デザインとプログラミング
p5.js Libraryを使う 1
p5.soundでサウンドプログラミング（2）

2020年11月27日

慶應義塾大学 総合政策学部・環境情報学部
田所淳

今後の授業スケジュール

今後の授業スケジュール

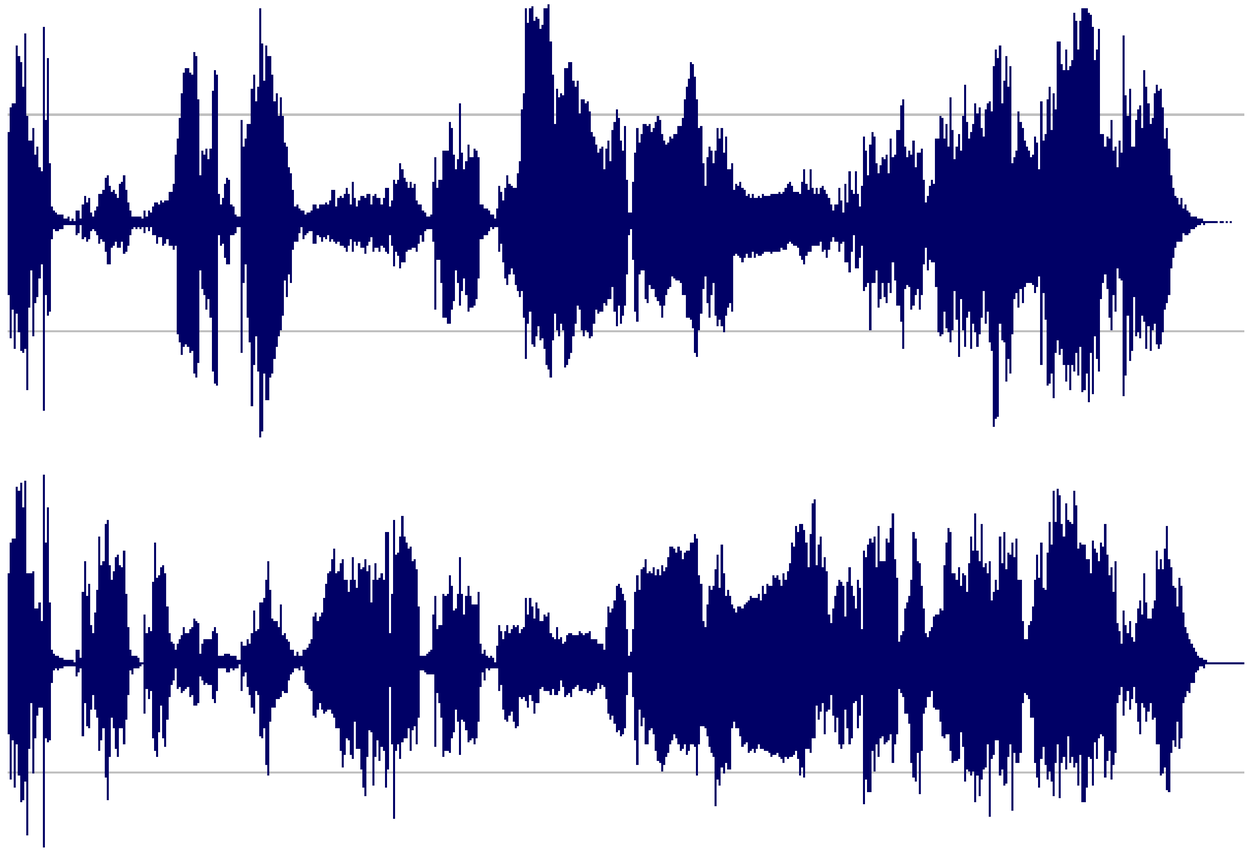
- 今後のスケジュールは以下の通りで進めていきます

- 11月27日（本日）サウンドプログラミング(2)、最終課題出題
- 12月4日 最終課題制作のヒント(オンデマンド) + 制作相談会（Zoomリアルタイム）
- 12月11日 最終課題発表会(1)（Zoomリアルタイム）※希望者のみ
- 12月18日 最終課題発表会(2)（Zoomリアルタイム）※希望者のみ
- 12月18日 最終課題最終締切(23:59)

音響の視覚化

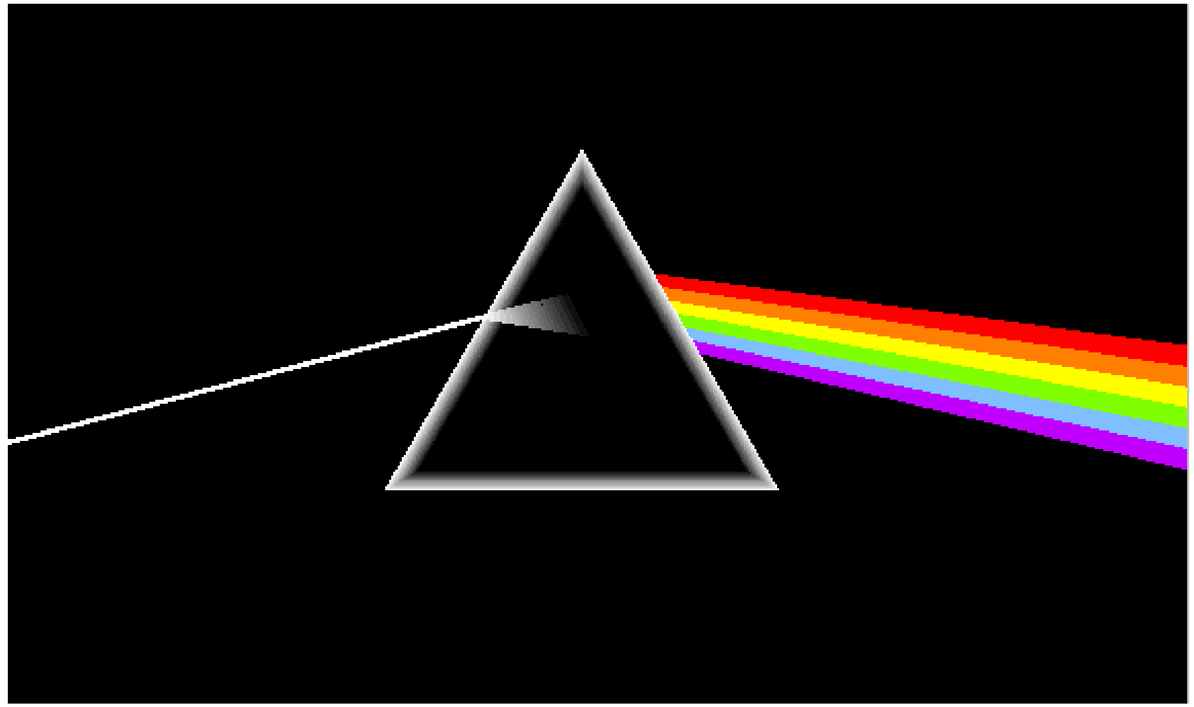
Audio into Visual.

