Making Hit Pop Songs

Adventures with Sonic Pi

Making A Hit Song

Possible goal of the project

- Create a song which will go straight to Number 1 in the Pop Charts
- Must be an original composition to avoid copyright problems

Any musical talent?

- My day job is as an actuary (working in a finance department of an insurance company) and started my programming life with the BBC Micro computer some time ago.
- I've been using Raspberry Pi's since they first came out, and attended a few Jams.
- Musical talents are as follows:
  - Vocals: very iffy
  - Drums: whilst I'm good at hitting things, not so good at doing it in time
  - Guitar: I can play Guitar Hero on easy, but that's my limit
  - Keyboards and other instruments: very little skill
- Overall score 1 / 10

A More Realistic Goal

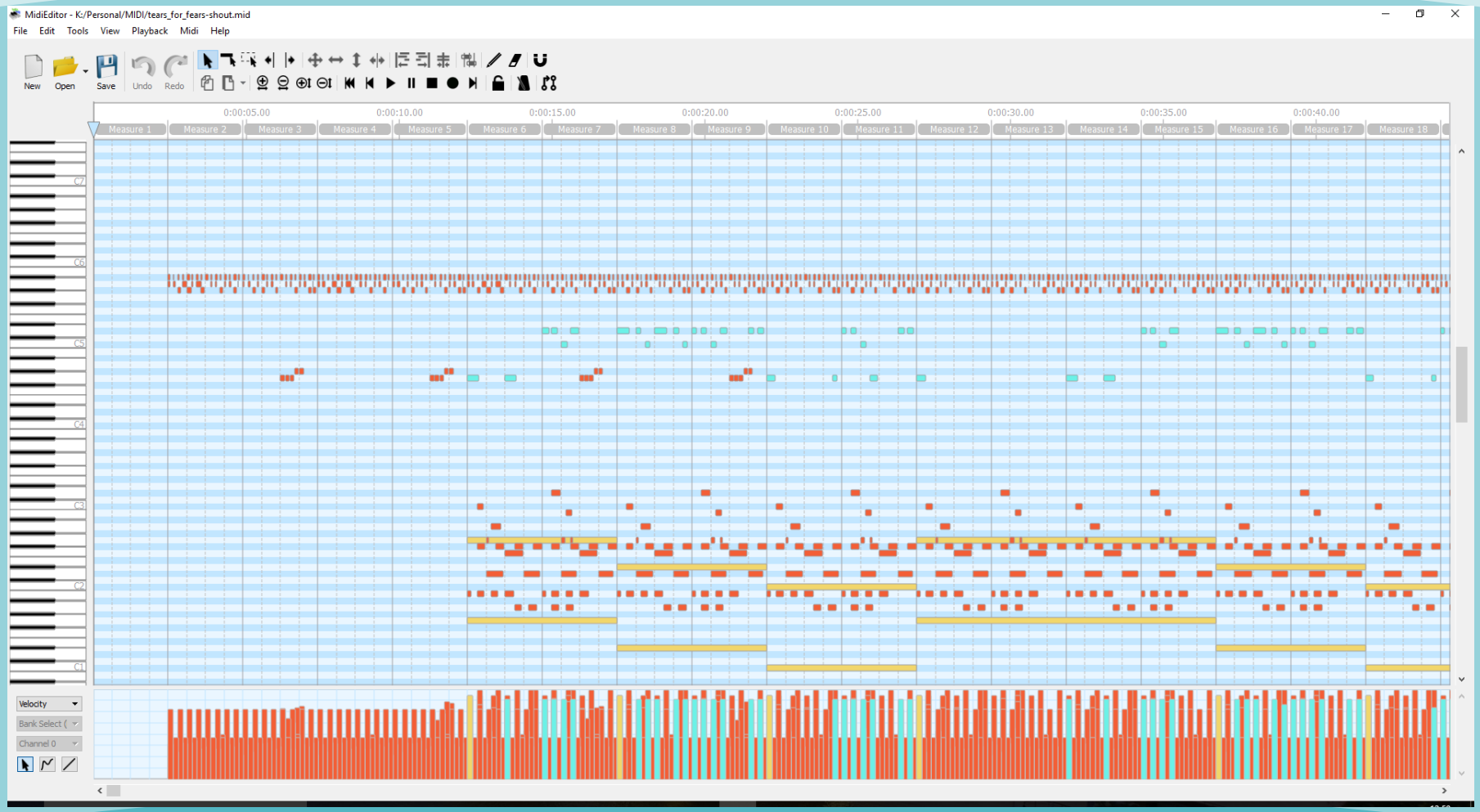
Given lack of musical talent, a more realistic goal of the project is to:

- Use programming to randomly generate sounds …
- … but needs to be tuneful, not a horrible mix of random notes
- Create a song which doesn’t sound too bad so that you won’t immediately want to switch it off!

Where Do We Start ?

