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 (1)

2020年11月20日

慶應義塾大学 総合政策学部・環境情報学部
田所淳

今日の内容

- p5.jsでライブラリーを扱ってみる
  - ライブラリーとは?
  - p5.soundライブリーについて
  - サウンドファイルの再生から、音のビジュアライズまで

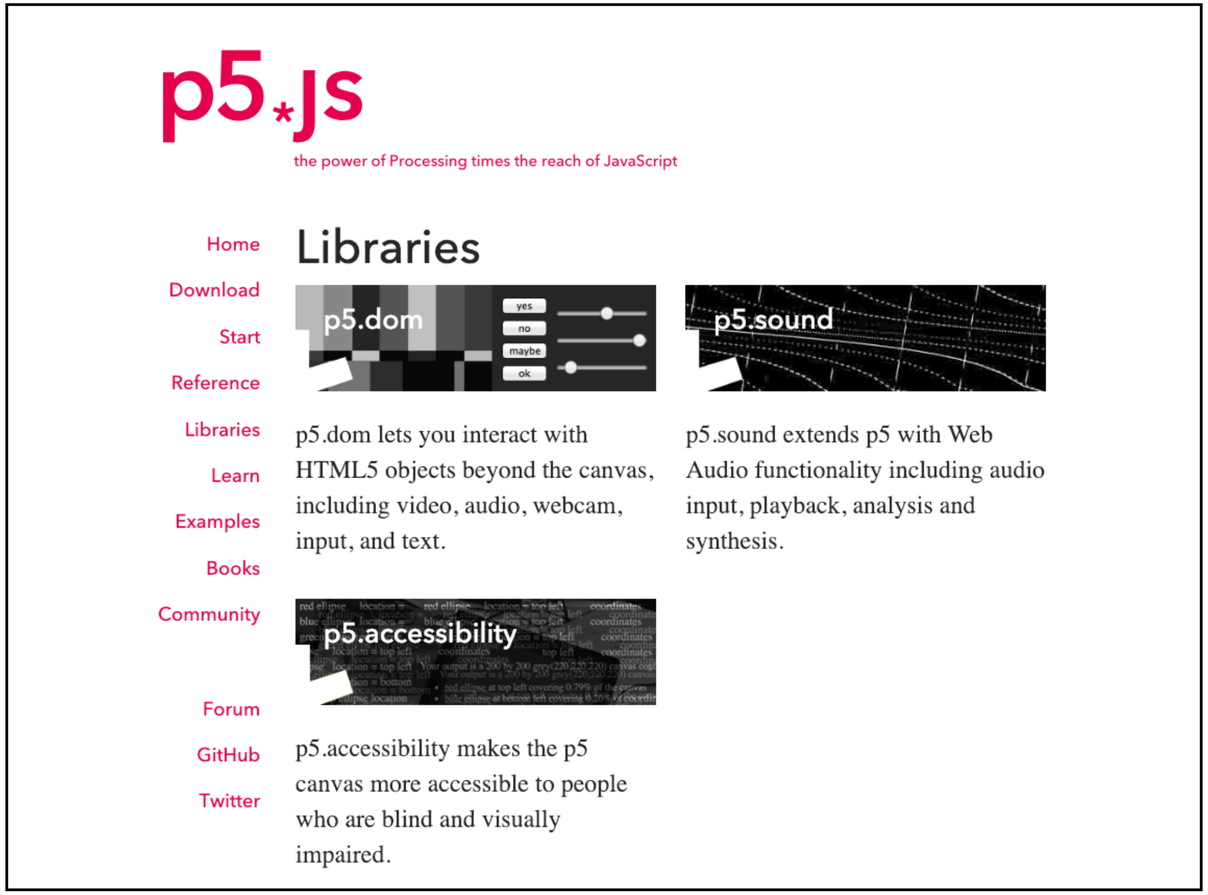
ライブラリー（Library）とは

ライブラリー（Library）とは

- ライブラリー（Library）
- 汎用性の高い複数のプログラムを再利用可能な形でひとまとまりにしたもの
- p5.jsにライブラリーを使用することで様々な機能を取り込むことができる
- p5.js自体もライブラリーの一つ

