Live Coding

How to

Live Code Manifesto from Toplap

We demand:

- Give us access to the performer's mind, to the whole human instrument.
- Obscurantism is dangerous. Show us your screens.
- Programs are instruments that can change themselves
- The program is to be transcended - Artificial language is the way.
- Code should be seen as well as heard, underlying algorithms viewed as well as their visual outcome.
- Live coding is not about tools. Algorithms are thoughts. Chainsaws are tools. That's why algorithms are sometimes harder to notice than chainsaws.

Live Code Manifesto from Toplap

We recognise continuums of interaction and profundity, but prefer:

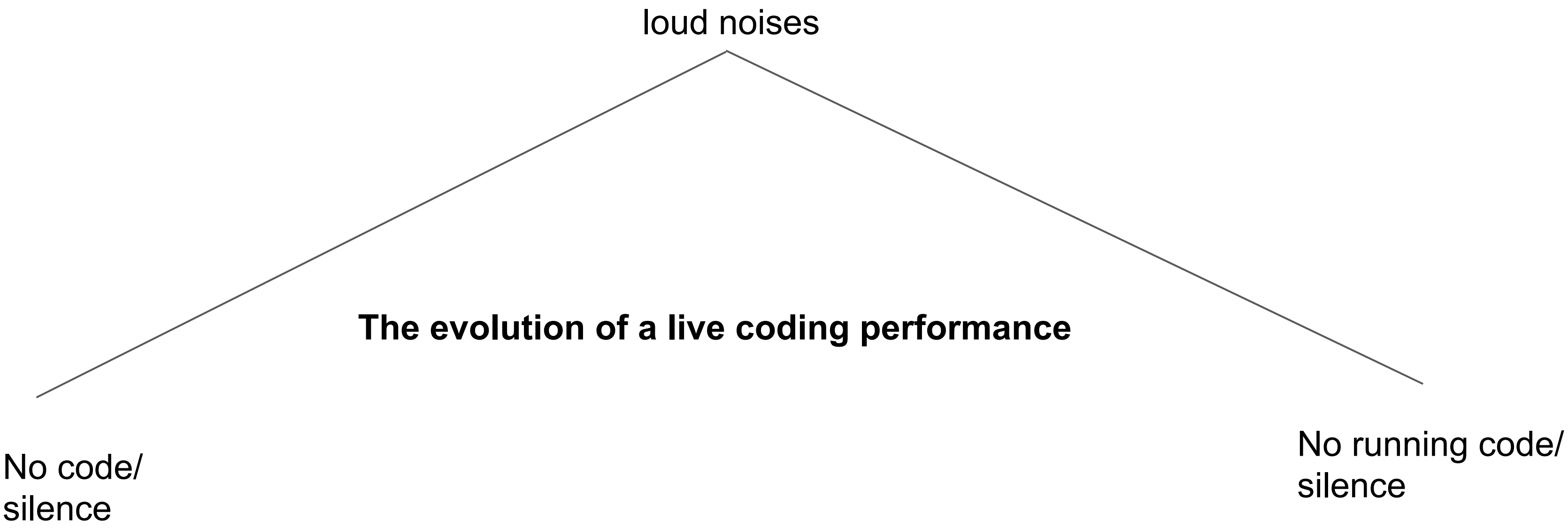
- Insight into algorithms
- The skillful extemporisation of algorithm as an expressive/impressive display of mental dexterity
- No backup (minidisc, DVD, safety net computer)

Live Code Manifesto from Toplap

We acknowledge that:

- It is not necessary for a lay audience to understand the code to appreciate it, much as it is not necessary to know how to play guitar in order to appreciate watching a guitar performance.
- Live coding may be accompanied by an impressive display of manual dexterity and the glorification of the typing interface.
- Performance involves continuums of interaction, covering perhaps the scope of controls with respect to the parameter space of the artwork, or gestural content, particularly directness of expressive detail. Whilst the traditional haptic rate timing deviations of expressivity in instrumental music are not approximated in code, why repeat the past? No doubt the writing of code and expression of thought will develop its own nuances and customs.

