seigaiha

A simple norns script that plays arpeggios while drawing a seigaiha pattern on the screen.

recommended listening

Set the clock to between 80 and 120 bpm, add some reverb and maybe play a field recording in the norns tape.

how it works

The step function uses the MusicUtil library to select a chord at random from the pentatonic scale with a random root note. It then waits until the next sync quantum of 4 beats before it plays the notes in that chord using the polyperc engine with a half beat pause between each note. The process is then repeated with a new chord played each time.

To draw the pattern a table 'bows' is used which can contain up to 36 rainbows to draw on the screen. The step function finds an empty slot when choosing a new chord and as the notes are played the value of that rainbow in the bows table increases from 0 to 1. The redraw function loops through the table and calculates the x and y coordinates for each rainbow before drawing them as 4 arcs.

extending the script

The following are a few suggestions to add functionality to the script. The awake script does a lot of these and is a great resource.

- add some engine options (release, cutoff, pw etc) and the scale as params so they can be altered and saved as presets
- change how fast the notes are played or how long the pause is after each chord. Add rests, a chance of not playing a note - make it more musical (or less)!
- select a handful of root notes when the script starts and only use those or use the sequins library to repeat a pattern
- add some softcut effects like the awake halfsecond delay
- output midi or crow, switch in a different engine or write your own
- draw a different pattern by exploring the screen library - lines, circles, the world is your geometric oyster

```
01: -- seigaiha
02: engine.name = 'PolyPerc'
03: MusicUtil = require "musicutil"
04: scale = MusicUtil.SCALES[11]
05:
06: bows = {}
07: counter = 0
08: max = 36
09: root_note = 43
10:
11: function init()
12:   engine.release(4)
13:   engine.pw(0.6)
14:   clock.run(step)
15: end
16:
17: function step()
18:   while true do
19:     -- if screen is full, restart
20:     if counter == max then
21:       bows = {}
22:       counter = 0
23:     end
24:
25:     -- find an unused rainbow
26:     bow = nil
27:     repeat
28:       bow = math.random(max) - 1
29:     until bows[bow] == nil
30:     bows[bow] = 0
31:     counter = counter + 1
32:
33:     -- select a random chord from the scale and play through the notes
34:     chord_idx = math.random(#scale.chords[1])
35:     chord = MusicUtil.CHORDS[scale.chords[1][chord_idx]]
36:     chord_root = root_note + scale.intervals[math.random(#scale.intervals)]
37:     clock.sync(8)
```

```
38:     for i=1, #chord.intervals do
39:       freq = MusicUtil.note_num_to_freq(chord_root + chord.intervals[i])
40:       engine.hz(freq)
41:
42:       bows[bow] = bows[bow] + (1 / #chord.intervals)
43:       redraw()
44:       clock.sync(0.5)
45:     end
46:   end
47: end
48:
49: function redraw()
50:   radius = 10
51:   margin = 2
52:   screen.clear()
53:   screen.aa(1)
54:
55:   for b = 0, max - 1 do
56:     if bows[b] ~= nil then
57:       x = (b % 6 * ((radius * 2) + 1)) + margin
58:       y = ((math.floor(b / 6) + 1) * (radius - 1) + margin)
59:
60:       if y % 2 > 0 then x = x - radius - 1 end
61:
62:       -- loop for first rainbow on alt rows that repeats at start and end of row
63:       while true do
64:         screen.move(x, y)
65:         for i=0, radius / 3 do
66:           screen.arc(x + radius, y, radius - (i * 2), math.pi, math.pi + (math.pi * bows[b]))
67:           screen.stroke()
68:         end
69:         if x < 0 then x = 128 - radius else break end
70:       end
71:     end
72:   end
73:
74:   screen.update()
75: end
```