ライブラリー（Library）とは

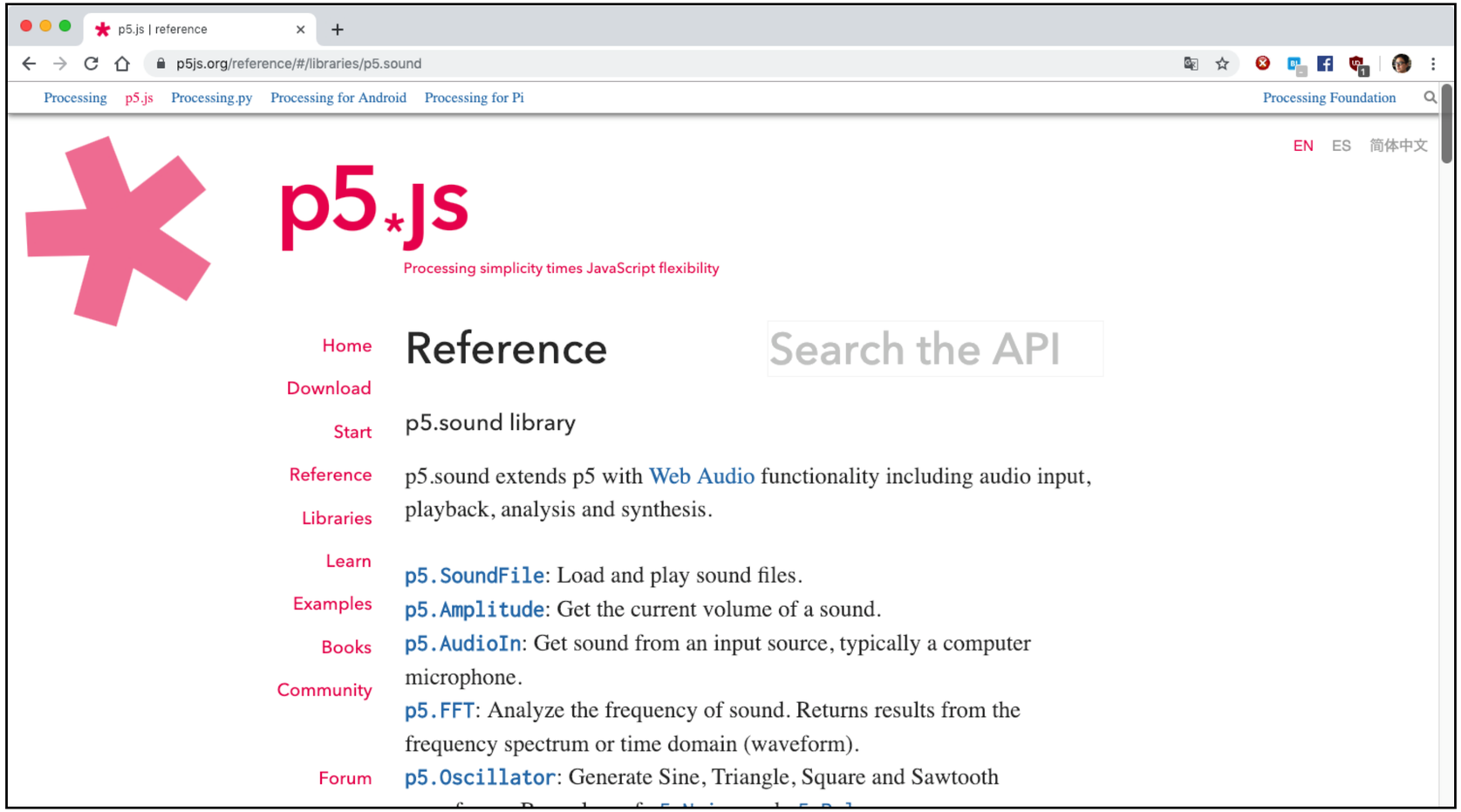
- p5.jsで使用可能なライブラリー
- https://p5js.org/libraries/



