using the Alternate Timing Training System implemented by the prototype software.

3.2.1 Assessment

In order to assess the efficacy of the two different timing training systems a method of assessing an individual's sense of time was needed. As discussed in Chapter 2, a musician develops two related skills when it comes to sense of time: sensitivity to changes in tempo (tempo modulation), and the ability to subdivide beats (temporal period) into smaller, evenly spaced periods. An assessment was designed that could be consistently applied to participants at different stages of the experiment, yielding data that were used in the results analysis.

How does one measure timing accuracy? A simple method is to record the subject playing some specified notes on his instrument in time to an isochronous reference click, just as he would while practicing with any standard metronome. The recording can then be analysed for timing errors, giving an indication of the subject's sense of time.

Synchronising to slower tempi is harder than synchronising to medium or faster tempi [Repp, 2005a, Mates et al., 1994]. Therefore an assessment of timing accuracy should cover a range of tempi, and sensibly move from easier tempi (medium) towards more difficult (slower). To fulfil this requirement an audio click 'template' was created (see Figure 3.1) consisting of 7 different tempi ranging from 100 bpm (medium tempo) down to 20 bpm (extremely slow tempo), each playing for between 20 and 50 seconds. Each tempo was separated by a

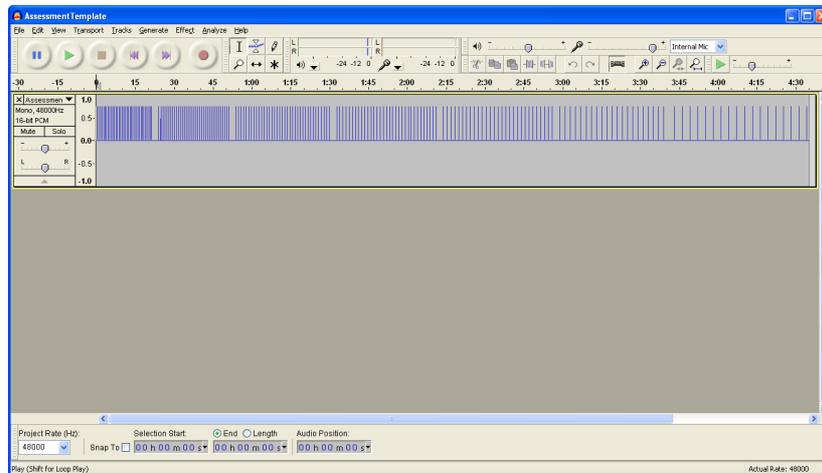


Figure 3.1: Click Template

period of silence of approximately 2-3 seconds, indicating to the subject that one tempo was complete and that a new tempo was about to commence. In summary, the assessment click template consisted of:

- 22 seconds at 100 bpm
- 28 seconds at 80 bpm
- 36 seconds at 60 bpm
- 39 seconds at 50 bpm
- 43 seconds at 40 bpm
- 41 seconds at 30 bpm
- 52 seconds at 20 bpm

Similar to the way a listener would clap along to music, subjects were recorded playing single notes that coincided with each click of the template. Although it seems that this would not test a subject's ability to subdivide the beat, once the assessment progresses to the slower tempi the subject is forced to mentally subdivide the larger temporal distances between clicks in order to predict the next reference click. See Figure 3.2 for a screenshot of an assessment recording. Note that the upper track is the click template and the lower track is the subject's attempt to match the template.

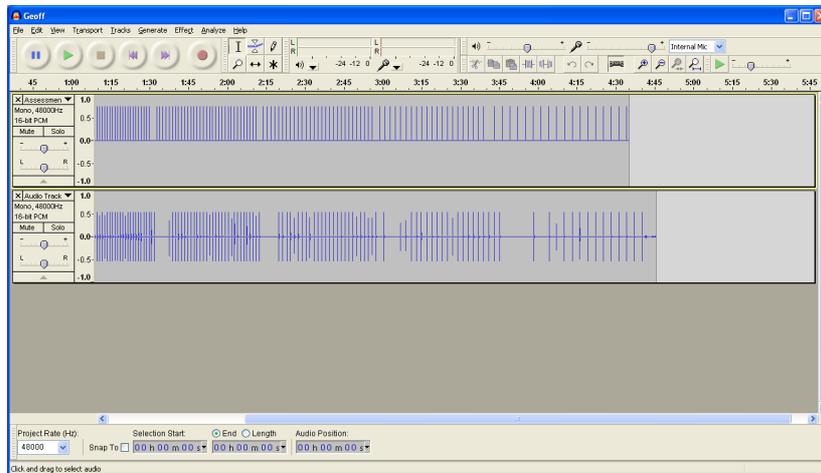


Figure 3.2: A recorded assessment

Recording Equipment

Equipment used for recording the assessments consisted of a laptop computer running Windows XP, an M-Audio Fastrack Pro digital audio interface and a Samson C01 condenser microphone. Performances were captured using the open-source multi-track digital recording software *Audacity* (<http://audacity.sourceforge.net>). A standard wooden drum practice pad was inverted and placed at a comfortable height chosen by the performer, who was asked to use the provided plastic-tipped drumsticks for the recording. The microphone was positioned approximately 20cm from the wooden surface of the practice pad, and a little to one side so as not to interfere with the beating of the sticks. Every effort was made to replicate this setup as closely as possible for every recording.

Data Extraction

In order to extract timing accuracy data from the assessment recordings a note onset detection plugin was used from Paul Brossier's *Aubio* library (<http://aubio.org>). Used within Audacity, the audio recording application used for the assessments, it produces a *label track*, a synchronised parallel track of text labels (Figure 3.3). Each label corresponds to a detected note, with the text of the label indicating the precise time at which the note occurred, measured in seconds and milliseconds from the beginning of the track. These labels can then be exported for further manipulation and analysis.

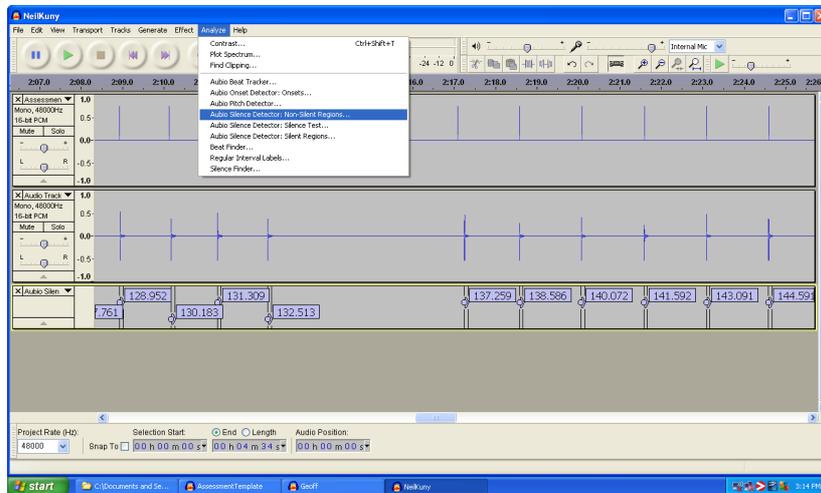


Figure 3.3: Note onset label track

In order for the onset detection algorithm to work reliably, and for the resulting data to be accurate, it is important for recorded notes to have a definite, consistent onset. This was achieved by requiring all assessment recordings to be made using a plastic-tipped drumstick against a piece of wood. Each subject used the same drumstick and piece of wood for his assessment recording, producing notes with a clearly visible (and detectable) onset. Figure 3.4 illustrates the sharp attack of a recorded note (bottom track).

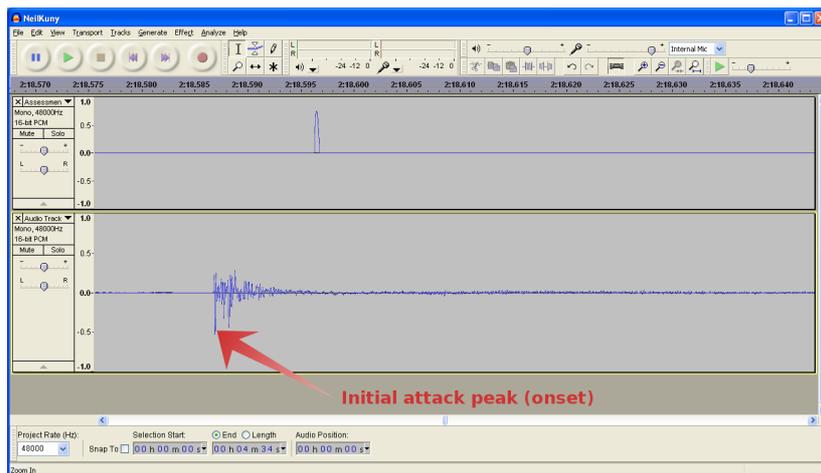


Figure 3.4: Zoomed in on recorded note

3.2.2 Usage Control

To assess the efficacy of the two timing training systems it was necessary to track the length of time spent using them. This was achieved by building usage logging functionality into the software. This requirement also necessitated the development of a standard software metronome that included the necessary logging functionality. The ability to email the generated log files directly to the author was provided via a button at the bottom of the user interface (see Figure 3.5).

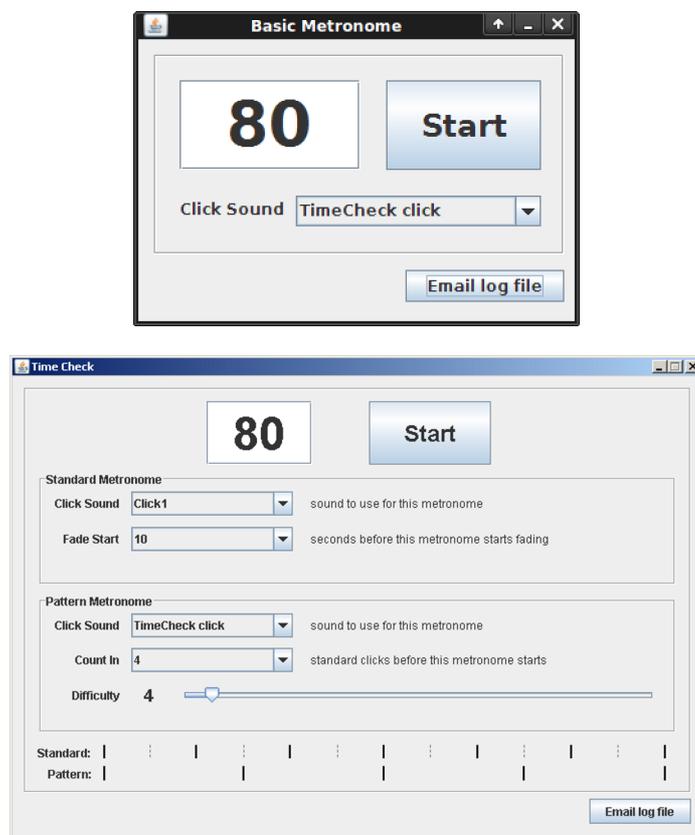


Figure 3.5: Basic Metronome and Alternate Timing Training System showing the ‘Email log file’ buttons.

The learning that takes place when executing motor-sequences in order to develop both motor skills and the inner clock is subject to the *massed vs. distributed practice effect*. Factors such as physical and mental fatigue mean that spending small amounts of time spread over a number of days is more effective than spending a larger amount of time all at once [Parncutt and McPherson,

2002]. To try to reduce this effect on the experiment subjects were limited to spending 20 minutes per day with whichever training system they were currently using. This was also an attempt to get all subjects to spend a similar amount of time with the training systems every day. The time limit was enforced by the software by monitoring the generated log files to calculate the total time spent on any given day. If at any point the time spent was greater than or equal to 20 minutes the user was presented with a message and the application terminated. 20 Minutes was selected as a reasonable time to expect subjects to spend without becoming fatigued in any way.

Usage tracking consisted of recording timestamped application events in a text file on the subject's local file system. Events recorded included application start-up and shutdown and the starting and stopping of the audio clicks. Current settings were recorded along with the events, such as tempo selection, level, pattern number and subdivision (discussed in Chapter 4).

The following is an example of the logging generated by the standard software metronome:

```

2009-10-13 08:13:33,593  INFO                Main: Basic Startup
2009-10-13 08:14:06,562  INFO          BasicMetronome: START  Tempo: 100.00
2009-10-13 08:14:09,953  INFO    AbstractMetronome: STOP
2009-10-13 08:14:15,859  INFO          BasicMetronome: START  Tempo: 080.00
2009-10-13 08:33:42,140  INFO    AbstractMetronome: STOP
2009-10-13 08:33:49,578  INFO    Main$ShutdownHook: Basic Shutdown

```

A fixed-width format was used to allow for easier parsing when extracting data for analysis. Each application event was recorded with a timestamp and information relevant to the event. Four types of event were recorded:

1. Application startup
2. Application shutdown
3. Training run start - in addition to the timestamp, this item recorded additional information regarding the state of the application, such as the current tempo
4. Training run end

An example of the logging generated by the Alternate Timing Training System software implementation:

```

2009-11-22 18:59:13,468  INFO                Main: TimeCheck Startup
2009-11-22 19:00:13,843  INFO    TimeCheckMetronome: START  Tempo: 080.00

```

```
Level: 009 Pattern: 02 Subdivision: 03
2009-11-22 19:07:42,843 INFO AbstractMetronome: STOP
2009-11-22 19:07:59,046 INFO TimeCheckMetronome: START Tempo: 090.00
Level: 013 Pattern: 07 Subdivision: 03
2009-11-22 19:19:20,390 INFO AbstractMetronome: STOP
2009-11-22 19:19:31,343 INFO Main$ShutdownHook: TimeCheck Shutdown
```

The logging generated by the Alternate Timing Training System software followed a similar format to that of the standard software metronome, only that extra state information was recorded with the training run start events.

The log files facilitated both the recording of each subject's usage of the software and the enforcement of the 20 minute daily time limit. This was achieved purely with the timestamp data – the state information turned out to be unnecessary and was not used.

3.2.3 Subject Requirements

To better control some of the variables in the experiment subjects were selected based on certain requirements.

- Subjects needed to be drummers. The constraints of the assessment recording required subjects to make use of a plastic-tipped drumstick against a piece of wood. To minimise the effects of lack of motor skills when manipulating a drumstick, subjects needed to already be comfortable using drumsticks.
- Subjects needed to be comfortable playing to a click already. This implies some level of musical competency, removing beginners from the sample. Seeing as learning rates are unlikely to be linear, it made sense to eliminate the possibly increased rate of improvement resulting from learning to simply synchronise to an external isochronous click. A musician unable to synchronise to a click would perform very poorly on his or her first timing assessment. After spending two weeks synchronising to a standard metronome the second assessment would show an artificially marked improvement in relation to the first, thereby potentially skewing the results. Even though we are using synchronisation to an external click as a means of assessment, we are not researching an individual's ability to synchronise to an external click, but rather attempting to assess the development of the individual's internal clock and their ability to subdivide the beat.
- Subjects needed to be physically located in the Johannesburg area in order

for them to be recorded for the three assessments using precisely the same recording equipment.

- Subjects needed regular access to a personal computer equipped with an audio interface in order to make use of the software.

A number of music industry professional drummers were approached directly as potential subjects, and a reputable tertiary music institution was approached for more subjects. Other online media such as mailing lists and social networking sites were also canvassed for subjects.

Directly after the recording of their first assessment subjects were provided with a CD containing installers for the standard metronome software (both Mac and Windows). Subjects were asked to use the software metronome for 20 minutes every day for two weeks. After two weeks subjects were recorded again for their second assessment and provided with a second CD containing installers for the Alternate Timing Training System software. The CD also included a short instruction manual and a short demonstration video showing suggested ways of using the software. Subjects spent the next two weeks with the Alternate Timing Training System software, again for 20 minutes a day, after which they were recorded for the final assessment.

A total of 15 volunteers were able to meet the requirements and were recorded for the first assessment. Of the 15, 12 subjects recorded the second assessment and a final 9 subjects recorded the final assessment. 6 Subjects were unable to complete the experiment either by choice or circumstance. Analysis of the log files eliminated a further 4 subjects from those suitable for analysis due them not using the training systems for a sufficient time, resulting in the very small sample size of 5.

3.2.4 Limiting Factors

The motor learning component of timing training does not take place at a linear rate, and is subject to many variables, not least of which is that it takes place in a biological system. Individual subjects will acquire skills at different rates, influenced by many factors such as practice conditions, fatigue, S-curve learning rates, motor-skill transfer, and numerous others beyond the scope of this research (see “Time Scales in Motor Learning and Development” [Newell et al., 2001]). This is one of the arguments for the *within-subjects* design of the experiment, as an individual subject’s rate of improvement needs to be measured and compared with a previously measured rate of improvement for the same individual subject.

A further limitation of the experiment, evident in the results discussed in Chapter 5, is that definite measurable timing improvement only takes place over the long term [Drake and Palmer, 2000].

As experiment subjects were predominantly music students or professional musicians, it is reasonable to presume that they spend a significant amount of time practicing their instruments, possibly using a standard metronome as part of their practice routine. This is likely to have had an effect on the results, but it is impossible to determine whether the effect is significant or not.

3.2.5 Data Extraction

For each subject taking part in the experiment there were two relevant sets of data generated: the software usage data extracted from the log files, and the timing accuracy data extracted from the assessment recordings. To facilitate easier analysis of the extracted data it was parsed and written to a MySQL database. A normalised database design was created that would partition the data to enable the extraction of different trends and features (Figure 3.6).

The unifying relation in the database is obviously the subject, so a *User* table was created. In terms of the usage data that was logged by the software for each subject, the following information was available in the log files:

- time of start-up and shutdown of the application, represented by the *Session* table.
- the type of software being used (either standard metronome or Alternate Timing Training System), represented by the *SessionType* table.
- time of starting or stopping a training run (starting or stopping the click), represented by the *Run* table. Other attributes of a training run were also stored in this table, such as *tempo*, and in the case of the Alternate Timing Training System software, *level*, *pattern* and *subdivision*. Ultimately, recording these attributes was unnecessary as they were not used in the usage analysis.

In short, each user could have a number of sessions (maximum 20 minutes a day), each containing zero or more training runs.

A simple log file parser was developed to extract the data from the log files, generating a series of SQL ‘insert’ statements that were written to a file (see Appendix A, section A.2 for source code). These were then executed against the database to populate the tables.

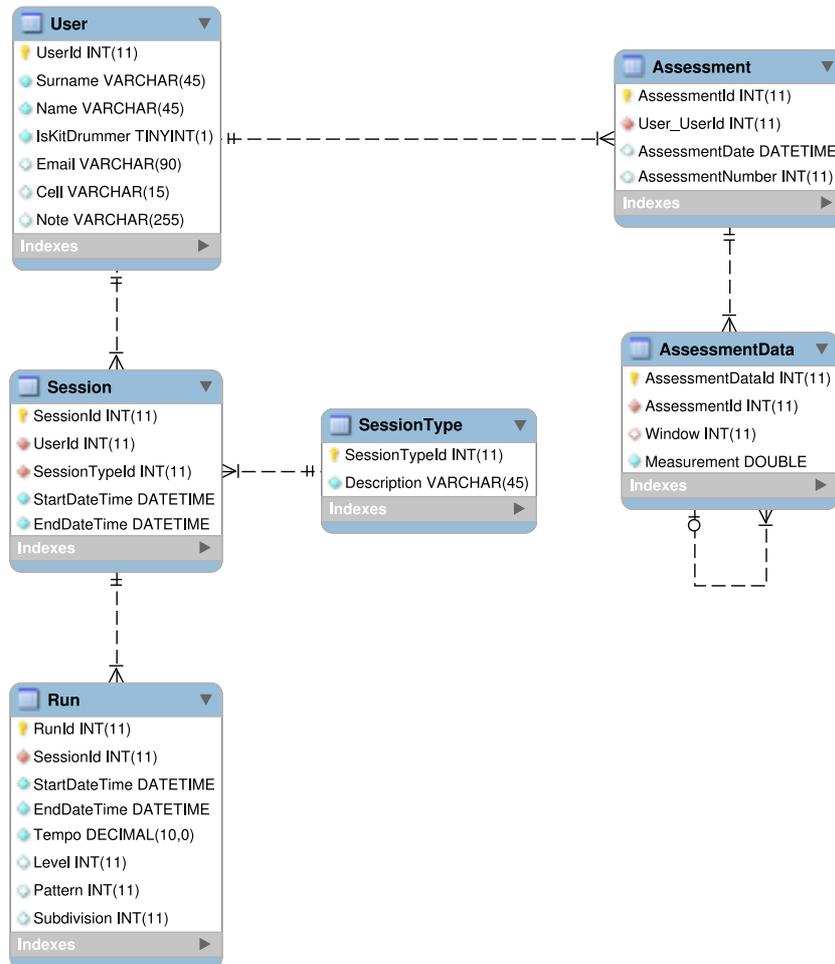


Figure 3.6: Entity Relationship Diagram

As described earlier, a note onset detection algorithm was applied to all the recorded assessments. Applying the algorithm, implemented as an Audacity plugin, creates a *label track* in Audacity. This is simply a placeholder for timeline markers, where each marker has a start time, an end time and a textual description (the label). The algorithm inserts a marker at the start time of each detected note onset, resulting in a marker for each note recorded. Exporting the generated label track from Audacity produces a text file containing an entry for each marker, indicating the exact time of each note onset as a temporal offset from the beginning of the recording, specified in seconds and milliseconds.

This is a short extract from an exported label track (headers have been added for clarity):

start time	end time	text
2.410333	2.476333	2.410
2.999333	3.052333	2.999
3.598333	3.671000	3.598
4.183000	4.268333	4.183
4.792333	4.865667	4.792
5.394333	5.463000	5.394
...		

Using a text editor (<http://www.vim.org/>) and a series of macros the exported label tracks for each assessment were transformed into SQL ‘insert’ statements in order to populate the *Assessment* and *AssessmentData* tables in the database (Figure 3.6).

In order to determine the accuracy with which each note was performed in relation to the reference click it was necessary to create and export the note onsets for the reference click as well, so as to create a ‘perfect’ reference performance. All recorded assessments were compared against this reference performance in different ways to determine timing accuracy (Chapter 5).

The database model for the assessment data consisted simply of an assessment relation representing each recorded assessment for a user (the *Assessment* table), and a relation for the assessment data belonging to each assessment (the *AssessmentData* table). The number of each assessment was recorded (first, second or third assessment), along with the date, and a timing measurement for each recorded note.

Each recorded note has an associated measurement – the instant at which the note occurred, captured as a temporal offset from the beginning of the

recording, measured in seconds and milliseconds. This measurement, however, does not indicate anything about the timing accuracy of the note. The timing accuracy needs to be assessed by measuring how close the recorded note is to its counterpart in the reference performance. In order to determine a recorded note's counterpart in the reference performance it was necessary to infer a temporal window period for each reference click (Figure 3.7). Window periods do not overlap. Any given recorded note should fall within a single window period, thereby indicating which reference performance note it 'belongs' to. It then becomes possible to measure the difference between recorded notes and their reference performance counterparts, and so determine an indication of accuracy. Inferring a window period for a given reference performance note was simply a case of determining the window period size, and then calculating the start and end times from the reference note as window mid-point. Window period size was determined by calculating the absolute difference between the reference note and either the previous or the next note in the sequence. This was achieved firstly by including an extra self-referencing column in the *AssessmentData* table to indicate which other note to use in determining the window period size, and secondly by creating a database view for the reference performance that included the extra calculated columns *WindowSize*, *WindowMin* and *WindowMax*. Using SQL queries, the recorded performances can be compared in various ways to the reference performance for analysis (Chapter 5).

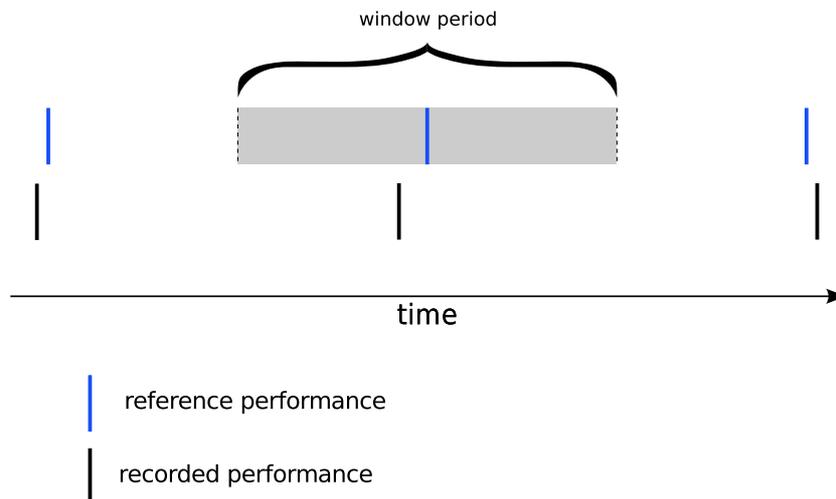


Figure 3.7: Temporal Window Period Surrounding Reference Note

3.2.6 Summary

The experiment conducted over the period of 4 weeks was an attempt to measure and compare the different rates of improvement in sense of time when using a standard metronome and a proposed Alternate Timing Training System as implemented by a prototype software application. Both usage data and assessment data were successfully extracted, transformed and written to a SQL database for analysis. The usable sample size was small, but the partitioning of the data allowed for a variety of options when querying and analysing the data (Chapter 5).

3.3 Survey

Due to the small number of volunteer subjects who completed the experiment it was decided to conduct a public survey to assess qualitative aspects of the Alternate Timing Training System. The intention of the survey was to evaluate the opinions of domain experts with regard to the Alternate Timing Training System implemented by the prototype software. Therefore the target demographic of the survey needed to be expert musicians and music teachers. Expert musicians with a vested interest in having a highly developed sense of time, such as percussionists and drummers, provided an even more specific target demographic, and so an attempt was made to recruit these types of musician for the survey.

Due to the relatively small population of target demographic, especially in terms of local physical access, it was decided that the survey should be offered online in an electronic format. A simple website was designed and hosted that offered the Alternate Timing Training System application for download and the 16 survey questions. Open source software was used to design and administer the survey (<http://www.limesurvey.org>), ensuring that survey responses were saved to a database and were easily exported to other formats for analysis. The website also included a disclaimer ensuring the protection of the participants' privacy and a simple user manual for the downloaded software. See Appendix E for screen captures of the website pages.

The survey was run for approximately 3 weeks. Participants were largely recruited from public online forums by posting a request for software testers and feedback. During the 3 week period a total of 41 responses were received, of which 33 were complete. Participants were all anonymous. See Appendix C for a textual version of the survey questions. Requests for respondents were posted on the following online forums:

- <http://www.drummerworld.com>
- <http://www.pearldrumsforum.com>
- <http://www.onlinedrummer.com>
- <http://www.drumsetconnect.com>
- <http://acapella.harmony-central.com>
- <http://www.drumforum.org>
- <http://www.freedrumlessons.com>
- <http://www.talkbass.com>
- <http://basschat.co.uk>
- <http://www.uglybassplayer.com>

3.3.1 Survey Design

Aside from survey questions directed specifically at opinion of the Alternate Timing Training System, it was necessary to assess both the eligibility of the respondents' opinions and whether the respondents had used the software for a sufficient period of time for their opinion to be valid. Thus three different categories of question were formulated:

- Competency assessment questions
- Application usage questions
- User opinion questions

3.3.2 Competency Assessment – 5 Questions

The answers to this category of question were used to assess the competency of the respondent in terms of their musical training and abilities. Answers could be used to qualify the respondent's opinion of the Alternate Timing Training System application.

Respondents were asked to rate their musical ability, and were asked about their previous timing training experience, as well as their opinions on the value of having a good sense of time. Finally, they were asked what instrument they play as a means of determining whether they fall into the preferred demographic of drummers and percussionists.

3.3.3 Application Usage Assessment – 3 Questions

In order to offer an informed opinion of the Alternate Timing Training System, a respondent would have needed to spend a reasonable amount of time with it. It is also possible that some of the fundamental intentions and concepts underlying the application are not immediately obvious to a first time user. A short instruction manual was made available for download along with the application software, providing basic operation instructions and some usage recommendations. Questions in this category were aimed at determining how much time was spent with the application, including with the respondent's chosen instrument, and whether they had gained an understanding of the underlying intentions by reading the manual. Again, answers in this category could be used to assess validity of answers in the 'Opinion' section.

