Tangle User Manual

February 6, 2014

Contents

1 Quick start instructions

1. Server

- **i.** Check all the desired instruments are connected correctly.
- **ii.** Start up the server. Once it has booted, log in either remotely—

  ```
  %> ssh sonics@192.168.33.1
  ```

  or using the login screen. If you log in locally to the server, open a terminal window.
- **iii.** Check `Instruments` directory—

  ```
  cd ~/Network/Tangle
  ls Instruments/
  ```

  make sure files are present for each of the instruments you intend to use.
- **iv.** Run `Master.ck` – in the shell this is (assuming we are still in the root project directory)

  ```
  %> chuck src/Master
  ```
- **v.** It will produce a lot of text, double check to make sure no errors were emitted. You could run the server from the miniAudicle, but this is often less stable over long periods of time.

2. On your own machine

- **i.** If you want to use your own OSC
  - **a.** Double check the address patterns the server printed at the end of its initialisation, you should be good to go.
- **ii.** If you want to use MIDI run `Client.ck` – it needs some information to function correctly. The minimum is:
  - **a.** An address for the server; hopefully 192.168.33.1 or `leoadmins-Mac-mini.local` will work [1]
  - **b.** An IP address for the client machine
  - **c.** Using this information, run `Client.ck`. The information above must be passed as arguments. For a server at the address 192.168.33.1, a client IP address of 192.168.33.3 and MIDI on the port "IAC Driver 1 Bus 1" the command would be as follows:

    ```
    %> chuck Client.ck:server=192.168.33.1:self=192.168.33.3:\
                midi=IAC Driver 1 Bus 1
    ```

    A full list of options an be found in section 3.

2 Start Up Procedure

What actually happens when the above instructions are followed:

1. Server starts running

- **a.** Searches `Instruments` directory, attempts to load files (ignores directories)
- **b.** Constructs list of instruments from files
  - i. MidiInstrument class sets up MIDI output and translation from OSC

[1]To find the IP address the server is using for the ethernet either look in the settings or use `ifconfig` to find the IPv4 address.

ii. Base class Instrument sets up OSC listeners according to what is defined in file.
iii. All instruments have two default messages they expect if nothing is specified – `/<name>/note,ii` and `/<name>/control,ii`

c. Starts listening for new clients

2. Client starts

a. Sends `/system/addme,si` to Server (with own hostname and port)

b. Server responds with a series of `/system/instruments/add,s` messages which list the instruments constructed earlier by name.

c. Any instruments with possible messages beyond the basic two send `/system/instruments/extend,ssi` to the client which contain the name of the instrument, the pattern for the message and the MIDI status byte to transform.

d. All instruments send any information they know about themselves in `/system/instruments/note,ss` messages where the first string is the name of the instrument and the second is a note about the instrument, probably defined in the data file.[2] The notes get displayed by the client to give the user any information the instrument's designer feels useful.

e. Client uses this data to construct a table of MIDI input to OSC output and prints details about the instruments connected to the server to the console.

f. Client checks if any latency calibration has been specified, if so sends to server. Blocks until server indicates it is complete.

g. Client checks if any test patterns have been asked for, if so sends to server, blocking until notified of completion.

h. Client listens for MIDI input on specified port and translates appropriately.

3 Client.ck options

Options for `Client.ck` are specified via a colon separated list of arguments. All arguments are specifed in the form `<key>=<value>`. A full list of options is in table 1.

An example of a likely command to run the client might then be:

```
%> chuck Client.ck:self=192.168.33.3:server=192.168.33.1:\
midi=IAC Driver 1 Bus 1:test=all:delay=on
```

4 Adding New Instruments

The server discovers instruments by searching the `Tangle/Instruments`. Each instrument requires a file which tels the server how to talk to it. This can be a configuration file to be read in or a ChucK source file which contains a sub-class of Instrument.

[2]Confusion between `/<name>/note,ii` and `/system/instrument/note,ss` should be avoidable given the different typetags and the `/system` prefix, although it is an unfortunate homonym.

Table 1: `Client.ck` options

key	values	Description
server	url or numerical IP in the format AAA.BBB.CCC.DDD	Tells the client the IP address of the server.
self	same as for server	Gives the server a return address to send information about connected instruments.
in	an integer	Port on which Client listens for communication from server (default 50001).
out	an integer	Port which server is listening on (default 50000).
midi	an integer or the name of a MIDI port	MIDI port on which the client listens, defaults to 0. It is better to use a name as the order of these can shift. Available MIDI ports can be found by running `chuck -probe`.
test	a comma separate list of names of instruments (ignores case)	Tells the client whether or not to ask the server to run a test pattern, and if so on which instruments. An empty string or `none` will not cause any tests, `all` will ask the server to test all instruments connected.
delay	a comma separate list of names, as for test.	Tells the client whether or not to ask the server to begin the delay calibration process. Special values `on` or `off` tell the server to calibrate all or no instruments respectively.