メディアアート・プログラミング Ⅰ
第7回: 関数によるモジュール化とバリエーション

2020年6月26日

東京藝術大学芸術情報センター（AMC）
田所淳

今日の内容

- 今日のテーマ：関数（function）

- 変数（variable）と対になって、次回のオブジェクト指向プログラミングのキーに
- しっかり理解しましょう

関数とは？

関数とは?

- 関数（function）
  - コンピュータプログラミングにおいて、プログラム中で意味や内容がまとまっている作業をひとつの手続きとしたもの
  - サブルーチン（subroutine）
  - 値(引数、arguments)を元に何らかの計算や処理を行い、結果を呼び出し元に返す

関数とは?

- 実は、ここまでの内容で関数を頻繁に用いてきた!!
- p5.jsの関数の呼び出し

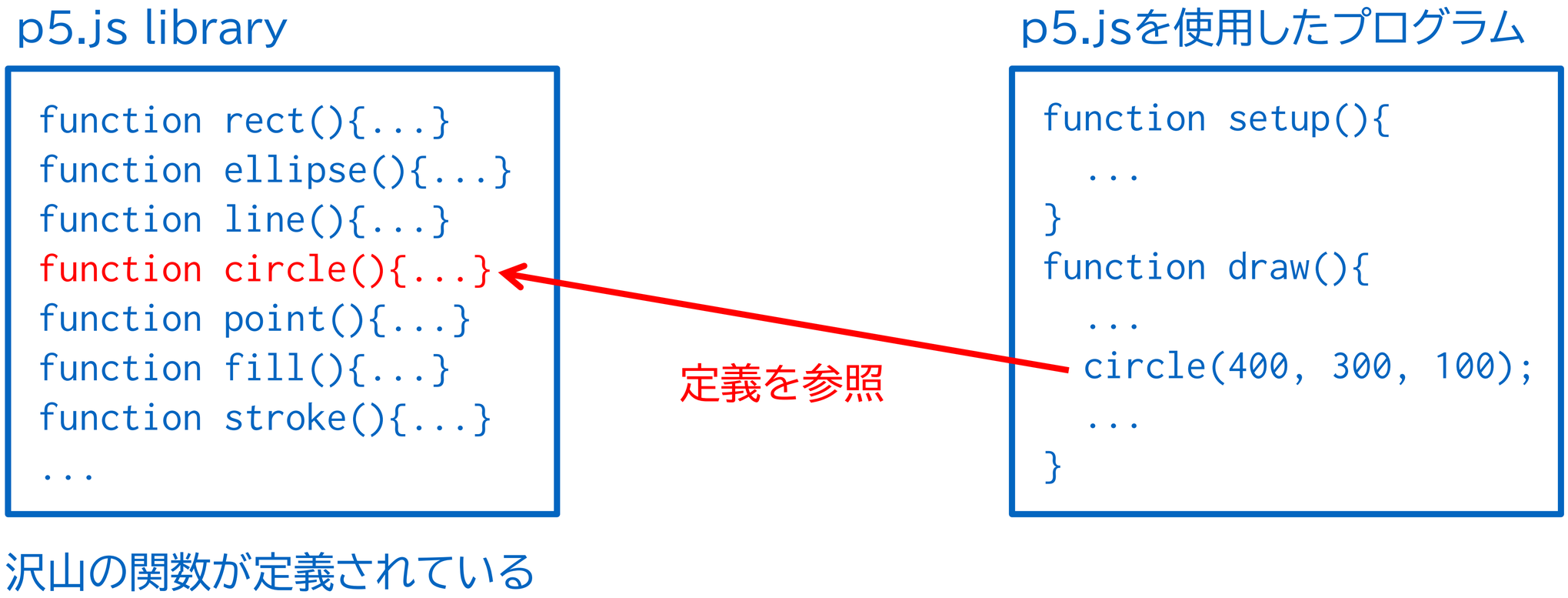
```
// circle関数の呼び出し
circle(400, 300, 100);
```

- p5.jsの関数の定義

```
// setup関数の定義
function setup() {
  createCanvas(windowWidth, windowHeight);
  frameRate(60);
}
```