p5.soundライブラリー

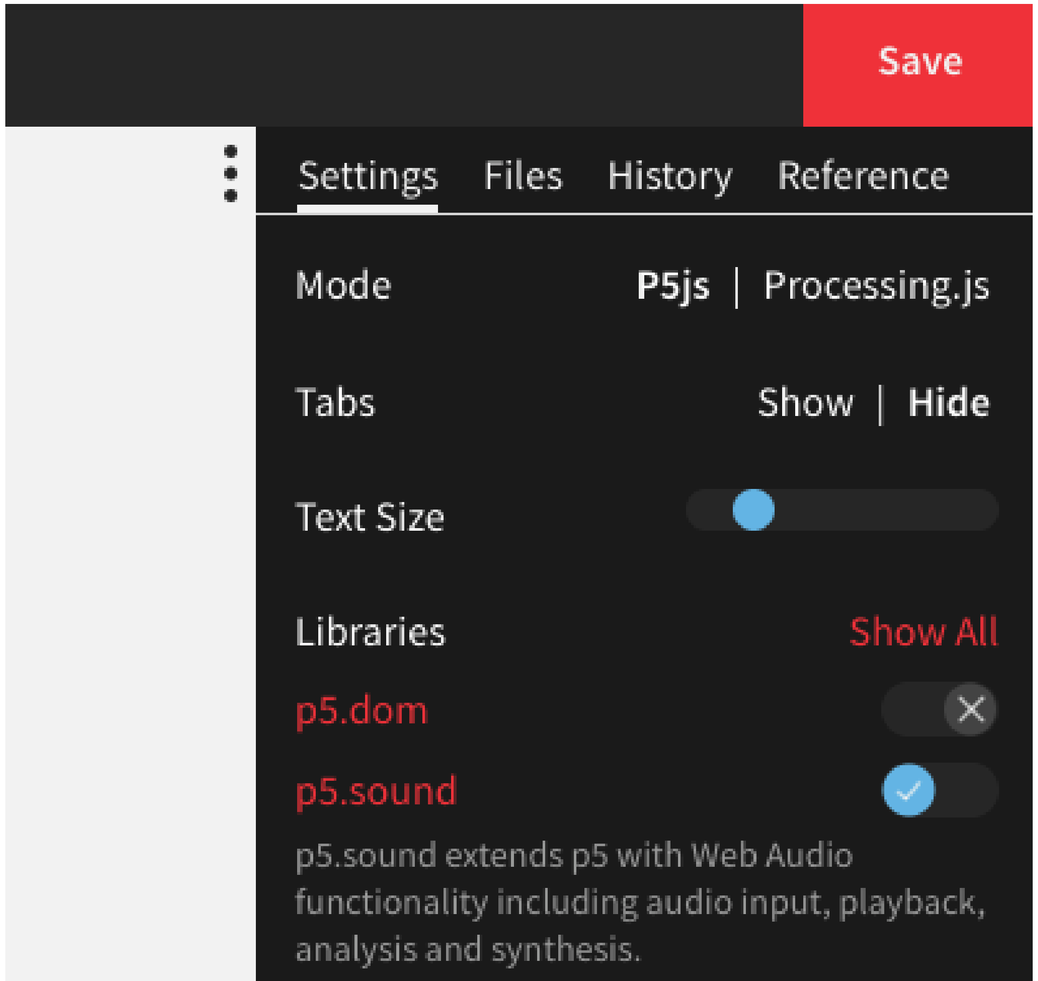
p5.soundライブラリー

- p5.soundライブラリーは、p5.jsで音を扱うためのライブラリー
- https://p5js.org/reference/#/libraries/p5.sound



p5.soundライブラリー

- OpenProcessingで利用するには?
- SketchのsettingsでP5.soundをONにする



サウンドファイルの追加と再生

サウンドファイルの追加と再生

- まず始めにサウンドファイルを読み込んで再生する簡単なサンプルを作成
- 再生するためのサウンドファイルを用意

- OpenProcessingの場合
  - ファイルのアップロードメニューからファイルをアップロード

- Atomエディターなどでローカルで作成する場合
  - sketch.jsと同じ場所に配置

サウンドファイルの追加と再生

- サウンドファイルの準備
- 以下からダウンロードして使用（beat.wav）

https://bit.ly/2UIXtx8

サウンドファイルの追加と再生

- まずはシンプルにサウンドファイルを再生してみる
- ループ再生するように

サウンドファイルの追加と再生

- サウンドファイルの再生

```
let sound;

//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);
  background(0);
  //サウンド再生
  sound.play();
}
```