- 音を視覚的に捉えるには？
- 波形から音を想像できるか？



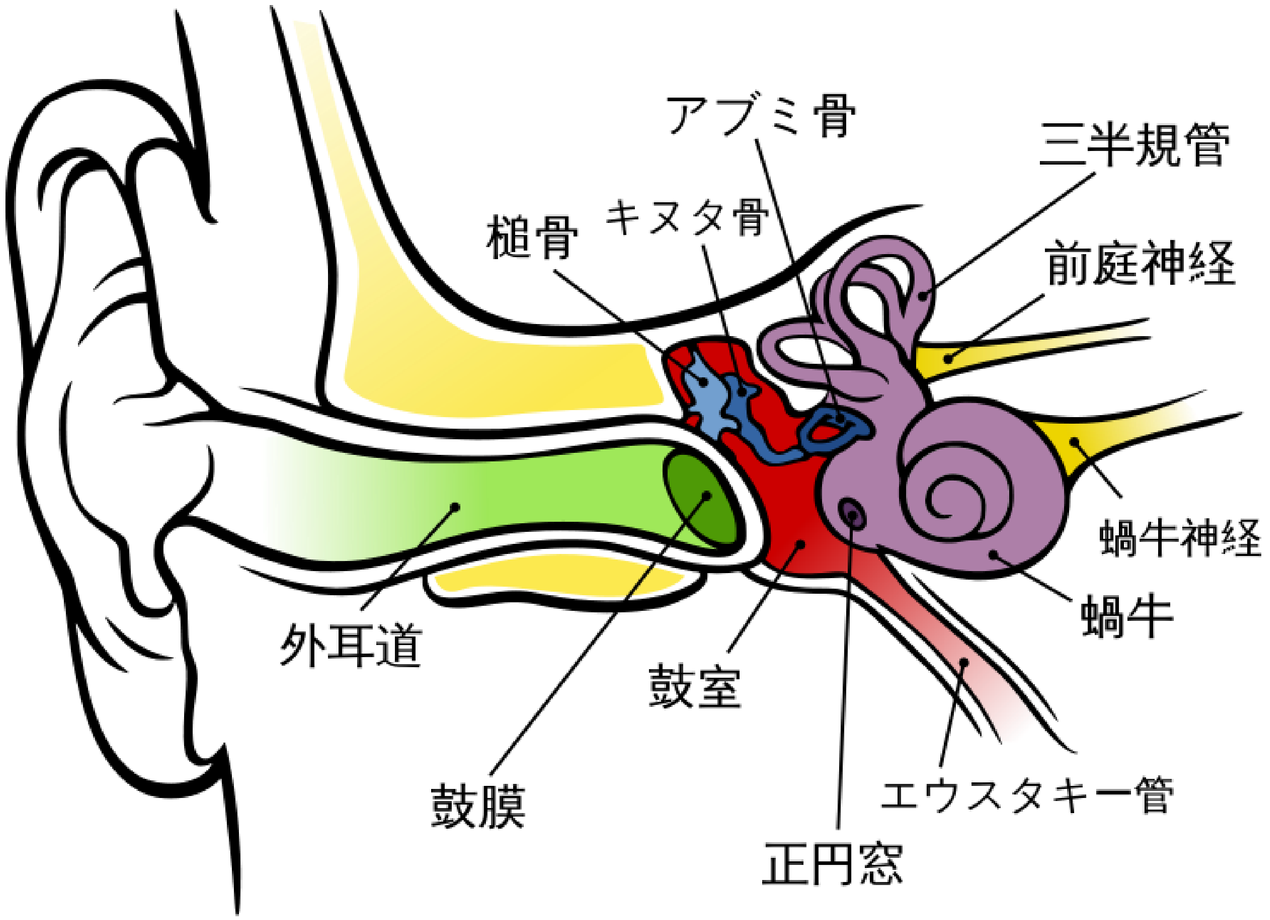
Audio into Visual.

- 我々は、音波の振動の形（音波）から直接音を聞いているわけではない
- 音に含まれる周波数の成分ごとに分析して聞いている
- 耳の中には周波数解析器のような機能が備わっている



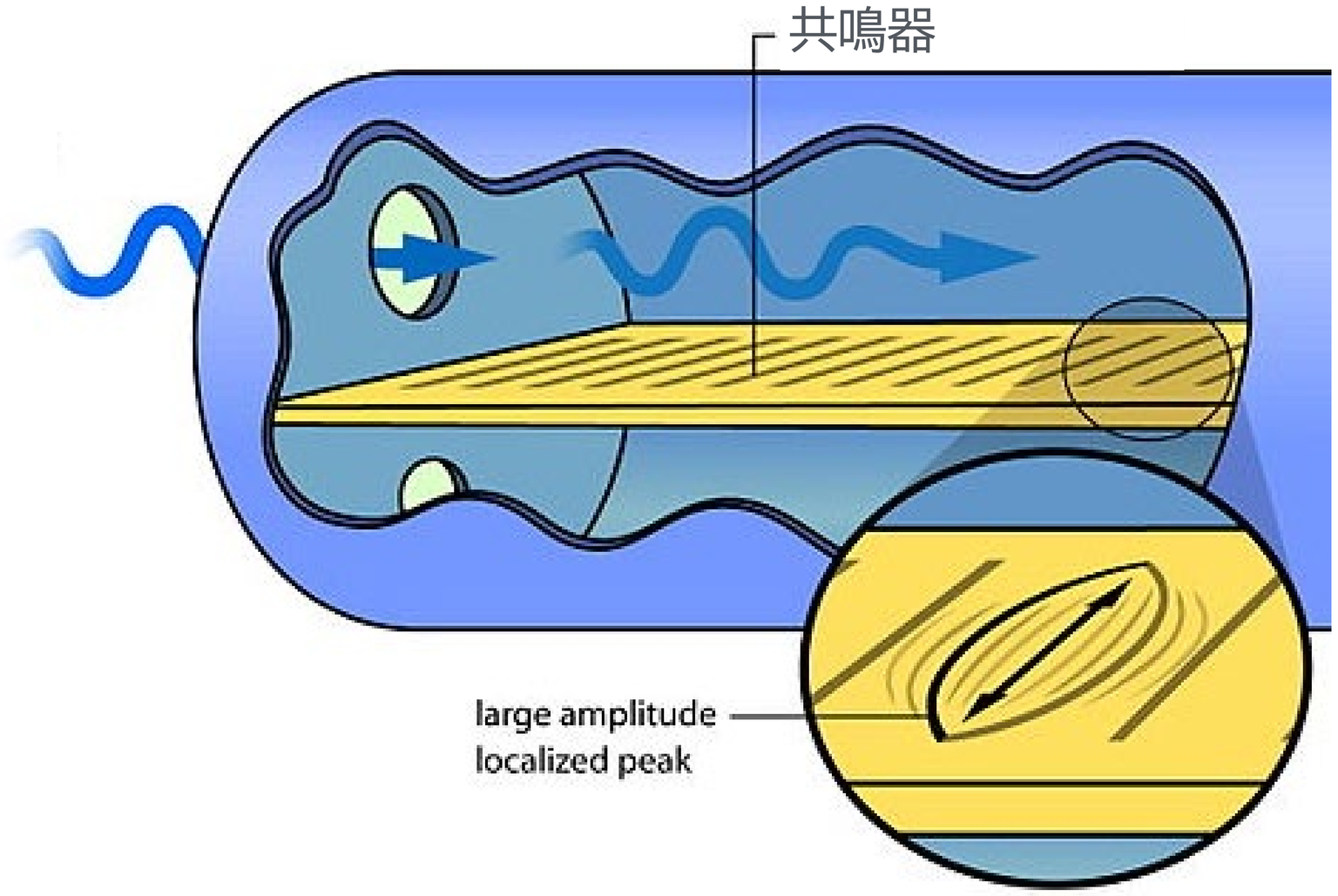
Audio into Visual.

- 耳の構造



Audio into Visual.

- 蝸牛の構造



Audio into Visual.

- では、耳が聞いているように音を見るにはどうすれば良いか？
- 音を周波数成分に分解する
- フーリエ変換: 音に含まれる周波数成分を数学的に解析できる

$$F(\xi) = \int_{-\infty}^{\infty} f(x) e^{-i2\pi\xi x} dx$$

$$f(x) = \int_{-\infty}^{\infty} F(\xi) e^{i2\pi\xi x} d\xi$$

Audio into Visual.

- では、耳が聞いているように音を見るにはどうすれば良いか？
- 音を周波数成分に分解する
- フーリエ変換: 音に含まれる周波数成分を数学的に解析できる

フーリエ変換：波形 → 周波数成分

$$F(\xi) = \int_{-\infty}^{\infty} f(x) e^{-i2\pi\xi x} dx$$

$$f(x) = \int_{-\infty}^{\infty} F(\xi) e^{i2\pi\xi x} d\xi$$

Audio into Visual.