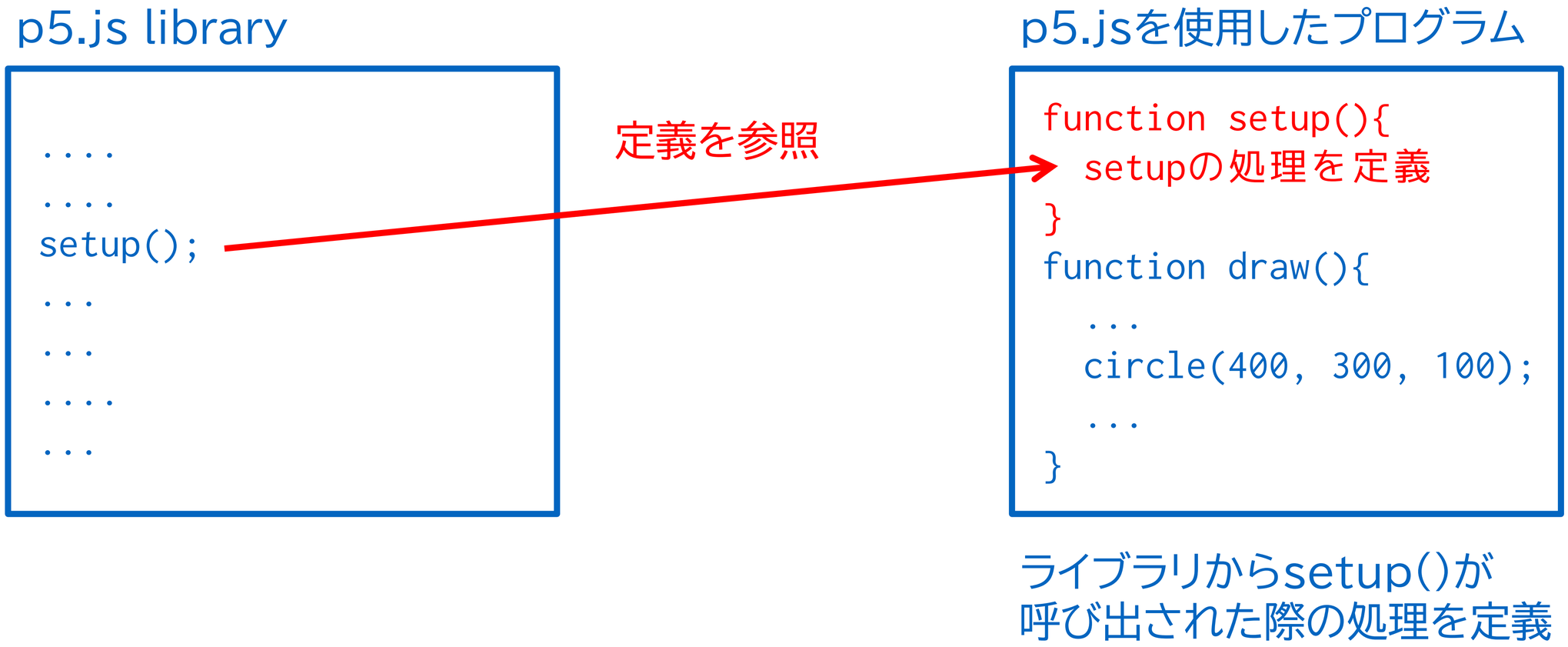
関数とは?

- p5.jsの関数の呼び出しのイメージ



関数とは?

