

**LIVE  
LODGING  
ONATOON**



**AMAGİTAKAYOSI**



@amagitakayosi

 Hatena

 Kyoto.js



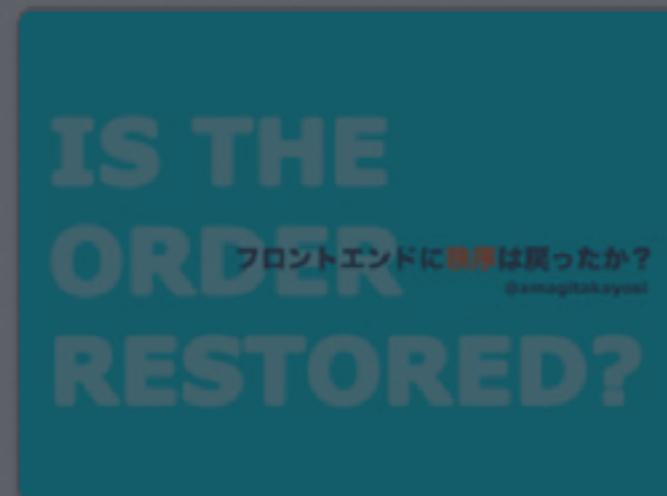
# たまにフロントエンドっぽい発表します

Talks by fand/amagitakayosi



Reactで学ぶ！いまどきの  
Web開発

Mar 18, 2017 by  
fand/amagitakayosi



フロントエンドに秩序は  
戻ったか？

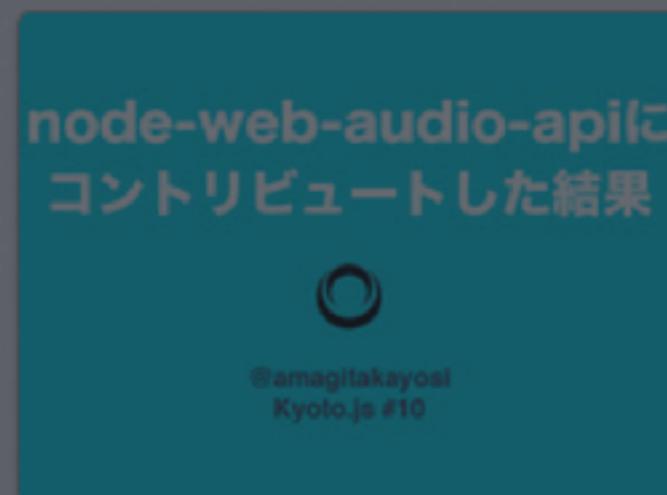
Jan 21, 2017 by fand/amagitakayosi



はてな教科書JavaScript編  
Oct 22, 2016 by fand/amagitakayosi



Spectacleについて  
Oct 22, 2016 by fand/amagitakayosi



node-web-audio-apiにコ  
ントリビュートした結果  
May 8, 2016 by fand/amagitakayosi



Designing in React: Why  
and How  
Mar 5, 2016 by fand/amagitakayosi

# Live Coding...?



ライブ・コーディング

読み：らいぶこーでいんぐ

英語：live coding

別名：

意味：

ライブコーディングとは、その場でコーディングすること。

プレゼンテーションや勉強会?で行われることがあります。

その場の雰囲気や上級者の書き方などを生で感じることができます。

今日はAtomを使って  
ライブコーディングで  
DJ/VJする方法を  
紹介します！！！！

**ついでに僕が作った  
ライブコーディング用  
Atomパッケージを  
自慢します！！！**

# Systems for Live Coding

# 音楽用は沢山ある

- SuperCollider
- CoffeeCollider
- Sonic Pi
- TidalCycles

# SuperCollider



- ・ 元祖ライブコーディング環境
- ・ Smalltalkっぽい文法
- ・ 他のライブコーディング用ソフトも  
SuperColliderを呼び出してたりする

# Sonic Pi



- ・ Rubyで書く
- ・ IDEがしっかりしてる
- ・ MinecraftのMODとして使うこともできる  
(?????????)

