Blog

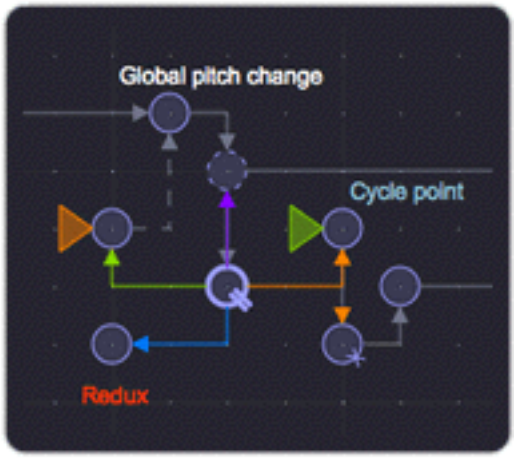
this blog has been compiled of iPhone memos and notes left on paper, I've merged it into one document and attempted to piece together the timeline.

Building a generative instrument in pure data

Research period
during the week i hope to identify key players in the field. i aim to identify what exactly i would like the software to be able to do.

throughout several weeks i have experimented with different apps, my favourites so far are Wotja and nodal

Wotja allows the user to create generative pieces based on loops and has a great looking GUI (the cable joining reminds me of early reason). Its relatively quick to learn and has some great preset samples. It does lack the ability to simply select some notes and make a pattern however.



Nodal is slightly more obscure and has a similarity to the Reactable, its very fun to use and can be connected to any Midi instrument the design is great for mobile devices, however controllability is quite random.

Wotja can be downloaded at
http://www.wotja.com

Nodal can be downloaded at
http://nodalmusic.com

What to make

I've decided to make my patch extremely controllable, that may seem odd as the app is supposed to be generative, however i believe making a choice of notes that are picked at random is still random right ? The application will be designed to tailor for the electronic producer that may not be extremely confident with instruments or phrasing. I want the user to be able to scan a track, receive note values and tempo and then be able to create a track with similar rules. I also want the app to be capable of humanising, i don't want it to sound to robotic, so velocity and length will be explored.

Starting the build

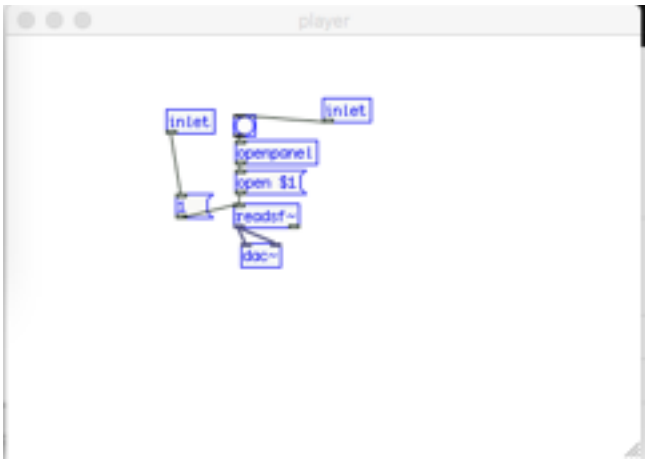
Aubio
// The first thing to build is definitely going to be the analyser, I've found an external library for pure data called Aubio which can be downloaded here https://aubio.org

// So after countless attempts i have finally managed to get the audio library to work with PureData, the trick was to download the pre built version available through the link, then transfer said file into the PD library through finder, in pure data simply create an [importaubio] object and it should be up and running.

// With audio finally running its time to start building the patch

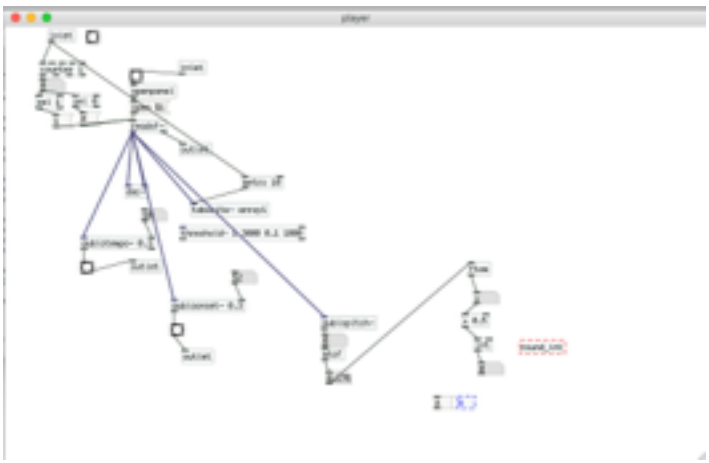
Importing the sound file

// So the first step in the patch is importing the sound file, this requires the use of Pure Data's [open] object.