3.3.4 Opinion – 8 Questions

The remaining survey questions were aimed at assessing public opinion of the Alternate Timing Training System implemented by the prototype software as a training tool, as a possible teaching tool and in comparison to the training offered by a standard metronome. The questions comparing the Alternate Timing Training System to a standard metronome included questions on possible benefits offered by the Alternate Timing Training System. The final question was an optional free form question asking the respondent for any feedback, criticism or suggestions with regard to the Alternate Timing Training System.

3.3.5 Summary

An online survey was conducted to assess expert opinion of the Alternate Timing Training System as compared to the training system implemented by a standard metronome. Survey questions were designed to assess the expertise of the respondents, their usage of the Alternate Timing Training System and their opinion of the Alternate Timing Training System. The survey was completed by 41 individuals, of which 33 answered all mandatory questions. An analysis and discussion of responses is presented in Chapter 5, along with reference to some discussions which took place with users via email and on public forums.

Chapter 4

Implementation

4.1 Design

At a functional level, the way in which a standard metronome works is that it provides the musician with regular temporal reference points in the form of an isochronous click. As the musician plays, he or she makes continual tiny timing adjustments to each note, moving them very slightly earlier or later so as to synchronise as exactly as possible with the reference points. The musician is effectively error-correcting his performance temporally to an external reference. When the time comes for the musician to perform in public, the metronome is not available as a metrical reference, and the musician must rely on his own inner clock for temporal reference. Two different activities have been described here: synchronising a performance to an external reference, and synchronising a performance to an internal reference. Ultimately, it is the musical performance that takes place without the metronome that is the end goal, so it is therefore desirable for the musician to develop the internal reference, or his inner clock.

Practicing with a metronome is the traditional way to improve a musician's inner clock, his sense of time. Through many hours of synchronising performances to an external reference the musician's sensitivity to tempo and subdivision gradually increases and the accuracy of the inner clock improves.

Instead of having two distinct phases – practice with a metronome and performance without – a proposed alternative is to progress gradually from the one to the other. The musician is initially provided with many reference points, and over a period of time the frequency of reference points is gradually decreased. Effectively, this means the musician must gradually move from relying on the metronome for reference to relying on his inner clock for reference. This is

the basic premise on which the following Alternate Timing Training System is designed.

The gradual progression from external reference towards internal reference provides the opportunity for some form of difficulty grading. The effect of decreasing the frequency of external reference points is that the gaps between reference points become larger and larger, and therefore synchronisation more and more difficult. This forces the musician to accurately subdivide the bigger temporal gaps, relying on and so fine-tuning his inner clock.

An Alternate Timing Training System was therefore designed with the following criteria in mind:

- facilitate the progression from reliance on an external reference (metronome) to reliance on an internal reference
- provide a graded system of ‘difficulty levels’

The primary goal of the system described below is to improve the user’s sense of time. It attempts to do this by training and conditioning the user’s ability to internally perceive the primary pulse of the target music, as well as to internally subdivide the primary pulse. The more accurate his internal perception, the better a musician’s sense of time. The training is achieved through varying the audio references given to a user, both in terms of temporal placement, and in terms of the frequency of provided reference points.

4.2 Principles and Concepts

The passage of time is fundamental to the existence of music. Music can be described as a collection of events that occur at specific points in time, such as notes and note durations. In much contemporary music, these events conform to a temporal grid, occurring at regularly spaced intervals, or at some multiple or subdivision of a base temporal interval. Music usually has a fundamental pulse or beat upon which the placement of events such as notes is based.

The way humans experience time has been the subject of much research in the field of psychology, see Chapter 2 for an overview of sensorimotor synchronisation and temporal control of the motor system.

4.2.1 Reference Points

The primary function of a standard metronome that consists of an isochronous click is to provide regular temporal reference points to the listener. It is a way for

the musician to check and adjust his or her performance so that it occurs earlier or later, depending on how he or she perceives the clicking of the metronome. Using the metronome as a reference, the musician is able to tell if he or she is speeding up or slowing down. The terms ‘reference point’ and ‘click’ will be used interchangeably in this thesis.

The Alternate Timing Training System uses audible clicks as reference points in the same way as a standard metronome, but facilitates the use of reference points not necessarily coinciding with the primary pulse of the target music or musical exercise. An algorithmic system was developed to vary an isochronous click in relation to the primary pulse that results in a collection of rhythmic patterns. The patterns are automatically organised into a series of roughly graded difficulty levels.

Subdivisions – where musical events occur in time

Another aspect of timing that the Alternate Timing Training System attempts to address is that of improving a musician’s ability to mentally subdivide the intervals between musical events. Most music has an underlying pulse or beat around which musical events occur. These events (usually musical notes) tend to either coincide with the underlying pulse, or to occur at some regular subdivision of the pulse (see Figure 4.1). Subdivisions in western music are usually multiples of 2 or 3, so this is what the Alternate Timing Training System will focus on.

A common way in which musicians synchronise to very slow tempi is by mentally and/or physically subdividing the long interval into smaller, or shorter evenly spaced time intervals. This effectively superimposes another, faster pulse onto the original slow pulse, making it easier for the musician to synchronise his actions correctly.

4.2.2 Methods of Pattern Generation

An isochronous click can be compared to a periodic wave, and has two basic properties that can be manipulated for the purposes of pattern generation:

- *period*, or temporal interval between clicks. This is what determines the frequency, or tempo of the clicks, measured in beats per minute (bpm).
- *phase*, or offset from some reference point in time, usually the primary pulse of the target music.

The inclusion of subdivisions as possible reference points instead of only using the primary pulse greatly increases the possible pattern variations.

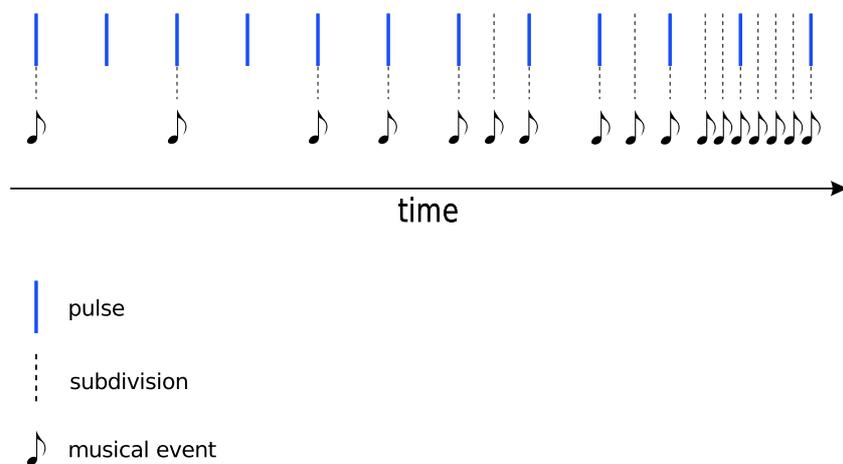


Figure 4.1: Musical Events

What follows is a more detailed discussion of how manipulation of the two properties, period and phase, can be used to generate patterns.

Note that for the purposes of this discussion the duration of musical events is ignored, i.e. the primary focus is on note onset, and not on note decay time or note duration.

Temporal Interval – slower is more difficult

Synchronising to an isochronous click that is beating at 100 beats per minute (bpm), for example by clapping to the beat, is fairly easy for most people. But the longer the time duration between events, the harder it becomes to predict exactly when the next beat will occur [Repp, 2005a, Mates et al., 1994]. Repp and Doggett [2007] have shown that trained musicians are better at slower tempi than non-musicians. Synchronising accurately to a click at 50bpm is more difficult, and synchronising to a click running at 10bpm (once every 6 seconds) becomes almost impossible except for those with a highly developed sense of tempo and subdivision (Figure 4.2).

Typically, a musician will set the tempo on the metronome he is using to match the underlying pulse of the piece of music or exercise he wishes to practice. This will provide him with reference points that occur at every pulse point in time. Once a musician is comfortable with performing the chosen music or exercise with the metronome, i.e. he is able to successfully synchronise his performance with the metronome click, a possible method for increasing the difficulty of the task is to increase the size of the interval between clicks, typically

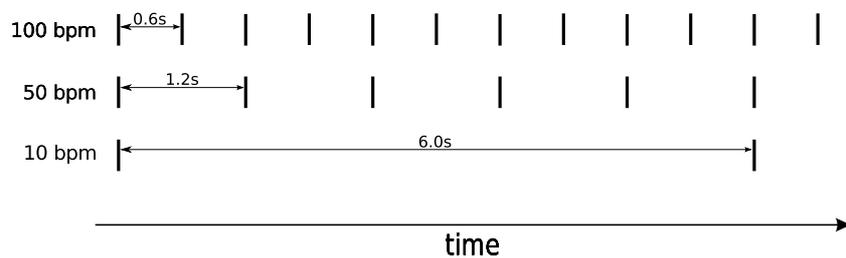


Figure 4.2: Visual representation of different tempi

by the period of the primary pulse. The interval is effectively doubled by adding the value of a primary pulse period, resulting in a metronome reference that runs at half the tempo of the primary pulse (Figure 4.3). This is effectively the same as removing every second reference point, making it more difficult because the musician has to estimate a period that is twice as long before he is given the next reference point by the metronome.

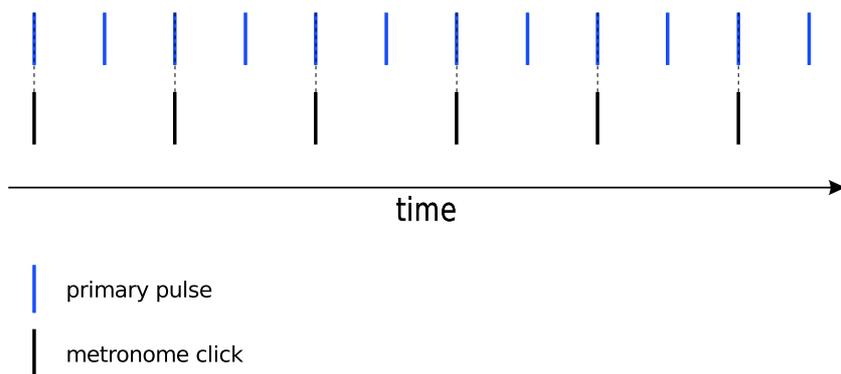


Figure 4.3: Half-time – metronome clicking every second beat

To further increase the difficulty of predicting the next reference point, the same technique can be repeatedly applied, that is, by increasing the temporal distance between clicks by a single primary pulse period each time.

A simple analogy could be the way in which a child learns to ride a bicycle. Initially, the bicycle is fitted with balance wheels, a pair of small wheels that prevent the bicycle from falling over. These wheels are in constant contact with the ground. As the child's sense of balance improves, the balance wheels are raised slightly so that they are no longer in constant contact with the ground, but only provide support when the bicycle leans sufficiently for them to make contact. This process of raising the balance wheels is repeated over time until

the balance wheels are no longer necessary, and the child is able to balance unaided. The balance wheels are analogous to the reference points provided by a metronome - as a musician's sense of time improves, the isochronous click can be adjusted to provide fewer and fewer reference points by increasing the size of the period.

To provide variation on the pattern shown in Figure 4.3 the second wave-like property of an isochronous click, phase, can be adjusted. Shifting the phase of the pattern by the value of one full primary pulse period results in a new pattern, similar in that the temporal interval between clicks (the period) is still the same, but the clicks are occurring at a different point in relation to the primary pulse. Figure 4.4 shows the half-time pattern from Figure 4.3 phase-delayed by the value of one full primary pulse period, effectively a 180° shift. Shifting by a further primary pulse period is ineffective in providing further variation as it is obviously another 180° shift, and will return the pattern to its original, unshifted state.

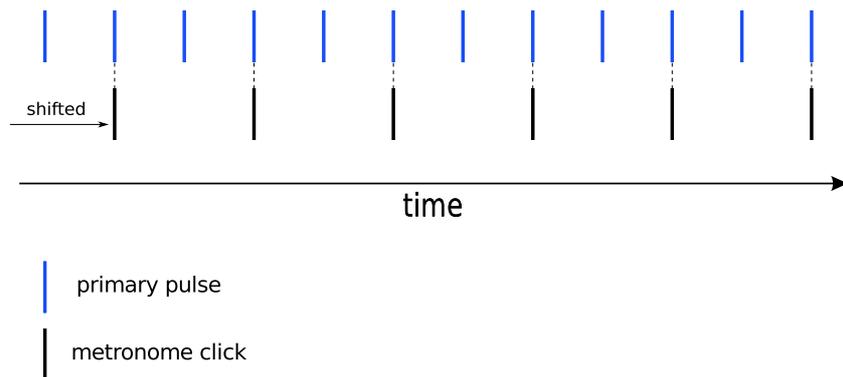


Figure 4.4: Half-time phase-shifted by 1 primary pulse period

Continuing with this system, another primary pulse period can be added, and the resulting pattern shifted as many times as possible without causing duplicates. Figure 4.5 shows this set of three patterns.

The problem that becomes evident when continuing with this method on its own is that increasing the temporal interval by the period of the primary pulse every time quickly results in very large temporal intervals between clicks. The number of possible patterns is consequently severely limited.

To solve this problem the use of subdivisions as the base interval is introduced, both for growing and for shifting the metronome reference. Instead of increasing the temporal interval by the primary pulse period each time, a sub-

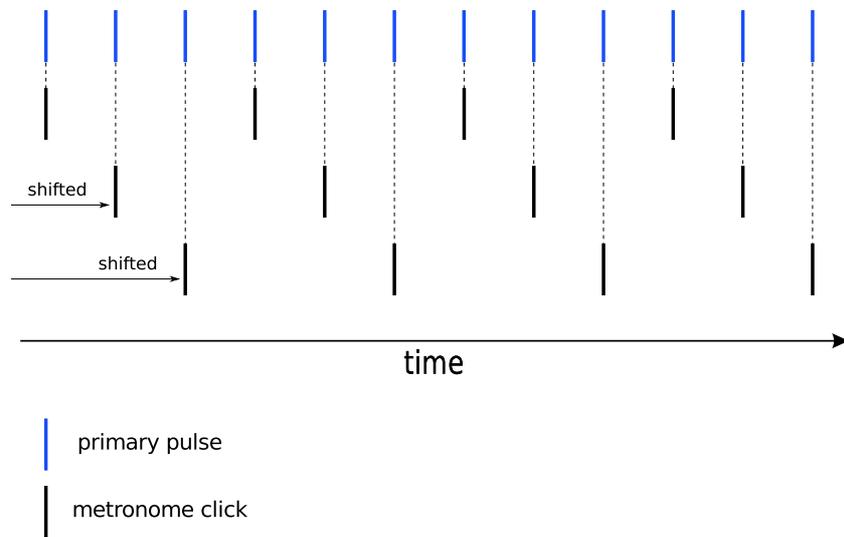


Figure 4.5: Three-beat interval, phase shifted

divided portion of the primary pulse period is used. This same period is then used to perform the phase shifting to create all possible patterns based on the chosen subdivision. An example of this can be seen in Figure 4.6, where the primary pulse period has been subdivided by two to create the base interval, which is then subsequently multiplied by three.

The use of subdivisions when generating isochronous clicks results in patterns containing click events that do not always coincide with primary pulse points. This is desirable because it encourages the user to become accustomed to using reference points that are not necessarily primary pulse points, and so enhances the user's ability to subdivide accurately.

By adjusting the two wave-like properties – varying temporal interval based on subdivision, and pattern displacement – one can arrive at a sufficiently large and varying number of patterns to use within a timing training system, all using an isochronous click as provided by a simple standard metronome. The fundamental action of the musician synchronising his playing to the clicking of the metronome remains the same, but the use of patterns and displacement to determine when the clicks occur can provide the musician with new and interesting challenges.

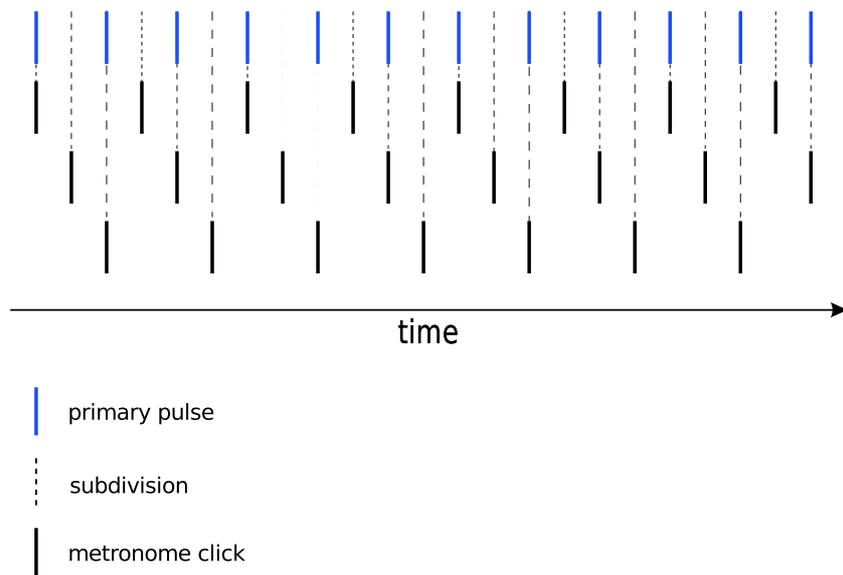


Figure 4.6: Subdivided by 2, multiplied by 3 and phase shifted

Caveat – the metronome always starts first

An important aspect not illustrated by the preceding explanation with its diagrams is the fact that the musician always synchronises to the metronome, and not the other way around. In other words, the metronome is started by the musician, who once comfortable with the pulse provided by the metronome begins to play. When using an isochronous click that does not match the primary pulse of the music it becomes very difficult for the musician to feel the primary pulse in relation to the click. While one of the intentions of the Alternate Timing Training System is to encourage the musician to develop his ability to feel the primary pulse internally, to feel the primary pulse right from the start is very difficult.

To remedy this problem a second isochronous click is introduced. This click always represents the primary pulse of the music, and functions in the same way as a standard metronome would when used in the traditional way. The reason for this extra click is to help the musician establish the primary pulse, and to hear how the pattern sounds and feels in relation to the primary pulse. An isochronous audio click only becomes an auditory pattern when heard in relation to another cyclical reference, which in the current context is the isochronous primary pulse. It is therefore necessary to initially provide the user with the primary pulse click in addition to the pattern click. This is so that the relationship

between the two can be clearly heard before gradually removing the primary pulse, forcing the user to rely only on the pattern for reference. This would be achieved by decreasing the volume of the primary pulse over time until it is no longer audible.

In the section on pattern generation the possible patterns that are available using the described methods of adjusting period and phase will be investigated in more detail. There are many other methods that could be used for pattern generation, including pseudo-random and other methods that would result in non-isochronous click-patterns. It was decided for this study to use only isochronous patterns for a number of reasons:

- isochronous patterns lend themselves to algorithmic generation
- isochronous patterns can be more easily categorised, facilitating a graded system of difficulty levels
- a standard metronome is isochronous, and is familiar to most musicians
- predicting temporal events such as clicks is easier when the events are isochronous

4.2.3 Summary

Metronomes are traditionally used to improve a musician's timing, helping him or her perform more consistently, especially when performing within the context of a musical group or ensemble. Using the same underlying training principles offered by a standard metronome, that of the musician synchronising his or her actions to an isochronous click, an Alternate Timing Training System was designed based on isochronous patterns that are varied by adjusting two wave-like properties of isochronous clicks: that of increasing the temporal interval between reference points (*period*) and that of displacing the reference points by different subdivision amounts (*phase*). The intention of increasing the period is to develop the musician's ability to predict temporal intervals; the intention of adjusting the phase is to develop the musician's sensitivity to temporal subdivision. Combining the two properties allows for the generation of a large number of isochronous clicks that appear as patterns when viewed in relation to the primary pulse, thereby providing more variety and challenge to the user.

4.3 Algorithmic Pattern Generation

In this section we will further explore the patterns that can be generated by adjusting the two wave-like properties discussed in the previous section, and show how they can be algorithmically generated.

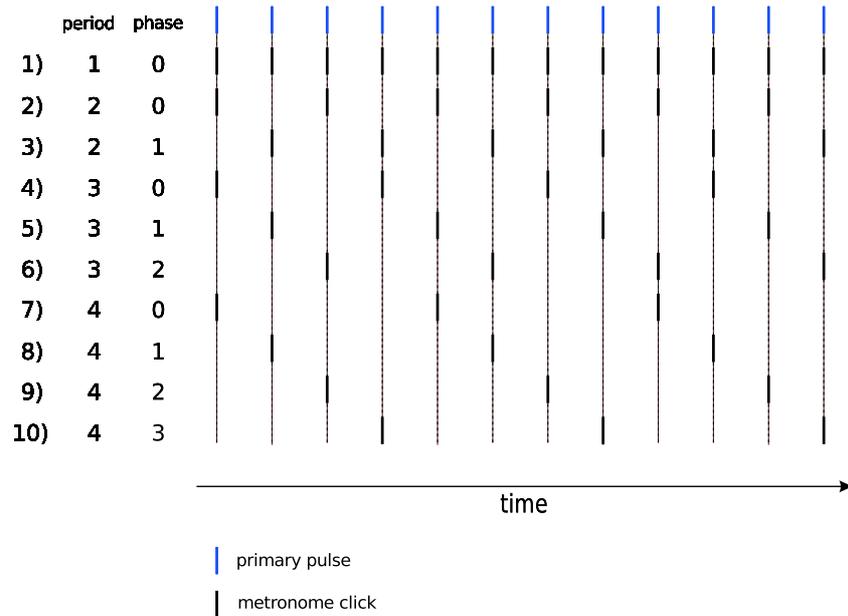


Figure 4.7: Pattern Series

Temporarily ignoring the variations offered by the use of subdivisions as a base interval, the first few patterns in the series of unique patterns that can be generated by manipulating period and phase is shown in Figure 4.7. A collection of numbered patterns roughly graded by difficulty is required for the Alternate Timing Training System. Given the pattern number as input, a method needs to be derived of generating the correct pattern by calculating the required period and phase-shift in relation to the base interval.

pattern numbers	number of patterns	period
1	1	1
2-3	2	2
4-6	3	3
7-10	4	4
11-15	5	5
16-21	6	6

Table 4.1: Total patterns for period group

Table 4.1 shows the number of unique patterns that can be generated for each increment of the metronome period. Upon closer inspection of both Figure 4.7 and Table 4.1 it will be noted that the relationship between the base interval and the number of patterns generated is in fact that of triangular numbers [Sierpiński and Schinzel, 1988, p. 44].

$$T_n = 1 + 2 + 3 + 4 + (n - 1) + n$$

$$T_n \equiv \sum_{k=1}^n k$$

$$= \frac{n^2 + n}{2}$$

To calculate the period of the target pattern the triangular root first needs to be calculated. This is a quadratic equation ($n^2 + n - 2T_n = 0$), and so can be solved for n using the standard quadratic formula to calculate the triangular root of a number, where x is the target pattern number and n is the calculated period. Negative roots are obviously not valid in the current context, and so are ignored.

$$n = \frac{-1 + \sqrt{8x + 1}}{2}$$

Calculating the triangular root for ‘imperfect’ triangular numbers gives non whole number results. To get the required period for a target pattern, the triangular root needs to be ‘rounded up’ to the nearest natural number, but only if it is not already a natural number.

$$period = \lceil \frac{-1 + \sqrt{8x + 1}}{2} \rceil$$

The period of the target pattern is the factor by which the base interval will be scaled to achieve the temporal interval between the clicks of the pattern.

Having calculated the period, the next step is to calculate the phase-shift (θ) required for the pattern. The phase-shift is the difference between the pattern number and the nearest lower ‘perfect’ triangular number, less 1. The nearest lower triangular number is easily attained by triangulating $period - 1$ as calculated previously.

$$\theta = x - \frac{(period - 1)^2 + period - 1}{2} - 1$$

where x is the pattern number. Here is an example of applying these formulae to calculate the period and phase-shift for pattern number 8:

$$\begin{aligned} period &= \lceil \frac{-1 + \sqrt{8(8) + 1}}{2} \rceil \\ &= \lceil 3.531128874 \rceil \\ &= 4 \end{aligned}$$

$$\begin{aligned} \theta &= 8 - \frac{(4 - 1)^2 + 4 - 1}{2} - 1 \\ &= 1 \end{aligned}$$

Both the period and the phase are expressed in relation to the base interval. The period is the factor by which the base interval is scaled to create the new temporal interval of the isochronous click. The phase is the number of base intervals by which the period is to be shifted in relation to the primary pulse.

Two problems emerged during the implementation of this algorithm, the first being that as soon as subdivisions were included as a means of choosing the base interval, duplicate patterns were generated. Combining patterns generated from different base intervals resulted in duplicates when patterns were compared in relation to the primary pulse. The simplest solution was to check for and remove any duplicates. The other problem was that for every base interval, including the primary pulse, one of the generated patterns was an exact match to the primary pulse. This pattern has no value over a standard metronome, and so was also treated as a duplicate and removed from the series.

4.4 Implementation

In order to evaluate the Alternate Timing Training System a software implementation was required that could be provided both to the subjects in the

experiment and to the survey respondents. The following section details some of the design and implementation decisions.

4.4.1 Requirements

There are many existing variations on the traditional metronome. A traditional mechanical metronome merely provides an isochronous click at the desired tempo to which the musician will synchronise his playing. This is also true of the simpler electronic and software metronomes available today. Although these metronomes are perfectly adequate and functional, they offer little in the way of variation to hold the interest of the musician. Modern electronic and software metronomes have added many features and enhancements to make the practice experience a little more interesting. They can be customised to suit a user's preferences for sound type, volume level and visual feedback, for example. Some of these ideas, such as the ability to choose custom click sounds and giving the user visual feedback of the current settings were incorporated into the implementation of the Alternate Timing Training System.

The primary requirement of the prototype application is to provide an implementation of the Alternate Timing Training System described in the preceding sections. This can be broken down:

- User selectable tempo – the user should be able to choose a working tempo.
- Start/stop – a means to start and stop the system.
- Pattern selection – the user should be able to choose a difficulty level/-pattern.
- Primary pulse fade – the user should be able to choose the point at which the primary pulse begins to fade out.

Along with these a few other non-essential requirements were identified that serve to improve the usability of the application:

- User selectable sounds – the user should be able to choose from a selection of click sounds for both the primary pulse and the pattern.
- Count-in – the user should be able to select an optional number of beats that establish the working tempo prior to the introduction of the pattern.
- Visual feedback – a visual representation of the primary pulse/pattern combination to assist the user in selecting a difficulty level.

- Built-in emailing of log files – not strictly functional, but a necessary requirement for the collection of data for the experiment.

One of the fundamental variables available to the user in most metronomes, if not all, is the ability to choose a working tempo. This is often the only way to vary the challenge of synchronising with the metronome. The design of the Alternate Timing Training System resulted in a series of roughly graded patterns that a user could choose from, with the varying difficulty of the patterns providing additional interest and challenge to the musician. While the prototype application attempts to offer an interesting musical experience to the user, it remains focused on the objective - the improvement of the user's timing by implementing the designed Alternate Timing Training System.

The Alternate Timing Training System was implemented using Java and the Java sound API for all digital audio mixing and rendering. The application design is a simple layered architecture, with an audio engine, a click engine, an abstract metronome and a user interface. The audio engine is responsible for rendering the audio clicks using the sound API. The click engine is responsible for managing the isochronous clicks that drive the audio engine. The abstract metronome is realised by a concrete metronome that algorithmically creates and manages the click patterns. This is the core implementation of the Alternate Timing Training System. The user interface was created using Java Swing. Java was chosen as the target environment due to its cross-platform nature, and the application was tested successfully on Microsoft Windows, Linux and Mac OSX.

See Appendix A for the source code listing and Appendix B for a simplified Unified Modeling Language (UML) class diagram of the application architecture.

4.4.2 User Interface

A formal user interface design methodology such as user-centred design requires a number of iterations involving various activities, such as interaction with potential users, creation of prototypes and interface mock-ups and the collection of user evaluations. Given the project's time constraints and the focus on evaluation of the Alternate Timing Training System, it would not have been possible to properly implement such a methodology. User interface design and evaluation is left to future work.