サウンドファイルの追加と再生

- 少しだけインタラクションを付加してみる
- マウスボタンを押したら再生を開始するように

サウンドファイルの追加と再生

- サウンドファイルの再生

```
let sound;
let playing = false;

//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);
}

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();
  }
}
```

サウンドファイルの追加と再生

- さらにもう一工夫
- マウスを押している時はループ再生
- マウスを離したら停止
- 再生と停止で背景色を変化させる

サウンドファイルの追加と再生

- サウンドファイルの再生

```
let sound;
let bgColor;

//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);
  bgColor = color(0, 0, 255);
}

function draw() {
  background(bgColor);
}

function mousePressed(){
  bgColor = color(255, 0, 0);
  if(sound.isPlaying() == false) {
    sound.loop();
  }
}

function mouseReleased() {
  bgColor = color(0, 0, 255);
  sound.stop();
}
```

再生スピードの操作

再生スピードの操作

- SoundFileクラスサウンドの再生に関する関数
  - play()：サウンドファイルを一度だけ再生
  - loop()：サウンドファイルをループ再生
  - jump()：サウンドファイルの指定した場所(秒)に移動して再生
  - rate()：再生スピードを変更
  - amp()：再生する音量を変更
  - stop()：再生の停止

- これらを活用してみる
- 例えば、マウスの位置で再生スピードを変更

再生スピードの操作

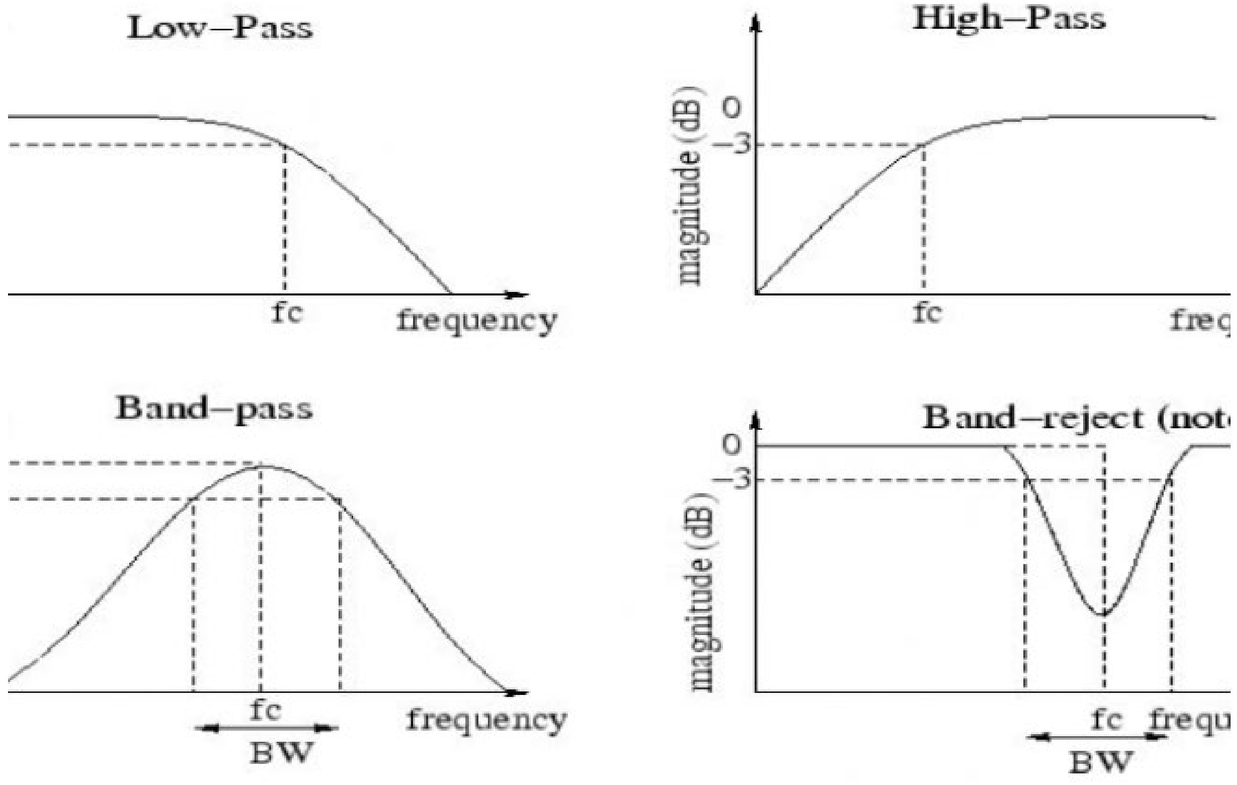
- マウスで再生スピードを変更

```
let sound;
let bgColor;

//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);
  bgColor = color(0, 0, 255);
}

function draw() {
  background(bgColor);
  //マウスのx座標で再生スピードを変化させる
  let speed = map(mouseX, 0.1, width, 0, 2);
  speed = constrain(speed, 0.01, 4);
  sound.rate(speed);
}

function mousePressed(){
  bgColor = color(255, 0, 0);
  if(sound.isPlaying() == false) {
    sound.loop();
  }
}

function mouseReleased() {
  bgColor = color(0, 0, 255);
  sound.stop();
}
```