- p5.jsの関数の呼び出しのイメージ



関数によるモジュール化

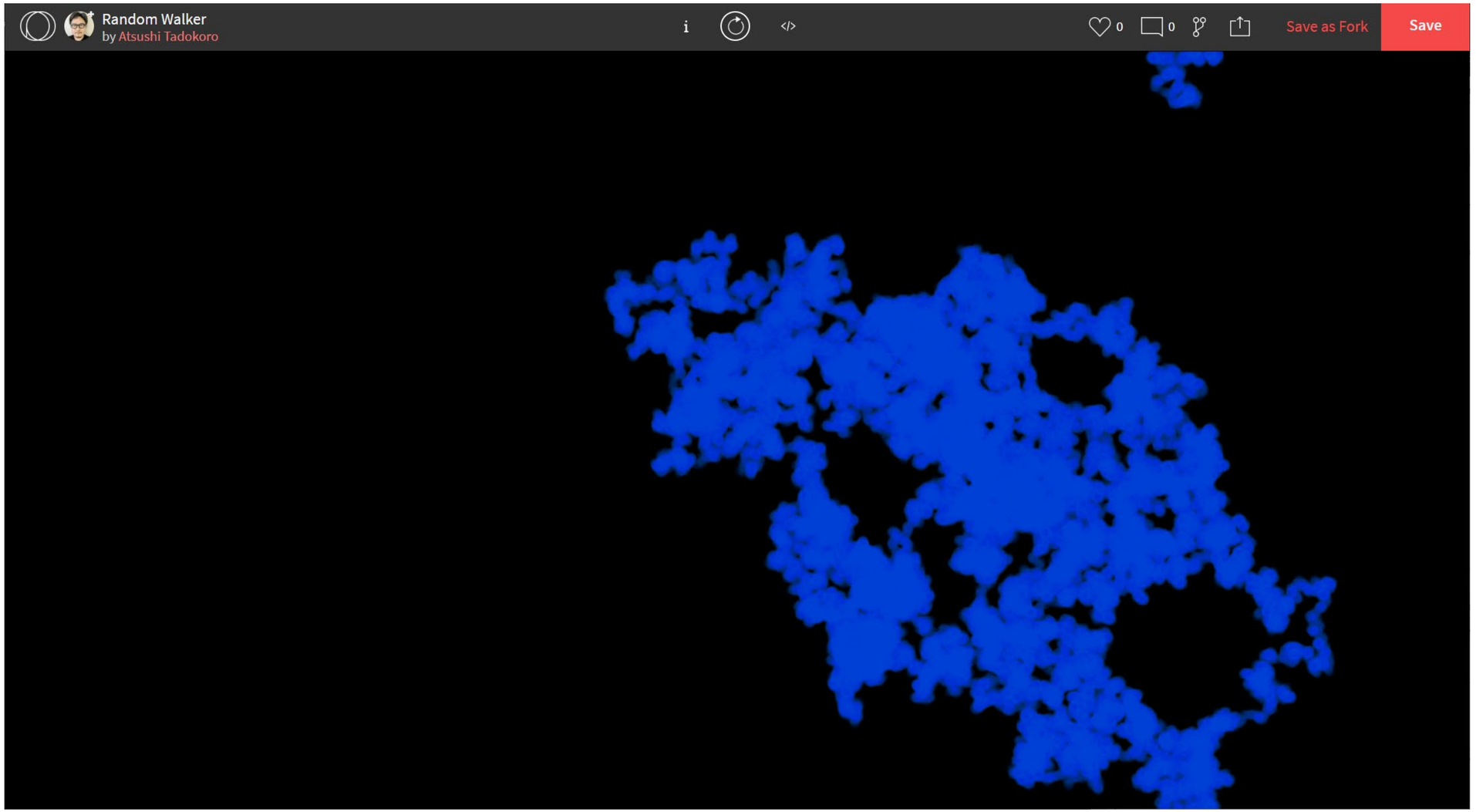
関数によるモジュール化

- 今回はこのプログラムからスタート
- forkして使用する

https://www.openprocessing.org/sketch/726079

関数によるモジュール化

- 雲のような模様を描くプログラム



関数によるモジュール化

- このプログラムはいくつかの手続から構成されている
- 初期化
  - 位置、大きさ、動く速さ、半径を設定
- 円の位置の移動
  - ランダムな方向へ動く
- 画面の端に来た際の処理
  - 画面の端に来たら反対側から出現
- 円を描く

関数によるモジュール化

- それぞれの処理を関数にしてみる
  - オリジナルの関数を定義する
  - 関数名は独自に決めてよい
  - できるだけ処理の内容がわかる関数名をつけるよう心掛ける

- 例:
  - 初期化 → init();
  - 円の位置の移動 → move();
  - 画面の端に来た際の処理 → throughWall();
  - 円を描く → display();

関数によるモジュール化

- draw()の下に、まずは関数定義の枠組みを記述

```
function init(){
}

function move(){
}

function throughWall(){
}

function display(){
}
```

