Introduction

* There are many ways of making music (documented in some capacity)
* DAWs exist and primarily allow users to work on a project (usually in the same location and machine, although it is possible to share project files in order to collaborate)
* There exist less ways of documenting a collaboration remotely, say over network
* Ableton provides Link for their DAW and other compatible software – yet this still restricts users to being part of the same local network. Briefly explain Link
* We aim to provide a Link-inspired functionality that works in a wide area network whilst still keeping the live performance element of it – introduces challenges involving dealing with networking issues such as latency and jitter, as well as (very critically) the problem of making sure everyone hears the same thing (a distributed sense of coherence)
* Briefly explain Sonic Pi and why we use it – well established software and language that already allows support for messages to be transmitted across a network

Starting Point

* Relevant courses – Networking and Distributed/Concurrent Systems
* Existing Projects – Quintet.net, Simulus
* Access to Ableton Link’s developer resources (including GitHub repository with some source code)

Substance and Structure *i.e. Key concepts, Major work items + their relations and relative importance, Data structures and algorithms*

* Explain Link properly
* We use Sonic Pi’s built in OSC receiving/sending capabilities
* Distributed related points – known algorithms that deal with coherence that could maybe be used
* Evaluation: look at timestamps for OSC messages between Ruby and SuperCollider and just check they’re the same across different users
* Make sure there is the same order of messages when they hit SuperCollider

Success Criteria

* Fully implement a way for users to setup a connection and play via Sonic Pi
* Ensure coherence of each user’s notion of the performance (when network conditions are acceptable) so all users hear the same thing
* Make it able for at least 2 users to be able to play together without (or with minimal) problems

Possible Extensions

Different schemes of getting coherence

Different implementation approaches Quintet.net seem to have a concept of a server, client (and conductor) instead of a peer to peer approach and I guess it probably works

WebRTC instead of server

Tasks

* Build module in between Sonic Pi and Network module
  + so users can specify what connection they want to make
  + this would ideally have a nice GUI (maybe MAX or Pure Data would work, although I know little about them, could sure learn)
  + This then connects to a network module to get incoming OSC messages
  + Handles logic regarding telling Sonic Pi when to act on said messages as well as making sure it is doing the correct thing (i.e. all ‘clients’ work together to ensure agreement) – dealing with OSC ‘merge conflicts’
* Build network module
  + handles outgoing and incoming OSC messages
  + WebRTC as a method of transport (or server approach)
* Point
  + Sub-point
* Point
  + Sub-point

Timetable

Weeks 1-2

* Read Lots

Weeks 3-4

Weeks 5-6

Weeks 7-8

Weeks 9-10

Weeks 11-12

Weeks 13-14

Weeks 15-16

* Finish progress report
* Prepare presentation
* Start dissertation planning

Weeks 17-18

Weeks 19-20

Weeks 21-22

Weeks 23-24

Weeks 25-26

Weeks 27-28

Resources Required

Timeline

Ableton Link

Ableton’s Link functionality is designed to allow musicians using separate instances of Link-compatible software, to play in sync in a local network, by keeping a global notion of time – this is done by synchronising tempo, beat and phase alignment. To understand Link, it is first necessary to define what is meant by playing “in time”.

The starting point in allowing two or more users to play in time, is the synchronisation of tempo. It is possible to describe music as being bound by beats, signifying points at which notes should (or should not) be played. The rate at which these beats occur in relation to time, is deemed to be the tempo of the musical piece – this rate is sensibly measured in beats per minute (BPM for short). A collaborative performance, requires all musicians to adhere to a predefined tempo, so that there is no divergence of the rate of the individual contributions – any divergence leads to an incoherent piece as components sound out of place.

It is not sufficient to synchronise tempo in order for users to play in time, as it is possible to imagine a scenario in a collaboration where differences in starting points exist amongst users – such that one user starts their contribution slightly before or after the others. Even with a synchronised tempo, it is likely that the beats of each individual user don’t match up – even though they may be occurring at the same rate. The solution to this issue is making sure that the beats of a user’s contribution, align (occur at the same time) with the beats of all other users – implicitly, ensure that this alignment is done for all users. This can be further clarified if we use integers to number each beat (starting from 1 denoting the first beat). Beat alignment aims to ensure that the difference between the current beat number of any two users, is itself always an integer, such that it is possible for one user to be at beat 4 whilst another is at beat 2, but it is not possible for third user to be at beat 2.5 (halfway between beat 2 and 3).

The final step to playing in time involves phase synchronisation. The need for this comes from the idea of music being made up of bars consisting of a specified number of beats – Ableton’s software also has the notion of loops made up of bars – and the problems that arise when bars or loops are not aligned. For example, assuming synchronised tempo, if one user starts playing a 4-beat loop at beat number 1 (in a global sense) and another starts to play a 4-beat loop one beat later (at global beat number 2), even though the beats are aligned, the loops start and end one beat apart. This is often problematic as the start and end of bars/loops, often assign a sense of meaning to a musical piece. Phase synchronisation deals with making sure that the start or end of bars or loops are somehow aligned by synchronising the beat position of individual contributions. This means that different users playing a 4-beat loop will always observe the same start and end to their loops (even though their separate beat numbers may be different e.g. if one user starts their loop 4 beats before the other). For different sized loops, phase synchronisation still works if the loops in question are multiples of each other, such that a 4-beat loop will be synchronised to start either at the start or the middle of an 8-beat loop.

OSC

OSC defines a message-based protocol that allows communication between different multimedia devices such as computers or synthesisers. OSC messages can be used to encode descriptions of sounds that can then be used in various scenarios.

SonicPi

WebRTC