Both the functional and usability requirements specified in the previous section contributed to the layout and design of the user interface. Tempo entry and start/stop functionality were placed prominently on the interface using large,

visible controls. The option to choose custom sounds for the two isochronous reference clicks prompted the decision to provide separate sections for each reference, with reference-specific controls and functionality grouped together. The means for selecting the difficulty pattern was logically placed close to the visual representation of the pattern. A slider control was chosen for selecting the difficulty pattern due to the ease with which one can move quickly through multiple patterns while viewing the visual representation.

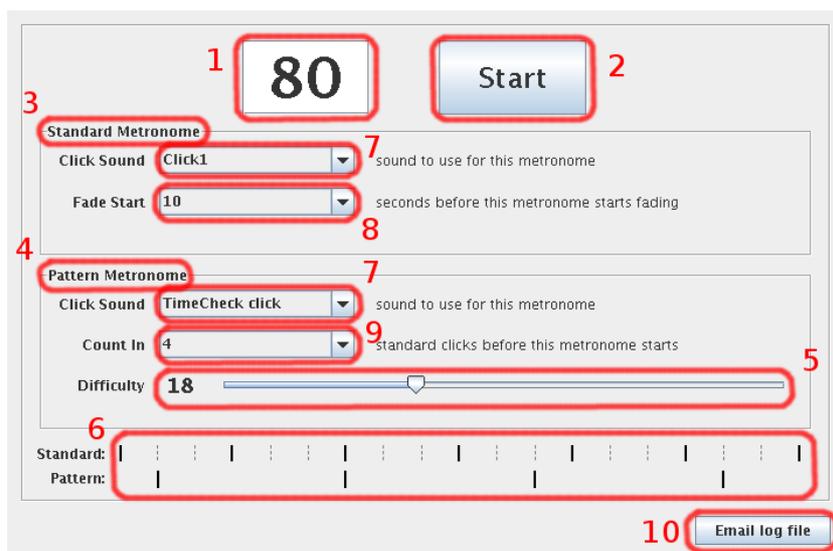


Figure 4.8: Screenshot

Figure 4.8 shows a screenshot of the implementation with annotations that will be used to explain each feature below. The term ‘reference’ is used as a substitute for ‘isochronous click’.

1. The tempo text box. This is used to input the tempo of the primary pulse reference, just as one would for a standard metronome.
2. The Start/Stop button. Used to start and stop the system.
3. These are the controls for the standard metronome reference, that is, the primary pulse.
4. These are the controls for the pattern reference.
5. The difficulty slider. This is used to choose a pattern for the pattern reference, with patterns being roughly graded by difficulty. Easier patterns

are on the left, more difficult patterns on the right. In addition to using the mouse to drag the slider left and right, the arrow keys can be used for incrementing and decrementing the pattern number.

6. This is a visual representation of what the two references will sound like together. The top row represents the primary pulse reference, with reference clicks indicated by solid lines and base interval subdivisions indicated by dashed lines. The bottom row represents the pattern reference, again with solid lines indicating the clicks, and shows where the pattern clicks will occur in relation to the primary pulse.
7. Drop down boxes are used to choose the individual sounds of the different references. Each reference can, and should, have a different sound.
8. Fade start. Once the user clicks the “Start” button, this is the number of seconds that both references will run simultaneously before the primary pulse reference begins to fade out.
9. Count in. This is the number of primary pulse reference clicks that will play before the pattern reference is started. Note that the visual representation of the two references (6) does not show the count-in clicks.
10. Email log file. Clicking this button emails the log file generated by the application directly to the author. Users were encouraged to click it after every session. Note that this functionality was not included in the version made available online for the survey.

The application was designed to include a number of different sounds that can be used as reference clicks. This allows for user preference, and allows each reference to have a different sound, making them more easily distinguishable from one another.

Although marked ‘Difficulty Level’, the pattern numbers are not a true reflection of how difficult it might be to use a particular pattern as a temporal reference. The size of the period of the chosen pattern is the only factor which can be said to correspond to the difficulty of using the pattern as a reference. The larger the period, the further apart the references and therefore the more difficult. The perceived difficulty level of each pattern in a set that share the same period is influenced by other factors, especially that of the musician’s familiarity with the rhythmic aspects of the pattern. This in turn can be influenced by factors such as the musician’s culture and the rhythms he has been exposed to (see [Hannon and Trainor, 2007] for a discussion of enculturation

in the section *Learning Rhythmic Structure: from movement to culture-specific metrical structure* and also [Drake and Heni, 2003]).

An early prototype of the application included the option to manually choose the subdivision used to generate the difficulty level patterns. Feedback from a few pilot users indicated that this was confusing, and so the pattern factory was reworked to automatically choose the base temporal interval. By specifying a threshold for the maximum size of the period, a new base interval is selected each time the threshold is exceeded. This results in a continuous series of patterns that include different subdivisions.

4.4.3 Usage

The Alternate Timing Training System can be used as a direct substitute for a standard metronome. Typically, a musician will use a standard metronome as an aid when practicing a piece of music or a musical exercise. The metronome will be set to the correct tempo and started. The musician will then synchronise his performance to the regular clicks of the metronome. The Alternate Timing Training System is intended to function in the same way as a standard metronome, and should be used in the same way, that is, after choosing a tempo and starting the metronome, the musician will perform the target music or exercise in sync with the metronome clicks. The difference is that the user must choose which pattern to use as a reference in addition to choosing the tempo.

Chapter 5

Results and Discussion

5.1 Experiment Results Analysis

Of the initial 15 subjects who volunteered to take part in the experiment, only 5 provided a sufficiently complete set of data. This was ascertained by querying the usage data that was parsed and extracted from the Alternate Timing Training System's and standard metronome's application log files. See Chapter 3 for a discussion of the database design and an entity relationship diagram (Figure 3.6). All SQL queries used in this chapter are included in Appendix A, section A.3.

Querying the usage data for the total time spent by each subject with each timing training system produced the result shown in Table 5.1. It can be seen that a number of subjects did not use either or both of the training systems sufficiently to provide usable data for analysis. Using a threshold of one hour as a minimum total time spent with each training system, a number of subjects were removed (*Nic, Jonathan, Mark, Keegan, Marcio, Avzal, Kerry, Ana*). Subjects who did not spend sufficient time with both systems were also removed (*Abbey, Laura*). Although the final sample of 5 subjects is small, the available data can be examined in a number of ways in an attempt to expose any trends.

The unit under scrutiny is a note played by a subject. The accuracy of the note is measured by determining how far the note is from the reference click in time. This error can be interpreted in one of two ways: either as an absolute value in milliseconds, or as an error rate which takes the current tempo into account. The first is simply achieved by subtracting the reference performance measurement from assessment measurement and multiplying by 1000 for a millisecond value. The sign of the result is indicative of whether the

Name	Training System	Session Count	Total Time
Geoffrey	Alternate System	15	7:01:00
Geoffrey	Standard Metronome	17	5:36:44
Wynand	Standard Metronome	11	3:51:01
Neil	Alternate System	11	3:21:39
Neil	Standard Metronome	12	3:11:34
Laura	Standard Metronome	9	2:53:44
Brett	Alternate System	13	2:47:09
Wynand	Alternate System	7	2:11:29
Brett	Standard Metronome	7	2:08:14
Abbey	Standard Metronome	12	1:49:18
Travis Marc	Alternate System	5	1:20:51
Travis Marc	Standard Metronome	5	1:01:00
Avzal	Alternate System	7	0:47:26
Avzal	Standard Metronome	7	0:42:22
Marcio	Alternate System	7	0:20:02
Marcio	Standard Metronome	3	0:14:34
Keegan	None	0	0:00:00
Mark	None	0	0:00:00
Jonathan	None	0	0:00:00
Nic	None	0	0:00:00
Kerry	None	0	0:00:00
Ana	None	0	0:00:00

Table 5.1: Training system usage statistics for each user, showing the training system used, the number of sessions (application startup to shutdown) and the total time spent (hours, minutes and seconds).

note was played early or late in relation to the reference click. The Error Rate is calculated as a ratio of the measured error to the maximum possible error for the current tempo. This gives a signed value between 0 and 1, again with the sign indicating whether the note was played before or after the reference click. For the following analyses the Error Rate will be used because it takes the current tempo into account. This makes sense because slower tempi are more difficult to play accurately than faster tempi due to the larger temporal intervals involved. Effectively, the slower the tempo the smaller the Error Rate for the same absolute millisecond error. As a musician settles in to a new tempo, there is a higher likelihood of him making timing errors until his inner clock is set to the tempo. This may result in statistical outliers in the data. Examination of the data gathered from the assessment recordings revealed only 2 outlying measurements with an Error Rate greater than 0.5. To mitigate the effect of these the dataset is filtered to exclude Error Rate values greater than 0.5 and smaller than -0.5 . Figure 5.1 is a screenshot showing a small segment of the full dataset, approximately 2760 measurements. The data consists of the following columns:

- ID – a unique number, used by the statistics application.
- Reference – the reference measurement, that is, a perfect assessment against which to compare. Recorded as elapsed seconds from the start-time of the assessment recording.
- Measurement – the measurement extracted from the recorded assessment. Recorded as elapsed seconds from the start-time of the assessment recording.
- ErrorRate – calculated error rate, the ratio between the Millisecond Error and the maximum possible millisecond error for a given tempo.
- MillisecondError – the absolute error, measured in milliseconds.
- Tempo – the current tempo.
- Assessment – the assessment number, 1, 2 or 3.
- Name – the subject’s name.

	A	B	C	D	E	F	G	H
1	ID	Reference	Measurement	ErrorRate	MillisecondError	Tempo	Assessment	Name
2	404	1.199	1.197667	-0.004443	-1.333000000000	100	1	Wynand
3	405	1.799	1.811	0.04	12	100	1	Wynand
4	406	2.399	2.401333	0.00777667	2.33300000000014	100	1	Wynand
5	407	2.999	2.978	-0.07	-21.000000000000	100	1	Wynand
6	408	3.599	3.618667	0.06555667	19.6669999999997	100	1	Wynand
7	409	4.199	4.167333	-0.105557	-31.667000000000	100	1	Wynand
8	410	4.8	4.812333	0.0410416	12.3329999999999	100	1	Wynand
9	411	5.399	5.375	-0.080134	-24	100	1	Wynand
10	412	5.999	5.962667	-0.12111	-36.333000000000	100	1	Wynand
11	413	6.599	6.58	-0.063333	-19.000000000000	100	1	Wynand
12	414	7.199	7.197667	-0.004443	-1.333000000000	100	1	Wynand
13	415	7.799	7.803333	0.01444333	4.33299999999992	100	1	Wynand
14	416	8.399	8.400333	0.00444333	1.33300000000069	100	1	Wynand
15	417	8.999	8.996667	-0.007777	-2.333000000000	100	1	Wynand

Figure 5.1: Screenshot showing dataset and columns.

The assessments recorded by subjects during the experiment are structured in such a way as to allow different aspects and areas of the data to be inspected. Apart from comparing average accuracy between assessments 1, 2 and 3, it is also worth partitioning the results by tempo. Each of the 7 different tempi recorded can be examined in isolation, but splitting the tempi into two groups, faster and slower, is sufficient to assess whether rates of improvement are different for different tempi. Slower tempi force the performer to subdivide the larger temporal intervals, thereby testing the accuracy of the performer’s ability to subdivide the beat. A further aspect that can be investigated is that it would normally take at least a few beats for a musician to settle into a new tempo. Instead of analysing all notes played for any given tempo, a certain number of beats at the beginning of a new tempo section can be ignored. This gives the musician time to set his inner clock to the new tempo. Four different datasets are analysed in the following sections:

- the full dataset, unfiltered.
- faster tempi - 100, 80 and 60 beats per minute (bpm).
- slower tempi - 50, 40, 30 and 20 bpm.
- the full dataset, but excluding the first 10 beats of each tempo.

The same approach is taken with each dataset - the mean and standard deviation of the Error Rate (the Dependent Variable) are calculated and examined for each assessment, and thereafter an Analysis of Variance (ANOVA) is

performed to assess whether the differences in the means are statistically significant. The mean is an indication of overall accuracy, the closer to zero the more accurate. The standard deviation shows the spread of values, with lower values indicating a smaller spread. A smaller spread may be an indication of a more consistent performance. Should the ANOVA test show any statistically significant difference in the mean Error Rate for each assessment, further *post hoc* analysis can be used to determine the specific differences between groups and identify possible trends in timing accuracy [Spatz, 2010, p231].

Assessment 1 is a baseline assessment, recorded before any training had taken place. Assessment 2 was recorded two weeks subsequent to the subject’s use of a standard metronome for training. Assessment 3 was recorded a further two weeks later, subsequent to the subject’s use of the Alternate Timing Training System. The respective training systems, as represented by the second and third assessments, are the statistical Independent Variable.

See Appendix A, Section A.3 for the SQL queries used to retrieve the data. Data were imported into a spreadsheet application (*Gnumeric*, <http://projects.gnome.org/gnumeric/>) for filtering to the different datasets and to create the histograms. *Sofa Statistics* (<http://www.sofastatistics.com>) was used to perform the ANOVA tests.

5.1.1 Full Dataset

The full dataset was used for the following analysis.

Assessment	Number of Measurements	Mean	Standard Deviation
1	918	-0.027	0.056
2	921	-0.03	0.052
3	919	-0.032	0.057

Table 5.2: Descriptive Statistics for Error Rate - all data

The third column in Table 5.2 shows the mean Error Rate for each assessment. A small increase in Error Rate can be observed, indicating a small decrease in timing accuracy for each subsequent assessment. The standard deviation decreases slightly for the second assessment, but then increases again for the third assessment. A histogram depicting the three assessments can be seen in Figure 5.2. The shape of the graph is very similar for each assessment’s Error Rate.

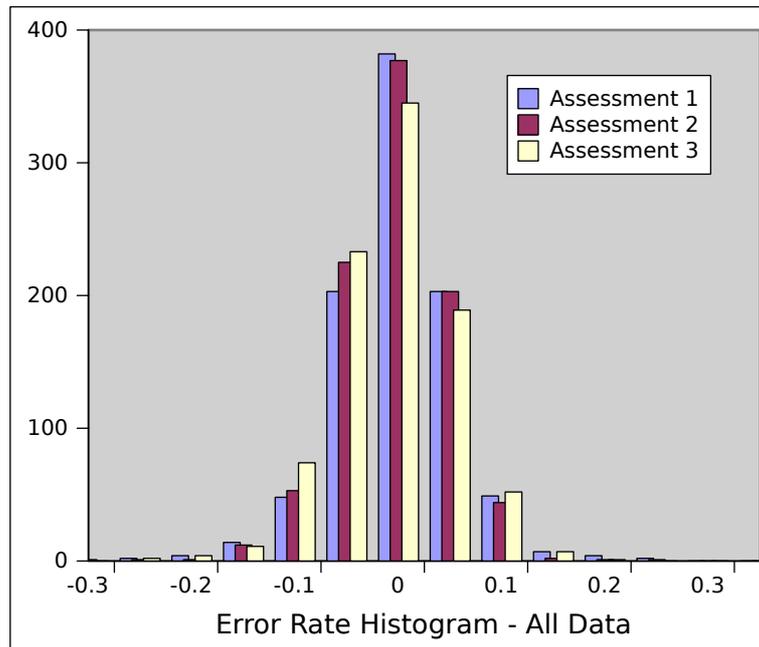


Figure 5.2: Histogram of Error Rate with the X-axis representing the Error Rate (bin size 0.05) and the Y-axis representing the Error Rate count per bin.

The results of the ANOVA test are shown in Table 5.3. There was no statistically significant effect of the timing training systems on the Error Rate, and therefore the subjects' timing accuracy, at the $p < .05$ level for the three assessments [$F(2, 2755) = 2.254, p = 0.105$] when considering all the data across all tempi.

	Sum of Squares	df	Mean Sum of Squares	F	p
Between assessments	0.014	2	0.007	2.254	0.105
Within assessments	8.351	2755	0.003		

df = degrees of freedom.

F = F-test result, the ratio of variance between 'Between-assessments' and 'Within-assessments' groups.

p = Statistical Significance. Values below 0.05 are considered statistically significant.

Table 5.3: Analysis of Variance of Error Rate for all measurements.

5.1.2 Faster Tempi

The full dataset was filtered to include only measurements recorded at the following tempi: 100, 80 and 60 bpm. Measurements recorded at the slower tempi (50, 40, 30 and 20 bpm) were excluded for the following analysis and are analysed in the next section.

Assessment	Number of Measurements	Mean	Standard Deviation
1	501	-0.026	0.054
2	494	-0.029	0.049
3	487	-0.034	0.055

Table 5.4: Descriptive Statistics for Error Rate - faster tempi

Table 5.4 shows the mean and standard deviation for each assessment filtered for faster tempi (100, 80 and 60 bpm). The means show an increase in Error Rate, indicating a decrease in accuracy. The pattern for standard deviation is similar to that of the analysis of the complete dataset - decreasing slightly and then increasing again. Figure 5.3 shows each assessment's Error Rate producing a similarly shaped histogram.

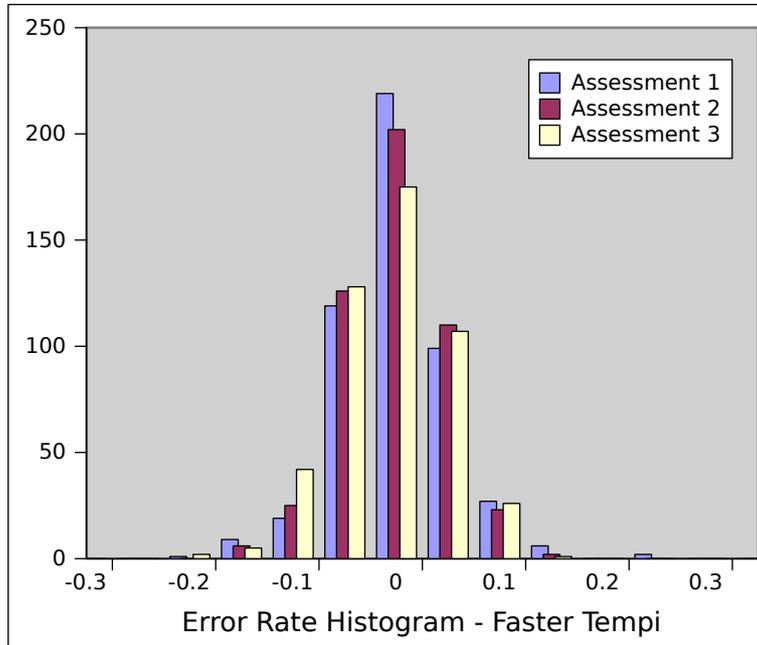


Figure 5.3: Histogram of Error Rate with the X-axis representing the Error Rate (bin size 0.05) and the Y-axis representing the Error Rate count per bin.

The results of the ANOVA test are shown in Table 5.5. There was no statistically significant effect of the timing training systems on Error Rate at the $p < .05$ level for the three assessments [$F(2, 1479) = 2.82, p = 0.06$] when considering the data for faster tempi.

	Sum of Squares	df	Mean Sum of Squares	F	p
Between assessments	0.016	2	0.008	2.82	0.060
Within assessments	4.097	1479	0.003		

df = degrees of freedom.

F = F-test result, the ratio of variance between 'Between-assessments' and 'Within-assessments' groups.

p = Statistical Significance. Values below 0.05 are considered statistically significant.

Table 5.5: Analysis of Variance of Error Rate for faster tempi.

5.1.3 Slower Tempi

The same analysis was performed on a dataset that included only the slower tempi: 50, 40, 30 and 20 bpm. Accurate performance at slower tempi is more difficult due to the larger temporal intervals between reference clicks, requiring the subject to mentally subdivide the larger intervals in order to keep time. As one of the potential benefits of the Alternate Timing Training System is to enhance the subjects' subdivision accuracy, measurements recorded at slower tempi were analysed separately from measurements recorded at faster tempi.

Assessment	Number of Measurements	Mean	Standard Deviation
1	417	-0.027	0.058
2	427	-0.031	0.056
3	432	-0.03	0.059

Table 5.6: Descriptive Statistics for Error Rate - slower tempi

The descriptive statistics shown in Table 5.6 are similar to those for the faster tempi dataset, with means and standard deviations very close together across the different assessments. The histogram in Figure 5.4 depicts this, with the shape and spread of each assessment closely matched.

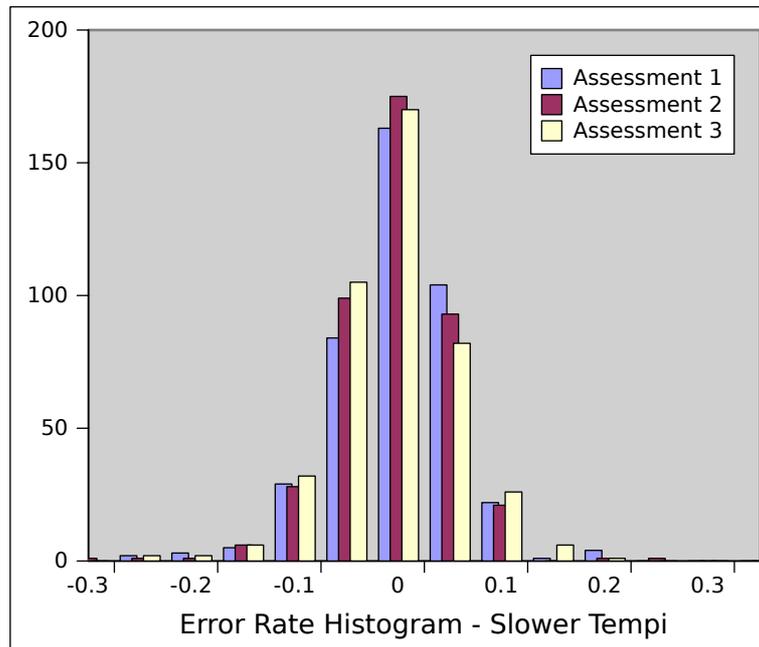


Figure 5.4: Histogram of Error Rate with the X-axis representing the Error Rate (bin size 0.05) and the Y-axis representing the Error Rate count per bin.

The ANOVA test results in Table 5.7 confirm that there was no statistically significant effect of the timing training systems on Error Rate at the $p < .05$ level for the three assessments [$F(2, 1273) = 0.529, p = 0.590$] when considering the data for slower tempi.

	Sum of Squares	df	Mean Sum of Squares	F	p
Between assessments	0.004	2	0.002	0.529	0.590
Within assessments	4.248	1273	0.003		

df = degrees of freedom.

F = F-test result, the ratio of variance between 'Between-assessments' and 'Within-assessments' groups.

p = Statistical Significance. Values below 0.05 are considered statistically significant.

Table 5.7: Analysis of Variance of Error Rate for slower tempi.

5.1.4 First 10 Beats per Tempo Removed

The first ten recorded beats were removed from each tempo for this analysis.

Assessment	Number of Measurements	Mean	Standard Deviation
1	848	-0.027	0.053
2	921	-0.03	0.052
3	919	-0.032	0.057

Table 5.8: Descriptive Statistics for Error Rate - first 10 beats removed

Table 5.8 above shows the summary of descriptive statistics. The same pattern as for the other datasets is evident, with a slight decrease in accuracy and fairly closely matched spreads across assessments. This is once again clearly depicted in Figure 5.5, with each assessment producing a histogram with a very similar shape and spread.

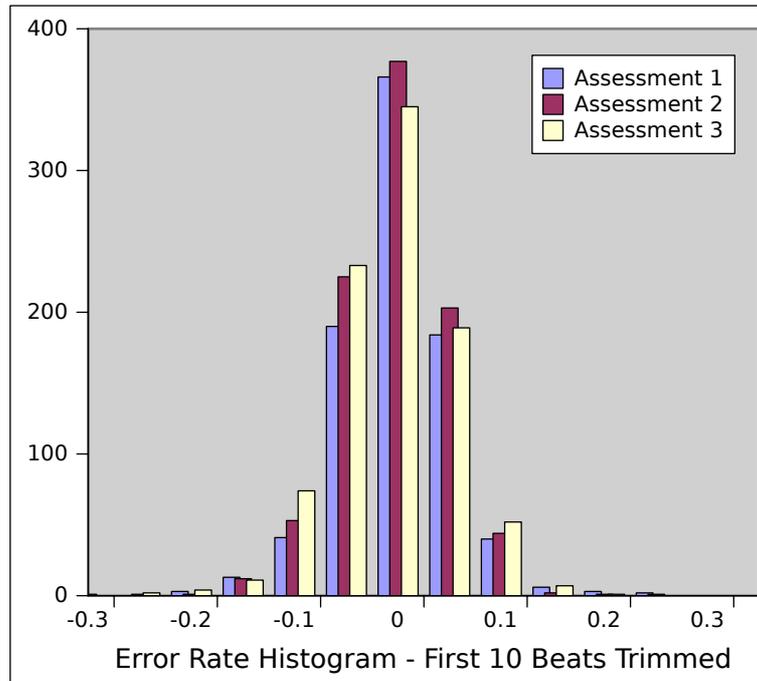


Figure 5.5: Histogram of Error Rate with the X-axis representing the Error Rate (bin size 0.05) and the Y-axis representing the Error Rate count per bin.

Once again, the results of the ANOVA test indicate that there was no statistically significant effect of the timing training systems on Error Rate at the $p < .05$ level for the three assessments [$F(2, 2685) = 2.12, p = 0.120$] when considering data that excluded the first 10 beats for each tempo.

	Sum of Squares	df	Mean Sum of Squares	F	p
Between assessments	0.012	2	0.006	2.12	0.120
Within assessments	7.902	2685	0.003		

df = degrees of freedom.

F = F-test result, the ratio of variance between ‘Between-assessments’ and ‘Within-assessments’ groups.

p = Statistical Significance. Values below 0.05 are considered statistically significant.

Table 5.9: Analysis of Variance of Error Rate with first 10 beats removed for each tempo.

5.1.5 Discussion

None of the datasets analysed showed any statistically significant change in timing accuracy over the duration of the experiment, not for either timing training system.

A standard metronome has been the de facto standard for timing training for decades, and should have produced at least some statistically significant results. This could be due to a number of factors, including the initial skill level of the subjects and the duration of the experiment. Timing improvement is known to take place over months and years [Drake and Palmer, 2000], rather than over the short term.

Another factor that may have played a part is the skill level of the experiment subjects. It was a requirement for subjects to already be somewhat skilled at synchronising to an external click so as to mitigate the effect of S-curve learning rates [Newell et al., 2001]. Unfortunately subjects may have been too skilled, meaning that further improvement in already well developed timing accuracy would only take place over a longer and possibly more rigorous training period. This is supported by the low mean Error Rate of -0.027 calculated from the first assessment recordings (Table 5.2).

Although the sample size of subjects that produced usable data was very small (5 subjects), the low standard deviation values indicate a narrow spread of measurements. This can be interpreted as consistency in terms of physical performance, meaning the data is at least fairly representative.

While the experiment did not yield statistically significant results, the methodology and the subsequently evident limitations can contribute significantly to informing future work. This is discussed in more detail in the following chapter.

It is interesting to note the evidence of Negative Mean Asynchrony (NMA) as discussed in Chapter 2. Not well understood, NMA is the tendency of the subject to play slightly earlier than the external reference click, and has been reported in many cognitive studies on sensorimotor synchronisation (see [Repp, 2005b] for a review and discussion). The means calculated from the assessment data were consistently negative, as is clearly seen in the histograms as well.

5.2 Survey Response Analysis

As discussed in Chapter 3 in the section on survey design, the sixteen survey questions were grouped into 3 sections: *Competency Assessment*, *Application Usage* and *Opinion*. Each section will be handled in turn, with the section's responses being presented and analysed and then briefly discussed. A more in-depth discussion is presented in Section 5.2.4.

The survey was completed by 41 individuals, of which 33 answered all mandatory questions. Examination of the incomplete surveys revealed that respondents answered all questions until they reached the questions on application usage. Failure to progress further indicates that these respondents elected to complete the survey without having used the Alternate Timing Training System software, and so were unable to answer the usage questions. Incomplete surveys will therefore not be considered here.

For the most part, the responses to individual questions will be examined using simple frequency charts. Where appropriate, multivariate frequency distribution analysis will be used, along with tabular presentation. It must be noted that some of the survey respondents were individuals that had also taken part in the previous experiment.