- では、耳が聞いているように音を見るにはどうすれば良いか？
- 音を周波数成分に分解する
- フーリエ変換: 音に含まれる周波数成分を数学的に解析できる

フーリエ変換：波形 → 周波数成分

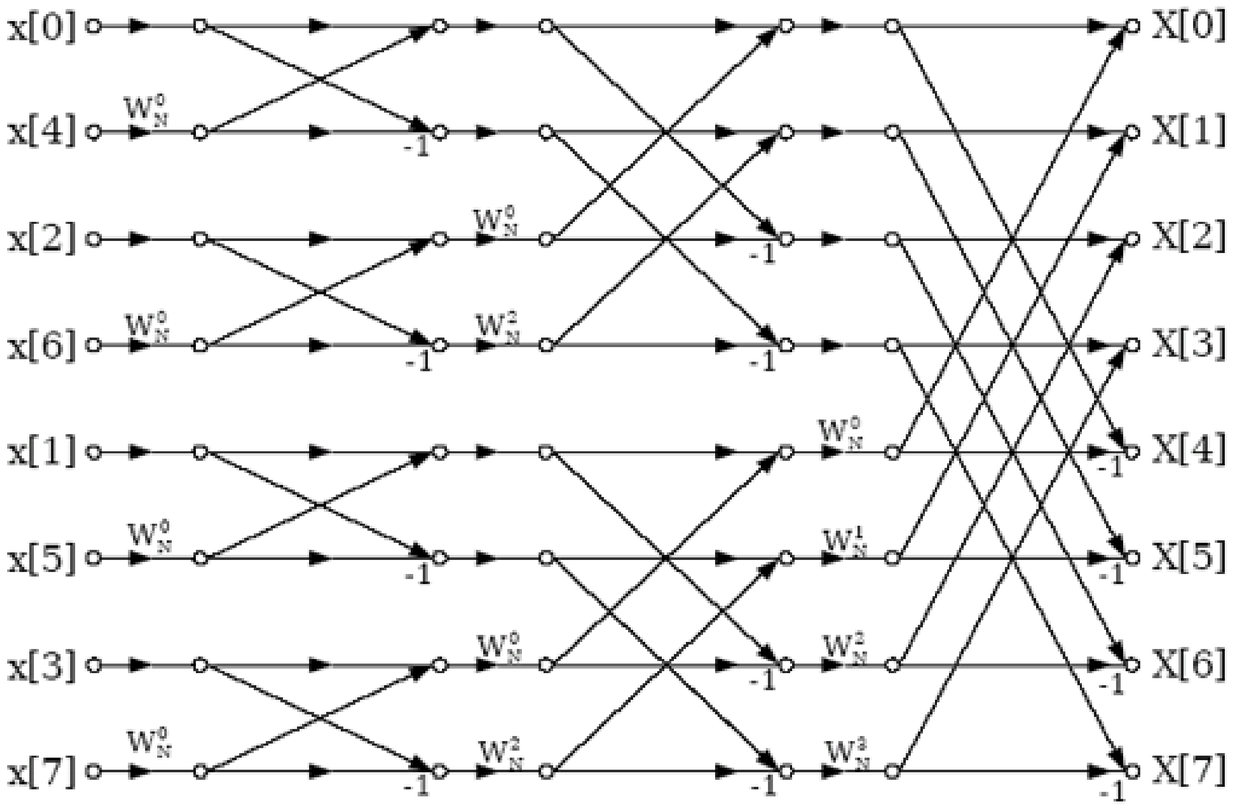
$$F(\xi) = \int_{-\infty}^{\infty} f(x) e^{-i2\pi\xi x} dx$$

フーリエ逆変換：周波数成分 → 波形

$$f(x) = \int_{-\infty}^{\infty} F(\xi) e^{i2\pi\xi x} d\xi$$

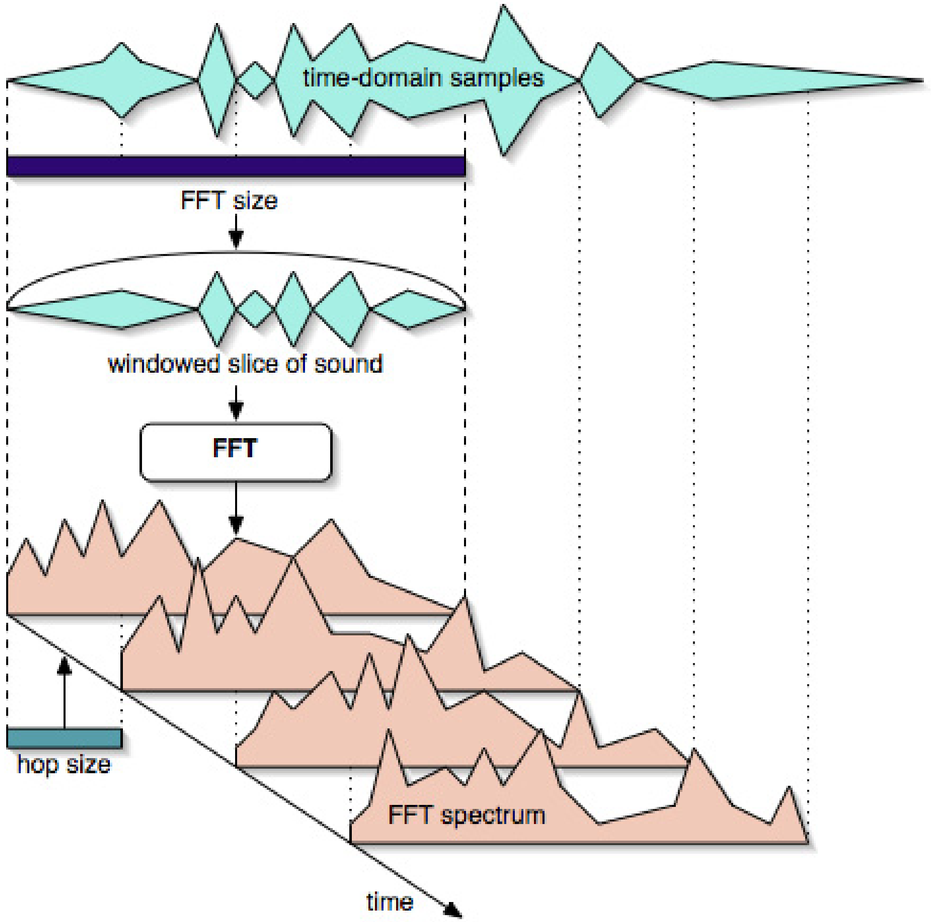
Audio into Visual.