フィルターをかける

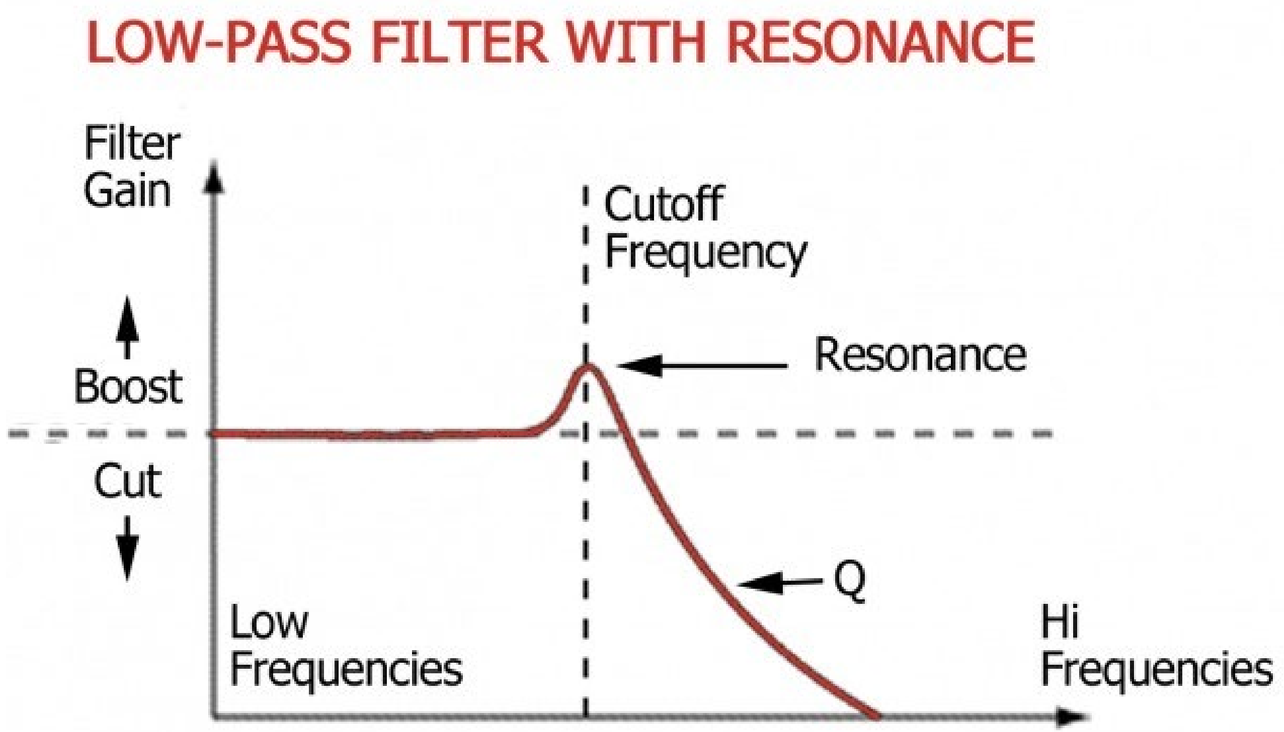
フィルターをかける

- フィルター：音の周波数成分を操作して特定の周波数帯のみを通過させる
  - ローパスフィルター：低音成分のみを通過（高音をカット）
  - ハイパスフィルター：高音成分のみを通過（低音をカット）
  - バンドパスフィルター：特定の帯域のみを通過



フィルターをかける

- マウスの位置でローパスフィルターをかけてみる
  - x座標：フィルター周波数
  - y座標：レゾナンス



再生フィルターをかけるの操作

- ローパスフィルター

```
let sound;
let bgColor;
let filter;

//サウンドファイルをプリロード
function preload() {
  sound = loadSound('./beat.wav');
  bgColor = color(0, 0, 255);
}

function setup() {
  createCanvas(windowWidth, windowHeight);
  //ローパスフィルターを初期化
  filter = new p5.LowPass();
  //サウンドの出力をローパスフィルターに切り替え
  sound.disconnect();
  sound.connect(filter);
}

function draw() {
  background(bgColor);
  //マウス座標ローパスフィルターをかける
  //x:カットオフ周波数, y:レゾナンス
  let filterFreq = map(mouseX, 0, width, 10, 22050);
  let filterRes = map(mouseY, 0, height, 40, 1);
  filter.set(filterFreq, filterRes);
}

function mousePressed(){
  bgColor = color(255, 0, 0);
  if(sound.isPlaying() == false) {
    sound.loop();
  }
}

function mouseReleased() {
  bgColor = color(0, 0, 255);
  sound.stop();
}
```

音量を視覚化

音量を視覚化

- SoundライブラリのAmplitudeクラス
- 再生している音の音量（ボリューム）をリアルタイムに解析することができる
- この機能を利用することで、再生している音の大きさに反応する簡単な視覚化が可能
- 音量はそのままの値ではなくRMS（二乗平均平方根）をとったほうが感覚と一致する
  - https://ja.wikipedia.org/wiki/%E4%BA%8C%E4%B9%97%E5%B9%B3%E5%9D%87%E5%B9%B3%E6%96%B9%E6%A0%B9