関数によるモジュール化

- 該当する処理をコピペして移動する

```
function init() {
  ball.x = random(width);
  ball.y = random(height);
  ball.radius = 10;
  ball.speed = 800;
  background(0);
}

function move() {
  ball.x += random(-1, 1);
  ball.y += random(-1, 1);
}

function throughWall() {
  if (ball.x > width) {
    ball.x = 0;
  }
  if (ball.x < 0) {
    ball.x = width;
  }
  if (ball.y > height) {
    ball.y = 0;
  }
  if (ball.y < 0) {
    ball.y = height;
  }
}

function display() {
  noStroke();
  fill(31, 127, 255);
  circle(ball.x, ball.y, ball.radius);
}
```

関数によるモジュール化

- コピペ後のsetup()とdraw()

```
function setup() {
  createCanvas(windowWidth, windowHeight);
  frameRate(60);
}

function draw() {
  background(0, 4);
  for (let i = 0; i < ball.speed; i++) {

  }
}
```

関数によるモジュール化

- 該当する場所から、関数を呼び出す

```
function setup() {
  createCanvas(windowWidth, windowHeight);
  frameRate(60);
  init();
}

function draw() {
  background(0, 4);
  for (let i = 0; i < ball.speed; i++) {
    move();
    throughWall();
    display();
  }
}
```

引数による関数のバリエーション

引数による関数のバリエーション

- ここまでの関数定義の例では、常に同じ処理をしていた
- 関数には引数（arguments）でパラメータを渡すことができる
- 引数を使ってバリエーションをつけてみる

引数による関数のバリエーション

- 同心円を描く関数 guruguru() を作ってみる!

- 想定されるパラメータ
  - 中心位置（circleLocation）※p5.vector
  - 大きさ（radius）
  - 同心円の数（numCircles）
  - 色（circleColor）

- まずは関数を使わずに、そのまま作成してみる

引数による関数のバリエーション

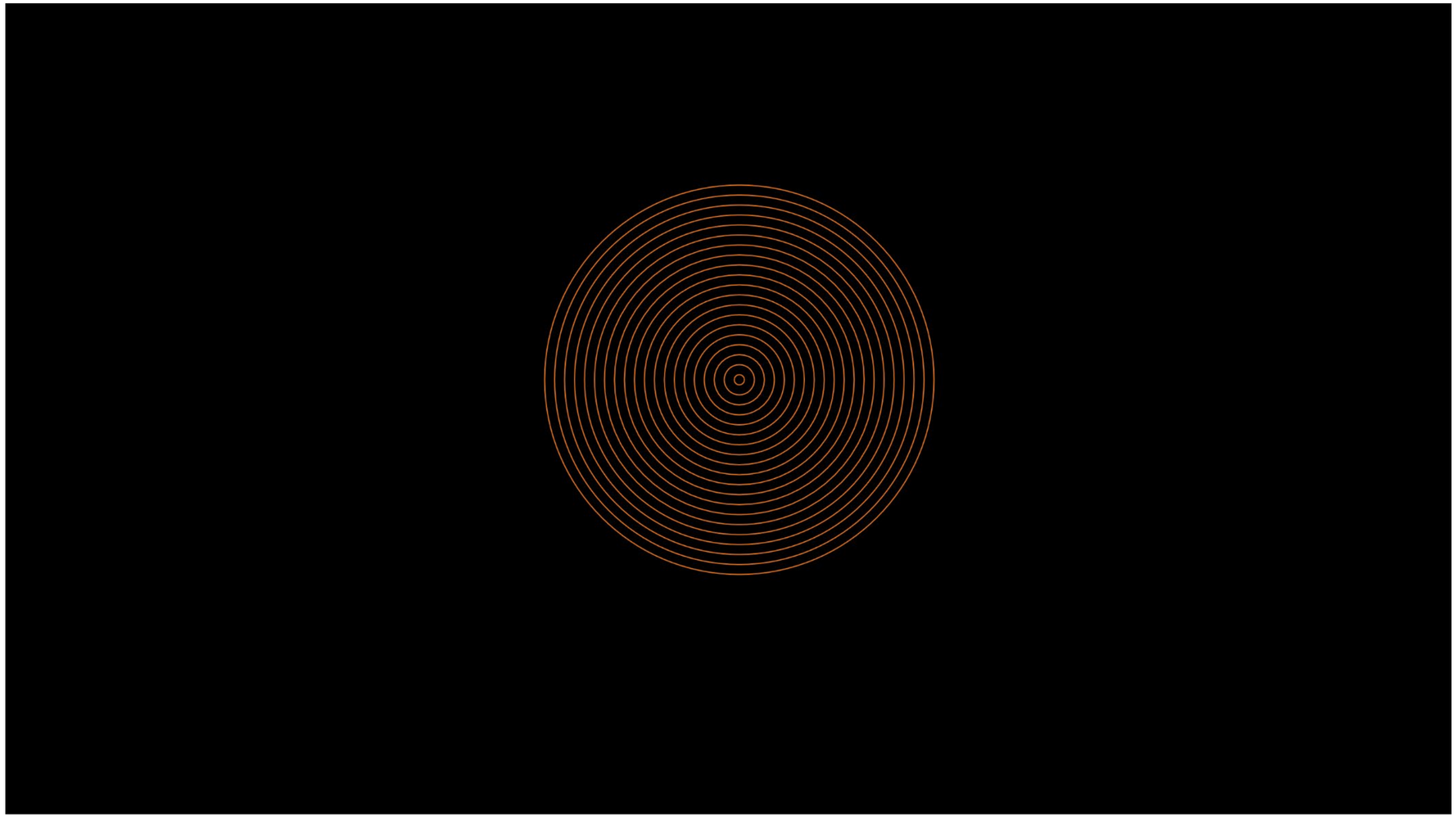
- 関数を使わないバージョン

```
function setup() {
    createCanvas(windowWidth, windowHeight);
    frameRate(60);
}

function draw() {
    background(0);
    let circleLocation = createVector(width/2, height/2);
    let numCircles = 20;
    let radius = 400;
    let circleColor = color(255, 127, 31);
    noFill();
    stroke(circleColor);
    for(let i = 0; i < numCircles; i++){
        let currentRadius = map(i, 0, numCircles, 10, radius);
        circle(circleLocation.x, circleLocation.y, currentRadius);
    }
}
```