// This allows the user to open a WAV file thats located on their machine, once this process is done, its time to put Aubio to work.

Analysing the Sound file



// To analyse the rhythm of the track I'm using the object [aubiotempo] it gives a reading in bangs, which will allow us to convert that to BPM

// To get a pitch reading I'm using the [aubiopitch] object, it gives a reading in hz so it can be converted to midi within pure data using the [ftom] object.

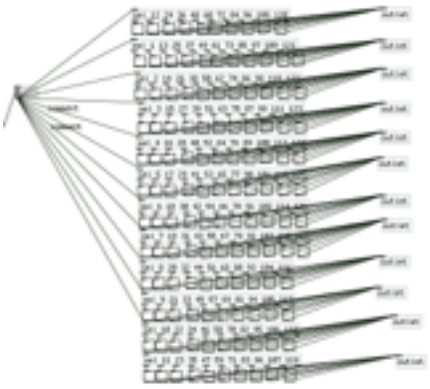
// Its very difficult to get an accurate pitch reading using only the pitch object, i managed to get a clearer reading when filtering the track for this reason i have added both a high pass and low pass filter. Although not the most elegant solution it allows the user to find a sweet spot where the reading is fairly accurate.

// I have found the [aubioonset] object, it basically measures silence, this can add to the rhythmic analysis and i will hopefully use this to trigger some phrases.

The Metronome

// If the project is going to sequence anything at all its going to need a metronome, on previous projects i have always kept the project in PD's native millisecond however as i want this to be user-friendly i have found a great patch to implement and it can be found at http://www.pdpatchrepo.info/hurleur/superTap1.pd.

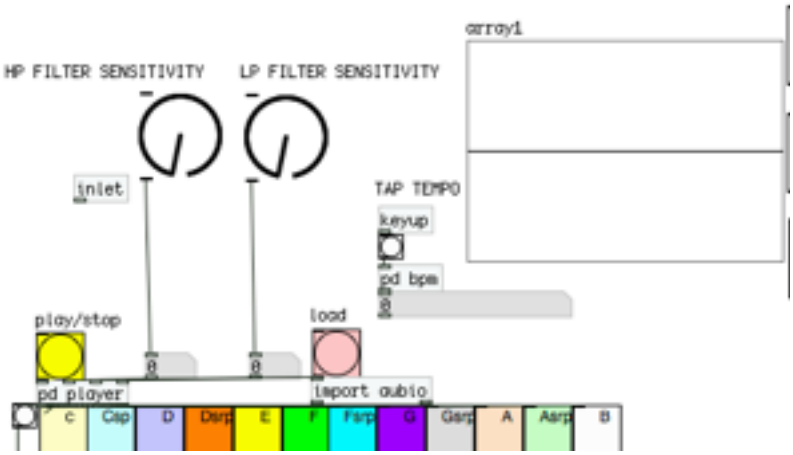
// with the metronome implemented the next stage is displaying the notes that have been identified.



//Using this selection process to identify the notes, the notes are shown in a toggle box in the main window

// when a note has been identified it will show up like this

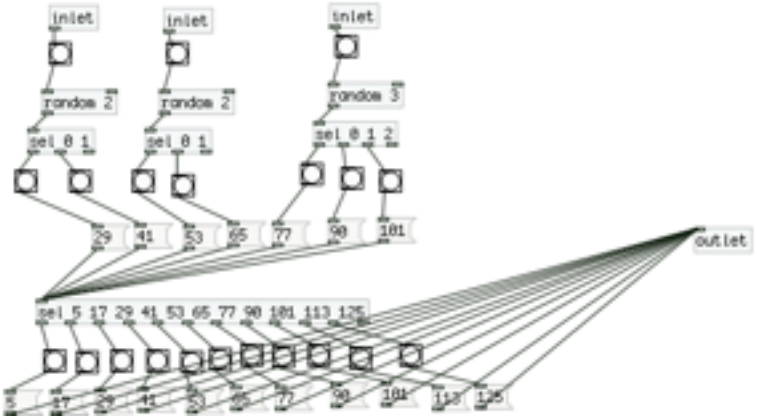
// It seemed appropriate for the user to have visual feedback that the sound file is playing and being analysed so a tab write object was added