# TidalCycles



- ・ Haskellで書く
- ・ めっちゃ短く書ける
- ・ AtomやVS Codeのパッケージとして  
実装されている

# TidalCyclesやってます

@amagitakayosi

TidalCycles、めっちゃ簡単に  
Breakcore/Mashcore作れてテンション上がっ  
てきた

```
01.tidal 03.tidal 02.tidal
1 cps (120 / 120)
2
3 hush
4
5 d2 $ s (samples "amen*8" (irand 12))
6   # crush 4.5
7   # cut 1
8   # up (rand * 10)
9
10 d1
11   $ s (samples "anime*8" (irand 22))
12   # cut 1
13   # up 2
14   # gain 0.9
15   # crush 3
16
17 d3 $ s "808bd:0(5, 8)" # crush 3
18
0:34 ■■■ d4 $ s "808bd:1" # crush 3
```

20:26 - 2017年6月23日

50件のリツイート 134件のいいね



Q 1

50



134



kayosi

GLSL and TidalCycles. GLSL  
[github.io/webgl-study/](https://github.io/webgl-study/)

[akayosi.hatenablog.com](http://akayosi.hatenablog.com)

9月に登録

# VJ用はあんまりない.....

- Fluxus
- Quil
- **GLSL**



<https://www.flickr.com/photos/hellocatfood/34531567721>

# GLSLはシェーダーの一種

- ・ 3DCGで陰影を計算するためのプログラム
  - ・ HLSL (DirectX)
  - ・ Metal (macOS)
  - ・ GLSL (OpenGL)

# 言語の特徴

- ・ C-likeな文法
- ・ 行列計算に便利なビルトイン関数
  - ・ float dot(vec2 a, vec2 b) とか

# シェーダーアート

- ・ GLSLはGPUパワーをフル活用できる
- ・ 「GLSLだけで複雑なグラフィック表現を描画できるのでは？」
- ・ 一部の人々が大興奮

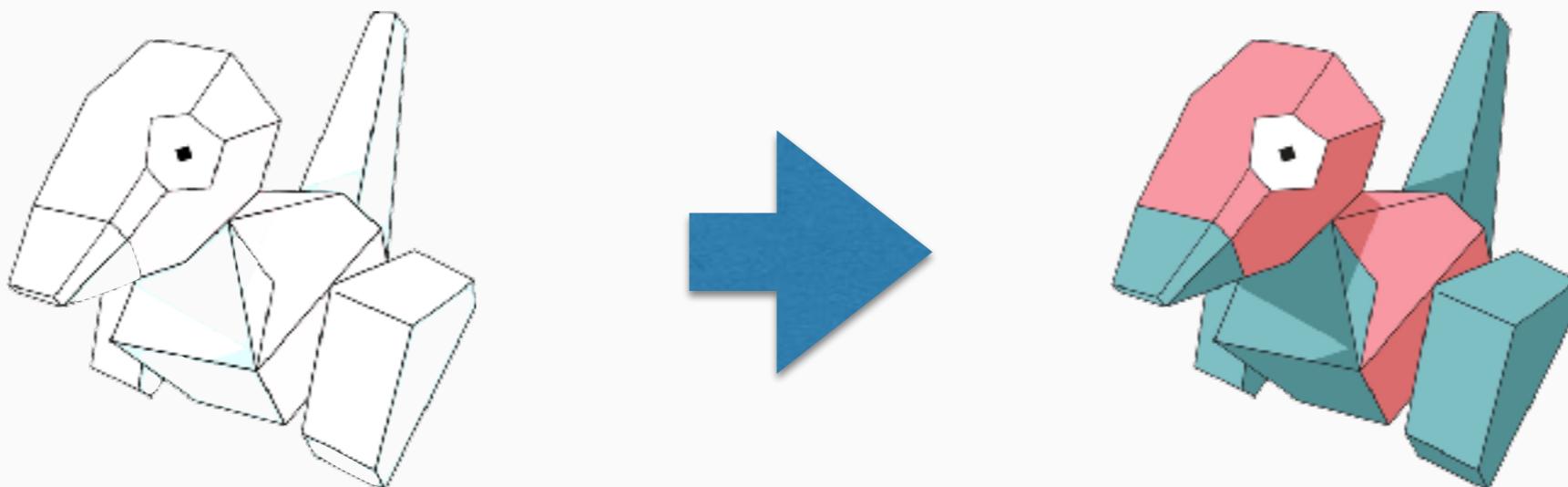
参考: 4k intro “Elevated”



