Click Tracker: Performance/composition tool for metrically complex scores

João Pais[*]
Friedenstraße 14
21098 Berlin
Germany
jmmmpais@googlemail.com

Pedro Lopes[†]
Department of Information Systems and Computer Science
INESC-ID/IST/Technical University of Lisbon
R. Alves Redol, 9, 1000-029 Lisboa, Portugal
pedro.lopes@ist.utl.pt

Abstract

Click Tracker is a tool designed for composers, conductors and instrumentalists working with modern music. It allows users to prepare an accompanying click track of any musical score, independently of its complexity. Also, several of its multimodal features take advantage of both visual and aural feedback; making it suitable for musical study or learning contexts. The software is available for public download and has been tested by several professional instrumentalists and composers, which results are hereby discussed.

Keywords

Metronome, Composers assistance software, Music performance, Pure Data, Modern music

1 Introduction

Click Tracker is a program based upon Pure Data[3] designed for instrumentalists, conductors and composers working with modern music. The main goal of the software is to prepare a click track of any score, no matter how complex it is.

Programmable metronomes and musical notation software are nowadays widespread, allowing musicians to create basic training scores. But these tools do not account the needs of contemporary composition, whereas the complexity of the score or the learning context cannot be addressed. Our solution addresses these challenges and provides a simple tool, suitable for both realtime performance as well as training and composing situations.

Performers, conductors and composers working with new music often face challenges throughout score preparation of rhythmically demanding pieces. Amongst these challenges we identified: lack of aural feedback besides traditional metronome "clicks", irregularity of beat durations, changing bars, tempi variations, repetitions, and so forth.

The Click Tracker reads the rhythmical structure of a piece, written in a specific syntax, and reproduces this structure aurally. Many of the features necessary for the work in modern music are covered and will be presented in the next sections. Further, we will address technical aspects of Click Tracker's features. Finally, we describe the numerous usages of Click Tracker by performers and composers; closing with our conclusions and depict upcoming features and improvements.

[*]Click Tracker main developer and project leader.

[†]HCI research, collaboration for GUI and this document.

2 Click Tracker

In this section we describe Click Tracker from two perspectives: user (who is interacting with it) and technical aspects of the feature implementation. Firstly we describe the dynamic metronome, that enables users to reproduce complex click accompanying tracks, following current trends on modern music. Followed by a description of user-generated content for the creation of the click tracks, the score syntax, and how to prepare language sound files. Also from the user perspective the Graphical User Interface (GUI) is depicted and explained. Finally Networking and Multi-user capabilities are discussed.

2.1 Complex Metronome

The Click Tracker feeds itself from a text file with the specific syntax to generate a score (for detailed example please refer to Appendix A). The commands include:

- ▷ beat durations: as integers, floats, or fractional symbolic notation
- ▷ regular bar structures: including untraditional bars and bars using irrational numbers
- ▷ tempo changes: sudden or gradual
- ▷ repetitions
- ▷ pick up bars
- ▷ aural output, language and recording format definitions

The aforementioned commands give the user a tool for manipulating durations, accommodating even

the needs of unorthodox scores, as depicted in Figure 1.

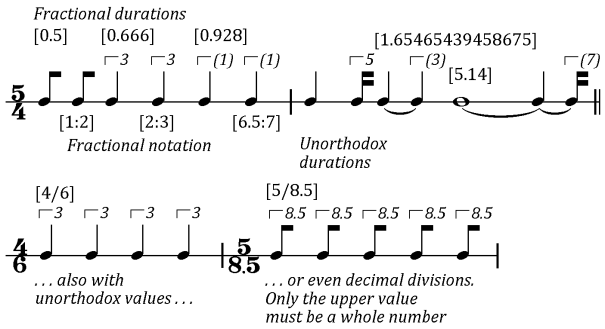

Figure 1: Some duration features from the tutorial. Numbers between [] indicate the original score syntax.

The score is described as a list of individual events with specific durations: beats, or beat groups, in the case of regular bars. Modifier commands (such as tempo or other structural commands) precede the beat(s) they affect. This information is interpreted by the patch, and organized into a group of arrays, which together generate a map of the score. These arrays, as depicted in Figure 2, store information regarding: beat duration, bar reference, tempi and bar database.