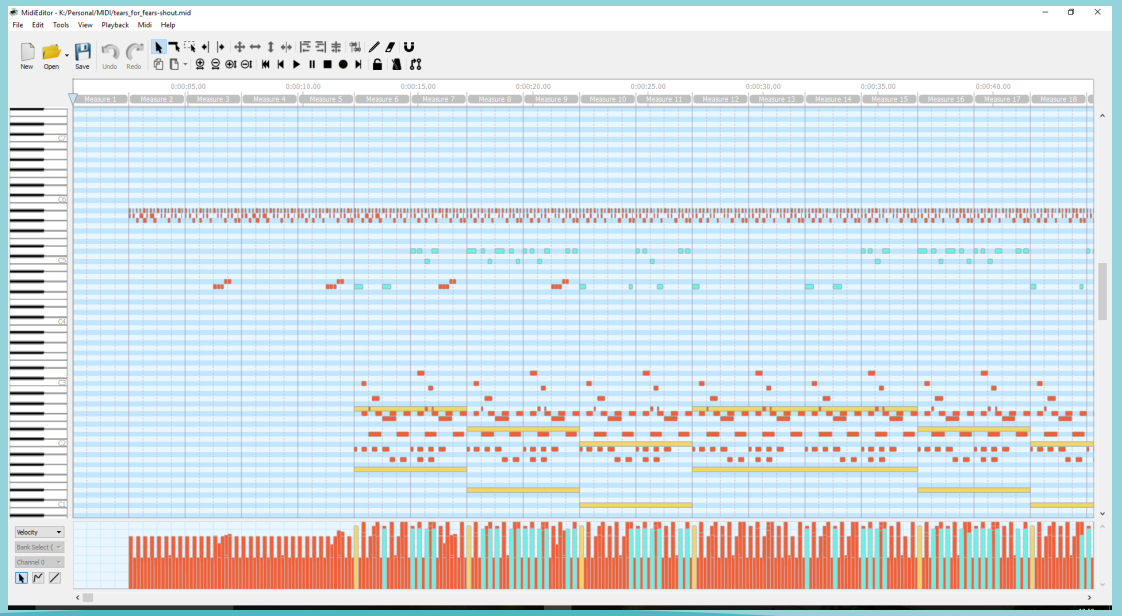
- I’ve tried to analyse some hit songs simply by listening to them, but working out what is going on can be tricky.
- On the internet, you can find MIDI files for songs (I’m not sure where these stand in terms of copyright). These MIDI files can be used to generate the “backing track” for a particular song.
- So I’ve cheated by using at these MIDI files to look at the structure of a particular song.
- I used a program called MidiEditor to view the MIDI file, as follows:

MIDI Editor


MidiEditor - K:/Personal/MIDI/tears_for_fears-shout.mid
File Edit Tools View Playback Midi Help
New Open Save Undo Redo
0:00:05.00 0:00:10.00 0:00:15.00 0:00:20.00 0:00:25.00 0:00:30.00 0:00:35.00 0:00:40.00
Measure 1 Measure 2 Measure 3 Measure 4 Measure 5 Measure 6 Measure 7 Measure 8 Measure 9 Measure 10 Measure 11 Measure 12 Measure 13 Measure 14 Measure 15 Measure 16 Measure 17 Measure 18
C7 C6 C5 C4 C3 C2 C1
Velocity
Bank Select (
Channel 0

What does this tell us?

- Some sort of timing (horizontal axis)
- Different notes are played at different times (vertical axis)
- Multiple instruments doing different things (different colours)
- Recurring patterns of notes
- Different stages in the song when patterns change



Project tasklist (first go)

1. Want to set up some sort of timing within Sonic Pi
2. Set up an instrument to generate a drum beat (aka drum machine)
3. Set up another instrument to generate a pattern of low notes (aka bass line)
4. Set up another instrument to generate a pattern of mid notes (aka tune)

How can this be coded in Sonic Pi?

My initial solution:

- Recurring patterns = use live_loops
- Multiple instruments = multiple live_loops
- Need a global way to keep everything in time:
  - Cue :timer, in one loop
  - Sync :timer, in all other loops

Music theory: timing

- Music notation expressed notes in a bar
- A bar is made up of a fixed number of beats (which are determined by the time signature)
- For example, a bar could be made up of 4 beats, so when counting we have:
  - Bar 1: beat 1, beat 2, beat 3, beat4
  - then
  - Bar 2: beat 1, etc.

Timing in Sonic Pi?

My solution:

- Create a loop to deal with the timing
- Create a counter to keep track of the beats
- Create another counter to keep track of the bars
- To allow for notes which are “off-beat”, I’m actually counting each half beat
- So loop every half beat, and define a time interval between each half beat:
  - For example, if the interval between half beats is 0.25 seconds
  - Then that means 2 beats per second = 120 beats per minute
- Other song elements need to know where they are in the song, so create global counters by prefixing variable names with $ (e.g. $beatcounter)

Project tasklist (first go)

1. Want to set up some sort of timing within Sonic Pi ✓
2. Set up an instrument to generate a drum beat (aka drum machine)
3. Set up another instrument to generate a pattern of low notes (aka bass line)
4. Set up another instrument to generate a pattern of mid notes (aka tune)

Developing a drum machine

- Features of a simple drum machine:
  - Pattern of when parts of the drum kit are hit (and reproduced electronically)
  - Pattern is repeated from one bar to the next
- I started with a simple drum kit with two parts:
  - Cymbal
  - Snare drum