- 高速フーリエ変換(FFT): フーリエ変換をコンピュータで高速に解析するためのアルゴリズム



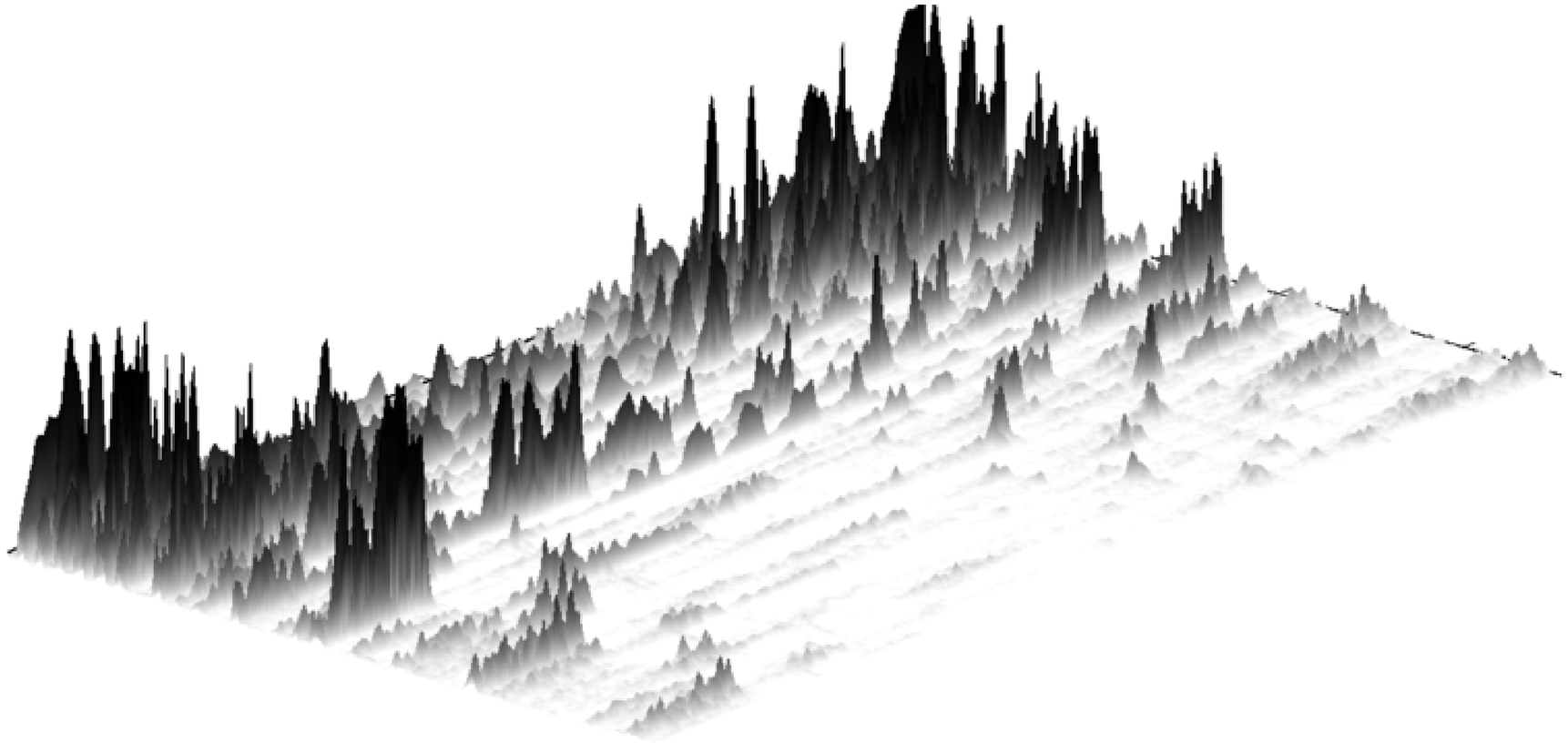
Audio into Visual.

- 短時間フーリエ変換(STFT): 波形を短かい区間で区切ってFFTを行い、周波数成分の時間的変化を分析



Audio into Visual.

- スペクトル(Spectrogram): 時間軸上での周波数成分の分析結果をプロットしたもの



FFT解析を行う

FFT解析を行う

- p5.soundライブラリには高速フーリエ変換のためのクラスFFTが用意されている
- FFTクラスを利用することで簡単に音を周波数成分に分解して視覚化可能

- AudioInによるサウンド入力でキャプチャーした音をFFTを使用して周波数成分に分解
- その結果をグラフに描画してみる

FFT解析を行う

- FFT解析(グラフ化)

```
let sound;
let fft;

//サウンドファイルをプリロード
function preload() {
  sound = loadSound('./beat.wav');
}

function setup() {
  createCanvas(windowWidth, windowHeight);
  //ループ再生
  sound.loop();
  //FFTを初期化
  fft = new p5.FFT();
  //サウンドファイルをFFTの入力に
  fft.setInput(sound);
}
```

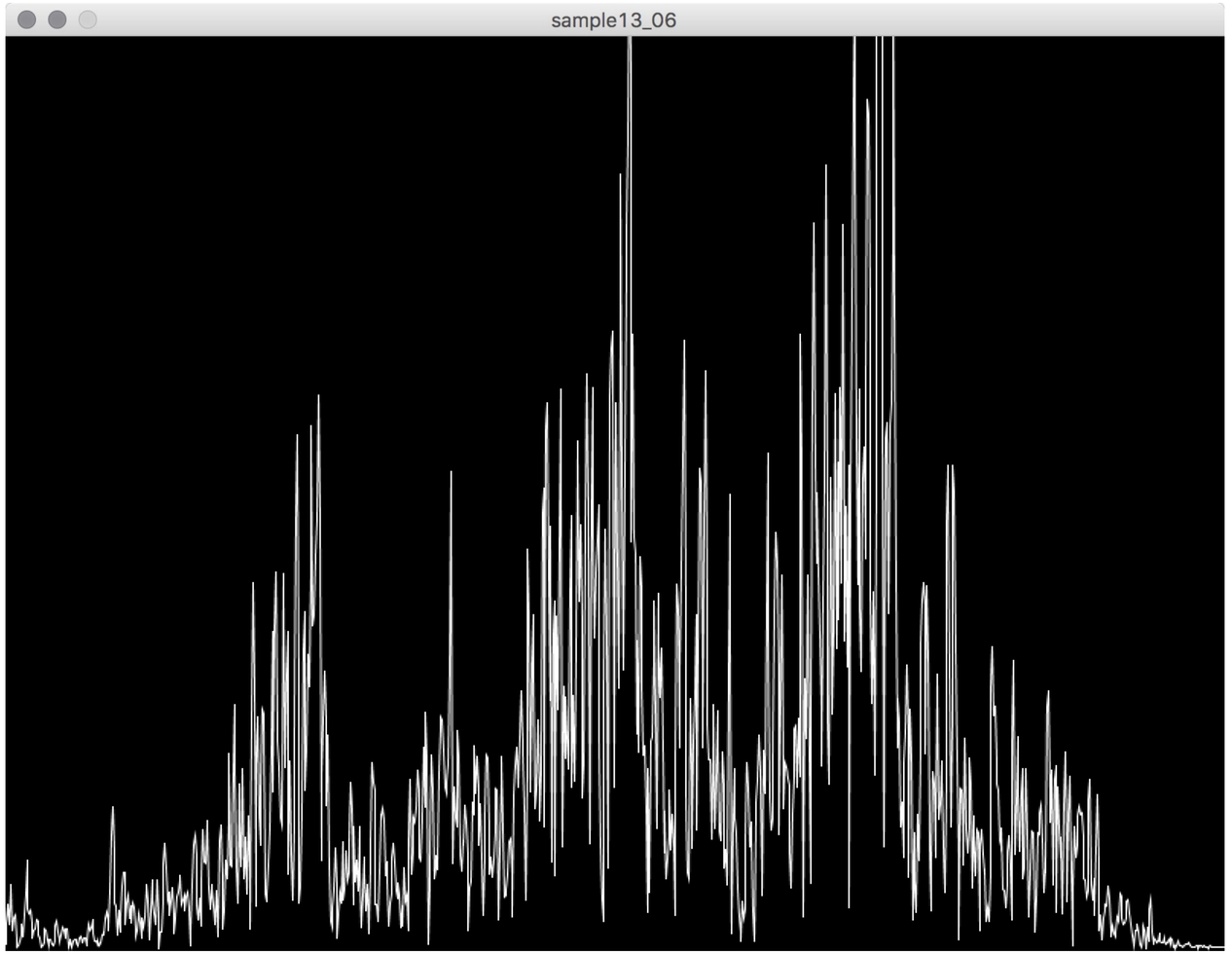
FFT解析を行う

- FFT解析(グラフ化)

```
function draw() {
  background(0);
  stroke(255);
  noFill();
  //FFT解析
  let spectrum = fft.analyze();
  //結果をグラフで描画
  beginShape();
  for (i = 0; i < spectrum.length; i++) {
    let x = map(i, 0, spectrum.length - 1, 0, width);
    let y = map(spectrum[i], 0, 255, height, 0);
    vertex(x, y);
  }
  endShape();
}
```

FFT解析を行う

- 周波数の成分がグラフ化された!