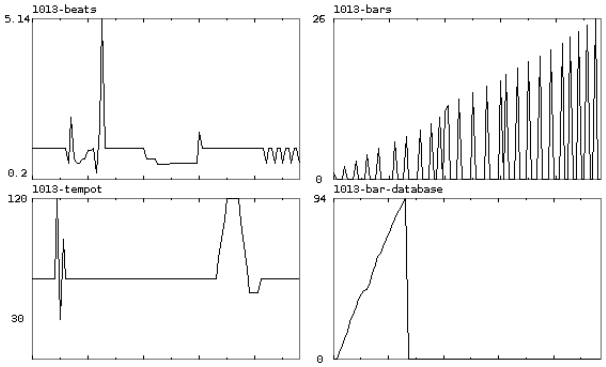

Figure 2: Internal arrays displaying the file *score-tutorial.txt*, sent with the program.

For each event (each beat) of the score is created an index in the first three arrays. These are responsible for storing the required information regarding current beat duration and tempo (after the tempo changes have been pre-calculated). For external reference, a fourth array contains the bar database, which lists at which index each bar begins.

To playback the score, the duration of each beat is read sequentially, considering its original or scaled tempo. A delay object is used to time the durations.

2.2 Aural output

Click Tracker distinguishes itself from hardware/software metronomes or conventional scoring software because it also allows several aural capabilities apart from conventional clicks. Figure 3 depicts the overall GUI of the current version, in the third section of the GUI (Figure 3.3) the user can choose how he wants the software to read the score:

- ▷ conventional metronome click
- ▷ voice (counting bar numbers) and click
- ▷ only voice, counting beat numbers in the bar
- ▷ voice, counting bar numbers plus beat numbers

Furthermore a wide range of languages is supported (English, German, Italian, Spanish and Portuguese). Other languages are being prepared - as the production process is very simple, ideally all languages can be integrated into the software.

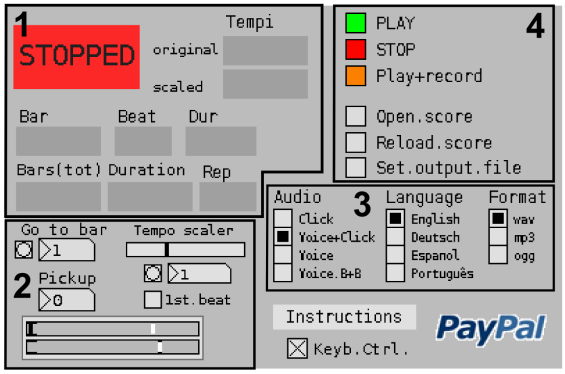

Figure 3: Click Tracker GUI components: **1)** Performance display; **2)** Performance control; **3)** Output settings; **4)** Transport and file control.

2.3 Score format

The Click Tracker supports a wide number of score formatting options, amounting to complex options which our testers confirmed to be suitable for performing contemporary music. These features are listed as follows:

- ▷ Any duration for individual beats (integer of float);
- ▷ Beat fractional notation in format x:y (e.g: 1:2 or 3:7,8);
- ▷ Bar symbolic notation in format x/y (e.g.: 4/4 or 3/9.17);
- ▷ Tempo change at each individual beat;
- ▷ Gradual tempo modifiers (*accellerando* / *ritardando*) in any amount of beats;

▷ Repetition of any fragment (independent from the bar structure);

Besides that, it is possible to bind the following performance parameters with each score:

▷ Number of pickup bars

▷ Audio setting

▷ Language

▷ Recording format

Please refer to appendix A for a detailed example of score format, further details regarding notation syntax are provided in[4], or in the tutorial score packaged with the program.