The Drum Machine

- Express the patterns for the cymbal and snare parts into:

  0 = do nothing, 1 = play sample

Half beat	1	2	3	4	5	6	7	8
Cymbal	0	0	1	0	0	0	1	0
Snare	1	0	0	0	1	0	0	0

- In Sonic Pi, create an array with these values and then read the value for the particular half beat

Project tasklist (first go)

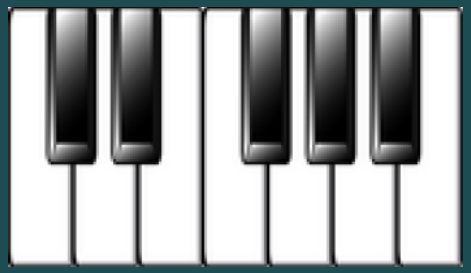
1. Want to set up some sort of timing within Sonic Pi ✓
2. Set up an instrument to generate a drum beat (aka drum machine) ✓
3. Set up another instrument to generate a pattern of low notes (aka bass line)
4. Set up another instrument to generate a pattern of mid notes (aka tune)

Developing a bass line

- Could just try to play any note at random
- But you quickly realise that this can sound really odd
- So need some structure to help us develop the bass line …

Music theory: scales

- Music theory tells us that there are 12 possible notes in what is called an octave.
- These are the white and black keys on a piano:



- Each of these 12 notes is given a name:
  - C, C#, D, D#, E, F, F#, G, G#, A, A#, B
- A piano has keys which repeat this pattern of 12 notes multiple times
- Each of these 12 notes has a MIDI number, such as the note C is number 60

Music theory: key

- Music theory talks about choosing a set of 7 out of the 12 notes
- This set of 7 notes sounds better together than playing all 12 notes
- In Music Theory, you talk about this set of 7 notes as a key.
- The key is defined by the first note and then the rule to choose the other 6 notes.
- In Music theory, the key is made up of the first note followed by (for example) “major” or “minor” – e.g. C Major

Keys in Sonic Pi

- We set the base note in the scale as a MIDI number (e.g. 60 = C)
- We create the pattern to convert the “major” or “minor” key into a set of numbers to be added to the base note – for example:

Pattern to add	**0**	**2**	**4**	**5**	**7**	**9**	**11**	**12**
MIDI note	60	62	64	65	67	69	71	72
Actual note	C	D	E	F	G	A	B	C

Developing a bass line, contd

- So instead of just trying to play any note at random, we could limit the choice of note to one of the 7 notes in the chosen key
- But you quickly realise that whilst this sounds better, we are not there yet!
- So need more structure to help us develop the bass line …

Music theory: progressions

- Whilst not strictly music theory, there are some rules as to what are “good” progressions of notes
- For example, if playing the base note of the key, then you can play any note in the key next.
- However, if playing the 2nd note of a major key, then you should follow this by the base note, 5th note or 7th note in the key

Progression rules in Sonic Pi

- We can define the progression rules mathematically in Sonic Pi
- I've done this by creating a set of arrays for each note (which doesn't look to be very efficient):

Note	1st (base)	2nd	3rd	4th	5th	6th	7th
If major key	2,3,4, 5,6,7	1,5,7	1,2,4, 6	1,2,3, 5,7	1,6	1,2,3, 4,5	1,3
If minor key	2,3,4, 5,6,7	1,3,5, 7	1,4,6, 7	1,5,7	1,6	1,3,4, 5,7	1

Project tasklist (first go)

1. Want to set up some sort of timing within Sonic Pi ✓
2. Set up an instrument to generate a drum beat (aka drum machine) ✓
3. Set up another instrument to generate a pattern of low notes (aka bass line) ✓
4. Set up another instrument to generate a pattern of mid notes (aka tune)

Developing a tune

- The development of the bass line tells us that we need some structure for choosing notes for the tune …

Music theory: triads

- Music theory takes each of the 7 notes in the key and adds two more notes to create a triad (or chord)
- We can use these three notes to create a tune to play along with the bass line
- There are a few ways to create these triads for each note, which are also referred to as “major” or “minor”
- The progression rules can be extended to cover the triads which work together, and for a major key, this gives us one triad to worry about per note

Triads in Sonic Pi

- We can define the rules to create the triads mathematically in Sonic Pi
- I've done this by creating an array for a major triad, and another for a minor triad:

Note	Triad type
Base	Major
2nd	Minor
3rd	Minor
4th	Major
5th	Major
6th	Minor
7th	Minor

Note	1	2	3
If major triad	0	+4	+7
If minor triad	0	+3	+7

Project tasklist (first go)

1. Want to set up some sort of timing within Sonic Pi ✓
2. Set up an instrument to generate a drum beat (aka drum machine) ✓
3. Set up another instrument to generate a pattern of low notes (aka bass line) ✓
4. Set up another instrument to generate a pattern of mid notes (aka tune) ✓

So let’s hear the result …

Any questions ???

Check out the code, together with comments and music theory at:
www.github.com/hammaman/sonicpi