Sculpting an Arc



loud noises

The evolution of a live coding performance

No code/
silence

No running code/
silence

Key techniques

sample

live loops+parallelism

comments

Randomness (pitch, time, amp)

Scales + ticks

effects/filters

Syncopation

bpm

Samples and synths

live loops

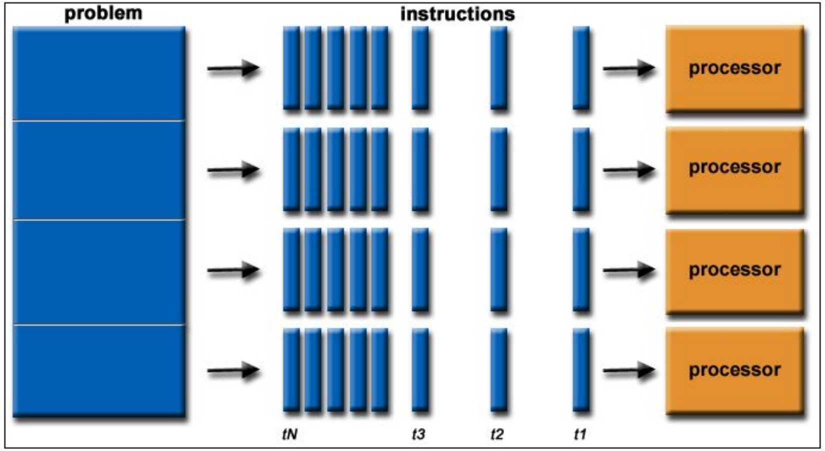
```
live_loop :bassDrumLine do
    sample :bd_haus, amp: 1.5
    sleep 1
    sample :sn_dolf
    sleep 1
end
live_loop :hihatLine do
    sample  :drum_cymbal_closed
    sleep 0.25
end
```

Threads and parallel computing

Computations that use multi-processor computers and/or several independent computers interconnected in some way, working together on a common task.

Each live loops is working together to make music.

Key problem - how to they stay in sync?



sync’ing live loops

```
live_loop :bassDrumLine do
   sample :bd_haus, amp: 1.5
   sleep 1
   sample :sn_dolf
   sleep 1
end
live_loop :hihatLine do
   sync "/live_loop/bassDrumLine"
   sample  :drum_cymbal_closed
   sleep 0.25
end
```

sync’ing live loops

live_loop restarts when it completes its current iteration.

This can lead to some strange behavior...

```
live_loop :testSync do
   4.times do
      sample :drum_bass_hard
      sleep 0.25
   end
end
live_loop :testSync2 do
   sync :testSync
   4.times do
      sample :drum_cowbell
      sleep 0.25
   end
end
```

comments in live programming

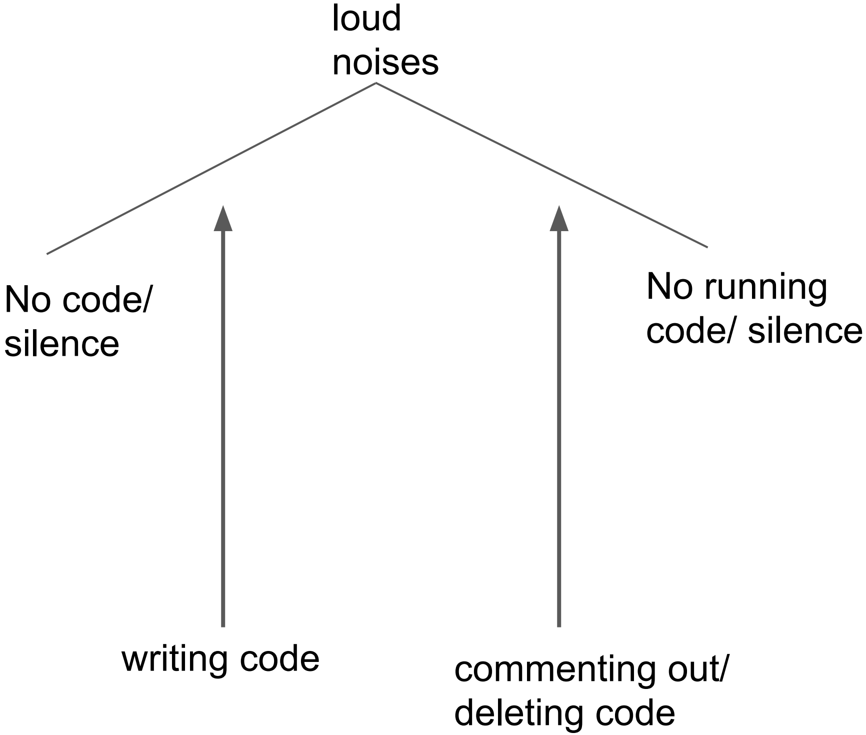
Commenting code “out” is generally bad practice.