2.4 Structure and preparation of language samples

Each sample is provided as a *wav* file, stored in the folder with the sub-patches. To make the samples as efficient and lightweight as possible (the bulk of the patch size is taken by the samples), only whole numbers from 1 to 100, and then every hundredth from 200 to 1000, are available. So, although the numbers 36, 136 and 236 aurally sound as "36", after a first run-through it is quite clear which number is meant to be playing - especially if the player is also looking at the GUI while rehearsing/performing. Besides the greater efficiency while preparing the audio files, it is also of advantage to have the numbers as short as possible. In some languages, such as German, numbers with more than 2 digits become very long words, which can't be spoken fast enough in a fast tempo.

The preparation and incorporation of new languages is a relatively simple process. A native language speaker with a clear diction - preferably someone with some acting or speech-related background - is better suited to read the numbers in a very precise, short, rhythmical, and yet understandable, way. After cleaning as much background noise as possible, the samples are synchronised aurally in a DAW against a reference click track at constant intervals. The playback is made by playing a specific number (that is, excerpt) from a specific file, directly from the disk. By not loading any sample to the memory buffer makes it easier to spare resources, without interruption of any ongoing DSP chain.

For now, the samples are recorded in mono CD-quality *wav* format. In the future, they will be downsampled to smaller sampling and bit rates - this will make the patch even lighter (and easier to use in mobile platforms), without any noticeable loss in the audio quality. Currently, the patch demands a tiny amount of processing power.

2.5 Networking and Multiple-users

The Click Tracker supports networking via OSC protocol[5; 1], using it to communicate with the external GUIs, such as the one proposed in this paper. All the parameters and features displayed in the GUI are available as OSC tags. Therefore, it's a simple task to configure new control surfaces to work bidirectionally with the Click Tracker.

Currently, the OSC messages are divided in a very simple way:

▷ GUI[1] → Pd: all user-controlled commands, sections 2-4 of Figure 3;

▷ Pd → GUI: realtime performance display elements, section 1 of Figure 3;

Since all realtime informations of a performance are available, it is a simple task to take any information - such as the current score location - and use it to control any OSC-compatible patches (e.g. in other software, such as Max/MSP, SuperCollider or Csound).

2.5.1 Pd GUI

The GUI in the original Pd patch is divided into four main sections (Figure 3), according to task separation:

· Performance display, where informations about the piece being played are displayed;

· Performance control (velocity, volume, and so forth);

· Output settings (language, audio and recording formats);

· Transport and file control (play+stop, file operations);

Furthermore, it is possible to control the Click Tracker's native GUI using keyboard shortcuts (refer to Appendix B) or turn the keyboard control off.

2.5.2 External GUI

Visualization and interaction with the click Tracker can be performed without recurring to the aforementioned Pd interface. In this case, an experimental interface was devised in Processing/Java, aiming for cross platform compliance and easier deployment amongst desktop and mobile scenarios. The external GUI takes advantage of Pd's networking capabilities to communicate through OSC messaging, as depicted in Figure 4. Any interaction on

[1] Or any other software/hardware controller, since the basis here is Client↔Server paradigm.

the Interface will be communicated to the Pd patch, while any changes of the current performance state (e.g. tempo, beat or bar numbers) will be communicated to the Interface for displaying purposes.

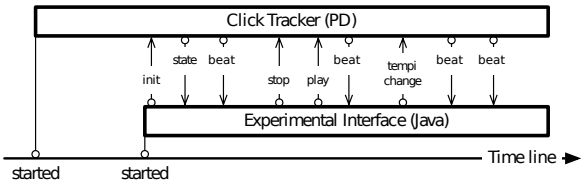

Figure 4: Communication between click tracker and an instance of the experimental interface.

The Interface is depicted in Figure 5, and follows the same organization of the original GUI, however it provides additional features, such as:

· hiding of content (such as minimizing the "Settings" tab);
· drag and drop of score files to anywhere on the interface, which will cause the new score to be loaded automatically;
· reset any parameter with a mouse right click (if a mouse is used as input device);
· operating remotely via OSC may facilitate in rehearsal situation, where the Pd patch can run on a computer and the Interface on a tablet or phone.

Figure 5: Experimental interface supports new feature, such as drag and drop scores.