5.2.1 Competency Assessment Questions

Question 1: How would you rate your musical ability?

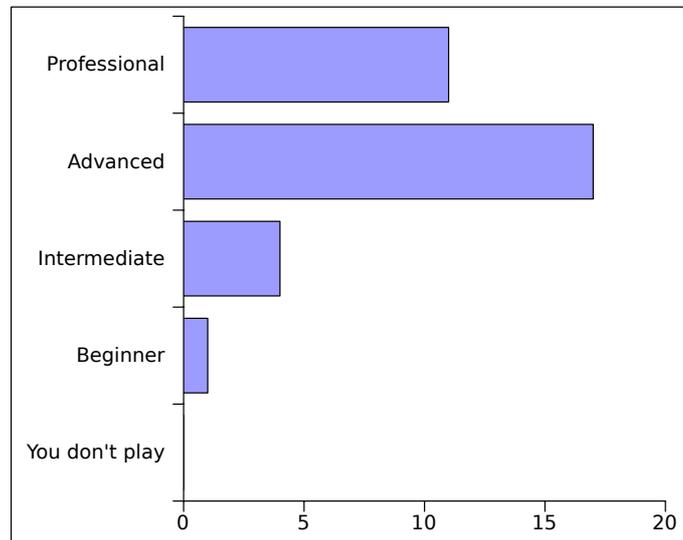


Figure 5.6: Musical Ability Rating.

For this question, respondents were asked to rate their musical ability in broad terms. 51.52% of respondents considered themselves to be musically advanced, with a further 33.33% claiming to be professional musicians or music educators (Figure 5.6). This accounts for over 80% of the respondents, with the remaining 5 respondents rating themselves at either intermediate or beginner level. This satisfies the preferred target demographic of professional musicians and music teachers. There were no respondents who do not play a musical instrument at all.

Question 2: Have you used a standard metronome before (software or hardware)?

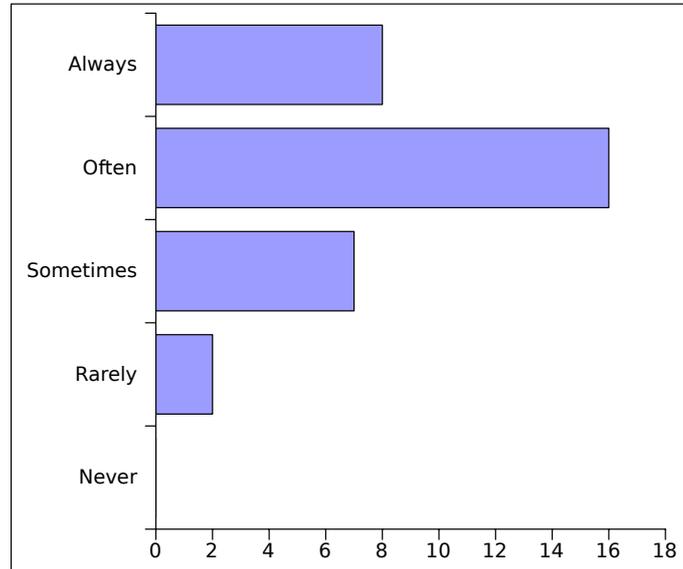


Figure 5.7: Standard Metronome Use.

This question is intended to assess the respondents previous experience with timing training. Due to the nature of the training methods employed by the Alternate Timing Training System, a user unfamiliar with timing training using a standard metronome would be unlikely to realise any benefits from the Alternate Timing Training System. The Alternate Timing Training System developed for this research is targeted at intermediate to advanced musicians already familiar with synchronising to a standard metronome. The previous graph shows that 16 of the 33 respondents (48.48%) use a metronome ‘often’, 8 respondents use a metronome ‘always’ (24.24%) and the remaining 9 respondents using a metronome ‘sometimes’ or ‘rarely’ (Figure 5.7). A substantial percentage of respondents are therefore familiar with the use of a metronome as a timing training aid.

Question 3: Please rate the importance of a good sense of time within the following musical settings (5 is most important, 1 is not important):

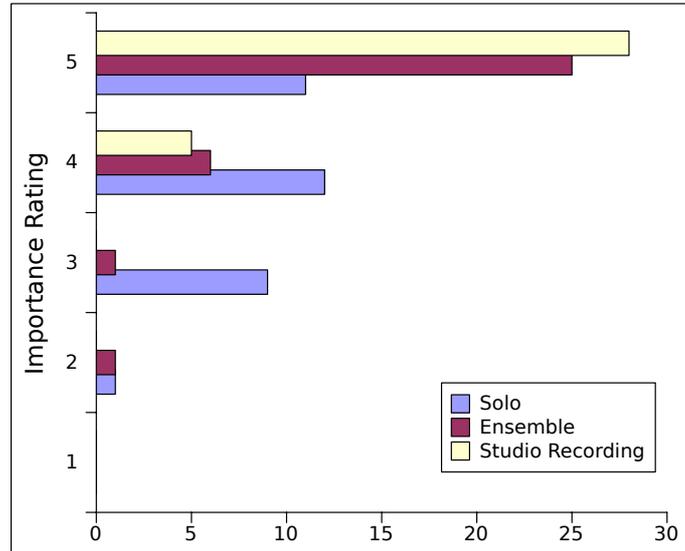


Figure 5.8: Importance of Sense of Time Rating.

This matrix-type question asks the respondent to rate the importance of good timing within three different musical performance scenarios, namely *Solo Performance*, *Ensemble Performance* and *Recording Studio Performance*. The above graph (Figure 5.8) summarises the response frequencies for all three scenarios. In all three scenarios having a good sense of time was considered important by respondents.

	Mean	Standard Deviation
<i>Solo Performance</i>	4.0000	0.8660
<i>Ensemble Performance</i>	4.6667	0.6922
<i>Recording Studio Performance</i>	4.8485	0.3641

Table 5.10: Mean and Standard Deviation

Table 5.10 presents some descriptive statistics, the mean and standard deviation for each set of responses. It can be seen that there is an increase in the mean response and a decrease in the standard deviation for each successive scenario, indicating a corresponding increase in the perceived importance of a good sense of time. A good sense of time is considered least important in solo

performances, but with a wide spread of responses. A good sense of time during a recording studio performance is considered to be very important by almost all respondents.

Question 4: How would you rate ‘sense of time’ in comparison to other musical skills such as theory knowledge, technical skill and ‘a good ear’?

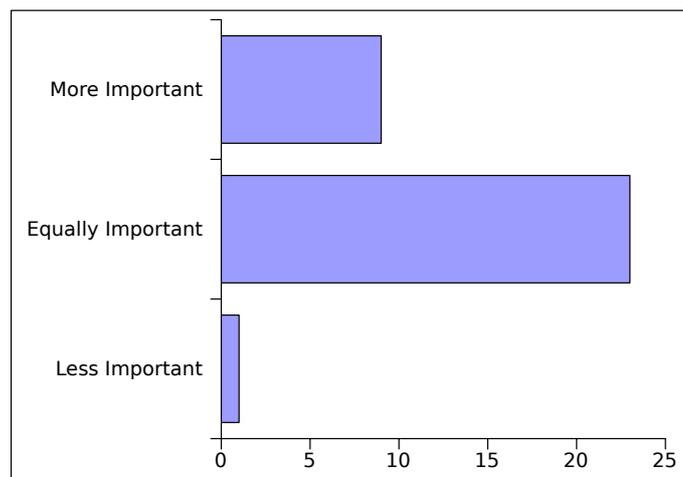


Figure 5.9: Sense of Time Compared with Other Musical Skills.

By examining the previous graph (Figure 5.9), we can see that the majority of respondents (69.7%) rate sense of time as equally important to other musical skills. 27.27% of respondents rated sense of time as more important than other musical skills. This is somewhat unexpected until the responses to this question are cross-tabulated with the responses to question 5: ‘What instrument do you play?’. This is discussed along with the next question.

Question 5: What instrument do you play?

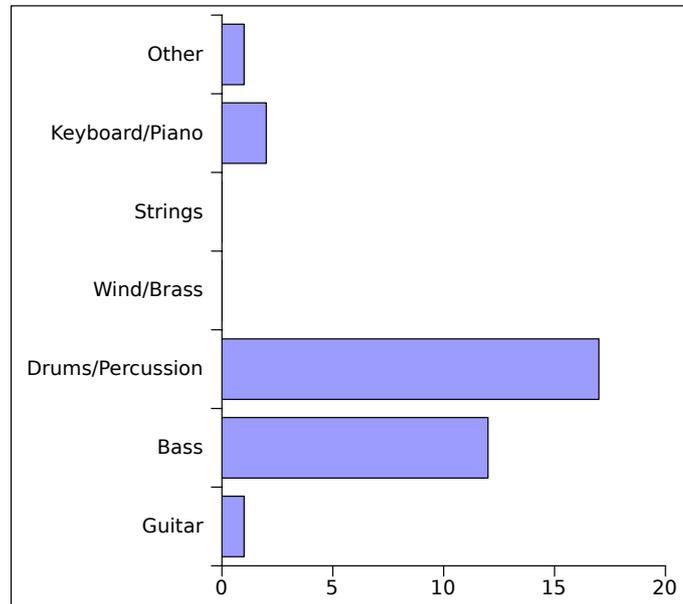


Figure 5.10: Subjects' Instruments.

From this graph (Figure 5.10) we can see that the specialised target demographic of drummers and percussionists forms the main body of respondents with 51.52% of responses. Bass players form the next significant percentage of respondents with 36.36%. The reason for such high representation by drummers and bass players is due to the postings placed on public online forums seeking survey participants. The forums chosen are specifically targeted at drummers and bass players. Other responses came from 1 guitarist, 2 keyboard players and 1 in the 'other' category.

The importance of sense of time grouped by instrument preference

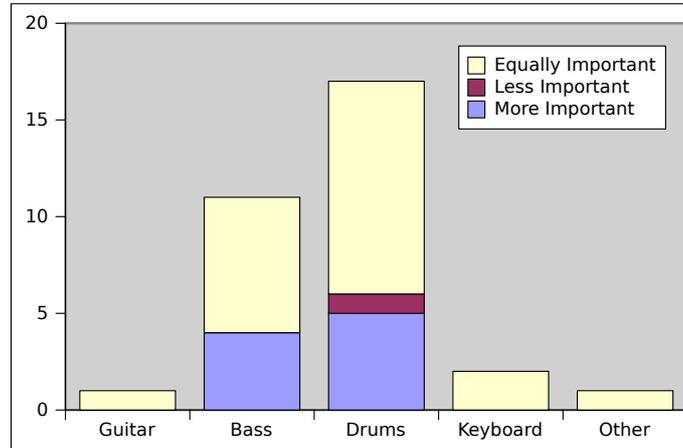


Figure 5.11: Importance of Sense of Time Grouped by Subject's Instrument.

An analysis of perceived importance of sense of time grouped by preferred musical instrument reveals how opinions and preferences are affected by the function of different instruments within a musical group.

The above graph (Figure 5.11) shows a visual representation of a contingency table showing a cross-tabulated breakdown of survey question 4 (“How would you rate ‘sense of time’ in comparison to other musical skills such as theory knowledge, technical skill and ‘a good ear’?”) with question 5 (“What instrument do you play?”).

Of the 33 respondents, 23 rated having a good sense of time as equally important to other musical skills. 9 rated sense of time as more important, all of them either bass players or drummers, meaning that their primary musical function is to keep time. Interestingly, there was one respondent, a drummer, who rated sense of time as less important than other musical skills, but this may have been an input error when completing the question.

Analysis indicates that on the whole, musicians value sense of time as a musical skill, and that a few drummers and bass players rate it as the most important musical skill.

Competency Assessment Discussion

Any committed musician, whether amateur or professional, needs to develop a range of skills and knowledge to support him in different musical situations. These include things such as technical facility on his chosen instrument, aural skills such as relative pitch, sight reading of music scores, knowledge of the the-

ory of music and a good sense of time. These skills are all important, although some more so than others depending on the given musical situation. Improvising, for example, does not necessarily require sight reading skills. The function of certain instruments within an ensemble situation may also place emphasis on some skills more than others. The rhythm section in a modern ensemble, usually consisting of bass and drums, has the responsibility of maintaining the tempo and timing of the group, and as such drummers and bass players require a more developed sense of time.

Examining the responses to the survey questions in the *Competency Assessment* section shows that a significant proportion of respondents consider themselves competent or expert musicians. Most respondents have suitable experience with timing training using a standard metronome and consider a good sense of time to be an important skill. Analysis of responses in this section indicate that respondents can offer informed opinion on timing training and on the Alternate Timing Training System.

5.2.2 Application Usage Assessment Questions

Question 6: How much did you use the timing training software?

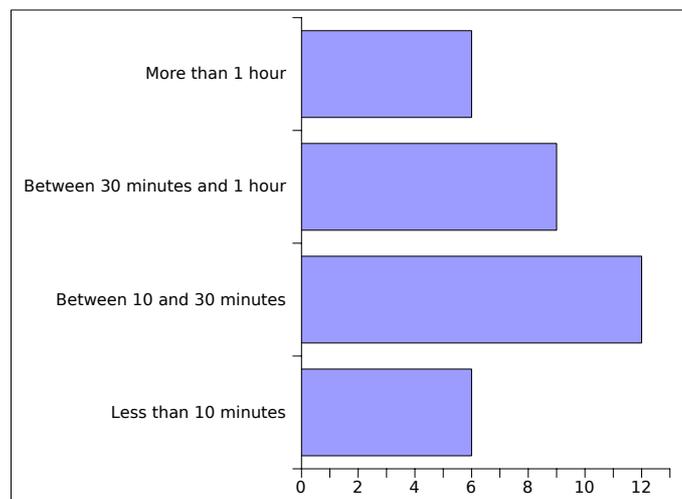


Figure 5.12: Alternate Timing Training System Usage.

This question is aimed at determining the quality of a respondent's answers, as the more time spent with the software the more qualified the opinion of the respondent. The above graph (Figure 5.12) shows that the highest percentage of users (36.36%) used the software for between 10 and 30 minutes before completing the survey. The standard deviation for this question is 1.00284, which

indicates a fairly broad spread of usage time by respondents.

Question 7: Did you read the mini manual?

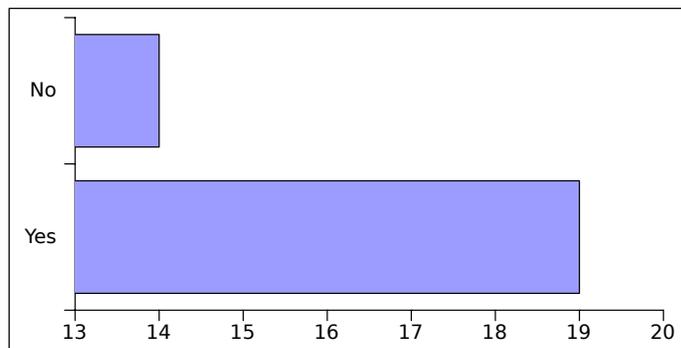


Figure 5.13: Did you read the mini manual?

This graph (Figure 5.13) indicates that 57.58% of respondents read the instruction manual made available on the website along with the software for download. The manual covered some of the underlying concepts driving the Alternate Timing Training System, as well as a user interface reference and some usage suggestions.

Question 8: Did you play along to the timing training software on your chosen instrument?

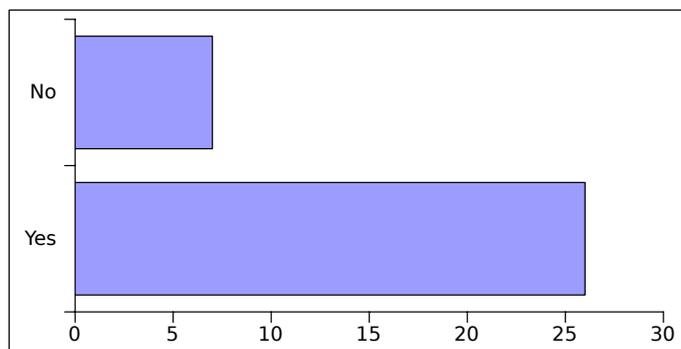


Figure 5.14: Alternate Timing Training System Usage in Conjunction with Subject's Instrument.

The concepts underlying the Alternate Timing Training System would not necessarily be evident to a musician should he not attempt to use it in a practice situation. The user may not realise the benefits or problems without actually playing something on his chosen instrument along to the Alternate Timing

Training System software. The previous graph (Figure 5.14) shows that 78.79% of respondents used the software in conjunction with their instruments.

Application Usage Discussion

The intention of this section of survey questions was to assess whether respondents had spent sufficient time with the Alternate Timing Training System, preferably with their instruments, to gain an understanding of the concepts and intention of the System. While quantitative evaluation of the efficacy of the Alternate Timing Training System would require far more than a few minutes, or even hours spent with the application, an understanding can be gained by reading the mini manual and playing along to some of the click patterns provided. This should not take more than about 20 or 30 minutes.

Analysis of responses to the *Application Usage* questions indicates that only 6 of the 33 respondents used the Alternate Timing Training System for less than 10 minutes. Opinions gathered in the following section would therefore be valid.

5.2.3 Opinion Questions

Question 9: Choose an option that best describes your opinion of the timing training software

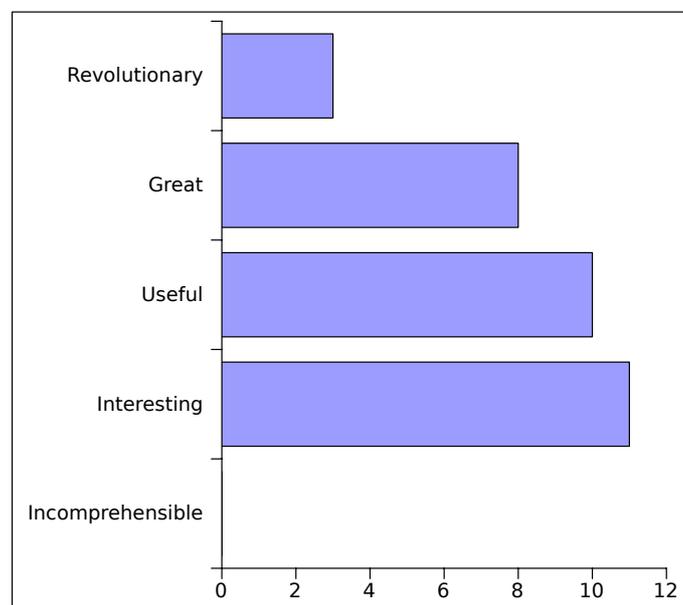


Figure 5.15: Adjectives to Describe the Alternate Timing Training System.

The first of the questions in the opinion category, question 9 presented a list

of adjectives to the respondent and asked them to choose the one they thought best described the Alternate Timing Training System. While the intention of this question was to provide respondents with a graded set of adjectives covering negative, neutral and positive opinions, the question is flawed due to adjectives being interpreted as mostly positive. Having said that, examining the graph above (Figure 5.15) shows results mostly concentrated around the centre (mean = 3.09375), indicating that users were neither overwhelmingly positive nor negative in their opinion. The graph shows the adjectives and the proportion of responses for each. The most popular option was *Interesting*, chosen by 11 out of 33 respondents (33.33%), with *Useful* the next most popular chosen by 10 respondents (30.30%). 1 Person elected not to answer the question, and no one chose *Incomprehensible*. 3 Chose *Revolutionary*.

Question 10: Would you use this software as a practice tool?

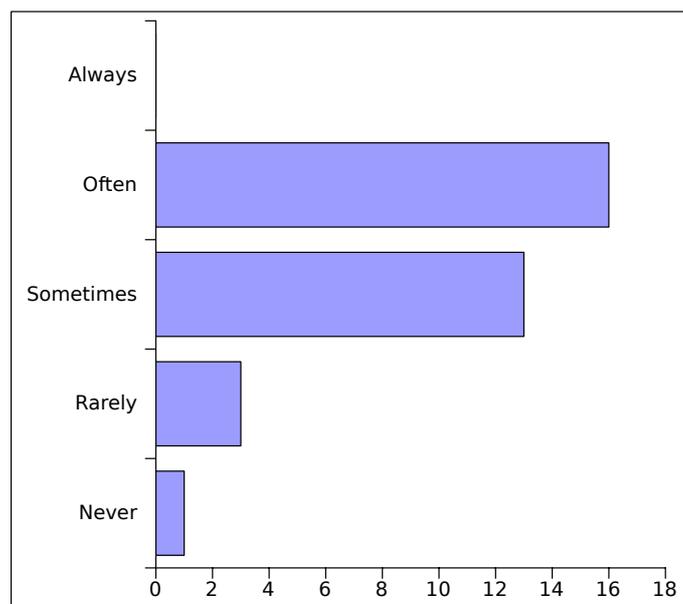


Figure 5.16: Would you use this software as a practice tool?

Responses to this question indicate that a large proportion of respondents believe they would use the software as a regular practice tool (Figure 5.16). 48.48% Believe they would use the software often, with a further 39.39% claiming they would use the software sometimes. Only 1 respondent said they would never use the software as a practice tool. Answers to this question indicate a willingness on the part of users to use timing training systems other than a standard metronome.

Question 11: Could the timing training software be used as a teaching tool?

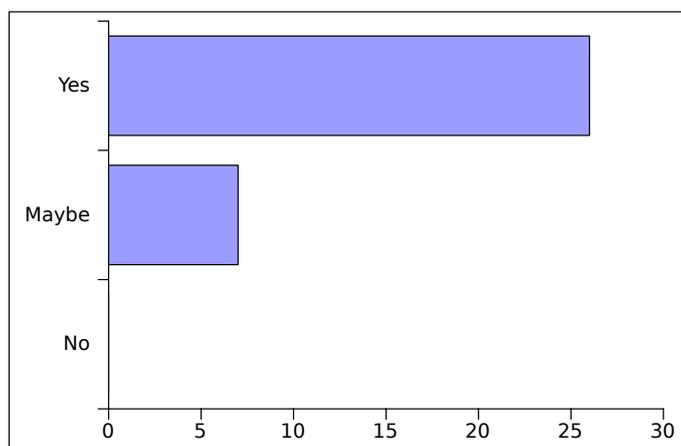


Figure 5.17: Potential as a Teaching Tool.

26 Of the 33 respondents believed that the software could be used as a teaching tool, with the remaining 7 respondents ambivalent about the possibility (see graph above, Figure 5.17). Although the Alternate Timing Training System was designed primarily as an aid to improving a musician's sense of time, the potential exists for the tool to be used in other ways. No one was against the idea of using the software as a teaching tool. While answers to this question do not directly inform the potential efficacy of the Alternate Timing Training System, they could inform future work.

Question 12: Could the same benefits offered by the timing training software be achieved with a standard metronome (hardware or software)?

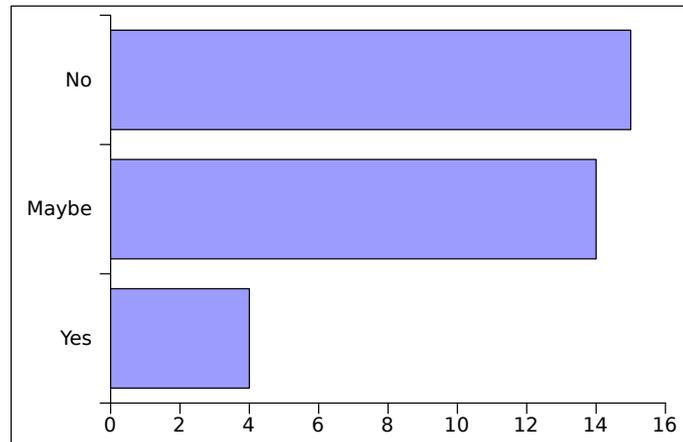


Figure 5.18: Could the same benefits be achieved with a standard metronome?

The Alternate Timing Training System offers a number of benefits over a standard metronome, as supported by the responses illustrated by the graph (Figure 5.18). Only a small proportion of respondents (12.12%) said that the same benefits could be achieved using a standard metronome, with 45.45% believing the same benefits could not be achieved using a standard metronome and the remaining 42.42% unsure. Using a standard metronome in the way that the Alternate Timing Training System makes use of an isochronous click is very difficult without the supporting functionality offered by the software, such as the accompanying primary pulse and the count-in functionality.

Question 13: Is the timing training software more musically interesting to use than a standard metronome (hardware or software)?

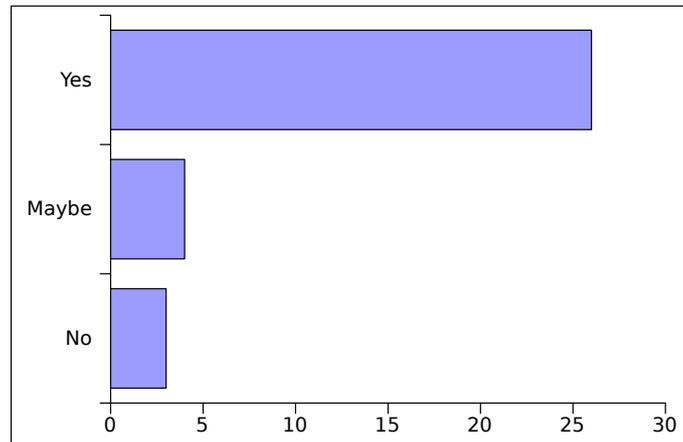


Figure 5.19: Is the Alternate Timing Training System more interesting to use?

By far the largest proportion of respondents agreed that the Alternate Timing Training System application was more musically interesting to use than a standard metronome. The previous graph (Figure 5.19) shows that 26 of the 33 respondents viewed the software as more interesting to use, while only 3 believed that the software was not more interesting to use. 4 Respondents were undecided. While ‘interesting’ is not a quantifiable term, future work could explore the benefits of providing varying and engaging ways in which to improve the user’s experience of what is normally repetitive drill-type training.

Question 14: How good is the specified metronome at improving a musician’s sensitivity to tempo changes? (1 is poor, 5 is excellent)

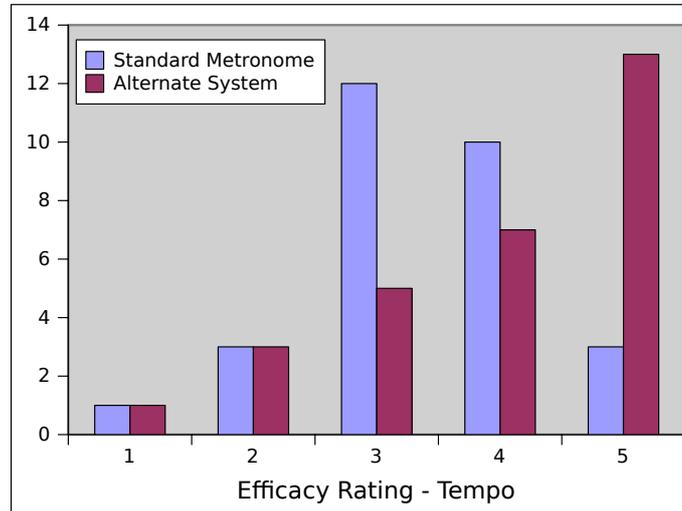


Figure 5.20: Potential Tempo Accuracy Improvement Rating.

This is one of the fundamental survey questions, and is asking respondents to compare the Alternate Timing Training System to a standard metronome. Aside from other benefits offered by software, such as convenience and variety, does the software perform its core function – that of helping to improve a musician’s sense of time? Specifically, this question is asking whether the research software is any better or worse at improving a musician’s sense of tempo when compared to a standard metronome. Respondents were asked to rate the efficacy of each type of training system on a scale of 1 to 5. Referring to the graph (Figure 5.20) above it can be seen that the highest proportion of respondents (36.36%) rate a standard metronome at level 3 and the Alternate Timing Training System at level 5 (39.39%). In both cases there is no obviously popular choice, with a fairly large spread. Comparing both the *mean* and the *standard deviation* (Table 5.11) shows the Alternate Timing Training System with a slightly higher *mean* but a larger *standard deviation*.