FFTでサウンドをビジュアライズ

FFTでサウンドをビジュアライズ

- 先週のFFTのグラフを描くプログラムをForkして改造していきましょう!
- 下記のURLからFork

https://www.openprocessing.org/sketch/630394

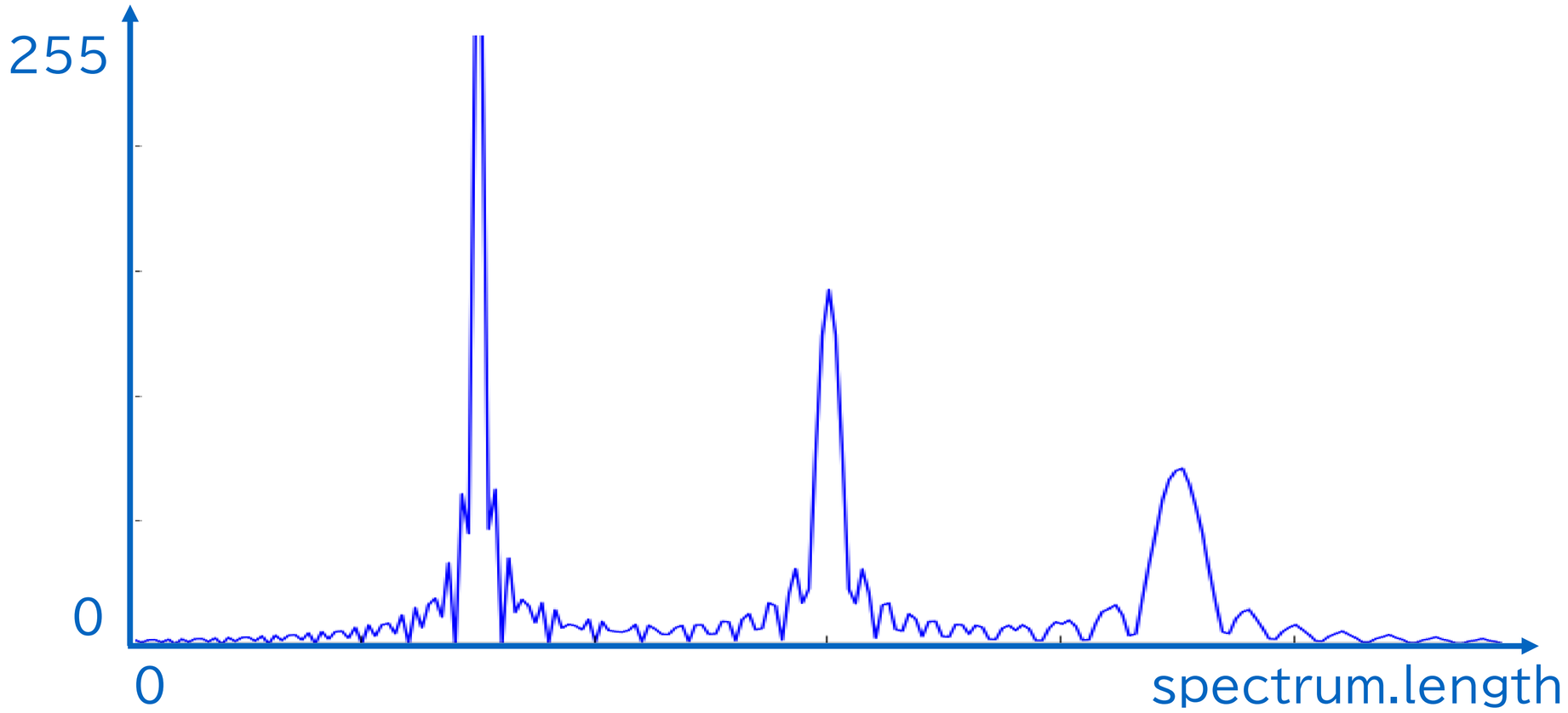
FFTでサウンドをビジュアライズ

- コードのポイントは、draw()関数内のfor文

```
//FFT解析
let spectrum = fft.analyze();
//結果をグラフで描画
beginShape();
for (i = 0; i < spectrum.length; i++) {
  let x = map(i, 0, spectrum.length - 1, 0, width);
  let y = map(spectrum[i], 0, 255, height, 0);
  vertex(x, y);
}
endShape();
```

FFTでサウンドをビジュアライズ

- 分析結果は、以下のように分析される → これをどうビジュアライズするか？



FFTでサウンドをビジュアライズ

- ビジュアライズ 1
  - まずはシンプルな実験
  - グラフではなく、グレースケールの色の濃度で塗り分けてみる
  - FFTのレベルが高い周波数ほど濃い色で
  - グラフを上からみているイメージ

FFTでサウンドをビジュアライズ

- ビジュアライズ1

```
let sound;
let fft;

//サウンドファイルをプリロード
function preload() {
  sound = loadSound('./beat.wav');
}

function setup() {
  createCanvas(windowWidth, windowHeight);
  //ループ再生
  sound.loop();
  //FFTを初期化
  fft = new p5.FFT();
  //サウンドファイルをFFTの入力に
  fft.setInput(sound);
}
```

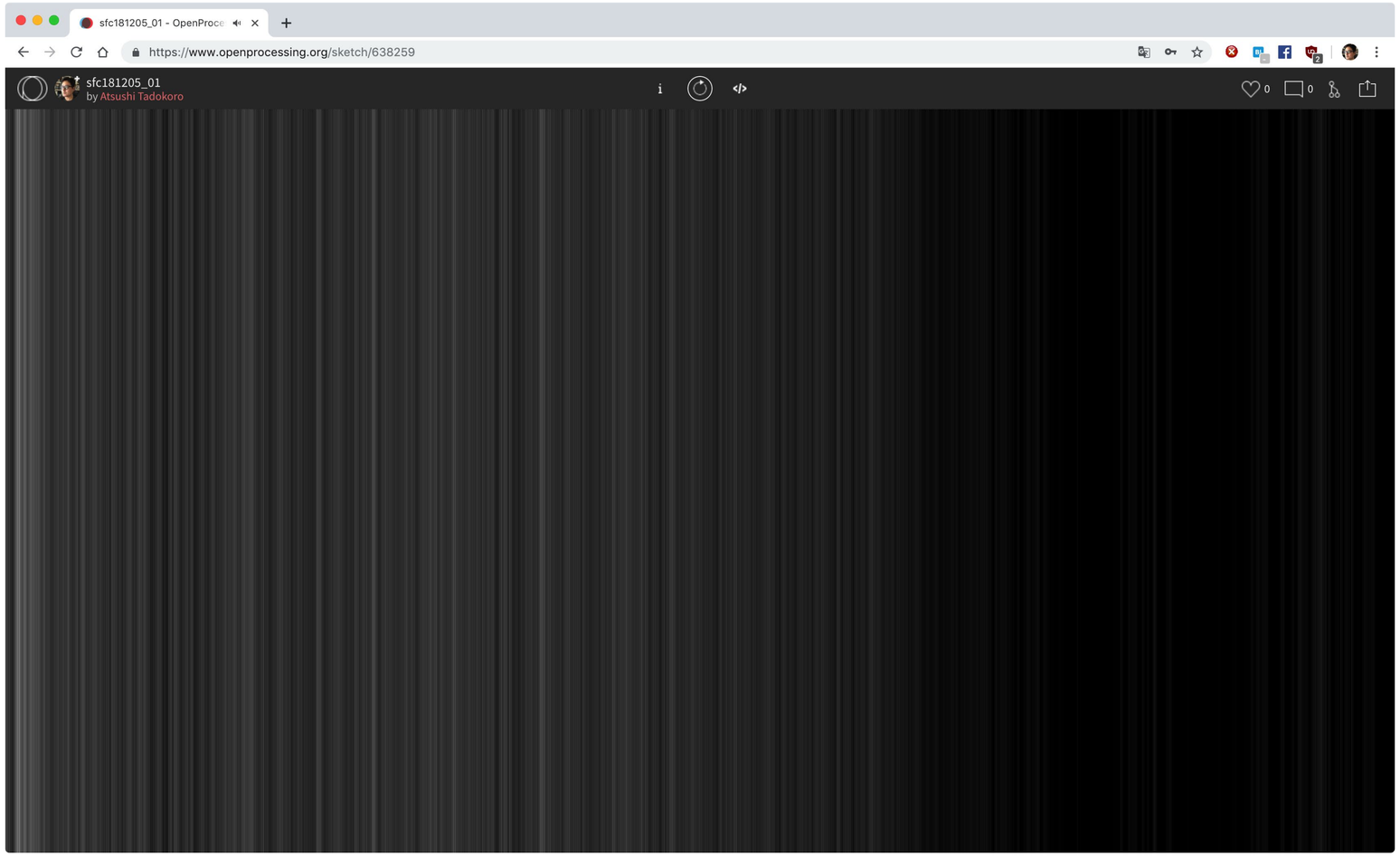
FFTでサウンドをビジュアライズ

- ビジュアライズ1

```
function draw() {
    background(0);
      //マウスクリックでサウンドループ再生
      if(mouseIsPressed && sound.isPlaying() == false){
        sound.loop();
      }
    noStroke();
    //FFT解析
    let spectrum = fft.analyze();
    //結果を色の濃さで描画
    for (i = 0; i < spectrum.length; i++) {
      let x = map(i, 0, spectrum.length - 1, 0, width);
      let col = spectrum[i];
          fill(col);
      rect(x, 0, 1, height);
    }
  }
```