3 In-performance Testing

The Click Tracker has been available as a private project for a couple of years. The first public presentation was made at the Darmstadt New Music Courses in 2010. Since then, several instrumentalists, ensembles and composers have joined the group of active users. The Click Tracker has been intensively used in all music-related contexts: for study, performance and composition of pieces.

The use of the program in varied real-life situations gave us a thorough insight about many small details to change, ranging from small user-friendly suggestions to bug reports. The usual feedback with musicians who used the software, as well as a small questionnaire of the known users allow us to draw out the following conclusions:

· musicians use the Click Tracker mainly for contemporary music, when the metrical or tempi structure is very complex. Composers such as B. Ferneyhough, B. Lang, E. Nunes, and E. Poppe (among others) fit better into this category;
· in pieces of B. Ferneyhough with a simple bar structure (usually x/8 at slow tempi), but very complex rhythms, usually with several tuplet layers, several musicians adopted the approach to prepare a metrical structure of strategic points - instead of the normal bar structure;
· it is used mostly at rehearsals, as musicians prefer to play without interference on stage - unless a precise coordination e.g. with electronics processing or tape is necessary;
· independently of the complexity of the musical piece, or of the computer experience level of the user, it is the general opinion that the current structure allows musicians to program (almost) all the elements necessary, while doing so in a very simple environment. In two particular cases, it took a composer (with computer experience) around 10 minutes to input the structure of his 15 minute piece; for a pianist with very few computer experience, it took him several hours to input a piece of B. Ferneyhough with a very complex rhythmical structure, while at the same time recalculating all tempi to fit one of the available tuplet layers (around 2 or 3 at any given time) of his choosing;
· the answer to the question "*did the program fulfil your needs and expectations*" varied between "*yes*" *to* "*completely*" (although, of course, still many improvements can be done);
· users said they advised the use of the Click Tracker to other musicians.

Among others, the program is now used by the following ensembles: Nadar Ensemble (Belgium), Jack Quartet (USA); performers: Pavlos Antoniadis (Greece), Anna d'Errico (Italy), Katrien Gielens (Belgium) and Matteo Cesari (Italy); and

the following composers: Aaron Cassidy (USA), Robert Dahm (Australia) and Mesias Maiguashca (Ecuador/Germany). These are just the active users, as many other musicians have benefited from click tracks made for the pieces performed.

Furthermore, although no users disagreed with either features and/or usability, it was still sometimes hard to make traditional musicians (even when they play untraditional music) to try out the software, or to integrate it into their workflow. This reveals that more musician/composer-friendly features, such as an appealing GUI, integration with commercial notation software, midi import/export or the mobile/web platforms maybe be worth researching into.

4 Conclusions and Future Work

Throughout this research we describe the major features of the proposed software, showing how it can assist users to create accompanying click tracks for complex scores. The expanding list of musicians that benefited from Click Tracker over the last years enable us to conclude upon its adequacy towards various musical scenarios: composing, performing and learning.

Latest research efforts have been made to create external networked interfaces through the integration of the OSC protocol, allowing the software to be controlled through an external GUI and accommodate (and synchronize) multiple users. Furthermore, the described GUI implementation (in Java) will enable it to be hosted entirely on mobile platforms, such as Android, as well as others were Pd is available[2] - including a Pd Webserver.

As for the future, many features are planned, either just extensions of existing features - e.g., the curve design of tempo changes; the integration of new languages (Norwegian and Flemish are already recorded); or new features - bar subdivisions, integration of ad-hoc samples; richer score syntax, and so forth.

An important improvement will be a more detailed accuracy by means of of audio-rate objects; although the current solution is not sample-accurate, it has shown satisfactory results for all musicians working with Click Tracker in their realtime performances.

An additional resource for score preparation could be a common online library, that would archive practice and performance scores created by musicians that use this software. This platform will allow musicians to share the work done between each other, and hopefully encourage others to use the program.

5 Acknowledgements

Our thanks go to the people who encouraged and/or helped the development of the program, either by trying it and reporting on it, and to the 1 composer, 1 performer and 1 ensemble who gave financial support for further development.