//Knob objects were added to the filters for aesthetic value

Adding Notes

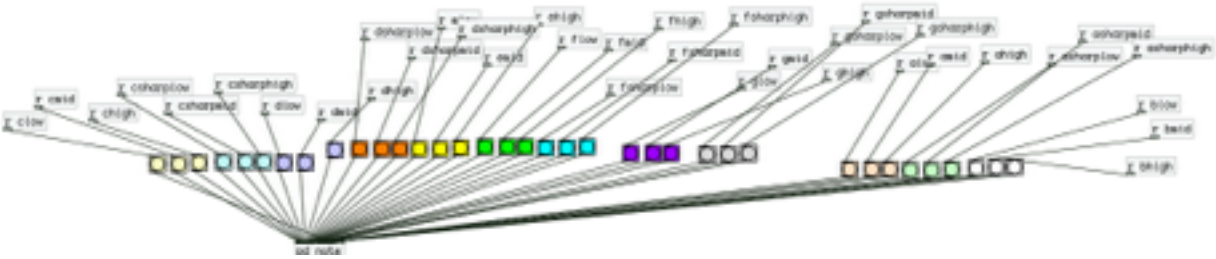
//once the midi values had been identified, it was important for them to be able to be sent out to the midi instrument



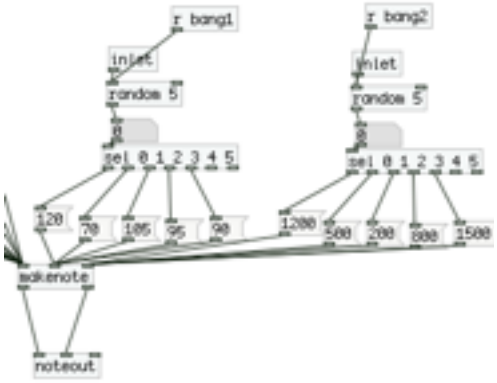
// Here is one example of a note patch, this is replicated for all 12 notes, the note values are adjusted.

// I have decided to make use of pure datas random object, the user can decide low mid or high but not the exact octave.

//Once the notes have been inputed they are put into a sub patch ready to be triggered.



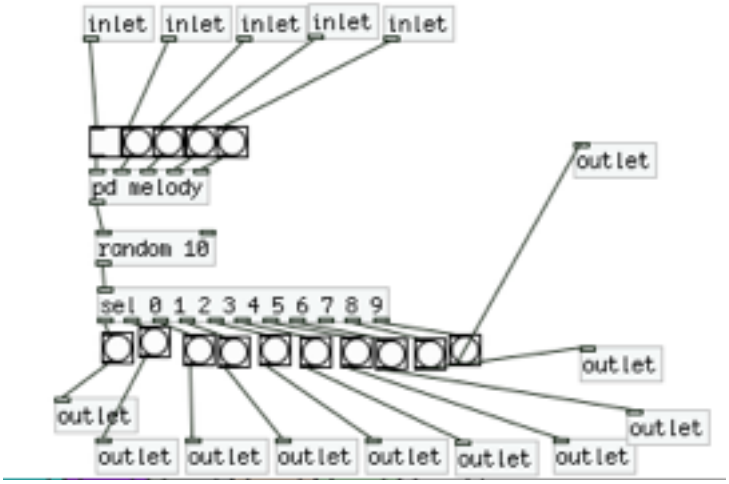
// The note is sent to the [makenote] and the [noteout] object. the right inlets control velocity and length, this is set at random also using the [random] and [sel] object.



// I've decided to use Native instruments Massive as the sound source, within logic pro the midi channel can be set which allows for plenty of sonic options.

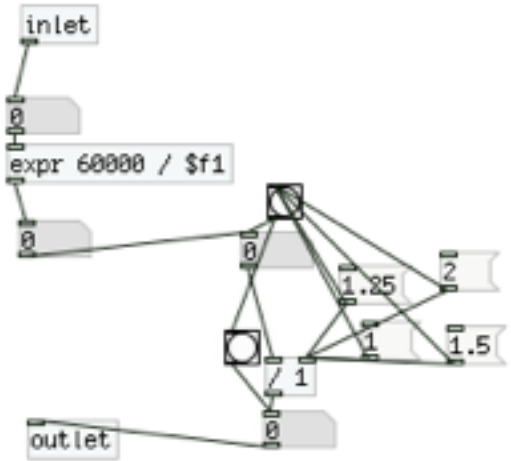
Sequencer

// The sequencer is built up of a random 10 object, this allows the user to control the chances of a note being played. If three bangs are set to low C, there is a 30 percent chance of low C being played.