FFTでサウンドをビジュアライズ

- 色の濃度でFFTをビジュアライズ!



FFTでサウンドをビジュアライズ

- ビジュアライズ 2
  - 解析結果をデザインしてみる
  - 左右対称にしてみたらどうなる？

FFTでサウンドをビジュアライズ

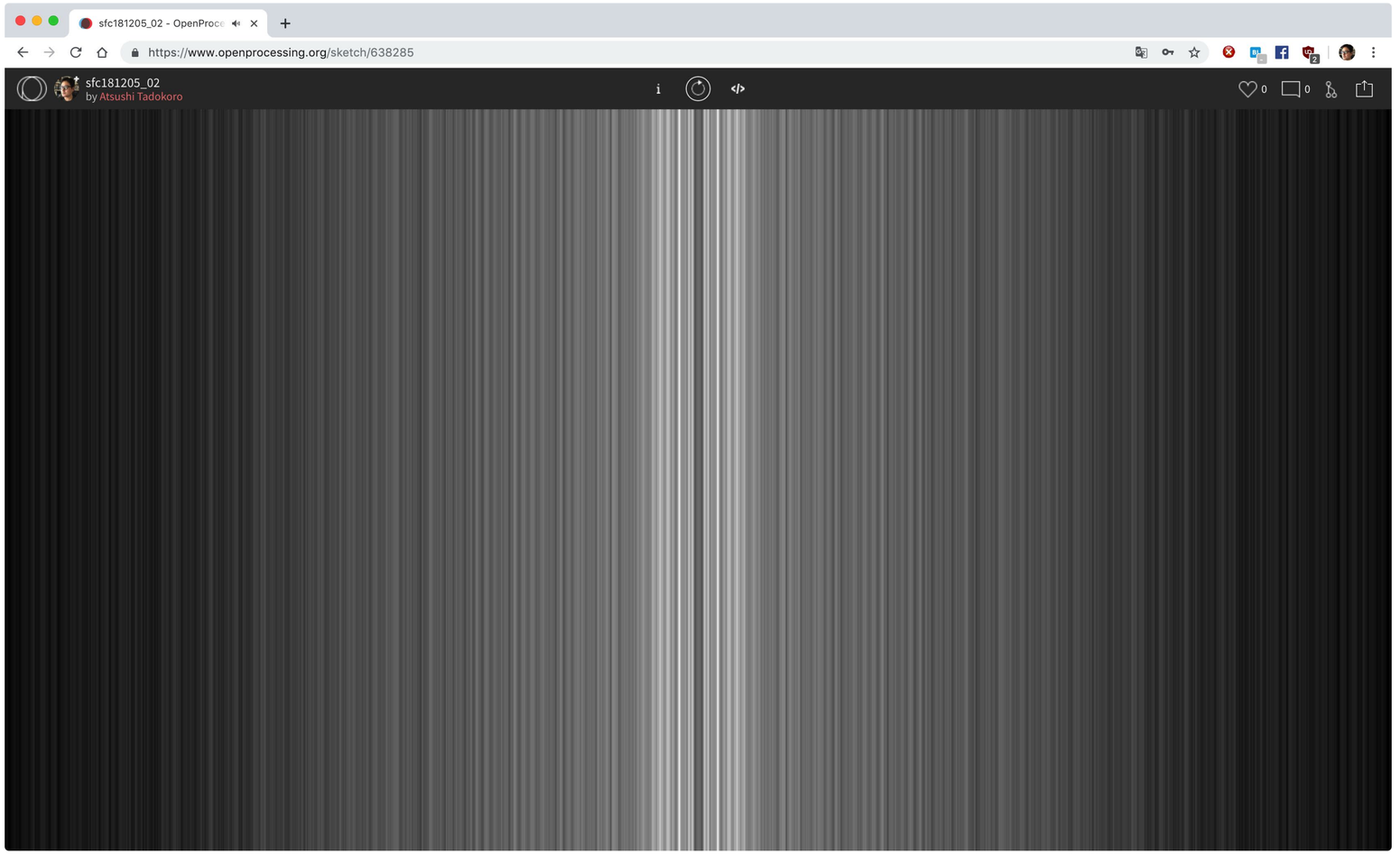
- ビジュアライズ 2

```
...(前略)...

function draw() {
  background(0);
  //マウスクリックでサウンドループ再生
  if(mouseIsPressed && sound.isPlaying() == false){
    sound.loop();
  }
  noStroke();
  //FFT解析
  let spectrum = fft.analyze();
  //結果を色の濃さで描画
  for (i = 0; i < spectrum.length; i++) {
    let col = spectrum[i];
    fill(col);
    //右半分
    let x = map(i, 0, spectrum.length - 1, width/2, width);
    rect(x, 0, 1, height);
    //左半分
    x = map(i, 0, spectrum.length - 1, width/2, 0);
    rect(x, 0, 1, height);
  }
}
```