References

[1] A. Chaudhary. Automated testing of open-source music software with open sound world and open sound control. In *ICMC 2005*, 1/11/2005 2005.

[2] PD. Pdanywhere. 2011. `http://sourceforge.net/projects/pd-anywhere/`.

[3] M. Puckette. Pure data: another integrated computer music environment. In *in Proceedings, International Computer Music Conference*, pages 37–41, 1996.

[4] C. T. recourses. `http://code.google.com/p/clicktracker` and `https://www.facebook.com/Click.Tracker`, 2011.

[5] M. Wright, A. Freed, and A. Momeni. Open sound control: State of the art 2003. In *International Conference on New Interfaces for Musical Expression*, pages 153–159, Montreal, 2003. OpenSound Control.

6 Appendixes

6.1 A: Score Syntax

The following is an overview of the score syntax used to create click tracks. The text after the # character is considered a comment and ignored by the parser.

t <float> *# tempo in MM, e.g., t 70*
b *# beat duration, a afterwards marks the beginning of a bar*
: *# fractional notation for duration, e.g., 2:3*
<integer>/ *# abbreviated input of regular bars, e.g., 4/4*
tc *# tempo change. First argument is new tempo, second is the remaining beats until change, e.g., tc 120 5. Beat duration of the tempo change must coincide with the sum of the beats it lasts.*
rep <integer> *# n repetitions of a fragment. First argument is the sum of the beats to repeat, second is the n value, e.g., rep 4 3*
pickup <integer> *# pickup bar repeats the 1st bar n times*
first <integer> *# number of the first bar of the score, e.g., first 35*
par <cl/clst/st/stbb> <english/deutsch/italiano/castellano/portugues> <wav/mp3/ogg> *# playback parameters*

6.2 B: Keyboard shortcuts

o - open score
l - reload score
space - play/stop
f - set output file
s - play + record

esc - turn keyboard control on/off

6.3 C: OSC tags

Pd → GUI
/tempo - Original tempo
/stempo - Scaled tempo
/bar <integer> - Current bar number
/beat <integer> - Current beat number
/dur - Current beat duration
/totbar <integer> - Total bars in score
/totdur - Score duration
/rep <integer> - Repetition number
GUI → Pd
/play - Play
/stop - Stop
/play+rec - Play+record
/score-l - Open score
/score-rl - Reload score
/o-file - Set output file
/go-to-bar <integer> - Go to bar x
/barbang - Reset go to bar
/pickup <integer> - Pickup bars
/tempi-sc - Tempo scaler
/tempobang - Reset tempo scaler
/1st-only <integer> - Play only first beat in bar
/aud <integer> - Audio settings
/lng <integer> - language settings
/rec-f <integer> - Format settings
/keybk <integer> - Keyboard control
/out-level-i - Audio volume

/state - sends back list with current data

6.4 D: Click Tracker tutorial

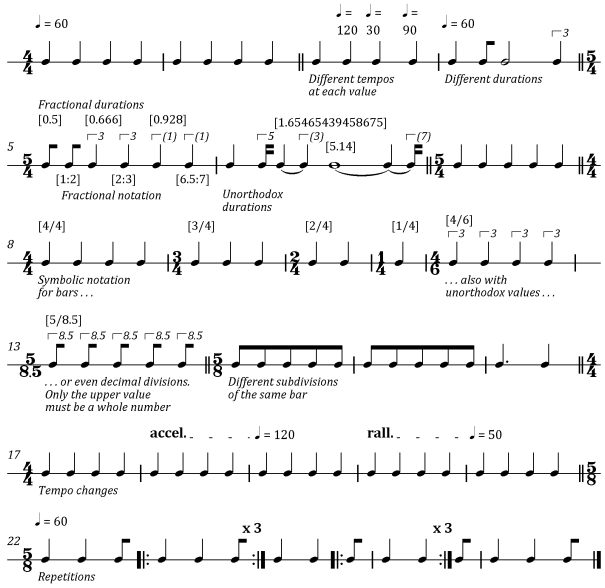

Figure 6: Score of the provided Tutorial, exemplifies supported features of click tracker.