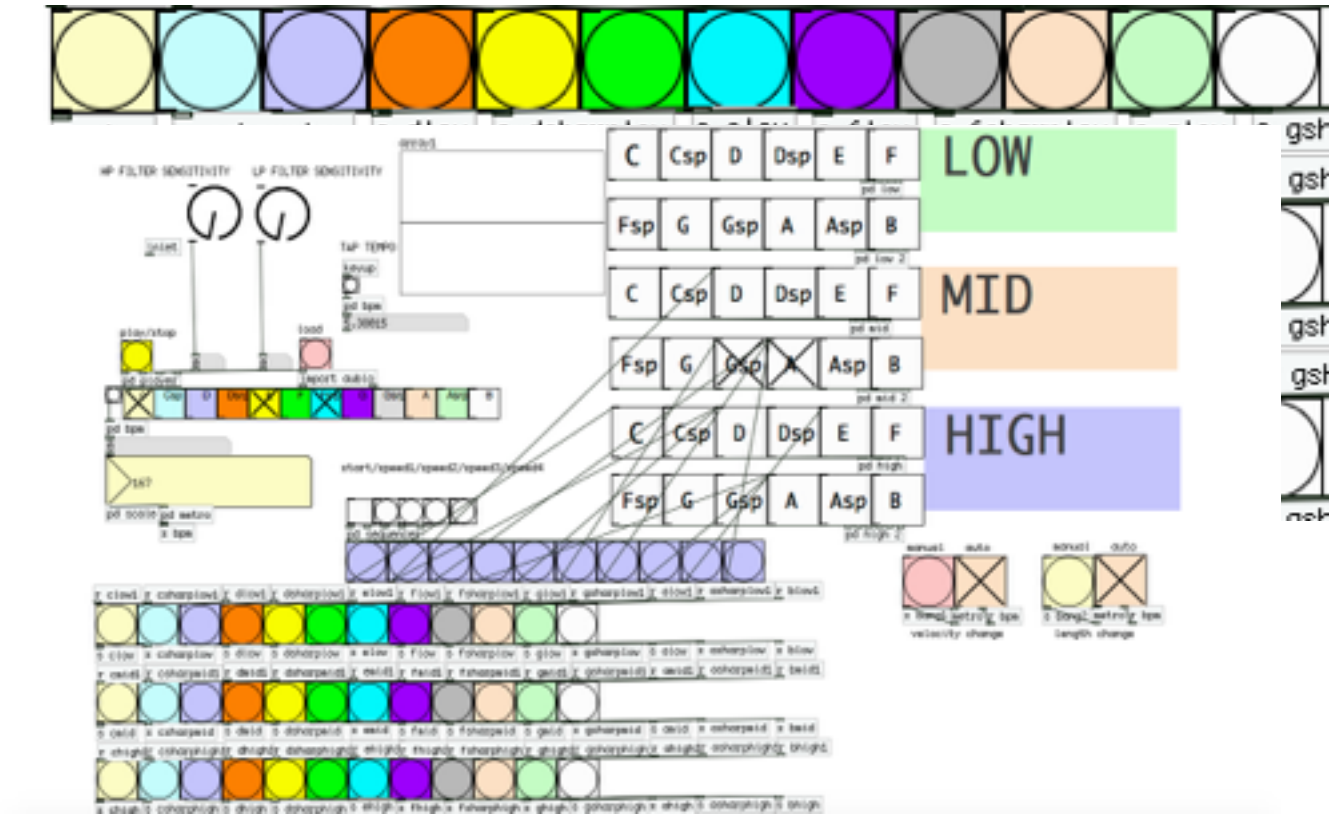
Metronome modification

// This allows the melody to become more complex and interesting. the messages create 16ths and 8ths which can be triggered but the user.



The Matrix

// This idea stemmed from the Launchpad, the visual feedback it creates shows what notes are being triggered live.



Visuals

// Following Microsoft's visual design guidelines I've made use of bright colours.

// The sequencer connects to the large toggle notes at the top right of the window, using send and receives to send notes. the two toggles on bottom left sequence velocity and length events.

You can see the patch in action in the video attached.