	Mean	Standard Deviation
<i>Standard Metronome</i>	3.3793	0.9416
<i>Timing Training Software</i>	3.9655	1.1797

Table 5.11: Mean and Standard Deviation

Grouping by Musical Ability

The following analysis is a means of examining the perceived efficacy of a standard metronome and of the Alternate Timing Training System with regard to tempo accuracy improvement, but with opinions grouped by musical ability (Question 1). It allows us to see whether different levels of musical experience result in significantly differing opinions. Due to Questions 14 being a matrix-type question, a separate graph is needed for both standard metronome ratings and for Alternate Timing Training System software ratings. These can then be compared.

The graphs below (Figure 5.21, 5.22) show a cross-tabulation of the questions “How would you rate your musical ability?” and “How good is the specified metronome at improving a musician’s sensitivity to tempo changes?”, firstly for a standard metronome, and secondly for the Alternate Timing Training System.

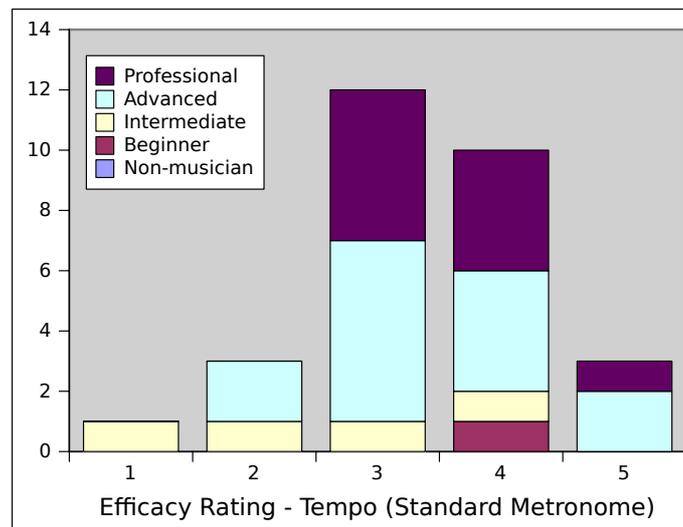


Figure 5.21: Tempo Accuracy Improvement Rating Grouped by Musical Ability – Standard Metronome.

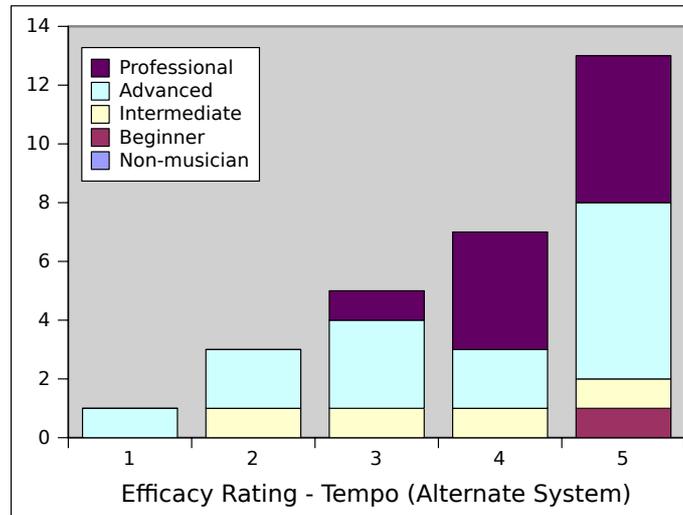


Figure 5.22: Tempo Accuracy Improvement Rating Grouped by Musical Ability – Alternate Timing Training System.

An examination of the results depicted in the first graph shows that the advanced and professional musicians mostly rate a standard metronome’s benefits at between 3 and 4 (19 respondents), with beginners and intermediates fairly evenly spread between 1 and 4. Overall there seems to be a fairly wide range of opinions with regard to how effective a standard metronome is at improving tempo sensitivity and accuracy.

In comparison to the ratings of the standard metronome, we see a marked shift in favour of the Alternate Timing Training System software, but with a slightly wider spread. This can clearly be seen in the second graph above.

Question 15: How good is the specified metronome at improving a musician’s sensitivity to subdivisions of the beat? (1 is poor, 5 is excellent)

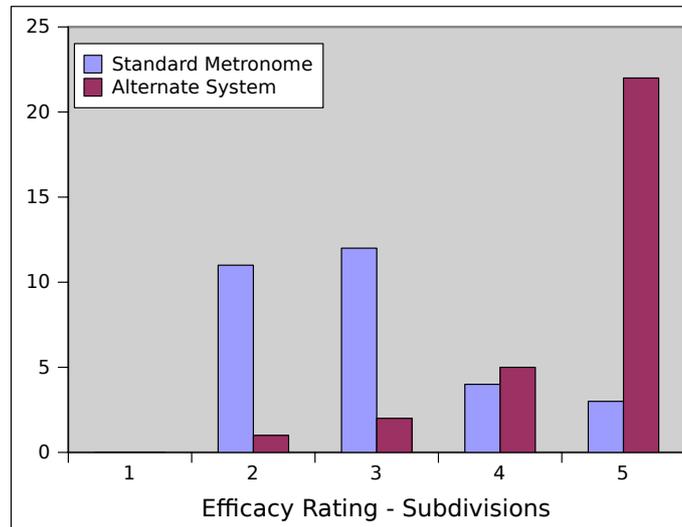


Figure 5.23: Potential Subdivision Accuracy Improvement Rating.

Similar to the previous question, this important survey question is asking respondents to compare specific benefits of using different types of timing training aids, namely a standard metronome and the Alternate Timing Training System research application. Respondents were asked whether a standard metronome is more or less effective than the Alternate Timing Training System software at improving a musician’s sense of subdivision¹. This was achieved by respondents rating the effectiveness of each system on a scale of 1 to 5. The previous graph (Figure 5.23) shows the results for both metronome types, with the highest proportion (36.36%) of respondents rating a standard metronome at level 3 and 69.70% of respondents for the software rating it at level 5. The means and standard deviations for both sets of results are presented in Table 5.12, with the Alternate Timing Training System showing a much higher mean and a smaller spread. The Alternate Timing Training System software is therefore perceived as being far more effective than a standard metronome at training a musician’s sense of subdivision.

As with Question 14 - “How good is the specified metronome at improving

¹Subdividing temporal intervals into smaller, evenly spaced intervals is a fundamental musical skill. For example, the ability to play two notes for every standard metronome click requires that the musician subdivide the temporal interval provided by the metronome into two exactly equal parts

	Mean	Standard Deviation
<i>Standard Metronome</i>	2.9667	0.9643
<i>Timing Training Software</i>	4.6	0.7701

Table 5.12: Mean and Standard Deviation

a musician’s sensitivity to tempo changes?” - the following section examines responses to Question 15 grouped by musical ability.

Grouping by Musical Ability

The following graphs (Figure 5.24, 5.25) show a cross-tabulation of the questions “How would you rate your musical ability?” and “How good is the specified metronome at improving a musician’s sensitivity to subdivisions of the beat? (1 is poor, 5 is excellent)”, both for a standard metronome and for the Alternate Timing Training System.

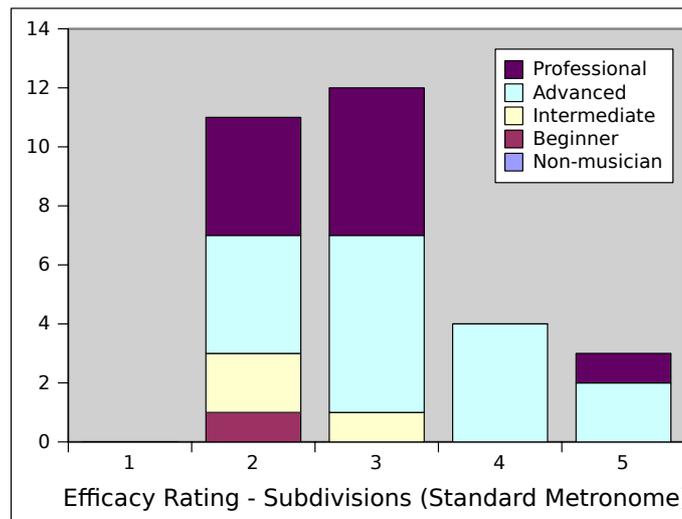


Figure 5.24: Subdivision Accuracy Improvement Rating Grouped by Musical Ability – Standard Metronome.

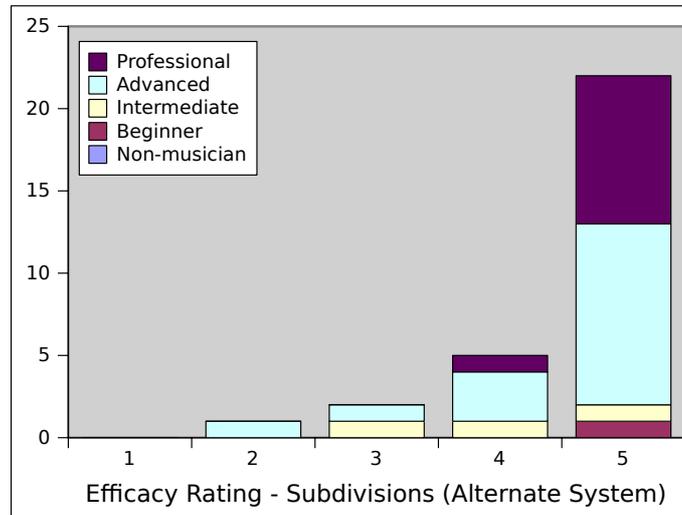


Figure 5.25: Subdivision Accuracy Improvement Rating Grouped by Musical Ability – Alternate Timing Training System.

22 Respondents rated the effectiveness of a standard metronome at improving subdivision sensitivity at 3 or below, with only 7 respondents giving ratings of 4 or 5. The few higher ratings were all given by advanced or professional musicians.

Only 8 of the 30 respondents gave a rating of less than 5 for the Alternate Timing Training System software as an effective means of improving subdivision sensitivity. Comparing the second graph to the ratings for the standard metronome shown in the first graph it is obvious that respondents are of the opinion that the prototype software (the Alternate Timing Training System) is far more effective than a standard metronome at improving a musician’s sensitivity to subdivisions of the beat.

Opinion Questions Discussion

Responses in this section reflected informed opinion of the Alternate Timing Training System as a practice tool and teaching tool, and more importantly, as an effective means of improving a musician’s sense of tempo and subdivision. Future work could explore the suitability of the System as a practice or teaching tool, especially in terms of grading the difficulty of the different patterns and how best to present them. Analysis of Questions 14 and 15 showed a significant preference for the Alternate Timing Training System over a standard metronome as an effective means of training tempo sensitivity and especially subdivision sensitivity.

Question 16: Any other comments, thoughts or criticisms on the timing training software?

As a final question, respondents were asked to provide any relevant feedback regarding their use of the software. The responses were in a free-form text format, with the question being optional. 22 Of the 33 respondents elected to provide feedback. Selected responses will be discussed briefly here, as not all comments added value. See Appendix D, Section D.2 for a complete list of responses.

“Is it possible to have the tempo adjusted by click & drag, like music apps do?”. This comment is referring to the usability of the Alternate Timing Training System software, and while valid, it was not the intention of the survey to assess the usability of the software, but rather the efficacy of the implemented Alternate Timing Training System.

“it’s not too bad...i dont think it will take over in popularity of the normal metronome but i like it...it’s making me focus on how out of time and in time i go :)” Although non-committal about the potential popularity of the system, the respondent’s claim that the system is enhancing his or her awareness of his or her sense of time suggests that the system is effective.

“its been a while since i last worked with the metronome but i remember saying at the time something about that it would help if the repetitions were in 4/4 groupings rather than arbitrary time signatures”. Because so much contemporary music is based in groups of 4 or 3, musicians tend to naturally ‘feel’ things in groups of 4 or 3. The respondent’s suggestion would be a useful feature to include in a real-world application.

“I would include a function that monitors the mouse buttons, so that you can ‘tap along’ on the mouse (or keyboard buttons) along with the beat. That way you can train your time keeping when you haven’t got an instrument on your hands. Also, I would have the beat fade in an out. The way it’s set up at the moment, you have no idea if you’re still keeping the right time after a while.” The suggestion by this respondent to have the primary click fade in again after a specified period of time was the feature most requested by users of the Alternate Timing Training System software. It would confirm to the user that he is still playing in time, as with some of the more difficult patterns it is easy to stray from the primary pulse. It would mean that the user does not have to stop and restart the click every time he or she makes a mistake. It would be important to

include this feature as part of a usability study, or in a real-world or commercial implementation.

“Great idea, I like it a lot. Definitely a different approach that I haven’t seen yet in a metronome. I currently use Y-Metronome, but I can definitely see your metronome getting a lot of use as well. A couple suggestions. Add a setting for 0 seconds in the fade start list. As well as an option to turn off the fade so it switches instantly. This way you can let the regular metronome go and then switch immediately to the ‘off beat’ metronome. This will increase the difficulty as you will never hear the two metronomes playing together. It will be one or the other. Another suggestion is to add an option to switch back and forth between the regular straight metronome and the off beat metronome after ‘X’ amount of measures (that you can set separately for both the normal and off beat metronome). This way you can ‘reset’ your timing so to speak if you get off time without having to stop playing and restart the metronome. Then to increase difficulty you can increase the number of measures for the offbeat metronome.”

These are good suggestions for a full-featured application, and indicate that the respondent had a good understanding of the concepts and intention of the Alternate Timing Training System.

“Show 8 beats in the bottom where it says ‘standard’, not 7. Also, have a distinctive sound for the ‘1’ of the beat in standard metronome, it could help greatly with getting a solid count to start with. It may also help if you put the 1 & 2 & 3 & 4 & somewhere in the diagrams at the bottom. Good stuff!”

Similar to one of the other comments, musicians prefer phrases that are grouped by 3 or 4 beats or bars.

“The instructions do not make clear enough the point of the software. It took a few reads to understand the idea is to hear the time pulse in your head at all times and play with that, not to play with the secondary pulse. It could be worded more clearly to better communicate the purpose. However, it is very good and I think it would be really useful. I think the object of most musicians should be to develop the clave sense, and I imagine this would help a lot. I do feel that a sense of perfect time is almost as rare as perfect pitch, but this software would help develop that skill and help a musician develop a very strong poly-rhythmic sense.”

This is one of two comments that suggested better/more documentation explaining the concepts underlying the Alternate Timing Training System. More

effort should have been invested into conveying the principles, perhaps with a selection of examples. Subjects who volunteered for the experiment were provided with a short demonstration video. This should have been made available online as well.

“The difficulty slider seemed like a odd name for it. I found setting 6 easier to play then 5. The number relates more to the beat then to its difficulty. Nice tool though”

A number of users of the Alternate Timing Training System software noted that the word ‘Difficulty’ was a poor term for the pattern selection slider, as certain patterns may be easy for some and more difficult for others.

“I found this to be excellent for establishing ones ‘internal clock’ when executing syncopated rhythms and would use it often. I didn’t understand question 14 so I’ll read the manual to see if I missed something. Not reading the manual first was intentional as to assess the initial ‘pick it up and use it’ ability and it passed with flying colors. I would like to see a snare sound but this comes from a 40 year bassist and I’m more used to that versus cowbells, etc.”

In contrast to two other respondents who felt that the documentation was lacking, this respondent thought that the intention of the software was self-explanatory.

“I found one glitch. When I had both clicks as the same sound, (woodblock1 for example) any bpm above 152 I was getting a click noise not a woodblock sound.”

This may be a bug in the Alternate Timing Training System software.

“Mini-manual should be a little less ‘mini’ – it would help to have a more complete explanation of the concepts involved, some practical examples of its usage to achieve specific goals, etc. Also, as far as the interface goes, would like to have a numeric stepper widget for the tempo box, and a lot more control for what’s going on the pattern section. Right now, the relationship between difficulty, standard, and pattern is a bit obscure. It’s genuinely interesting and has the potential to be very useful. Hope you keep working on it.”

This is the second respondent who would have preferred more documentation explaining the principles of the Alternate Timing Training System. Other suggestions were mostly to do with usability and configurability, functionality that could be included in a usability study or a full-featured version of the application.

“it doesn’t actually test what you can do - it’s a learning tool, right, not some-

thing you'd use on stage or in studio - learning is complex and needs stages and modules - a guided process. this is like a plug-in gadget - offers no program on how to take the student from A to B, how to grow etc. hope this comes soon ..."

This respondent emailed the author with these and other comments which are reproduced in Appendix D.

"If you could include a feature that helps people practice the subdivisions in odd-meter, it would make this an awesome tool. One can use it for odd meter (sort-of) as it is now, but it would be better (easier to use for this purpose) if there was a menu where you could choose the time signature, and work on subdivisions that are derived from that time signature."

The Alternate Timing Training System software does not impose any meter on the user, in the same way that the simplest standard metronomes do not have any indication of meter. The respondent may be requesting a feature that would indicate a chosen meter to the user, for example a different sound every four beats for a 4/4 meter, or every seven beats for a 7/4 meter. This would satisfy other requests for the ability to group beats and patterns into four bar sections.

"It would be a useful feature to be able to fade the standard metronome back in after a specified interval, in order to determine whether or not time has been kept accurately."

Another user who suggested the return of the primary pulse click after a specified period.

5.2.4 Discussion

Although answers to the first five survey questions allowed for the partitioning of responses by respondent expertise, this was unnecessary except in a few isolated cases (see the discussion of responses to questions 14 and 15). Over 80% of respondents considered themselves either advanced or professional musicians, indicating a high level of expertise. If there had been a broader spread of ability levels it may have been beneficial to partition survey responses by ability as part of the analysis, but with the bulk of respondents considered experts the data provided by non-experts were largely insignificant. Other questions in the *Competency Assessment* section (Questions 1 – 5) also indicated a general consensus in terms of the importance of having a good sense of time, as well as indicating that a large proportion of respondents were not only expert musicians, but were either bass players or drummers (well over 80%). Most respondents had sufficient experience with a standard metronome to qualify their opinions

of the Alternate Timing Training System.

Besides a sufficient level of expertise, it was desirable for respondents to spend sufficient time with the Alternate Timing Training System in order to form a valid opinion. Questions 6, 7 and 8 were designed to assess whether the respondent had used the Alternate Timing Training System software enough and whether they had an understanding of the training concepts implemented by the software. Over 80% of respondents used the Alternate Timing Training System software for more than 10 minutes, with just under 80% using the software with their instruments. Only 57% read the manual - a higher percentage would have been preferable, but overall it can be assumed that the majority of subjects acquired a good understanding of the functionality offered by the Alternate Timing Training System, enough to offer informed opinions of the efficacy of the system.

Survey questions 9 to 15 were designed to assess respondents' opinion of the Alternate Timing Training System. The core questions are 14 and 15 - the efficacy questions. These asked the respondent to rate the efficacy of the Alternate Timing Training System compared to a standard metronome, both for improving sensitivity to tempo and to subdivisions. For further clarification the responses were grouped by respondent expertise, as indicated by the responses to question 1. Respondents rated the Alternate Timing Training System as more effective than a standard metronome, both in terms of improving tempo sensitivity and in terms of improving sensitivity to subdivisions of the beat, with the Alternate Timing Training System rating very highly in terms of subdivision sensitivity development. Questions 9 to 13 asked the respondent's opinion on other possible uses and benefits of the Alternate Timing Training System. Although the answers to these questions did not contribute directly to the questions asked by this research, they indicated positive opinion of the Alternate Timing Training System as a possible practice or teaching tool, and could possibly inform further work.

5.2.5 Supplementary Discussion

Recruiting respondents for the survey resulted in a few interesting discussions between the author and the respondents, both via email and in online forums. These are included in Appendix D.

5.3 Summary

Data collected from the assessments recorded during the month-long experiment were analysed in four different ways using descriptive statistics and an Analysis of Variance test. Analysis showed no statistically significant effect of either standard metronome or Alternate Timing Training System on the timing accuracy of the subjects. Inconclusive results could be due to a number of factors, such as the initial skill level of the subjects and possibly insufficient training duration.

Survey responses were analysed using simple frequency distribution for each survey question. Multivariate frequency distribution analysis was performed on efficacy related questions. Probably the most significant results are those of the efficacy ratings analysis, with respondents showing a preference for the Alternate Timing Training System over a standard metronome for tempo accuracy development and overwhelmingly so for subdivision accuracy development. The response to Question 13 (“Is the timing training software more musically interesting to use than a standard metronome (hardware or software)?”) also indicated a majority preference for the Alternate Timing Training System software.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

Two approaches were taken in attempting to evaluate the efficacy, and potential efficacy, of a proposed Alternate Timing Training System, one quantitative and one qualitative. An experiment was conducted in which the timing accuracy of a number of test subjects was measured prior to and after the use of two different timing training systems – a standard metronome and the proposed Alternate Timing Training System. Data collected after a two week period spent with each system were analysed and compared in an attempt to determine the efficacy of each system. A public survey was also conducted in order to gather expert opinion on the Alternate Timing Training System. Responses were collected and analysed to compare opinion of the efficacy of a standard metronome to that of the Alternate Timing Training System.

Analysis of the data collected during the experiment produced inconclusive results due to a number of factors. These factors and their limitations could be used to inform future work in what is largely an unexplored area (see below). Limitations identified included:

- Limited availability of suitable candidates due to the requirement that subjects be skilled rhythm-section players familiar with the use of a standard metronome. This resulted in a very small sample size.
- Too short a training period.

- Subjects already had a well developed sense of time.
- Treatment order was not reversed using a second group.
- Limited method of measuring timing accuracy.

Training was limited to a two week period with each training system, possibly insufficient to produce statistically significant results in the timing accuracy of the subjects. The preferred demographic for experiment subjects was expert musicians, specifically drummers, percussionists and bass players, as these musicians are normally familiar with timing training and have a vested interest in having a good sense of time. However, it is possible that these subjects already have such a highly developed sense of time that any improvement would require more rigorous training over a longer period. The strictness of the demographic requirements also resulted in a limited number of available candidates.

The limited number of subjects meant that it was not possible for a second group to perform the treatments in reverse, that is to use the Advanced Timing Training System for the first two weeks and then the standard metronome for the second two weeks. This would have reduced the effects of practice effect and S-curve learning rates.

The method used for measuring a subject's timing accuracy was a further limiting factor, as it required some rudimentary skill with a pair of drum sticks. This was a further reason for the preferred demographic of drummers and percussionists.

As indicated by the responses to survey questions relating to tempo training and subdivision training, timing training targeted specifically at temporal subdivision abilities is far better addressed by the Alternate Timing Training System than by a standard metronome. This, in addition to the opinion of respondents that the Alternate Timing Training System is more interesting to use than a standard metronome, suggests that computers can be used in delivering effective and engaging training solutions for musicians wishing to improve their sense of time.

While analysis of the survey results indicates a preference for the Alternate Timing Training System implementation rather than for a standard metronome, a number of other factors should be taken into consideration: the demographic of the survey candidates selected is largely that of drummers and bass players who belong to an online community. Within a musical group, the primary function of bass players and drummers is to keep time. They therefore have a vested interest in possessing a good sense of time, and so may be more interested in timing training than would other types of musician. That being said, they may

also have a more informed opinion on timing and timing training. The fact that survey candidates were selected from online communities suggests candidates are at ease with the use of computers, and so may perhaps be more open to the use of computers as a training aid.

6.2 Future Work

Studies have shown that musical training does improve temporal accuracy [Franek et al., 1991], but investigations suggest that little research has been done into the specifics of effective training methods. Musical training is generally targeted at the improvement of technical proficiency on the chosen instrument. Other types of musical training exist that target specific musical skills, such as aural training for the improvement of relative pitch perception. Music practice that makes use of the standard metronome is the traditional method for improving a musician’s sense of time. Although this may be effective, the benefits offered by computers and software are largely unexplored at this time. The Alternate Timing Training System used in this research is potentially one of many possible systems, with further research needed to explore the effectiveness of different systems. The broader aspects of computer-assisted timing training systems could be explored, as well as specifics such as usability within different training environments.

The methodology applied in conducting the experiment for this research could inform future work in a number of ways. The choice of expert musicians as experiment subjects proved to be a limitation, both due to their already highly trained sense of time and due to the availability of suitable candidates. This ultimately resulted in a very small sample population of 5 subjects. The requirement for using expert musicians was an attempt to reduced any S-curve learning rates effect on the results, but this could be controlled in other ways. A within-subjects design would still be required, but a large enough sample of non-musicians or amateur musicians could be divided into two groups. One group could train with a standard metronome first, and then with an alternate system, the other group first with an alternate system and then with a standard metronome. Practice effect and S-curve learning rates would therefore be mitigated. Quantitative measurement of the actual efficacy of timing training would likely need to take place as part of a longer term study, but the use of non-expert subjects may result in measurable improvement that takes place in the shorter term, perhaps in a matter of weeks.

The method used in this research for measuring a subject’s timing accuracy,

while effective, is limited to the use of audio recordings made of subjects hitting a piece of wood with a drumstick. Other methods of measuring timing accuracy could be the subject of future work, such as the use of the MIDI protocol (Musical Instrument Digital Interface) for recording subjects' performances. This would allow the use of any instrument that supports MIDI to be used, opening up the preferred demographic to a wider variety of musicians. It may also allow for easier extraction of data for analysis.

The qualitative aspects of timing training can also be explored in future work, such as the use of a formal user interface design methodology to improve the usability of a given timing training system, whether it be standard metronome or an alternate system. The potential benefits of computer-aided timing training systems as teaching tools that deliver graded training is another aspect that needs further research.

Better understanding of the cognitive processes governing temporal perception and the sensory and motor systems could lead to better training methods, which in turn could be implemented using computers. Specifically, further research is needed into the way that musicians subdivide temporal intervals as, although related to tempo sensitivity, this is a different musical ability.

The Alternate Timing Training System implemented and evaluated for this research is but one of many possible timing training systems. Other alternate systems could make use of random, or pseudo-random placement of reference clicks, weighting of certain subdivisions, alternate time signatures and training sessions that increase in difficulty in real-time. Feedback systems could be developed that inform the user of the accuracy of their performance. The efficacy of these and other systems could be investigated through further research.

Bibliography

- [Aschersleben, 2002] Aschersleben, G. (2002). Temporal control of movements in sensorimotor synchronization. *Brain and Cognition*, 48(1):66–79.
- [Brady, 1969] Brady, J. P. (1969). Studies on the metronome effect on stuttering. *Behaviour Research and Therapy*, 7(2):197–204.
- [Brady, 1973] Brady, J. P. (1973). Metronome-conditioned relaxation: a new behavioural procedure. *The British Journal of Psychiatry: The Journal of Mental Science*, 122(571):729–730. PMID: 4716075.
- [Brady et al., 1974] Brady, J. P., Luborsky, L., and Kron, R. E. (1974). Blood pressure reduction in patients with essential hypertension through metronome-conditioned relaxation: A preliminary report. *Behavior Therapy*, 5(2):203–209.
- [Chan et al., 2006] Chan, L. M., Jones, A. C., Scanlon, E., and Joiner, R. (2006). The use of ICT to support the development of practical music skills through acquiring keyboard skills: a classroom based study. *Computers & Education*, 46(4):391–406.
- [Casper et al., 2009] Casper, S. M., Lee, G. P., Peters, S. B., and Bishop, E. (2009). Interactive metronome training in children with attention deficit and developmental coordination disorders. *International Journal of Rehabilitation Research*, 32(4):331–336.
- [Covington, 2005] Covington, K. (2005). The mind’s ear: I hear music and no one is performing. *College Music Symposium*, 45:25–41. ArticleType: research-article / Full publication date: 2005 / Copyright © 2005 College Music Society.
- [Crozier, 2005] Crozier, J. (2005). *Collins Discovery Encyclopedia*. Harper-Collins Publishers Limit, first edition.