<https://www.youtube.com/watch?v=jB0vBmiTr6o>

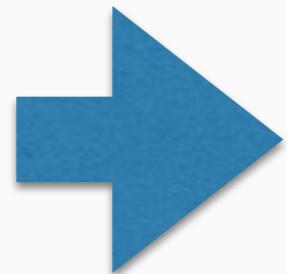
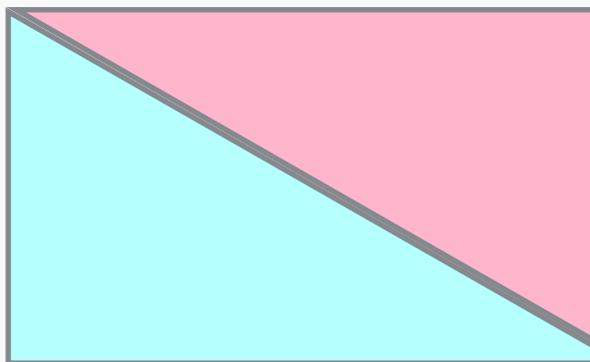
# 普通の3Dグラフィックス

- ・ 数千～数万ポリゴン + シェーダー



# 板ポリ

- ・ 三角形2個 + シェーダー

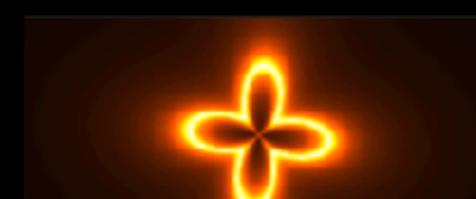
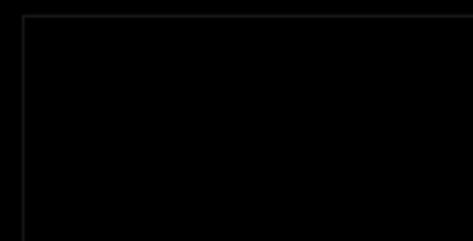
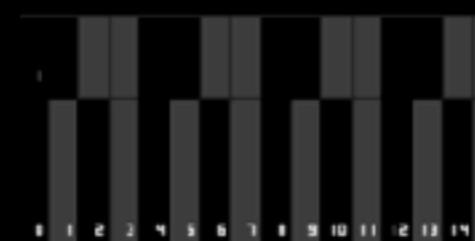
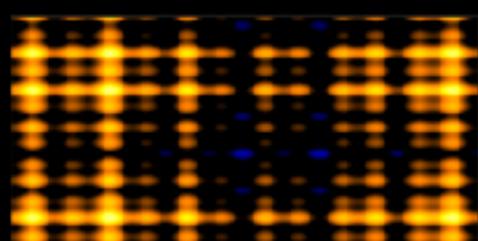
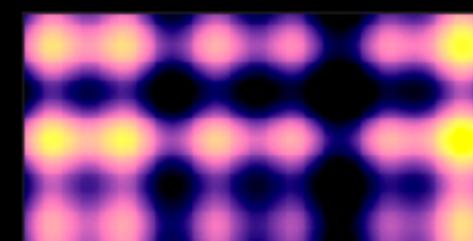
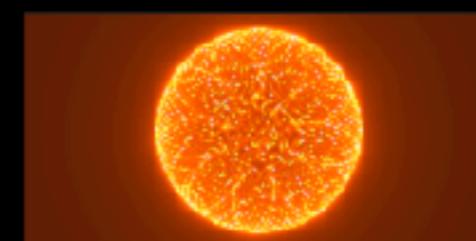
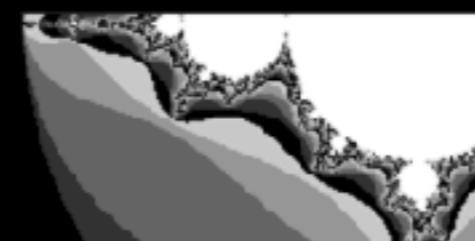
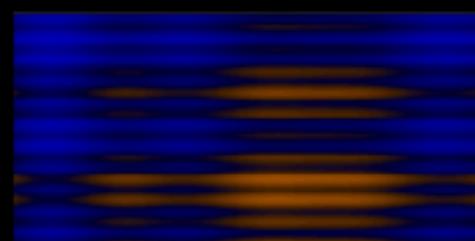
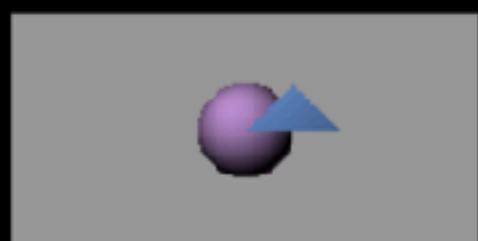