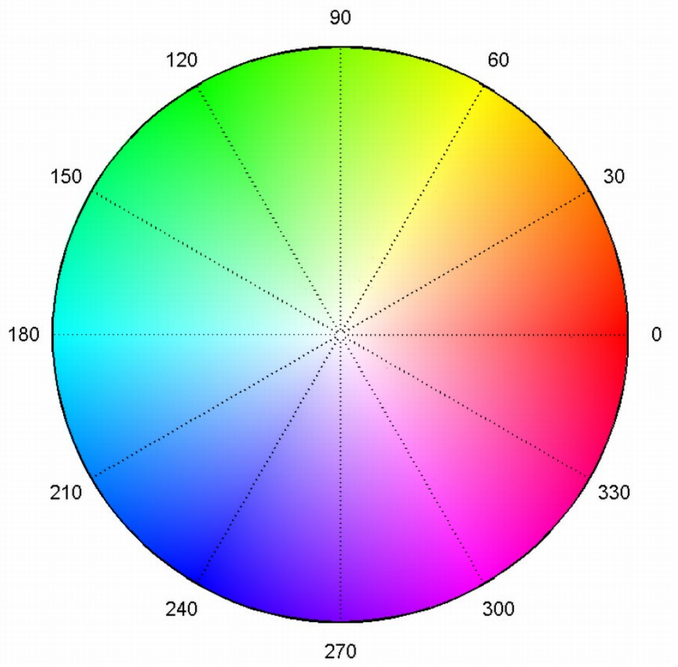
FFTでサウンドをビジュアライズ

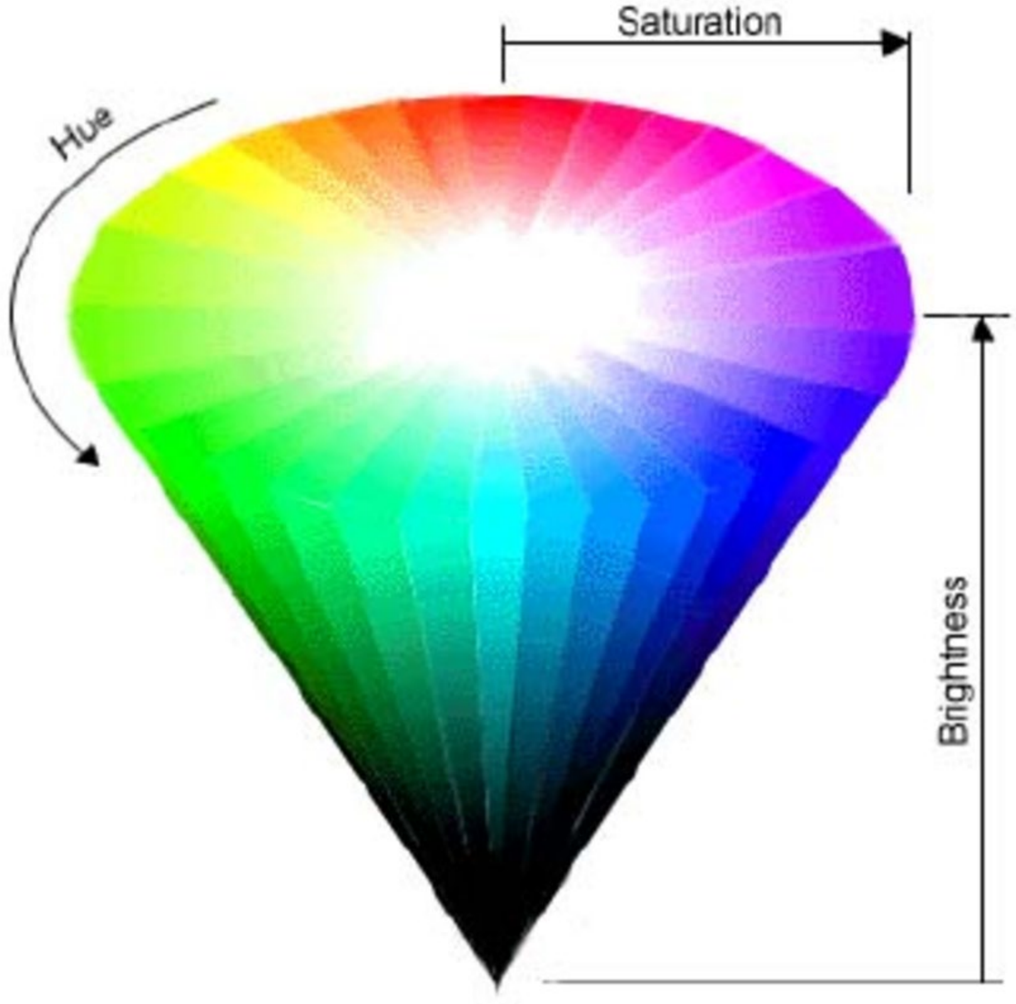
- 左右対称の美しいパターンに!



FFTでサウンドをビジュアライズ

- ビジュアライズ 3
  - 色を塗り分けてみる
  - RGBではなくHSB（色相、明度、彩度）で
  - 色相（Hue）を周波数に対応させてみる!





FFTでサウンドをビジュアライズ

- p5.jsでHSBを使用するには、colorModeで指定する

```
//例: HSBをそれぞれ 0〜255 で表現
colorMode(HSB, 255);
```

FFTでサウンドをビジュアライズ

- ビジュアライズ 3

```
let sound;
let fft;

//サウンドファイルをプリロード
function preload() {
  sound = loadSound('./beat.wav');
}

function setup() {
  createCanvas(windowWidth, windowHeight);
  //カラーモードをHSBに
  colorMode(HSB, 255);
  //FFTを初期化
  fft = new p5.FFT();
  //サウンドファイルをFFTの入力に
  fft.setInput(sound);
}
```

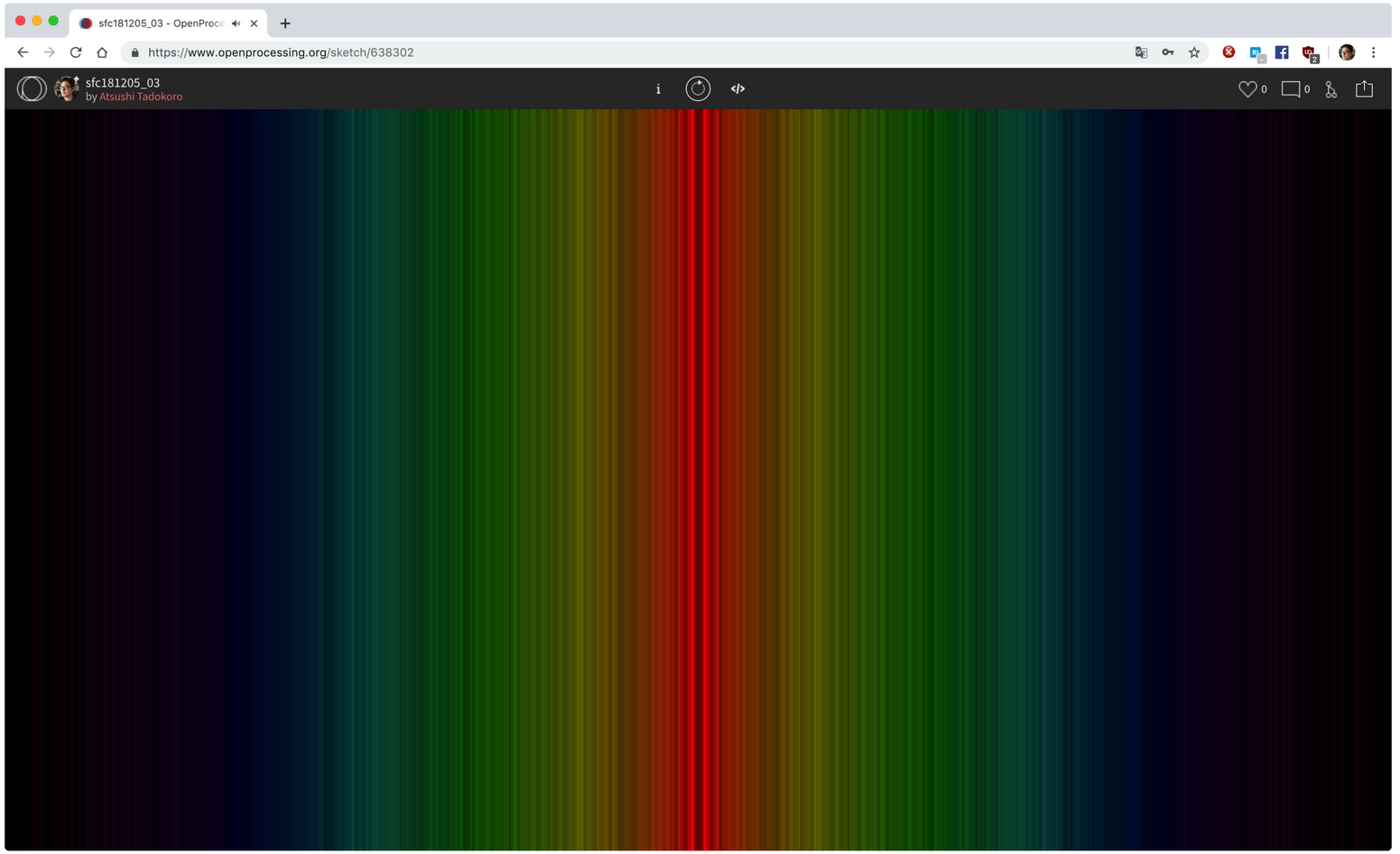
FFTでサウンドをビジュアライズ

- ビジュアライズ 3

```
function draw() {
  background(0);
  //マウスクリックでサウンドループ再生
  if (mouseIsPressed && sound.isPlaying() == false) {
    sound.loop();
  }
  noStroke();
  //FFT解析
  let spectrum = fft.analyze();
  //色相で塗り分ける
  for (i = 0; i < spectrum.length; i++) {
    //周波数色相を変化
    let hue = map(i, 0, spectrum.length - 1, 0, 255);
    //スペクトラムのレベルで明度を変化
    let brightness = spectrum[i];
    //HSBから色を生成
    fill(hue, 255, brightness);
    let x = map(i, 0, spectrum.length - 1, width / 2, width);
    rect(x, 0, 1, height);
    x = map(i, 0, spectrum.length - 1, width / 2, 0);
    rect(x, 0, 1, height);
  }
}
```

FFTでサウンドをビジュアライズ

- 周波数を色相で塗り分け!