- [Dalby, 1992] Dalby, B. F. (1992). A Computer-Based training program for developing harmonic intonation discrimination skill. *Journal of Research in Music Education*, 40(2):139–152.
- [Deihl and Zeigler, 1973] Deihl, N. C. and Zeigler, R. H. (1973). Evaluation of a CAI program in articulation, phrasing and rhythm for intermediate instrumentalists. *Bulletin of the Council for Research in Music Education*, (31):1–11. ArticleType: research-article / Full publication date: Winter, 1973 / Copyright © 1973 University of Illinois Press.
- [Drake and Heni, 2003] Drake, C. and Heni, J. B. E. (2003). Synchronizing with music: intercultural differences. *Annals of the New York Academy of Sciences*, 999:429–437. PMID: 14681167.
- [Drake and Palmer, 2000] Drake, C. and Palmer, C. (2000). Skill acquisition in music performance: relations between planning and temporal control. *Cognition*, 74(1):1–32.
- [Duke et al., 1991] Duke, R. A., Geringer, J. M., and Madsen, C. K. (1991). Performance of perceived beat in relation to age and music training. *Journal of Research in Music Education*, 39(1):35–45.
- [Franek et al., 1991] Franek, M., Mates, J., Radil, T., Beck, K., and Pöppel, E. (1991). Finger tapping in musicians and nonmusicians. *International Journal of Psychophysiology*, 11(3):277–279.
- [Friberg and Sundberg, 1993] Friberg, A. and Sundberg, J. (1993). Perception of just-noticeable time displacement of a tone presented in a metrical sequence at different tempos. *The Journal of the Acoustical Society of America*, 94(3):1859.
- [Hannon and Trainor, 2007] Hannon, E. E. and Trainor, L. J. (2007). Music acquisition: effects of enculturation and formal training on development. *Trends in Cognitive Sciences*, 11(11):466–472. PMID: 17981074.
- [Hirsh, 1959] Hirsh, I. J. (1959). Auditory perception of temporal order. *The Journal of the Acoustical Society of America*, 31(6):759–767.
- [Ho, 2007] Ho, W. (2007). Music students' perception of the use of multi-media technology at the graduate level in hong kong higher education. *Asia Pacific Education Review*, 8(1):12–26.

- [Hutchinson and Navarre, 1977] Hutchinson, J. M. and Navarre, B. M. (1977). The effect of metronome pacing on selected aerodynamic patterns of stuttered speech: Some preliminary observations and interpretations. *Journal of Fluency Disorders*, 2(3):189–204.
- [Kuhn, 1974a] Kuhn, T. L. (1974a). Discrimination of modulated beat tempo by professional musicians. *Journal of Research in Music Education*, 22(4):270–277.
- [Kuhn, 1974b] Kuhn, W. (1974b). Computer-Assisted instruction in music: Drill and practice in dictation. *College Music Symposium*, 14:89–101. ArticleType: research-article / Full publication date: Fall, 1974 / Copyright © 1974 College Music Society.
- [Libkuman et al., 2002] Libkuman, T., Otani, H., and Steger, N. (2002). Training in timing improves accuracy in golf. *The Journal of General Psychology*, 129(1):77–96.
- [Mates et al., 1994] Mates, J., Müller, U., Radil, T., and Pöppel, E. (1994). Temporal integration in sensorimotor synchronization. *Journal of Cognitive Neuroscience*, 6(4):332–340.
- [Meegan et al., 2000] Meegan, D. V., Aslin, R. N., and Jacobs, R. A. (2000). Motor timing learned without motor training. *Nat Neurosci*, 3(9):860–862.
- [Moses, 1989] Moses, B. (1989). *Drum Wisdom*. Modern Drummer Publications, Inc.
- [Nardo and Choe, 2010] Nardo, R. and Choe, E. J. (2010). Technology in the piano lab: Band-in-a-Box—An interview with E.J. Choe. *General Music Today*, 23(3):48–50.
- [Newell et al., 2001] Newell, K. M., Liu, Y.-T., and Mayer-Kress, G. (2001). Time scales in motor learning and development. *Psychological Review*, 108(1):57–82.
- [Palmer, 1997] Palmer, C. (1997). MUSIC PERFORMANCE. *Annual Review of Psychology*, 48(1):115–138.
- [Parncutt and McPherson, 2002] Parncutt, R. and McPherson, G. (2002). *The science & psychology of music performance: creative strategies for teaching and learning*. Oxford University Press.

- [Pascual-Leone, 2001] Pascual-Leone, A. (2001). The brain that plays music and is changed by it. *Annals of the New York Academy of Sciences*, 930:315–329. PMID: 11458838.
- [Repp, 2005a] Repp, B. H. (2005a). Rate limits of On-Beat and Off-Beat tapping with simple auditory rhythms. *Music Perception*, 23(2):165–188.
- [Repp, 2005b] Repp, B. H. (2005b). Sensorimotor synchronization: A review of the tapping literature. <http://pbr.psychonomic-journals.org/content/12/6/969.abstract>.
- [Repp and Doggett, 2007] Repp, B. H. and Doggett, R. (2007). Tapping to a very slow beat: A comparison of musicians and nonmusicians. *Music Perception*, 24(4):367–376.
- [Schlaug et al., 1995] Schlaug, G., Jäncke, L., Huang, Y., Staiger, J. F., and Steinmetz, H. (1995). Increased corpus callosum size in musicians. *Neuropsychologia*, 33(8):1047–1055. PMID: 8524453.
- [Shaffer, 1981] Shaffer, L. H. (1981). Performances of chopin, bach, and bartok: Studies in motor programming. *Cognitive Psychology*, 13(3):326–376.
- [Shaffer et al., 2001] Shaffer, R. J., Jacokes, L. E., Cassily, J. F., Greenspan, S. I., Tuchman, R. F., and Stemmer, P. J. (2001). Effect of interactive metronome® training on children with ADHD. *The American Journal of Occupational Therapy*, 55(2):155–162.
- [Sierpiński and Schinzel, 1988] Sierpiński, W. and Schinzel, A. (1988). *Elementary theory of numbers*. Elsevier.
- [Spatz, 2010] Spatz, C. (2010). *Basic Statistics: Tales of Distributions*. Cengage Learning.
- [Thaut et al., 1997] Thaut, M. H., McIntosh, G. C., and Rice, R. R. (1997). Rhythmic facilitation of gait training in hemiparetic stroke rehabilitation. *Journal of the Neurological Sciences*, 151(2):207–212. PMID: 9349677.
- [Thaut et al., 1996] Thaut, M. H., McIntosh, G. C., Rice, R. R., Miller, R. A., Rathbun, J., and Brault, J. M. (1996). Rhythmic auditory stimulation in gait training for parkinson’s disease patients. *Movement Disorders: Official Journal of the Movement Disorder Society*, 11(2):193–200. PMID: 8684391.
- [Voros, 1995] Voros, G. (1995). *Rhythm of the Head*. Big Drum Press, The.

[Willett and Netusil, 1989] Willett, B. E. and Netusil, A. J. (1989). Music computer drill and learning styles at the Fourth-Grade level. *Journal of Research in Music Education*, 37(3):219 –229.

Appendix A

Source Code

A.1 Alternate Timing Training System Implementation

The Alternate Timing Training System prototype application was implemented in Java using the open source Netbeans IDE. The Java Sound API was used for audio playback and the Swing framework for the user interface. The following listing does not include the user interface code. See also Appendix B for a simplified class diagram.

```
package clickengine;

import javax.sound.sampled.SourceDataLine;

/**
 * Runnable implementation that writes
 * audio data to a supplied SourceDataLine
 * @author David Manchip, MNCDAV002
 */
class AudioStreamWriter implements Runnable
{
    private SourceDataLine inputLine;
    private IAudioDataGenerator generator;
```

```

private Boolean stop = false;

AudioStreamWriter(SourceDataLine inputLine,
    IAudioDataGenerator generator)
{
    this.inputLine = inputLine;
    this.generator = generator;
}

@Override
public void run()
{
    inputLine.start();
    byte[] dataToWrite;

    while (!stop)
    {
        dataToWrite = generator.generate();
        inputLine.write(dataToWrite, 0, dataToWrite.
            length);
    }

    inputLine.stop();
    inputLine.flush();
    stop = false;
}

void setStop(Boolean stop)
{
    this.stop = stop;
}
}

```

```

package clickengine;

/**
 * Utilities for manipulating the arrays
 * used for the audio buffers
 * @author David Manchip, MNCDAV002
 */
final class BufferUtils
{

```

```

/**
 * An empty immutable byte array.
 */
static final byte[] EMPTY_BYTE_ARRAY = new byte[0];

/**
 * Recursive method that returns a loop of the source
 * buffer
 * given sufficient length.
 * @param source
 * @param startIndex
 * @param length
 * @return
 */
static byte[] getLoop(byte[] source, int startIndex, int
    length)
{
    int endIndex = length + startIndex;
    byte[] result = subarray(source, startIndex,
        endIndex);
    if ((length - result.length) > 0)
    {
        return addAll(result, getLoop(source, 0, (length
            - result.length)));
    }
    else
    {
        return result;
    }
}

/**
 * Multiply each element by the same factor
 * @param result
 * @param factor
 * @return
 */
static byte[] multiplyAll(byte[] result, float factor)
{
    for (int i = 0; i < result.length; i++)
    {
        result[i] = (byte) (result[i] * factor);
    }
}

```

```

    }

    return result;
}

/**
 * <p>Adds all the elements of the given arrays into a
 * new array.</p>
 * <p>The new array contains all of the element of <code>
 * >array1</code> followed
 * by all of the elements <code>array2</code>. When an
 * array is returned, it is always
 * a new array.</p>
 *
 * <pre>
 * ArrayUtils.addAll(array1, null)    = cloned copy of
 *   array1
 * ArrayUtils.addAll(null, array2)    = cloned copy of
 *   array2
 * ArrayUtils.addAll([], [])         = []
 * </pre>
 *
 * @param array1 the first array whose elements are
 *   added to the new array.
 * @param array2 the second array whose elements are
 *   added to the new array.
 * @return The new byte[] array.
 * @since 2.1
 */
static byte[] addAll(byte[] array1, byte[] array2)
{
    if (array1 == null)
    {
        return clone(array2);
    }
    else if (array2 == null)
    {
        return clone(array1);
    }
    byte[] joinedArray = new byte[array1.length + array2
        .length];

```

```

        System.arraycopy(array1, 0, joinedArray, 0, array1.
            length);
        System.arraycopy(array2, 0, joinedArray, array1.
            length, array2.length);
        return joinedArray;
    }

/**
 * <p>Clones an array returning a typecast result and
 * handling
 * <code>>null</code>.</p>
 *
 * <p>This method returns <code>>null</code> for a <code>
 * null</code> input array.</p>
 *
 * @param array the array to clone, may be <code>>null</
 * code>
 * @return the cloned array, <code>>null</code> if <code>
 * null</code> input
 */
static byte[] clone(byte[] array)
{
    if (array == null)
    {
        return null;
    }
    return (byte[]) array.clone();
}

/**
 * <p>Produces a new <code>byte</code> array containing
 * the elements
 * between the start and end indices.</p>
 *
 * <p>The start index is inclusive, the end index
 * exclusive.
 * Null array input produces null output.</p>
 *
 * @param array the array
 * @param startIndexInclusive the starting index.
 * Undervalue (<math><lt;0</math>)

```

```

*      is promoted to 0, overvalue (>array.length)
      results
*      in an empty array.
* @param endIndexExclusive  elements up to endIndex-1
      are present in the
*      returned subarray. Undervalue (< startIndex)
      produces
*      empty array, overvalue (>array.length) is
      demoted to
*      array length.
* @return a new array containing the elements between
*      the start and end indices.
* @since 2.1
*/
static byte[] subarray(byte[] array, int
    startIndexInclusive, int endIndexExclusive)
{
    if (array == null)
    {
        return null;
    }
    if (startIndexInclusive < 0)
    {
        startIndexInclusive = 0;
    }
    if (endIndexExclusive > array.length)
    {
        endIndexExclusive = array.length;
    }
    int newSize = endIndexExclusive -
        startIndexInclusive;
    if (newSize <= 0)
    {
        return EMPTY_BYTE_ARRAY;
    }

    byte[] subarray = new byte[newSize];
    System.arraycopy(array, startIndexInclusive,
        subarray, 0, newSize);
    return subarray;
}
}

```

```

package clickengine;

import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import java.util.Timer;
import javax.sound.sampled.*;

/**
 * Co-ordinates the various objects required
 * for creating and managing isochronous clicks
 * @author David Manchip, MNCDAV002
 */
public class ClickEngine
{
    private static final float DEFAULT_TEMPO = 80.0f;
    private static final int FADE_RATE = 600; //milliseconds
    private static final int MAX_FADE_START = 60000;

    private AudioStreamWriter audioStreamWriter;
    private Thread streamWriterThread;
    private SourceDataLine line = null;

    private List<Integer> fades = new ArrayList<Integer>();
    private Timer timer;
    private int fadeStart = 10000;

    private ArrayList<IAudioDataGenerator> clicks = new
        ArrayList<IAudioDataGenerator>();
    private Boolean isPlaying = false;

    public ClickEngine()
    {
    }

    @Override
    protected void finalize() throws Throwable
    {
        line.close();
    }
}

```

```

public int createClick(String resourceName) throws
    IOException, LineUnavailableException
{
    return this.createClick(resourceName, DEFAULT_TEMPO,
        0, 0, DEFAULT_TEMPO);
}

public int createClick(String resourceName, float tempo,
    float startOffset, int prerollBeats, float
    prerollTempo) throws IOException,
    LineUnavailableException
{
    Impulse impulse = new Impulse(resourceName);
    if (line == null) //only create the first time
    {
        line = AudioSystem.getSourceDataLine(impulse.
            getFormat());
        line.open();
    }

    IAudioDataGenerator click = new Click(impulse, tempo
        , startOffset, prerollBeats, prerollTempo);

    clicks.add(click);

    return clicks.indexOf(click);
}

public void setClickSound(int clickIndex, String
    resourceName) throws IOException,
    LineUnavailableException
{
    Click oldClick = (Click) clicks.get(clickIndex);
    Impulse impulse = new Impulse(resourceName);

    recreateClick(clickIndex,
        impulse,
        oldClick.getTempo(),
        oldClick.getStartOffset(),
        oldClick.getPrerollBeats(),
        oldClick.getPrerollTempo());
}

```

```

}

public void start()
{
    isPlaying = true;

    Mixer mixer = new Mixer();
    for (int i=0; i < clicks.size(); i++)
    {
        IAudioDataGenerator click = clicks.get(i);
        click.setVolume(1);
        mixer.add(click);
    }

    audioStreamWriter = new AudioStreamWriter(line,
        mixer);
    streamWriterThread = new Thread(audioStreamWriter);
    streamWriterThread.start();

    InitiateFades();
}

public void stop()
{
    timer.cancel();

    audioStreamWriter.setStop(true);

    for (int i=0; i < clicks.size(); i++)
    {
        clicks.get(i).reset();
    }

    isPlaying = false;
}

/**
 * @return the isPlaying
 */
public boolean isPlaying()
{
    return isPlaying;
}

```

```

}

public void setClickTempo(int clickIndex, float tempo)
{
    Click oldClick = (Click) clicks.get(clickIndex);
    recreateClick(clickIndex,
        oldClick.getImpulse(),
        tempo,
        oldClick.getStartOffset(),
        oldClick.getPrerollBeats(),
        oldClick.getPrerollTempo());
}

public void setClickOffset(int clickIndex, float offset)
{
    Click oldClick = (Click) clicks.get(clickIndex);
    recreateClick(clickIndex,
        oldClick.getImpulse(),
        oldClick.getTempo(),
        offset,
        oldClick.getPrerollBeats(),
        oldClick.getPrerollTempo());
}

public void setClickPrerollBeats(int clickIndex, int
    prerollBeats)
{
    Click oldClick = (Click) clicks.get(clickIndex);
    recreateClick(clickIndex,
        oldClick.getImpulse(),
        oldClick.getTempo(),
        oldClick.getStartOffset(),
        prerollBeats,
        oldClick.getPrerollTempo());
}

public void setClickPrerollTempo(int clickIndex, float
    tempo)
{
    Click oldClick = (Click) clicks.get(clickIndex);
    recreateClick(clickIndex,
        oldClick.getImpulse(),

```

```

        oldClick.getTempo(),
        oldClick.getStartOffset(),
        oldClick.getPrerollBeats(),
        tempo);
    }

    public void setFade(int clickIndex)
    {
        if (!fades.contains(clickIndex))
        {
            fades.add(clickIndex);
        }
    }

    /**
     * @return the fadeStart
     */
    public int getFadeStart()
    {
        return fadeStart;
    }

    /**
     * @param fadeStart the fadeStart to set
     */
    public void setFadeStart(int fadeStart)
    {
        if (fadeStart > MAX_FADE_START)
        {
            this.fadeStart = MAX_FADE_START;
        }
        else
        {
            this.fadeStart = fadeStart;
        }
    }

    private void InitiateFades()
    {
        timer = new Timer();
        for (int i = 0; i < fades.size(); i++)
        {

```

```

        FadeOut fade = new FadeOut(clicks.get(fades.get(
            i)));
        timer.scheduleAtFixedRate(fade, fadeStart,
            FADE_RATE);
    }
}

private void recreateClick(int clickIndex, Impulse
    impulse, float tempo, float startOffset, int
    prerollBeats, float prerollTempo)
{
    IAudioDataGenerator newClick = new Click(impulse,
        tempo, startOffset, prerollBeats, prerollTempo);
    clicks.set(clickIndex, newClick);
}
}

```

```

package clickengine;

import java.security.InvalidParameterException;

/**
 * IAudioDataGenerator implementation that creates an
 * isochronous click according to the supplied constructor
 * parameters
 * @author David Manchip, MNCDAV002
 */
class Click implements IAudioDataGenerator
{
    private Impulse impulse;
    private float tempo = 100;
    private byte[] buffer;
    private int position;
    private float startOffset;
    private byte[] preroll;
    private int prerollDuration;
    private int prerollPosition;
    private float prerollTempo;
    private float volume = 1.0f;
    private int prerollBeats;

    /**

```

```

* Constructor take setup params
*
* @param impulse to use at the start of the generated
  stream
* @param tempo
* @param startOffset - start position expressed as a
  fraction of tempo (0.0 - 1.0)
* @param prerollBeats - how many beats to wait before
  introducing the clicks
* @param prerollTempo - the tempo of the preroll beats
*/
Click(Impulse impulse, float tempo, float startOffset,
      int prerollBeats, float prerollTempo)
{
    this.impulse = impulse;
    this.tempo = tempo;
    if (startOffset != 0)
    {
        startOffset = 1.0f -startOffset;
    }
    this.startOffset = startOffset;
    this.prerollTempo = prerollTempo;
    this.prerollBeats = prerollBeats;

    if (impulse.getImpulseDuration() > -1)
    {
        float silenceDuration = calculateSilenceDuration
            ();
        byte[] silence = new byte[(int) silenceDuration
            - (int) silenceDuration % 2];
        buffer = BufferUtils.addAll(impulse.
            getImpulseData(), silence);

        //add preroll
        if (prerollBeats > 0)
        {
            prerollDuration = calculatePrerollDuration(
                prerollBeats);
            preroll = new byte[prerollDuration];
        }

        reset();
    }
}

```

```

    }
}

@Override
public float getVolume()
{
    return this.volume;
}

@Override
public void setVolume(float level)
{
    if (level < 0 || level > 1.0)
    {
        throw new InvalidParameterException("level must
            be between 0 and 1");
    }
    this.volume = level;
}

public void reset()
{
    float frameRate = impulse.getFormat().getFrameRate()
        ;
    position = (int) ((60 / getTempo() * frameRate * 2)
        * getStartOffset());

    int frameSize = impulse.getFormat().getFrameSize();
    position = position - (position % frameSize);
    prerollPosition = 0;
}

public byte[] generate()
{
    return generate(DEFAULT_BUFFER_SIZE);
}

public byte[] generate(int length)
{
    byte[] result;

    if (prerollPosition < prerollDuration)

```

```

{
    int endIndex = prerollPosition + length;
    result = BufferUtils.subarray(preroll,
        prerollPosition, endIndex);

    prerollPosition = prerollPosition + length;

    if (result.length < length) //when finished with
        the preroll
    {
        //add missing required data from click
        buffer
        int lengthStillRequired = length - result.
            length;
        result = BufferUtils.addAll(result,
            BufferUtils.getLoop(buffer, position,
                lengthStillRequired));
        position = ((position + lengthStillRequired)
            % buffer.length);
    }
}
else
{
    result = BufferUtils.getLoop(buffer, position,
        length);

    position = ((position + length) % buffer.length)
        ;
}

if (volume != 1.0)
{
    return BufferUtils.multiplyAll(result, volume);
}
return result;
}

private int calculatePrerollDuration(int prerollBeats)
{
    float frameRate = impulse.getFormat().getFrameRate()
        ;
}

```

```

        return (int) ((60 / getPrerollTempo() * frameRate *
            2) * prerollBeats);
    }

private float calculateSilenceDuration()
{
    float frameRate = impulse.getFormat().getFrameRate()
        ;
    return (60 / getTempo() * frameRate * 2) - impulse.
        getImpulseDuration();
}

/**
 * @return the impulse
 */
Impulse getImpulse()
{
    return impulse;
}

/**
 * @return the tempo
 */
float getTempo()
{
    return tempo;
}

/**
 * @return the startOffset
 */
float getStartOffset()
{
    return startOffset;
}

/**
 * @return the prerollTempo
 */
float getPrerollTempo()
{
    return prerollTempo;
}

```

```

    }

    /**
     * @return the prerollBeats
     */
    int getPrerollBeats()
    {
        return prerollBeats;
    }
}



---


package clickengine;

import java.util.TimerTask;

/**
 * TimerTask implementation that is responsible
 * for fading any audio data generator
 * @author David Manchip, MNCDAV002
 */
class FadeOut extends TimerTask
{
    private IAudioDataGenerator click;

    FadeOut(IAudioDataGenerator click)
    {
        this.click = click;
    }

    @Override
    public void run()
    {
        float targetLevel = click.getVolume() - 0.05f;
        if (targetLevel < 0)
        {
            click.setVolume(0);
            this.cancel();
            //System.out.println("Done...");
        }
        else
        {
            //System.out.println("Fading...");
        }
    }
}

```



```

*/
class Impulse
{
    private AudioFormat format;
    private int impulseDuration = 0;
    private static final int EXTERNAL_BUFFER_SIZE = 512 *
        1024;
    private byte[] impulseData;

    Impulse(String resourceName) throws IOException
    {
        AudioInputStream inputStream = null;
        try
        {
            URL url = this.getClass().getClassLoader().
                getResource(resourceName);
            inputStream = AudioSystem.getAudioInputStream(
                url);
        }
        catch (UnsupportedAudioFileException e)
        {
            e.printStackTrace();
            return;
        }
        catch (IOException e)
        {
            e.printStackTrace();
            return;
        }

        format = inputStream.getFormat();

        byte[] tmpData = new byte[EXTERNAL_BUFFER_SIZE];
        try
        {
            impulseDuration = inputStream.read(tmpData, 0,
                EXTERNAL_BUFFER_SIZE);
            impulseData = BufferUtils.subarray(tmpData, 0,
                impulseDuration);
        }
        catch (IOException e)
        {

```

```

        e.printStackTrace();
        return;
    }
    finally
    {
        inputStream.close();
    }
}

/**
 * @return the format
 */
AudioFormat getFormat()
{
    return format;
}

/**
 * @return the clickDuration
 */
int getImpulseDuration()
{
    return impulseDuration;
}

/**
 * @return the impulseData
 */
byte[] getImpulseData()
{
    return impulseData;
}
}
}



---




---


package clickengine;

import java.security.InvalidParameterException;
import java.util.ArrayList;

/**
 * Audio data generator used to mix other
 * audio data generators together

```

```

* @author David Manchip, MNCDAV002
*/
class Mixer implements IAudioDataGenerator
{
    private ArrayList<IAudioDataGenerator> generators = new
        ArrayList<IAudioDataGenerator>();

    void add(IAudioDataGenerator generator)
    {
        generators.add(generator);
    }

    @Override
    public byte[] generate()
    {
        return generate(DEFAULT_BUFFER_SIZE);
    }

    @Override
    public byte[] generate(int length)
    {
        byte[] result = new byte[length];
        for (int i = 0; i < generators.size(); i++)
        {
            result = mix(result, generators.get(i).generate(
                length));
        }

        return result;
    }

    /**
     * The crudest of mixing algorithms. Should really be
     * converting to float and then compressing overshoots.
     * Too hard for me...
     * @param byteStream1
     * @param byteStream2
     * @return
     */
    private byte[] mix(byte[] byteStream1, byte[]
        byteStream2)
    {

```

```

        if (byteStream1.length != byteStream2.length)
        {
            throw new InvalidParameterException("Streams
                must be the same length");
        }

        byte[] result = new byte[byteStream1.length];
        for (int i = 0; i < byteStream1.length; i++)
        {
            int mixed = byteStream1[i] + byteStream2[i];
            result[i] = (byte) (mixed * 0.5);
        }

        return result;
    }

    @Override
    public void reset()
    {
        throw new UnsupportedOperationException("Not
            supported yet.");
    }

    @Override
    public void setVolume(float level)
    {
        throw new UnsupportedOperationException("Not
            supported yet.");
    }

    @Override
    public float getVolume()
    {
        throw new UnsupportedOperationException("Not
            supported yet.");
    }
}

```

```

package metronome;

import clickengine.ClickEngine;
import java.io.IOException;

```

```

import javax.sound.sampled.LineUnavailableException;
import org.apache.log4j.Logger;

/**
 * Base class for metronome implementations.
 * Provides some common behaviour.
 * @author David Manchip, MNCDAV002
 */
public abstract class AbstractMetronome
{
    protected static Logger log = Logger.getLogger(
        TimeCheckMetronome.class.getPackage().getName());
    protected ClickEngine engine;
    protected float tempo = 80;

    public AbstractMetronome()
    {
        engine = new ClickEngine();
    }

    public String[] getAvailableClickNames()
    {
        return new String[] {
            "Click1", "Click2", "Click3", "Cowbell1", "
            Cowbell2", "Cowbell3", "D click", "MP click",
            "MO click", "Short cowbell", "TimeCheck
            click", "U click", "Woodblock1"
        };
    }

    /**
     *
     * @return the current tempo of the metronome
     */
    public float getTempo()
    {
        return tempo;
    }

    /**
     * Set the tempo of the metronome
     * @param tempo

```

```

    */
    public void setTempo(float tempo)
    {
        if (!engine.isPlaying())
        {
            this.tempo = tempo;
            resetClicks();
        }
    }

    /**
     *
     * @return if the metronome is playing, otherwise false
     */
    public boolean isPlaying()
    {
        return engine.isPlaying();
    }

    public void setClickSound(int clickIndex, String
        resourceName) throws IOException,
        LineUnavailableException
    {
        engine.setClickSound(clickIndex, "timecheck/
            resources/" + resourceName + ".wav");
    }

    public void start()
    {
        if (!engine.isPlaying())
        {
            engine.start();
        }
    }

    /**
     * Stop the metronome
     */
    public void stop()
    {
        engine.stop();
        log.info("STOP");
    }

```

```

    }

    protected void configureClick(int clickIndex, float
        tempo, float startOffset, int prerollBeats, float
        prerollTempo)
    {
        if (!isPlaying())
        {
            engine.setClickTempo(clickIndex, tempo);
            engine.setClickOffset(clickIndex, startOffset);
            engine.setClickPrerollBeats(clickIndex,
                prerollBeats);
            engine.setClickPrerollTempo(clickIndex,
                prerollTempo);
        }
    }

    protected abstract void resetClicks();
}

```

```
package metronome;
```

```

/**
 * Contract for click patterns. Used
 * by the TimeCheckMetronome to configure
 * the different clicks within the click engine.
 * @author David Manchip, MNCDAV002
 */
interface IPattern
{
    /**
     * returns the tempo of the pattern
     * @param sourceTempo - the tempo used to calculate the
     * target tempo
     * @return the tempo
     */
    float getTargetTempo(float sourceTempo);

    /**
     * overload of getTargetTempo that
     * takes an integer used as a multiplier
     * on the target tempo. Used for patterns

```

```

    * based on subdivisions of the beat
    * @param sourceTempo
    * @param subdivision
    * @return
    */
float getTargetTempo(float sourceTempo, int subdivision)
    ;

/**
 * The start offset of the pattern
 * expressed as a fraction of 1 beat
 * of the target tempo
 * @return
 */
float getStartOffset();

/**
 * The start offset expressed as
 * number of source tempo beats
 */
int getStartOffsetInBeats();

/**
 * used to identify the pattern
 * @return the pattern number
 */
int getPatternNumber();

/**
 *
 * @return the ratio of this pattern
 * to the sourcetempo.
 */
int getPatternRatio();
}

```

```

package metronome;

```

```

/**
 * Represents a difficulty level/pattern and
 * its associated pattern and subdivision
 *