In Live Coding, it is a key part of the performance. This is how you construct the second half of your arc.

Be careful when commenting out `sleep` commands

Commenting out a `live_loop` does not end it

M+○/ to (un)comment



Randomness

As with visual arts, an element of randomness helps add complexity.

Levels of randomness:

micro vs macro variation

Can be applied to:

pitch, amp, time, any parameter

```
.choose/choose()
.pick()
rrand()
rrand_i()
dice
```

Randomness (pitch and time) in Sonic Pi

```
 live_loop :randomMelodia do
   use_synth :chipbass
   play [60, 75, 67, 70].choose
   sleep 0.25
end
live_loop :randomSleep do
   sample :elec_blip, amp: 2
   sleep [0.25, 0.5, 0.75].choose
end
```

Randomness (parameters) in Sonic Pi

```
live_loop :trance do
    use_synth :tb303
    play [:C2, :C3].choose, cutoff: rrand(50, 120), release: 0.25
    sleep 0.25
end

live_loop :hihat do
    sample :drum_cymbal_closed, amp: rrand(0,2)
    sleep 0.25
end
```

Syncopation

```
live_loop :trance do
    use_synth :tb303
    play [:C2, :C3].choose, cutoff: rrand(50, 120), release: 0.25
    sleep 0.25
end

live_loop :hihat do
    sync :trance do
    3.times do
        sample :drum_cymbal_closed, amp: rrand(0,2)
        sleep 0.333
    end
end
```

FXS

```
live_loop :trance do
    use_synth :tb303
    with_fx :distortion do
        play [:C2, :C3].choose, cutoff: rrand(50, 120), release: 0.25
        sleep 0.25
    end
end

live_loop :hihat do
    sync :trance do
    3.times do
        sample :drum_cymbal_closed, amp: rrand(0,2)
        sleep 0.333
    end
end
```

use_bpm

Ambient 50–100 BPM

Hip-hop 70–95 BPM

Deep house 110–130 BPM

Trance / Techno 130–145 BPM

Hard dance/hardcore 145–170 BPM

Drum and bass 160–180 BPM

Build an arc with these techniques

sample

live loops+parallelism

comments

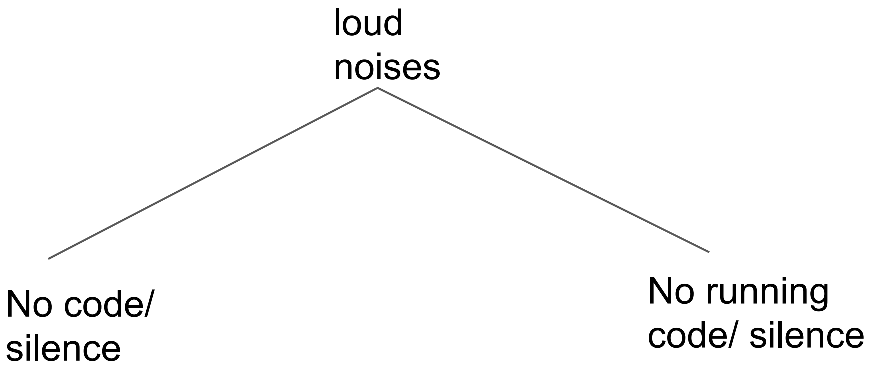
Randomness (pitch, time, amp)

Scales + ticks

effects/filters

Syncopation

bpm



loud noises

No code/ silence

No running code/ silence