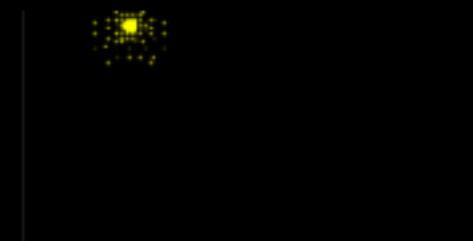
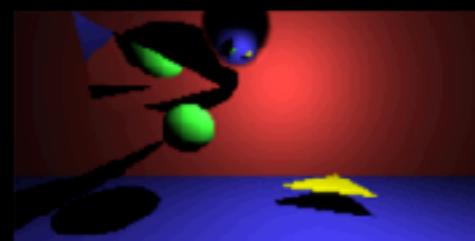
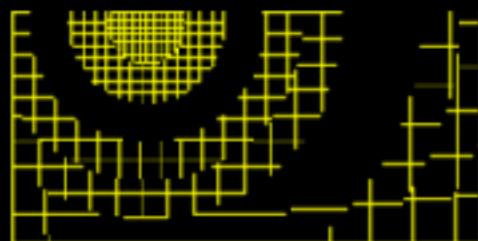
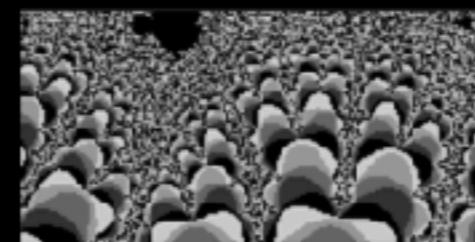
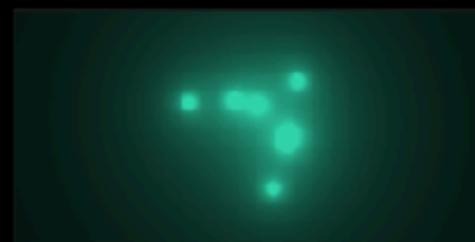
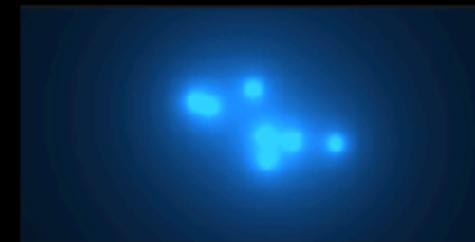
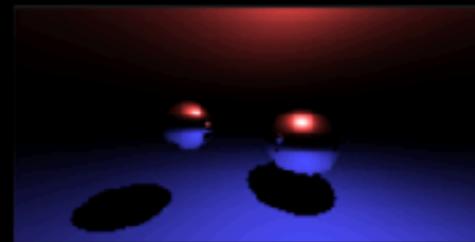


# GLSLお試し環境

- ・ GLSL Sandbox
  - ・ 気軽にGLSLを始められる
- ・ Shadertoy
  - ・ ガチ勢の力作がおおい

# GLSL Sandbox

Create new effect! / [github](#) / gallery by [@thevaw](#) and [@feiss](#) / editor by [@mrdoob](#), [@mrkishi](#), [@p01](#), [@alteredq](#), [@kusmabite](#) and [@emackey](#)