- 例えば、音量を円の直径に反映させて円を描いてみる

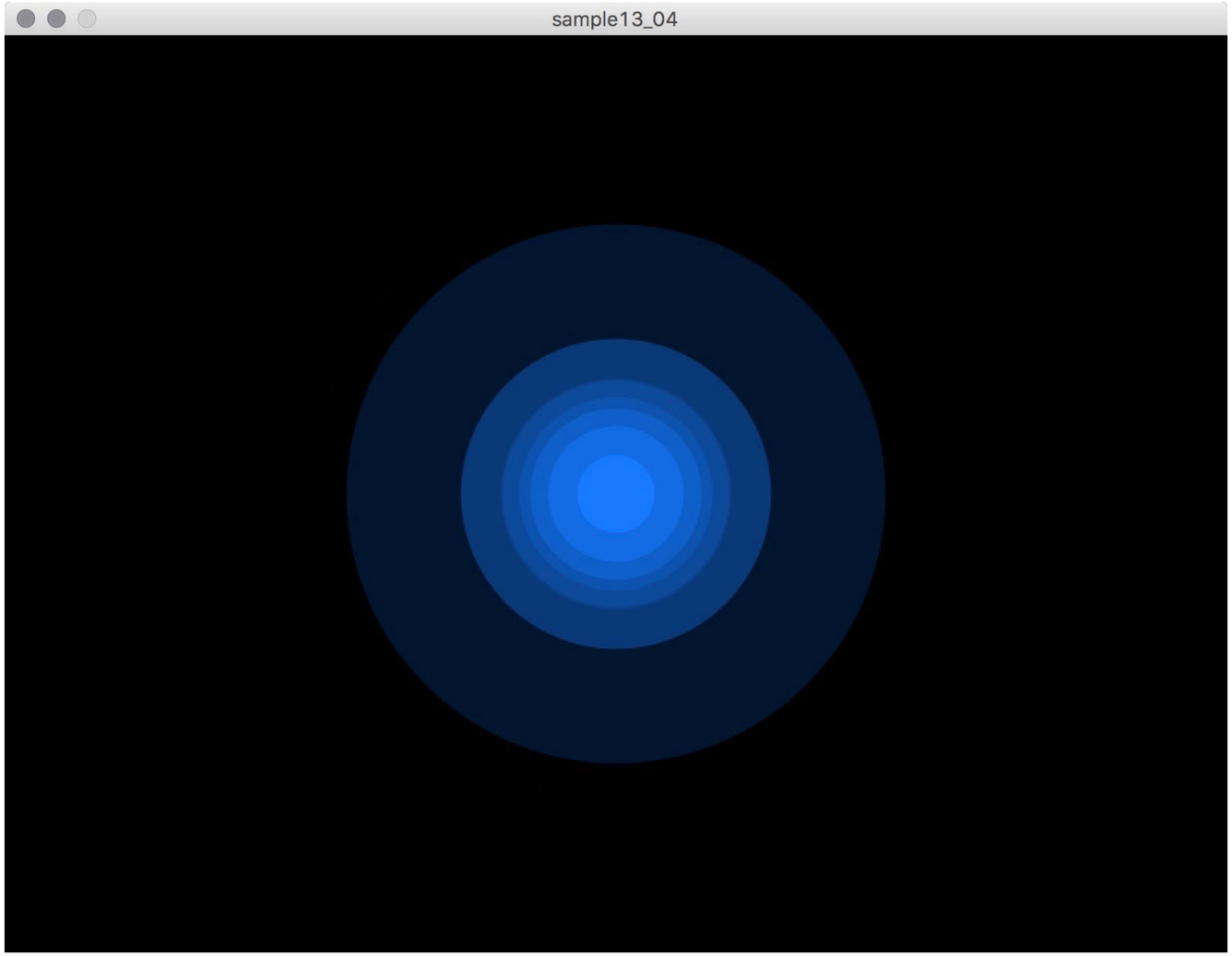
音量を視覚化

- 音量を円の直径に反映

```
let sound;
let analyzer;

//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);
  //音量アナライザーを初期化
  analyzer = new p5.Amplitude();
  //サウンドを音量アナライザーに入力
  analyzer.setInput(sound);
}

function draw() {
  background(0);
  //音量のRMS(二乗平均平方根)を計算
  var rms = analyzer.getLevel();
  fill(31, 127, 255);
  noStroke();
  //RMSの値を直径にして円を描く
  ellipse(width / 2, height / 2, rms * width);
}

function mousePressed(){
  if(sound.isPlaying() == false) {
    sound.loop();
  }
}

function mouseReleased() {
  sound.stop();
}
```

音量を視覚化

- 音の大きさに反応する円



本日の課題!

本日の課題!

- 本日作成した最後のサンプルプログラム（音量を円の半径で表現）をもとに
- 読み込んだサウンドファイルの音量で何かが変化するプログラムを作成
  - 色、大きさ、形 ...etc.
  - 読み込むサウンドは自由に変更しても良い
  - OpenProcessingに提出

- 本日のタグ sfcdesipro201120