```

```

* @author David Manchip, MNCDAV002
*/
class Level
{
    IPattern pattern;
    int subdivision;

    Level(IPattern pattern, int subdivision)
    {
        this.pattern = pattern;
        this.subdivision = subdivision;
    }

    /**
     * Convenience for getting the target tempo using the
     * current
     * level without seperately querying the pattern and
     * subdivision.
     * @param sourceTempo
     * @return
     */
    float getTargetTempo(float sourceTempo)
    {
        return pattern.getTargetTempo(sourceTempo,
            subdivision);
    }

    /**
     * Convenience for getting the StartOffset of the
     * the pattern.
     * @return
     */
    float getStartOffset()
    {
        return pattern.getStartOffset();
    }

    /**
     * Equal implementation: pattern.ratio / subdivision are
     * equal
     *
     * && pattern.offset / subdivision
     * are equal

```

```

    * @param otherLevel
    * @return
    */
@Override
public boolean equals(Object otherLevel)
{
    if (!(otherLevel instanceof Level))
    {
        return false;
    }

    IPattern otherPattern = (IPattern) ((Level)
        otherLevel).pattern;

    int thisPatternRatio = this.pattern.getPatternRatio
        ();
    int otherPatternRatio = otherPattern.getPatternRatio
        ();

    int thisStartOffset = this.pattern.
        getStartOffsetInBeats ();
    int otherStartOffset = otherPattern.
        getStartOffsetInBeats ();

    int thisSubdivision = this.subdivision;
    int otherSubdivision = ((Level) otherLevel).
        subdivision;

    boolean result = ((float)thisPatternRatio /
        thisSubdivision) == ((float)otherPatternRatio /
        otherSubdivision);
    result = result && ((float)thisStartOffset /
        thisSubdivision) == ((float)otherStartOffset /
        otherSubdivision);

    return result;
}

@Override
public int hashCode()
{

```

```

        int hash = 7;
        hash = 11 * hash + (this.pattern != null ? this.
            pattern.hashCode() : 0);
        hash = 11 * hash + this.subdivision;
        return hash;
    }
}

```

```

package metronome;

```

```

import java.util.ArrayList;
import java.util.List;

```

```

/**
 * This class indexes patterns and subdivisions
 * into a simple 'difficulty' hierarchy. It hides
 * the choice of subdivision from the user, hopefully
 * making the UI a little less confusing.
 * The idea is to add patterns (generated by the pattern
 * factory) up to a certain threshold. Then we start
 * from the beginning again, but this time using a different
 * subdivision, again up to a certain threshold. Rinse,
 * repeat.
 * The threshold is determined by a combination of
 * maxTriangularThreshold and the current subdivision.
 *
 * Unfortunately the maths behind indentifying duplicates
 * across subdivisions escapes me, so instead of simple
 * maintaining references to pattern numbers, we will have
 * to maintain references to the patterns themselves for
 * direct comparison purposes.
 *
 * @author David Manchip, MNCDAV002
 */

```

```

class LevelManager

```

```

{
    private List<Level> levels;

```

```

    /**
     * Also sets up some standard subdivisions as defaults
     * @param patterns
     */

```

```

LevelManager(int numberOfPatterns)
{
    this(numberOfPatterns, 4, new int[] { 1, 2, 3, 4, 6,
        8 });
}

LevelManager(int numberOfPatterns, int
    maxTriangularThreshold, int[] subdivisions)
{
    this(numberOfPatterns, maxTriangularThreshold,
        subdivisions, false);
}

LevelManager(int numberOfPatterns, int
    maxTriangularThreshold, int[] subdivisions, boolean
    includedDuplicates)
{
    List<IPattern> patterns = PatternFactory.
        createMultiple(numberOfPatterns);

    levels = createLevels(patterns, subdivisions,
        maxTriangularThreshold, includedDuplicates);
}

Level getLevel(int levelNumber)
{
    return levels.get(levelNumber - 1);
}

int getLevelCount()
{
    return levels.size();
}

/**
 * use triangularThreshold * subdivision
 * each time before incrementing the subdivision.
 * We also have to exclude effective duplicates.
 * @return
 */
private static List<Level> createLevels(List<IPattern>
    patterns, int[] subdivisions, int

```

```

maxTriangularThreshold, boolean includeDuplicates)
{
    List<Level> list = new ArrayList<Level>();

    for (int i=0; i < subdivisions.length; i++)
    {
        int maxPatternNumber = TriangularUtils.
            calculateTriangular(maxTriangularThreshold *
                subdivisions[i]);

        for (int patternNumber=0; patternNumber <
            maxPatternNumber; patternNumber++)
        {
            Level newLevel = new Level(patterns.get(
                patternNumber), subdivisions[i]);
            if (includeDuplicates)
            {
                list.add(new Level(patterns.get(
                    patternNumber), subdivisions[i]));
            }
            else if (!isDuplicate(list, newLevel))
            {
                list.add(new Level(patterns.get(
                    patternNumber), subdivisions[i]));
            }
        }
    }

    //Now we want to remove the level represented by the
    //first pattern
    //in each subdivision group. We do this afterwards
    //because of duplicate checking
    if (!includeDuplicates)
    {
        for (int i=0; i < subdivisions.length; i++)
        {
            Level newLevel = new Level(patterns.get(0),
                subdivisions[i]);
            list.remove(newLevel);
        }
    }
}

```

```

        return list;
    }

    private static boolean isDuplicate(List<Level> list,
        Level level)
    {
        return list.contains(level);
    }
}

```

```

package metronome;

import java.util.ArrayList;
import java.util.List;

/**
 * Generates patterns based on
 * a difficulty level using the core difficulty algorithm.
 * This is the heart of the pattern/difficulty engine.
 * @author David Manchip, MNCDAV002
 */
final class PatternFactory
{
    static IPattern create(int patternNumber)
    {
        int ratio = TriangularUtils.
            calculateNextTriangularRoot(patternNumber);

        int triangular = TriangularUtils.calculateTriangular
            (ratio - 1);
        triangular++;

        int numberOfBeatsToOffset = patternNumber -
            triangular;

        return new TimeCheckPattern(patternNumber, ratio,
            numberOfBeatsToOffset);
    }

    static List<IPattern> createMultiple(int
        numberOfPatterns)
    {

```

```

        List<IPattern> result = new ArrayList<IPattern>();

        for (int i = 1; i <= numberOfPatterns; i++)
        {
            result.add(create(i));
        }

        return result;
    }
}

```

```

package metronome;

/**
 * IPattern implementation that uses subdivisions
 * and ratios to represent the patterns
 * @author David Manchip, MNCDAV002
 */
class TimeCheckPattern implements IPattern
{
    private int patternNumber;
    private int ratio;
    private int numberOfBeatsToOffset;

    TimeCheckPattern(int patternNumber, int ratio, int
        numberOfBeatsToOffset)
    {
        this.patternNumber = patternNumber;
        this.ratio = ratio;
        this.numberOfBeatsToOffset = numberOfBeatsToOffset;
    }

    public float getTargetTempo(float sourceTempo)
    {
        return getTargetTempo(sourceTempo, 1);
    }

    public float getTargetTempo(float sourceTempo, int
        subdivision)
    {
        return (sourceTempo * subdivision) / ratio;
    }
}

```

```

public float getStartOffset ()
{
    return (float) numberOfBeatsToOffset / (float) ratio
    ;
}

public int getStartOffsetInBeats ()
{
    return numberOfBeatsToOffset ;
}

public int getPatternNumber ()
{
    return this.patternNumber ;
}

public int getPatternRatio ()
{
    return ratio ;
}
}

```

```

package metronome ;

import java.io.IOException ;
import javax.sound.sampled.LineUnavailableException ;

/**
 * AbstractMetronome implementation that uses
 * all the supporting classes to create an advanced
 * timing training system
 * @author David Manchip, MNCDAV002
 */
public class TimeCheckMetronome extends AbstractMetronome
{
    private int standardClickIndex ;
    private int patternClickIndex ;

    /**
     * How far apart can clicks get? In primary click beats
     */
}

```

```

private static int MAX_GAP_THRESHOLD = 4;
private LevelManager levelManager;
private int currentLevel;

private int countIn;
/**
 * This is where the subdivisions are decided. Choose
 * carefully...
 */
private int[] subdivisions = new int[] { 1, 2, 3, 4, 5,
    6, 7 };

/**
 * Instantiates a new metronome with default settings
 * @throws java.io.IOException
 * @throws javax.sound.sampled.LineUnavailableException
 */
public TimeCheckMetronome() throws IOException,
    LineUnavailableException
{
    int numberOfPatterns = EstimateNumberOfPatterns();
    levelManager = new LevelManager(numberOfPatterns,
        MAX_GAP_THRESHOLD, subdivisions);
    currentLevel = 1;

    standardClickIndex = engine.createClick("timecheck/
        resources/Click1.wav");
    patternClickIndex = engine.createClick("timecheck/
        resources/TimeCheck click.wav");

    engine.setFade(standardClickIndex);

    resetClicks();
}

@Override
public void start()
{
    if (!engine.isPlaying())
    {
        super.start();
    }
}

```

```

        log.info(String.format("START  Tempo: %1$06.2f
                                Level: %4$03d  Pattern: %2$02d  Subdivision:
                                %3$02d", tempo, getPatternNumber(),
                                getSubdivision(), currentLevel));
    }
}

protected void resetClicks()
{
    configureClick(standardClickIndex, tempo, 0, 0, 0);
    configureClick(patternClickIndex,
                    levelManager.getLevel(currentLevel).
                        getTargetTempo(tempo),
                    levelManager.getLevel(currentLevel).
                        getStartOffset(),
                    countIn,
                    tempo);
}

@Override
public void setClickSound(int clickIndex, String
                           resourceName) throws IOException,
                           LineUnavailableException
{
    super.setClickSound(clickIndex, resourceName);
    engine.setFade(standardClickIndex);
}

/**
 * Set the difficulty level.
 * Implicitly sets the pattern and subdivision.
 * @param levelNumber
 */
public void setLevel(int levelNumber)
{
    if (!engine.isPlaying())
    {
        currentLevel = levelNumber;
        resetClicks();
    }
}
}

```

```

/**
 *
 * @return the number of available levelManager
 */
public int getLevelCount()
{
    return levelManager.getLevelCount();
}

/**
 * The numbers of beats played by the
 * primary click before the pattern starts.
 * @return
 */
public int getCountIn()
{
    return countIn;
}

/**
 * The numbers of beats played by the
 * primary click before the pattern starts.
 * @param countIn
 */
public void setCountIn(int countIn)
{
    if (!engine.isPlaying())
    {
        this.countIn = countIn;
        resetClicks();
    }
}

/**
 *
 * @return the ratio of the current pattern
 */
public int getPatternRatio()
{
    return levelManager.getLevel(currentLevel).pattern.
        getPatternRatio();
}

```

```

public int getStartOffsetBeats ()
{
    return levelManager.getLevel(currentLevel).pattern.
        getStartOffsetInBeats ();
}

/**
 * @return the currentDifficultyLevel
 */
public int getPatternNumber ()
{
    return levelManager.getLevel(currentLevel).pattern.
        getPatternNumber ();
}

/**
 *
 * @return the whole number used to generate the
 * difficulty level patterns. Represents a typical
 * subdivision when multiplied by 4. eg 2 would mean
 * 8th notes, 3 triplets, 4 would be 16ths, etc.
 */
public int getSubdivision ()
{
    return levelManager.getLevel(currentLevel).
        subdivision;
}

/**
 *
 * @param fadeStart how many seconds until the
 * fade-out should start
 */
public void setFadeStart(int fadeStart)
{
    if (!engine.isPlaying())
    {
        engine.setFadeStart(fadeStart * 1000);
    }
}

```

```

/**
 * Estimates the number of patterns required by the
 * LevelManager
 * @return
 */
private int EstimateNumberOfPatterns()
{
    int maxSubdivision = 0;
    for (int i=0; i < subdivisions.length; i++)
    {
        maxSubdivision = subdivisions[i] >
            maxSubdivision ? subdivisions[i] :
            maxSubdivision;
    }

    return TriangularUtils.calculateTriangular(
        MAX_GAP_THRESHOLD * maxSubdivision);
}
}

```

```
package metronome;
```

```

/**
 * Basic utilities for working with triangular
 * numbers (1+2+3+n)
 *
 * @author David Manchip, MNCDAV002
 */
final class TriangularUtils
{
    /**
     *
     * @param n is the root
     * @return the triangular of n
     */
    static int calculateTriangular(int n)
    {
        return (n * n + n) / 2;
    }

    /**
     * This returns the closest

```

```

    * perfect triangular root (whole number) that is greater
      than or equal to the actual root.
    * Uses a simplified quadratic formula - only positive
      solutions and first
    * two coefficients are 1
    * @param n is any number
    * @return the closest larger whole number triangular
      root
    */
    static int calculateNextTriangularRoot(int n)
    {
        double root = (-1 + Math.sqrt(8 * n + 1)) / 2;
        return (int) Math.ceil(root);
    }
}
}
}

package timecheck;

import metronome.TimeCheckMetronome;
import java.io.IOException;
import javax.sound.sampled.LineUnavailableException;
import org.apache.log4j.Logger;
import usage.ExpiryTask;
import usage.TimeLimiter;

/**
 * Entry point for the advanced timing
 * training system. Ties the metronome and
 * the UI together
 * @author David Manchip, MNCDAV002
 */
public class Main
{
    private static Logger log = Logger.getLogger(Main.class.
        getPackage().getName());
    private static TimeCheckMetronome metronome;

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) throws
        IOException, LineUnavailableException

```

```

    {
        log.info("TimeCheck Startup");
        Thread hook = new Thread(new ShutdownHook());
        Runtime.getRuntime().addShutdownHook(hook);

        metronome = new TimeCheckMetronome();

        TimeCheckUI gui = new TimeCheckUI(metronome);
        gui.setVisible(true);
//        Uncomment to enable time limited usage (20 min)
//        ExpiryTask expiry = new ExpiryTask(gui, metronome)
;
//        TimeLimiter usageLimiter = new TimeLimiter();
//        usageLimiter.execute(expiry, "TimeCheck Startup",
"TimeCheck Shutdown");
    }

    private static class ShutdownHook implements Runnable
    {
        public void run()
        {
            if (metronome != null && metronome.isPlaying())
            {
                metronome.stop(); //purely for logging
                porpoises
            }
            log.info("TimeCheck Shutdown");
        }
    }
}

```

A.2 Log File Parser

The log file parser, also implemented in Java, modelled the target database entities using classes. The user is prompted for a log file using a standard File Chooser dialog. The file is parsed and the generated SQL insert statements are written out to a file. These were then executed against a MySQL database to capture the data.

```

package logfileprocessor;

import java.io.*;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import javax.swing.JFileChooser;
import org.apache.log4j.Logger;

/**
 * Parses user log files and generates sql inserts for the
 * usage database
 * @author David Manchip, MNCDAV002
 */
public class Main
{

    private static final int TIMESTAMP_LENGTH = 23;

    private static SimpleDateFormat timestampFormat = new
        SimpleDateFormat("yyyy-MM-dd HH:mm:ss");

    private static Logger log = Logger.getLogger(Main.class.
        getPackage().getName());

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) throws
        IOException, ParseException
    {
        log.debug(null);

        JFileChooser fileChooser = new JFileChooser();
        fileChooser.setFileFilter(new
            FileNameExtensionFilter("log files", "log"));
        if (fileChooser.showDialog(null, "Choose") ==
            JFileChooser.APPROVE_OPTION)
        {
            File logFile = fileChooser.getSelectedFile();
            String filePath = logFile.getAbsolutePath();

```

```

        User user = buildObjectGraph(filePath, "Startup"
            , "Shutdown");

        //exportSessions(user);
        createSql(user, logFile);
    }
}

static User buildObjectGraph(String filePath, String
    startToken, String endToken) throws IOException,
    ParseException
{
    User user = new User();
    Session session;

    try
    {
        LineNumberReader logFile = new LineNumberReader(
            new FileReader(filePath));
        try
        {
            while((session = getNextSession(logFile,
                startToken, endToken)) != null)
            {
                if (!session.isEmpty())
                {
                    user.sessions.add(session);
                }
            }
        }
        catch (ParseException ex)
        {
            log.error("Parse error at line: " + logFile.
                getLineNumber());
        }
        finally
        {
            logFile.close();
        }
    }
    catch (FileNotFoundException ex)
    {

```

```

        log.error(ex);
    }

    return user;
}

private static void exportSessions(User user)
{
    for (Session aSession : user.sessions)
    {
        System.out.println("Session start: " + aSession.
            startDateTime);
        for (Run run : aSession.runs)
        {
            System.out.println("\tRun start: " + run.
                start + " Tempo: " + run.tempo + " Run
                stop: " + run.stop);
        }
        System.out.println("Session end: " + aSession.
            endDateTime);
    }
}

private static void createSql(User user, File logfile)
    throws FileNotFoundException
{
    File outputFile = new File(logfile.getAbsolutePath()
        + logfile.getName() + ".sql");
    OutputStream output = new FileOutputStream(
        outputFile);
    PrintStream printOut = new PrintStream(output);

    System.setOut(printOut);

    System.out.println("\n\n
        -----\n");

    System.out.println("start transaction;\n");

    System.out.println("-- logfile: " + logfile.getName
        () + "\n");
}

```

```

System.out.println("-- insert correct user id here:"
);
System.out.println("set @user_id = ;\n");

for (Session aSession : user.sessions)
{
    System.out.println("
        -----");
    System.out.println("insert into Session (UserId,
        SessionTypeId, StartDateTime, EndDateTime)"
);
    ;
    System.out.println("values (@user_id, "
        + getSessionTypeId(
            aSession) + ", '
        "
        + timestampFormat.
            format(aSession.
                startDateTime) +
        "', '
        + timestampFormat.
            format(aSession.
                endDateTime) + "
        ');");
    System.out.println("set @sessionId :=
        last_insert_id();");
    for (Run run : aSession.runs)
    {
        System.out.println("insert into Run (
            SessionId, StartDateTime, EndDateTime,
            Tempo, Level, Pattern, Subdivision)"
);
        System.out.println("values (@sessionId, '
            +
            timestampFormat
                .format(run.
                    start) + "',
            '
            +
            timestampFormat
                .format(run.
                    stop) + "', "
            + run.tempo + ",
            "

```

```

        + run.level + ",
          "
        + run.pattern +
          ", "
        + run.
          subdivision +
          ");");
    }
}
System.out.println("-----"
);

System.out.println("\ncommit;");

System.out.println("\n\n
-----\n\n");
}

private static Session getNextSession(LineNumberReader
    logFile, String sessionStartToken, String
    sessionEndToken) throws IOException, ParseException
{
    String sessionStartTimestamp = null;
    String sessionEndTimestamp = null;
    Session session = null;
    Run run = null;

    String line = logFile.readLine();
    if (line != null)
    {
        if (!line.contains(sessionStartToken))
        {
            throw new ParseException("sessionStartToken
                expected: " + sessionStartToken, 0);
        }
        session = new Session();

        session.sessionType = line.contains("Basic") ?
            SessionType.BASIC_METRONOME : SessionType.
            TIMECHECK_ENHANCED_METRONOME;
        sessionStartTimestamp = line.substring(0,
            TIMESTAMP_LENGTH);
    }
}

```

```

session.startDateTime = timestampFormat.parse(
    sessionStartTimestamp);
session.startLineNumber = logFile.getLineNumber
    ();

while (true)
{
    line = logFile.readLine();
    if (line == null)
    {
        throw new ParseException("log ended
            without sessionEndToken: " +
                sessionEndToken, 0);
    }

    if (line.contains(sessionEndToken))
    {
        sessionEndTimestamp = line.substring(0,
            TIMESTAMP_LENGTH);
        session.endDateTime = timestampFormat.
            parse(sessionEndTimestamp);
        session.endLineNumber = logFile.
            getLineNumber();
        break;
    }

    if (session.sessionType == SessionType.
        TIMECHECK_ENHANCED_METRONOME)
    {
        run = getNextTimeCheckRun(line, logFile)
            ;
    }
    else if (session.sessionType == SessionType.
        BASIC_METRONOME)
    {
        run = getNextBasicRun(line, logFile);
    }

    session.runs.add(run);
}
}

```

```

        return session;
    }

    private static Run getNextTimeCheckRun(String line,
        LineNumberReader logFile) throws ParseException,
        IOException
    {
        String runStartTimestamp = null;
        String runStopTimestamp = null;
        Run run = new Run();

        if (!line.contains("START")) {
            throw new ParseException("START or Shutdown
                expected", 0);
        }
        runStartTimestamp = line.substring(0,
            TIMESTAMP_LENGTH);
        run.start = timestampFormat.parse(runStartTimestamp)
            ;
        //add other stuff here
        //System.out.println(line);

        String slice = line.substring(66, 72);
        run.tempo = Float.parseFloat(slice);

        slice = line.substring(81, 84);
        run.level = Integer.parseInt(slice);

        slice = line.substring(95, 97);
        run.pattern = Integer.parseInt(slice);

        slice = line.substring(112, 114);
        run.subdivision = Integer.parseInt(slice);

        line = logFile.readLine();
        if (line == null || !line.contains("STOP")) {
            throw new ParseException("log ended or STOP
                expected", 0);
        }
        runStopTimestamp = line.substring(0,
            TIMESTAMP_LENGTH);
        run.stop = timestampFormat.parse(runStopTimestamp);
    }

```

```

        return run;
    }

    private static Run getNextBasicRun(String line,
        LineNumberReader logFile) throws ParseException,
        IOException
    {
        String runStartTimestamp = null;
        String runStopTimestamp = null;
        Run run = new Run();

        if (!line.contains("START")) {
            throw new ParseException("START or Shutdown
                expected", 0);
        }
        runStartTimestamp = line.substring(0,
            TIMESTAMP_LENGTH);
        run.start = timestampFormat.parse(runStartTimestamp)
            ;

        String slice = line.substring(66, 72);
        run.tempo = Float.parseFloat(slice);

        line = logFile.readLine();
        if (line == null || !line.contains("STOP")) {
            throw new ParseException("log ended or STOP
                expected", 0);
        }
        runStopTimestamp = line.substring(0,
            TIMESTAMP_LENGTH);
        run.stop = timestampFormat.parse(runStopTimestamp);

        return run;
    }

    private static String getSessionTypeId(Session aSession)
    {
        return aSession.sessionType == SessionType.
            BASIC_METRONOME ? "1" : "2";
    }

```

```

    }
}

```

```

package logfileprocessor;

import java.util.ArrayList;
import java.util.List;

/**
 * This is the metronome user, essentially a dto
 * @author David Manchip, MNCDAV002
 */
class User
{
    String name;
    String surname;
    Boolean isKitDrummer;
    String email;
    String cell;
    String note;
    List<Session> sessions = new ArrayList<Session>();
}

```

```

package logfileprocessor;

import java.util.ArrayList;
import java.util.Date;
import java.util.List;

/**
 * Represents a user session with a metronome
 * @author David Manchip, MNCDAV002
 */
class Session
{
    SessionType sessionType;
    Date startDateTime;
    Date endDateTime;
    int startLineNumber;
    int endLineNumber;
    List<Run> runs = new ArrayList<Run>();
}

```

```

        boolean isEmpty()
        {
            return runs.size() == 0;
        }
    }

enum SessionType
{
    BASIC_METRONOME,
    TIMECHECK_ENHANCED_METRONOME
}

```

```

package logfileprocessor;

import java.util.Date;

/**
 * Represents a run of a metronome (within a session)
 * @author David Manchip, MNCDAV002
 */
public class Run
{
    Date start;
    Date stop;
    Float tempo;
    int level;
    int pattern;
    int subdivision;
}

```

A.3 SQL Queries

The following SQL query shows the number of sessions for each user, and the total time spent using both timing training systems:

```

select User.Name
       , coalesce(SessionType.Description, 'None') as '
           Training System'
       , count(Session.SessionId) as 'Session Count'

```

```

        , sec_to_time(sum(time_to_sec(ifnull(session_time .
            SessionDuration, 0)))) as 'Total Time Spent'
from User
left join Session
    on User.UserId = Session.UserId
left join SessionType
    on Session.SessionTypeId = SessionType.SessionTypeId
left join
    (select Run.SessionId
        , sec_to_time(sum(time_to_sec(timediff
            (Run.EndDateTime, Run.
            StartDateTime)))) as '
            SessionDuration'
        from Run
        group by Run.SessionId) as session_time
    on Session.SessionId = session_time.SessionId
group by User.Name, SessionType.Description
order by sum(session_time.SessionDuration) desc;

```

The SQL query used to retrieve a denormalised version of the experiment data suitable for import into a spreadsheet or statistical analysis application. A database view was created for convenience and reuse:

```

drop view 'TimeCheck.Usage'. 'FlatAssessments';
create view 'TimeCheck.Usage'. 'FlatAssessments' as
select ass_data.AssessmentDataId as ID
    , ref_perf.Measurement as ReferenceMeasurement
    , ass_data.Measurement
    , (ass_data.Measurement - ref_perf.Measurement) / (
        ref_perf.WindowSize / 2) as ErrorRate
    , (ass_data.Measurement - ref_perf.Measurement) * 1000
        as MillisecondError
    , round (ref_perf.Tempo) as Tempo -- to correct for some
        small errors, will be used for grouping
    , ass.AssessmentNumber
    , user.Name
from User user
join Assessment ass
    on user.UserId = ass.User_UserId
join AssessmentData ass_data
    on ass.AssessmentId = ass_data.AssessmentId

```

```
join ReferencePerformance ref_perf
  on ass_data.Measurement > ref_perf.WindowMin
  and ass_data.Measurement < ref_perf.WindowMax
where user.UserId <> 14 -- exclude ref performance
```

Using the previously created database view, the following retrieves a denormalised version of the data, but with the first 10 recorded measurements removed for each tempo:

```
select assessments.*
from FlatAssessments assessments
  inner join (select ass1.ID
              from FlatAssessments as ass1
              join FlatAssessments as ass2
                on ass1.Tempo = ass2.Tempo and ass1.ID
                ID >= ass2.ID
              group by ass1.Tempo, ass1.ID
              having count(*) > 10
              order by ass1.Tempo desc) as filtered
  on assessments.ID = filtered.ID
```

Appendix B

Class Diagram

Figure B.1 is a simplified class diagram of the Alternate Timing Training System software implementation used for this research. Please note that in order to keep the diagram simple, only elements essential to the static structure of the application are shown. Some non-essential elements such as operations, attributes and inter-class relationships have been omitted. User interface and entry point classes are also not shown.

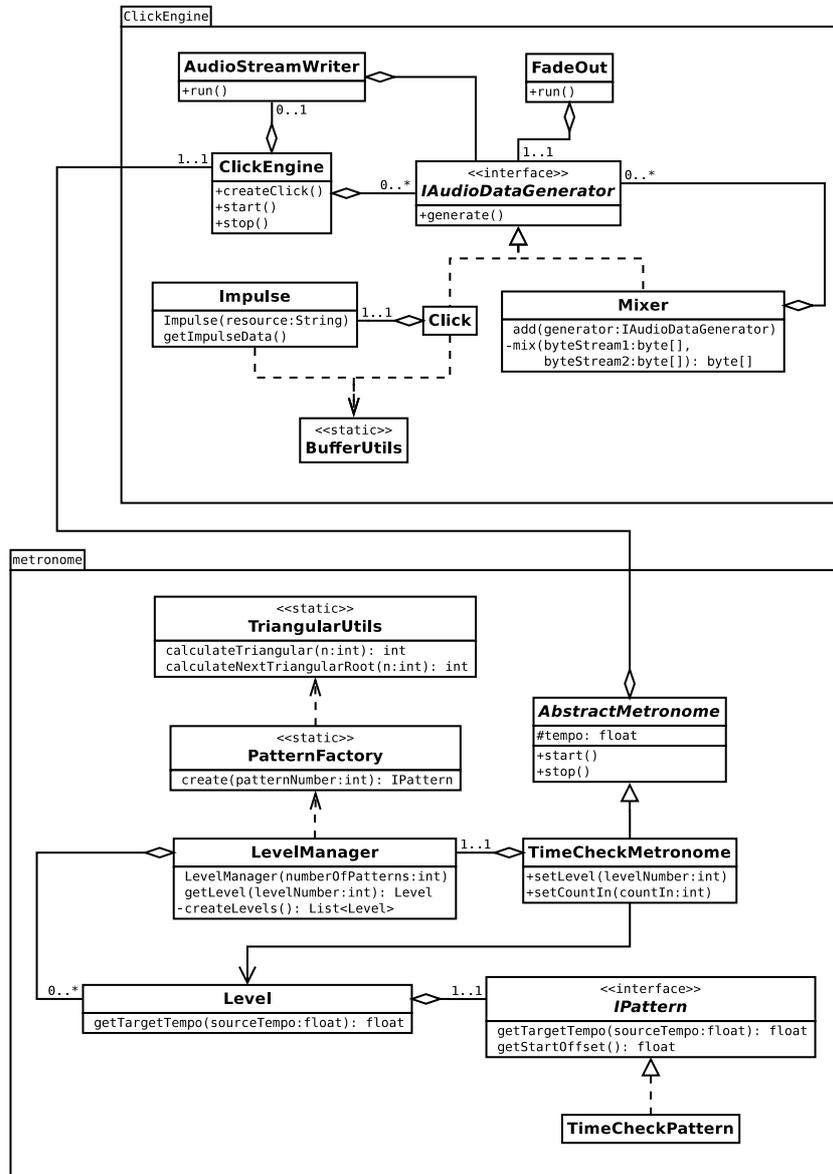


Figure B.1: Class Diagram

Appendix C

Survey Questions

TimeCheck Survey

Computer-assisted Timing Training for Musicians (website). Thank you for taking the time to complete this survey.