# ライブコーディング環境

- GLSL Sandbox
- Kodelife
- openFrameworksで自作してる人とか

circle.frag

circle2.frag

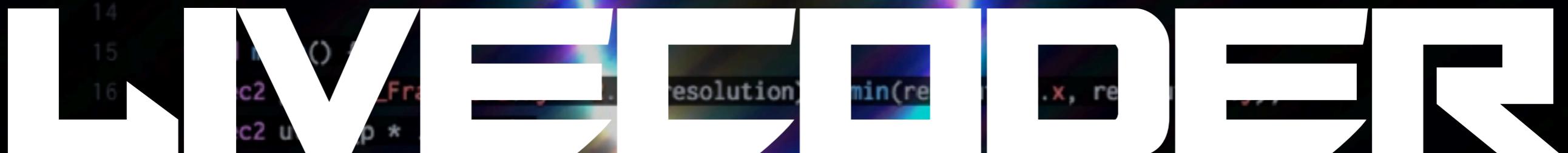
```
precision mediump float;
uniform float time;
uniform vec2 resolution;
uniform sampler2D backbuffer;
uniform sampler2D spectrum;
// You can use GLSL Sandbox style uniform variables.

vec2 rotate(in vec2 p, in float t) {
    return mat2(sin(t), -cos(t), cos(t), -sin(t)) * p;
}

// Hi there!
// This is a demo of glsl-livecoder!

float h = texture2D(spectrum, vec2(abs(p.x * 0.5), .0)).r;
float m = texture2D(spectrum, vec2(abs(p.x * 0.3), .0)).r;
float l = texture2D(spectrum, vec2(abs(p.x * 0.1), .0)).r;

gl_FragColor = vec4(h, m, l, 1.) * (.05 / abs(length(p) - .3));
}
```



# Atomパッケージとして実装

- ・ 自分の設定でエディタを使える
- ・ 既存のGLSL用パッケージも使える
  - ・ linter-glsl
  - ・ autocomplete-glsl

# 機能一覧

- ・ GLSL Sandboxの上位互換
- ・ 音声入力
- ・ MIDI入力
- ・ 画像/動画のロード

# GLSL Sandboxの上位互換

- ・ GLSL Sandboxのコードがそのまま動く
- ・ 画面サイズ、マウス位置、経過時間などを利用できるように

# 音声入力

- ・ Web Audio APIで取得
  - ・ 波形、ボリューム、スペクトラム
- ・ TidalCycles等と組み合わせると  
一人で DJ + VJ 可能

# MIDI入力

- ・ Web MIDI APIで取得
  - ・ MIDIイベント、ノート番号
- ・ MIDIコントローラーで  
VJエフェクトを制御できる！

# 画像ロード機能の実装案

- ・ 設定画面からロードする？
- ・ コマンドで設定する？
- ・ ロード用UIを作る？
- ・ 設定ファイルを作成する！

# .liverc

- ・ 作業ディレクトリに .liverc を設置
- ・ JSON5形式で画像のパスを書く
  - ・ ローカルファイルでも、URLでも可
  - ・ Electron最高～
- ・ chokidarで変更を監視

**DEMO**

# 課題

- ・ 実装が一部めっちゃ雑
- ・ パフォーマンス

# めちゃくちゃ雑なようす

```
this.style = document.createElement('style');
this.style.innerHTML =
  'body, atom-workspace, .header *, .footer *, atom-workspace-axis :not(.cursor):not(
    background: transparent !important;
    border: none !important;
    text-shadow: 0 1px 1px black;
    box-shadow: none !important;
  )
.line > span {
  background: rgba(0,0,0,0.7) !important;
}
.cursor {
  width: 0 !important;
  box-shadow: 0 0 3px #0FF !important;
  border-left: 4px solid #0FF !important;
}
autocomplete-suggestion-list, atom-overlay {
  background: rgba(0, 0, 0, 0.9) !important;
```

# めちゃくちゃ雑なようす

```
yle');

 eter *, atom-workspace-axis :not(.cursor):not(autocomplete-suggestion-list):not(atom-over)
;

cant;

;

cant;

overlay {
  portant;
```

# パフォーマンス

- WebGL canvasの上に要素を置くと  
極端に重くなる、らしい
- autocomplete-plus自体が重い
  - こっちの方がまだどうにかなる

# autocomplete-plusのパフォーマンス

- ・ issue立ってる (<https://github.com/atom/autocomplete-plus/issues/839>)
- ・ etchコンポーネントで書き直そうぜ、との事
  - ・ Atomチームが開発した  
VirtualDOMコンポーネントライブラリ
  - ・ ちょっと試したけどムズくて放置してる……

**ENDE**  
**ERGIE**  
**FINGER**