FFTでサウンドをビジュアライズ

- ビジュアライズ ４
  - さらに見た目を工夫してみる
  - 左右だけでなく上下方向にも描画してみる

FFTでサウンドをビジュアライズ

▸ ビジュアライズ 4

```
...(前略)...

function draw() {
  //ブレンドモードをBLEND(混色)に
  blendMode(BLEND);
  background(0);
  //マウスクリックでサウンドループ再生
  if (mouseIsPressed && sound.isPlaying() == false) {
    sound.loop();
  }
  noStroke();
  //FFT解析
  let spectrum = fft.analyze();
  //色相で塗り分ける
  //ブレンドモードをADD(加算)に
  blendMode(ADD);
  for (i = 0; i < spectrum.length; i++) {
    //周波数色相を変化
    let hue = map(i, 0, spectrum.length - 1, 0, 255);
    //スペクトラムのレベルで明度を変化
    let brightness = spectrum[i] / 2.0;
```

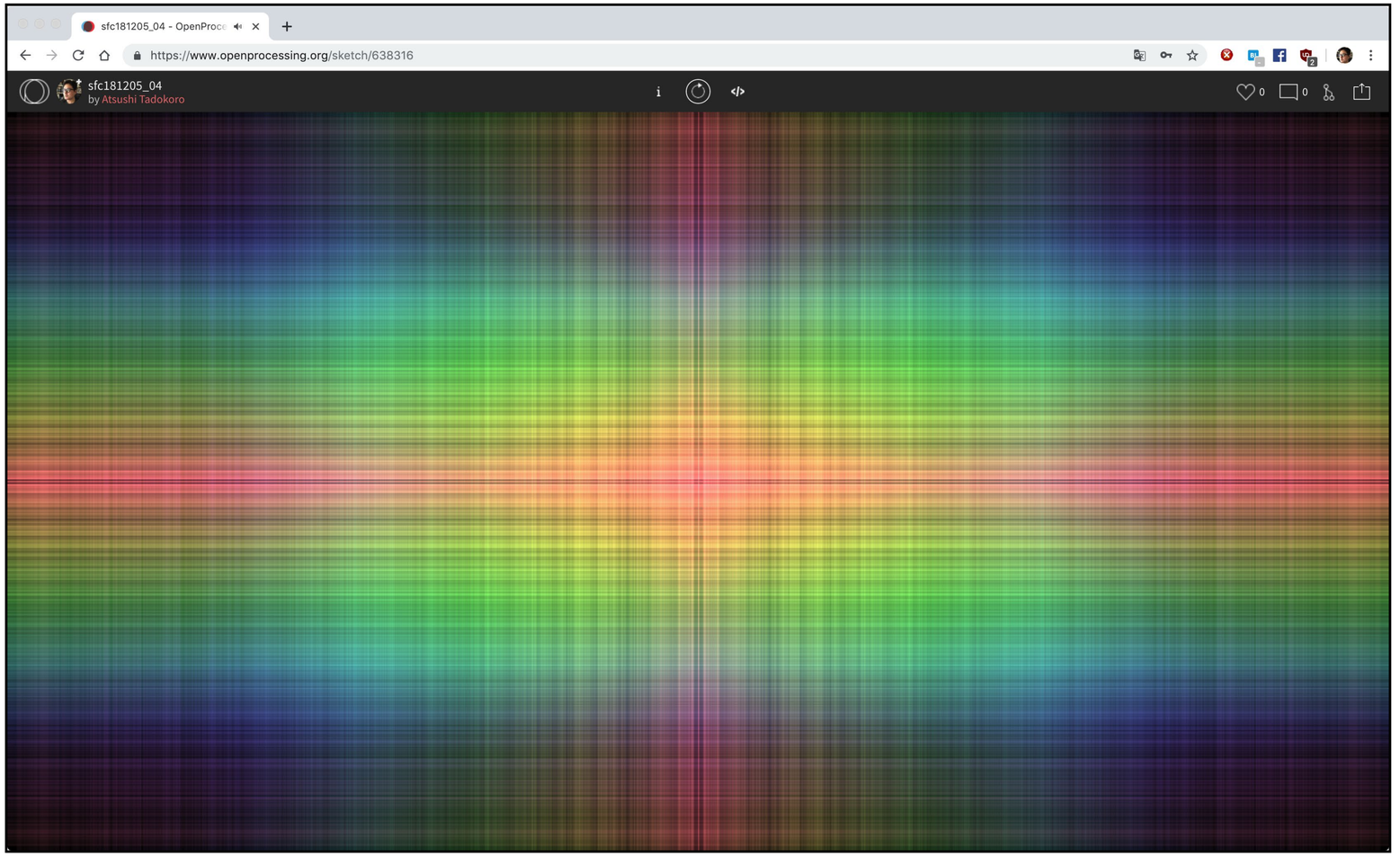
FFTでサウンドをビジュアライズ

- ビジュアライズ 4

```
    //HSBから色を生成
    fill(hue, 127, brightness);
    //横方向
    let x = map(i, 0, spectrum.length - 1, width / 2, width);
    rect(x, 0, 1, height);
    x = map(i, 0, spectrum.length - 1, width / 2, 0);
    rect(x, 0, 1, height);
    //縦方向
    let y = map(i, 0, spectrum.length - 1, height / 2, height);
    rect(0, y, width, 1);
    y = map(i, 0, spectrum.length - 1, height / 2, 0);
    rect(0, y, width, 1);
  }
}
```

FFTでサウンドをビジュアライズ

- 美しい格子模様に!



FFTでサウンドをビジュアライズ

- ビジュアライズ 5
  - 円の大きさで表現
  - 色はそのままで

FFTでサウンドをビジュアライズ

- ビジュアライズ 5

```
...(前略)...

function draw() {
  blendMode(BLEND);
  background(0);
  //マウスクリックでサウンドループ再生
  if (mouseIsPressed && sound.isPlaying() == false) {
    sound.loop();
  }
  noStroke();
  //FFT解析
  let spectrum = fft.analyze();
  //円の大きさで表現
  blendMode(ADD);
  for (i = 0; i < spectrum.length; i++) {
    //周波数色相を変化
    let hue = map(i, 0, spectrum.length - 1, 0, 255);
    //スペクトラムのレベルで円の直径を変化させる
    let diameter = map(spectrum[i], 0, 255, 0, height);
```

FFTでサウンドをビジュアライズ

- ビジュアライズ 5

```
    //HSBから色を生成
    fill(hue, 255, 15);
    let x = map(i, 0, spectrum.length - 1, width / 2, width);
    ellipse(x, height / 2, diameter, diameter);
    x = map(i, 0, spectrum.length - 1, width / 2, 0);
    ellipse(x, height / 2, diameter, diameter);
  }
}
```

FFTでサウンドをビジュアライズ

- 完成!!

最終課題!

最終課題!

- 音響をビジュアライズした作品を制作してください!
- 読み込むサウンドファイルは自由（できれば著作権フリーのもので）
- 音響を独自の視点でビジュアライズしてください

最終課題!

- 今後のスケジュールは以下の通りで進めていきます

- 11月27日（本日）サウンドプログラミング(2)、最終課題出題
- 12月4日 最終課題制作のヒント(オンデマンド) + 制作相談会（Zoomリアルタイム）
- 12月11日 最終課題発表会(1)（Zoomリアルタイム）※希望者のみ
- 12月18日 最終課題発表会(2)（Zoomリアルタイム）※希望者のみ
- 12月18日 最終課題最終締切(23:59)