This survey presumes that the advanced metronome software has been downloaded and used for a reasonable period of time (e.g. at least 30 minutes). If this is not the case, please return to the download page and try out the software for a time before completing this survey.

Please answer as many questions as possible as honestly as possible.

There are 16 questions in this survey

Ability

Questions about your musical and rhythmic ability

1 How would you rate your musical ability? *

Please choose **only one** of the following:

- You don't play at all
- Beginner - you've recently started playing an instrument
- Intermediate - you've been playing for a few years
- Advanced - you've been playing for many years
- Professional - you earn a living playing and/or teaching music

2 Have you used a standard metronome before (software or hardware)? *

Please choose **only one** of the following:

- Never
- Rarely
- Sometimes
- Often
- Always

3 Please rate the importance of a good sense of time within the following musical settings (5 is most important, 1 is not important):

Please choose the appropriate response for each item:

	1	2	3	4	5
Solo performance	<input type="radio"/>				
Ensemble performance, i.e. playing in a group	<input type="radio"/>				
Recording studio	<input type="radio"/>				

4 How would you rate 'sense of time' in comparison to other musical skills such as theory knowledge, technical skill and 'a good ear'? *

Please choose **only one** of the following:

- More important
- Less important
- Equally important

5 What instrument do you play? *

Please choose **only one** of the following:

- Guitar
- Bass
- Drums/Percussion
- Wind/Brass (saxophone, trombone, clarinet, flute, etc.)
- Strings (Violin, Cello, etc.)
- Keyboard (Piano, Organ, etc.)
- Other (anything that doesn't fit into any of the above, including vocals)

Usage

Questions regarding usage of the prototype software

6 How much did you use the advanced metronome software? *

Please choose **only one** of the following:

- Less than 10 minutes
- Between 10 and 30 minutes
- Between 30 minutes and 1 hour
- More than 1 hour

7 Did you read the mini manual?

Please choose **only one** of the following:

- Yes
- No

8 Did you play along to the advanced metronome software on your chosen instrument?

Please choose **only one** of the following:

- Yes
- No

Opinion

Questions regarding your opinion of the advanced metronome software

9 Choose an option that best describes your opinion of the advanced metronome software

Please choose **only one** of the following:

- Incomprehensible
- Interesting
- Useful
- Great
- Revolutionary

10 Would you use this software as a practice tool? *

Please choose **only one** of the following:

- Never
- Rarely
- Sometimes
- Often
- Always

11 Could the advanced metronome software be used as a teaching tool? *

Please choose **only one** of the following:

- Yes
- No
- Maybe

12 Could the same benefits offered by the advanced metronome software be achieved with a standard metronome (hardware or software)? *

Please choose **only one** of the following:

- Yes
- No
- Maybe

13 Is the advanced metronome software more musically interesting to use than a standard metronome (hardware or software)? *

Please choose **only one** of the following:

- Yes
- No
- Maybe

14 How good is the specified metronome at improving a musician's sensitivity to tempo changes (1 is poor, 5 is excellent)

Please choose the appropriate response for each item:

	1	2	3	4	5
Standard metronome	<input type="radio"/>				
Advanced metronome software	<input type="radio"/>				

15 How good is the specified metronome at improving a musician's sensitivity to subdivisions of the beat?

Please choose the appropriate response for each item:

	1	2	3	4	5
Standard metronome	<input type="radio"/>				
Advanced metronome software	<input type="radio"/>				

16 Any other comments, thoughts or criticisms on the advanced metronome software?

Please write your answer here:

Survey complete, thank you.

2010-07-21 19:49

Submit your survey.

Thank you for completing this survey.

Appendix D

Survey – Supplementary Discussion

D.1 Email and Online Forum Discussions

Recruiting respondents for the survey resulted in a few discussions between the author and the respondents, both via email and in online forums. These are included verbatim in this section, as questions and suggestions were addressed directly in the discussion.

One of the survey respondents is a professional musician who offered his comments directly via email, as well as via the survey question that invited suggestions and comments. This resulted in a lengthy email discussion, reproduced below.

On Tue, Jul 27, 2010 at 6:57 PM, David
<david@timecheck.asparagusaudio.co.za> wrote:

Howdy Daveman,

I'm canvassing for musicians to try out my advanced metronome software and fill in a short online survey for my masters. Would you mind checking it out for me?

<http://timecheck.asparagusaudio.co.za>

David

On Wed, Jul 28, 2010 at 12:22 AM, dave reynolds
<davemanreynolds@gmail.com> wrote:

completed your survey - played with the metronome - here's my comments
(not anonymous):

it doesn't actually test what you can do - it's a learning tool, right, not something you'd use on stage or in studio - but learning is complex and needs stages and modules - a guided process. this is like a plug-in gadget - offers no learning program on how to take the student from A to B, how to grow etc. hope this comes soon ...

daveman

The metronome is intended as a practice tool, just like any other metronome, so I guess it's a kind of learning tool. It's not like a guided programme of lessons, though, if that's what you mean. The intention is to provide something a little more interesting to practice with that offers benefits that are hard to get with a standard metronome, such as developing sensitivity to subdivisions.

On Wed, Jul 28, 2010 at 12:06 PM, dave reynolds
<davemanreynolds@gmail.com> wrote:

dave

i dig it, but i don't see

it as a metronome - sure it has similarities to a metronome, the way you have designed the knobs and stuff, selctr the tempo, choose the click sound etc, but essentially i see it as a software box with 50 exercises - although they are rated as "difficulty 1 - 50", for me they are just exercises and some folks might find number 13 more difficult than number 23, so "difficulty" is perhaps misleading. especially since the first few are compound time (6/8 feel) as opposed to simple time (4/4 feel) which for some people is more difficult 'cos they don't have the reference. people from zimbabwe twenty years old only listened to compound time, so it's all pap 'n vleis, but those who grew up in an exclusively 4/4 world can't even imagine tapping one straight beat with your hand and stamping another straight beat with your foot. i'm kinda suggesting that you consider packaging or presenting this as a rhythm program/series - how to improve your rhythm - etc like a drummer's book (of which there are a million out there), but not publishing it as a book, rather as a pdf with a software gadget (this advanced metronome) to go with it. just a suggestion, cos i'm not sure that your intention (ie. what the gadget is for) is super clear to people. for eg, graeme sax told me about this 6 months ago - what he told me and what i found last night are quite different.

dave

On Wed, Jul 28, 2010 at 1:31 PM, David Manchip
<david@timecheck.asparagusaudio.co.za> wrote:

I agree that the term "difficulty" for the slider was a bad choice, but I couldn't come up with anything better at the time. Perhaps "Variations", or something. This thing is part of my thesis, the research question essentially being "Can computers offer something more when it comes to timing training than a standard metronome", hence the comparison with a standard metronome. I believe that musicians tend to rely on the down beat being given to them as a bit of a crutch, i.e. it's easy to play along with a click. It doesn't really improve your 'inner clock' beyond a certain point because you are constantly given these obvious references. Take the down beat away (or have it fade out, as does with

the software), and you get the subdivisions as reference points, which forces you to 'hear' the pulse internally. You're supposed to play along with the down beat/internal pulse, playing whatever you like, and not necessarily accenting or following the reference pattern. So the intention is not to teach rhythm, although that might be a by-product. The intention is to improve one's sense of time. Just like a metronome.

On Wed, Jul 28, 2010 at 1:57 PM, dave reynolds
<davemanreynolds@gmail.com> wrote:

ok a thesis - i see ... well as to computers, why not design an online interview/survey for musicians who have and use computers for their regular practice. i for one use a computer everyday - actually it's a ten year habit now - i can hardly practice without it - i never practice to a click/metronome, though - i program a groove because (as you said) a metronome doesn't tell the whole story - for "inner timing" you need something else too. i think computers have made practice enjoyable for me - i actually look forward to being alone with my instrument because i'm not alone i have a sequencer and can program michel alibo lines, karim ziad grooves or whatever - any tempo, any style. metronomes are boring, they just go click click the whole time and you only have two parameters/choices - (1) on or off (2) fast or slow - not music, is it. i think metronomes were mainly invented to see what the tempo is - switch on, set the tempo at say 125, listen for a few seconds, internalise, then switch it off and play... that's how the classical musicians i studied with for my degree used it - and their timing's terrible (cos they're classical owens, they don't groove). they guys that do groove, and have a computer, at least many of the ones i know, they practice to a sequencer not a metronome (like me). the thing is this: a gadget has to make you wanna dance - then your timing improves - if sade makes you wanna dance or manu dibango, then practicing scales to their music will improve your sense of inner timing, cos you're inspired, you're enjoying yourself.

oh well - that's my mini thesis, hope it makes some part of you wanna dance, otherwise no hard feelings! LOL

daveman

On Wed, Jul 28, 2010 at 2:07 PM, David Manchip
<david@timecheck.asparagusaudio.co.za> wrote:

Unfortunately I know _many_ musicians who play with sequencers all the time who have terrible time. It's because your inner clock doesn't improve past a certain point if you're spoon fed all the notes, all the time. You need to take some stuff away to internalise it properly. Just my 2c.

Thanks for the discussion Daveman.

sure - we're on the same page - that's what i do with the sequencer - take stuff away all the the time. so it's 12-8, u have a four on the floor cowbell (dotted quarters), then you add a kick and snare somewhere (not like cheeseey backbeat or anything - a west african groove like "bikutsi" - the brice wassy stuff), then you add a high hat, not playing triplets but playing against the straight cowbell, then you add the guy nsangué style bassline. then you jam it til you're locked, then you jump in at the deep end and solo the bassline - see if you can jam along without getting lost. unlikely, so you add one other thing - not the cowbell cos that's the click, right, that's the crutch - but add, say just the kick ... and so on. for a whole day you're having fun, adding, subtracting, discovering what can be done, notching up the tempo, notching it down, transposing the bassline up or down - all the time improving a sense of inner timing - and having fun ... there's texture, there's groove, there's timbre (if you have good vst instruments which are two a penny these days), there's a mix too ... struggling without the cowbell on the beat, make it softer, fade it out, but not always over 10secs, sometimes over 6 mins ... my point: i need a sea of possibilities, the sequencer gives me that, the metronome doesn't, so i get bored. your metronome is fascinating because of the 50 variations, not because it's a metronome. i say it's selling point is not what it is, but you can do with it - the learning program ... the timing improvement exercises. i would consider a before-&-after-style pilot: find fifteen volunteers in east london area or wherever - get them on a program to improve their timing using the method u've invented - meet with them once

a week for three weeks (half an hour each), then give them another three weeks on their own and then evaluate. just an idea, could be explored in other directions too - got a bit of rock 'n roll in there - something to write about (if you're keen) ... theoretical possibilities are easier to write about, actual experience/experimentation runs along a different course.

gotta run, nuendo's calling ... dave

The following is an excerpt from a forum thread at <http://drumsetconnect.com>. The discussion centres around an explanation of the Alternate Timing Training System.

<http://www.drumsetconnect.com/drum-forum/play-drums-drum-talk/9703-volunteers-needed-test-timing-trainer-software.html>

doggit 07-21-2010 03:39 PM Volunteers needed to test timing trainer software

I am currently doing research into timing training for musicians. I'm looking for musicians who would be interested in downloading and testing some timing training software, and then filling in a very short online survey (16 point-and-click questions). Downloads and more information available here:

<http://timecheck.asparagusaudio.co.za>

Thanks, and please pass the link on to anyone who might be interested.

David Manchip

Der Trommler 07-29-2010 12:25 AM Re: Volunteers needed to test timing trainer software

At the risk of sounding like an jerk I fail to see why this software is of any value. What exactly was your goal here?

doggit 07-29-2010 03:35 PM Re: Volunteers needed to test timing trainer software

The goal was to create a metronome that is more challenging, more interesting and more effective than a standard metronome. Perhaps I have failed ;-)

Seriously, I believe that musicians tend to rely on the down beat being given to them too much. It's a bit of a crutch, i.e. it's easy to play along with a click. It doesn't really improve your 'inner clock' beyond a certain point because you are constantly given these obvious reference points. Take the down beat away (or have it fade out, as does with the software), and you get the subdivisions as reference points, which forces you to 'hear' the pulse internally. You're supposed to play along with the down beat/internal pulse, playing whatever you like, and not necessarily accenting or following the reference pattern.

Hope that makes some sense.

Der Trommler 07-29-2010 10:23 PM Re: Volunteers needed to test timing trainer software

I guess I am not following here, but if you are playing in 4/4 time (or any other time signature for that matter), how would you know when to start or count your time after you start without having a downbeat? Or in other words because music is simply mathematics, and mathematics is counting how do you get to 2 in 4/4 time without first counting one?

doggit 07-30-2010 02:03 AM Re: Volunteers needed to test timing trainer software

I probably shouldn't have said 'down beat' and rather just 'beat' or 'pulse'.

Let me try to explain with a couple of simple examples using a standard click:

Lets say you want to practice you favourite 4/4 groove, so you put on the metronome at 120bpm and start playing, no problem. The metronome is clicking on every beat, 1 2 3 4.

Now try setting the metronome to 60bpm, except now you're still going to play at 120bpm. So while you are playing the same groove at the same tempo, the metronome is only clicking on beats 1 & 3, i.e. every second beat. Not too hard, but already a good exercise because the gaps between clicks are longer, forcing you to feel the spaces between more accurately.

The next step is to flip the metronome around in your head so that it is only clicking on beats 2 & 4. So it's still clicking at 60bpm, you're still playing at 120bpm, but there is no more down beat. One way to do this is to starting counting from 2 (2 1 2 3 4 1 2 3 4 1...) until you can feel the down beat, even though it's not being supplied by the metronome.

The software takes these basic concepts a lot further using displacement and polyrhythms to provide all kinds of patterns to play with, not just 1 & 3 or 2 & 4.

Der Trommler 07-30-2010 09:19 AM Re: Volunteers needed to test timing trainer software

Very good explanation. Your goal is to use the metronome to teach variation of syncopation and even polyrhythmic variation. Great idea.

Have you considered adding additional function by having two metro's in one? An example would be one tone or click counting 4/4 while another

tone is clicking or counting a different time sig?

doggit 07-30-2010 09:31 AM Re: Volunteers needed to test timing trainer software

Um, that's what it is, two metronomes running together :)

Der Trommler 07-30-2010 03:08 PM Re: Volunteers needed to test timing trainer software

Hmm, I guess I need to take a closer look at it. Not sure how I missed that. :eek:

D.2 Survey Comments

The following comments were the responses to optional survey question 16 ‘*Any other comments, thoughts or criticisms on the timing training software?*’. A selection of these is addressed directly in Chapter 5, Section 5.2.3.

“A bit complex to use, but otherwise very interesting”

“Is it possible to have the tempo adjusted by click & drag, like music apps do?”

“Since I’ve used the software, there’s definitely an improvement and a much keener sense of time. And as a Pro, I thought I had very good time but now I have great time. Thanks for allowing me to be a part of this project..”

“None”

“it’s not too bad...i dont think it will take over in popularity of the normal metronome but i like it...it’s making me focus on how out of time and in time i go :)”

“its been a while since i last worked with the metronome but i remember saying at the time something about that it would help if the repetitions were in 4/4 groupings rather than arbitrary time signatures”

“I would include a function that monitors the mouse buttons, so that you can ‘tap along’ on the mouse (or keyboard buttons) along with the beat. That way you can train your time keeping when you haven’t got an instrument on your hands.Also, I would have the beat fade in an out. The way it’s set up at the moment, you have no idea if you’re still keeping the right time after a while.”

“Great idea, I like it a lot. Definitely a different approach that I haven’t seen yet in a metronome. I currently use Y-Metronome, but I can definitely see your metronome getting a lot of use as well.A couple suggestions. Add a setting for 0 seconds in the fade start list. As well as an option to turn off the fade so it switches instantly. This way you can let the regular metronome go and then switch immediately to the ‘off beat’ metronome. This will increase the difficulty as you will never hear the two metronomes playing together. It will be one or the other.Another suggestion is to add an option to switch back and forth between the regular straight metronome and the off beat metronome after ‘X’ amount of measures (that you can set separately for both the normal and off beat metronome). This way you can ‘reset’ your timing so to speak if you get off time without having to stop playing and restart the metronome. Then to increase difficulty you can increase the number of measures for the offbeat metronome.”

“Show 8 beats in the bottom where it says ‘standard’, not 7. Also, have a distinctive sound for the ‘1’ of the beat in standard metronome, it could help greatly with getting a solid count to start with. It may also help if you put the 1 & 2 & 3 & 4 & somewhere in the diagrams at the bottom. Good stuff!”

“Things to improve upon:The GUI is a little too plain and not visually appealing. I don’t like the fact that you can’t change any of the settings after you click ‘start.’ After you click ‘stop’, it’ll continue for 1 or 2 beats. When you bump the tempo to 420, 430, 440, 460 it makes a weird distorted click.”

“Really cool software, a metronome that is adaptable to the player. Nice”

“Brilliant tool, I’ve been waiting for something like this for quite a while! Keep it up!”

“The instructions do not make clear enough the point of the software. It took a few reads to understand the idea is to hear the time pulse in your head at all times and play with that, not to play with the secondary pulse. It could be worded more clearly to better communicate the purpose. However, it is very good and I think it would be really useful. I think the object of most musicians should be to develop the clave sense, and I imagine this would help a lot. I do feel that a sense of perfect time is almost as rare as perfect pitch, but this software would help develop that skill and help a musician develop a very strong poly-rhythmic sense.”

“The difficulty slider seemed like an odd name for it. I found setting 6 easier to play than 5. The number relates more to the beat than to its difficulty. Nice tool though”

“I found this to be excellent for establishing one’s ‘internal clock’ when executing syncopated rhythms and would use it often. I didn’t understand question 14 so I’ll read the manual to see if I missed something. Not reading the manual first was intentional as to assess the initial ‘pick it up and use it’ ability and it passed with flying colors. I would like to see a snare sound but this comes from a 40 year bassist and I’m more used to that versus cowbells, etc.”

“I found one glitch. When I had both clicks as the same sound, (woodblock1 for example) any bpm above 152 I was getting a click noise not a woodblock sound.”

“Mini-manual should be a little less ‘mini’ – it would help to have a more complete explanation of the concepts involved, some practical examples of its usage to achieve specific goals, etc. Also, as far as the interface goes, would like to have a numeric stepper widget for the tempo box, and a lot more control for what’s going on the pattern section. Right now, the relationship between difficulty, standard, and pattern is a bit obscure. It’s genuinely interesting and has the potential to be very useful. Hope you keep working on it.”

“Nice one Dave - Travis”

“it doesn’t actually test what you can do - it’s a learning tool, right, not something you’d use on stage or in studio - learning is complex and needs stages and modules - a guided process. this is like a plug-in gadget - offers no program on how to take the student from A to B, how to grow etc. hope this comes soon ...”

“If you could include a feature that helps people practice the subdivisions in odd-meter, it would make this an awesome tool. One can use it for odd meter (sort-of) as it is now, but it would be better (easier to use for this purpose) if there was a menu where you could choose the time signature, and work on subdivisions that are derived from that time signature.”

“very good program, really can’t fault it”

“It would be a useful feature to be able to fade the standard metronome back in after a specified interval, in order to determine whether or not time has been kept accurately.”

Appendix E

Survey Website

The following screen shots were taken from the website created to enable potential respondents to download the prototype software and complete the online survey (<http://timecheck.asparagusaudio.co.za>)

The screenshot shows a website titled "Timing Training for Musicians". The main content area is on the left, and there are three yellow sidebar boxes on the right. The main content includes a "Home" heading, an "Update" section, a paragraph about the research, a section about the software prototype, a "mini manual" section, a "survey" section, a contact section for David Manchip, and a privacy statement. The sidebar boxes contain a "Menu" with links to Home, Download, Manual, and Survey; a "Links" section with links to Web Design Directory and Free templates; and a "Design by Minimalistic Design" credit. At the bottom of the sidebar is a Creative Commons license logo (CC BY-NC-ND) and text stating that materials are available on the site belong to the University of Cape Town and are licensed under a Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported License.

Timing Training for Musicians

Home

Update: Thank you to all who participated for all the feedback and suggestions, the survey is now closed. I will leave the downloads up for a while for those who might be interested in testing the software anyway.

I am currently doing research towards a Masters degree through the University of Cape Town. The research is centred around timing training for musicians, which is traditionally done using a metronome. Please consider [downloading](#) the advanced metronome software, assessing it for approximately 30 minutes and then completing the online [survey](#). Ideally participants should already be comfortable playing with a standard metronome as the software is targeted at the intermediate to advanced musician, although this is not a requirement.

The software prototype is written in Java, and so should run on a variety of platforms, including Windows, Mac and Linux. More detail on the [download page](#).

There is also a 'mini' [manual](#) available to explain the concept and how to use the software. Please read it, it's less than 2 pages long.

The [survey](#) is an online survey consisting of 15 point and click questions. It should take under 10 minutes to complete.

Any queries or problems can be directed to david-at-timecheck/asparagusaudio/co/za (replace slashes with dots) and I'll do my best to respond.

David Manchip

Privacy: any participants will be treated as anonymous, all data collected is strictly for research purposes and will not be used in any other way

Menu

- Home
- Download
- Manual
- Survey

Links

- [Web Design Directory](#)
- [Free templates](#)

Design by [Minimalistic Design](#)

Materials available on this site belong to University of Cape Town and are licensed under a Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported License.

Timing Training for Musicians

Download

System Requirements

- Java Runtime Environment (JRE) minimum version **1.6** (download the latest version from <http://www.java.com/en/download/>)
- A sound card

Personal Requirements

The advanced metronome software is targeted at intermediate to advanced musicians, so it is recommended that participants are already comfortable playing with a metronome or click

Files

(right click and "Save Link As...")

- [setup.exe](#) 1.5mb - Windows installer, tested on XP and Vista, not yet tested on Windows 7 but should work
- [setup.exe](#) 20mb - Windows installer with bundled Java runtime
- [timecheck.jar](#) 1.5mb - cross-platform executable, mainly for Mac users. Save to your machine, double click should run. Linux users can execute 'java -jar timecheck.jar' in the same directory as the jar file
- [TimeCheck.exe](#) 1.5mb - standalone Windows executable (exe wrapped jar file)
- [MiniManual.pdf](#) 220kb - a pdf version of the mini [manual](#)

Note for Mac Users: some versions of OSX have the default Java version set as 1.5. This will have to be changed to 1.6

Important

Due to the fact that this software is part of research conducted through the University of Cape Town, the software and any ideas represented by the software are the intellectual property of the University of Cape Town. By downloading this software you are implicitly stating that you understand this fact.

Menu

[Home](#)
[Download](#)
[Manual](#)
[Survey](#)

Links

[Web Design Directory](#)
[Free templates](#)

Design by [Minimalistic Design](#)



Materials available on this site belong to University of Cape Town and are licensed under a Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported License.

Timing Training for Musicians

Manual

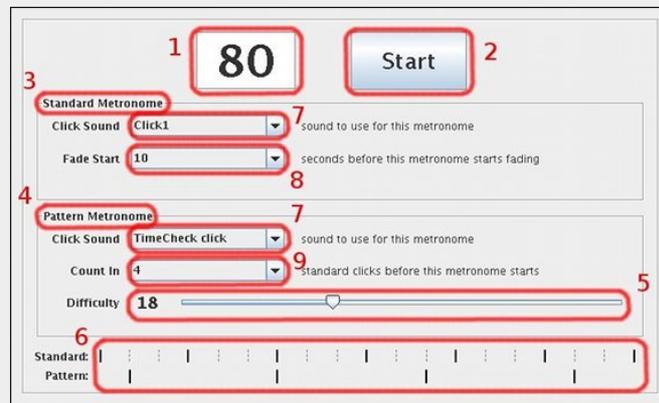
[Download](#) a pdf version of this 'mini' manual (right-click, "Save link as...").

Concept

When playing or practising with a traditional metronome one is always given all the 'beats'. This software will start off giving all the 'beats', but gradually the main pulse will fade out, leaving only occasional 'anchor' beats as reference points in time. The anchor beats occur as a repeating pattern that is related in some way to the main pulse. Learning to play with this software will force the user to develop their inner sense of time.

It is recommended that the user is comfortable with the use of a traditional metronome before attempting to use this software.

User Interface



1. The tempo box. Click and type to enter a new tempo.
2. The Start/Stop button. Click to start the metronome playing, click again to stop. No, really.
3. The controls for the Standard Metronome. See below for specifics of how these controls affect the Standard Metronome click.

Menu

- [Home](#)
- [Download](#)
- [Manual](#)
- [Survey](#)

Links

- [Web Design Directory](#)
- [Free templates](#)

Design by [Minimalistic Design](#)



Materials available on this site belong to University of Cape Town and are licensed under a Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported License.

4. The controls for the Pattern Metronome. See below for how these controls affect the Pattern Metronome click.
5. The difficulty slider. Use this to choose a pattern for the Pattern Metronome. Although not really an indication of difficulty, it should at least give a rough indication of how difficult it will be to play along with the pattern. If dragging the slider with the mouse is too sensitive, use the arrow keys.
6. This is a picture of what the two metronomes will sound like together before the Standard Metronome fades out. Note that this visual may not show the full repeating pattern, and may only show the first few measures.
7. Click sound drop down boxes are used to choose what sound to use for the metronomes.
8. Fade start. This is the number of seconds that pass after clicking "Start" before the Standard Metronome starts to slowly fade away.
9. Count in. This is the number of Standard Metronome clicks that will play before the Pattern Metronome joins in. Note that the pattern picture (6) does not show the count-in clicks.

Tips

- The most fundamental thing to remember when using this metronome is that you ALWAYS need to feel the original pulse, even after it has faded out. If you are a drummer, it is a good idea have the kick drum play this pulse while doing hand exercises until you are really comfortable with it. If you 'lose it', and are no longer sure where the pulse is STOP, and start again, perhaps with a longer fade out setting.
- Take it slow. Start with difficulty level 1 and get comfortable playing your favourite grooves, exercises and pieces of music at a variety of tempi. Only then increase the difficulty level. It is generally a good idea to only increase the level once you are comfortable with the current level.
- Playing at slow tempos is harder than playing at fast tempos, so work down, not up.

Frequently Asked Questions

How should I be using the advanced metronome?

In principle you should be using this metronome just as you would any other metronome: pick your tempo, hit 'start' and start playing your chosen exercise, piece of music or groove in time with the metronome clicks. The important thing is to feel the standard pulse, even after it has faded out. Over time you can start increasing the difficulty level to give yourself more challenging patterns to play with, but don't rush it.

Known Issues

- Rendering problems – on some systems the user interface cuts off a small piece of the bottom of the screen. This doesn't affect usability or functionality, it just looks bad.
- Metronome doesn't start up – this can be due to some other audio program locking the audio sub-system. Try to close all other audio-related applications before running the metronome.