引数による関数のバリエーション

- 同心円が描けた!



引数による関数のバリエーション

- Step 1 - 1
- 同心円を描いている部分をそのまま関数 guruguru() として外に出す

引数による関数のバリエーション

- guruguru() 関数（引数なし）

```
function guruguru(){
    let circleLocation = createVector(width/2, height/2);
    let numCircles = 20;
    let radius = 400;
    let circleColor = color(255, 127, 31);
    noFill();
    stroke(circleColor);
    for(let i = 0; i < numCircles; i++){
        let currentRadius = map(i, 0, numCircles, 10, radius);
        circle(circleLocation.x, circleLocation.y, currentRadius);
    }
}
```

引数による関数のバリエーション

- Step 1 – 2
- 引数無しの guruguru() を呼び出す

引数による関数のバリエーション

- guruguru() 関数の呼び出し

```
function setup() {
    createCanvas(windowWidth, windowHeight);
    frameRate(60);
}

function draw() {
    background(0);
    guruguru();
}
```

引数による関数のバリエーション

- Step 2 - 1
- 引数を渡せるようにしてみる
  - 中心位置（circleLocation）※p5.vector
  - 大きさ（radius）
  - 同心円の数（numCircles）
  - 色（circleColor）

引数による関数のバリエーション

- guruguru() 関数（引数あり）

```
function guruguru(circleLocation, numCircles, radius, circleColor){
    noFill();
    stroke(circleColor);
    for(let i = 0; i < numCircles; i++){
        let currentRadius = map(i, 0, numCircles, 10, radius);
        circle(circleLocation.x, circleLocation.y, currentRadius);
    }
}
```

引数による関数のバリエーション

- Step 2 - 2
- 引数を指定して、guruguru() 関数を呼び出してみる

引数による関数のバリエーション

- 引数を指定して guruguru() 関数を呼び出し

```
function setup() {
    createCanvas(windowWidth, windowHeight);
    frameRate(60);
}

function draw() {
    background(0);
    guruguru(createVector(width/2, height/2), 20, 400, color(255, 127, 31));
}
```

引数による関数のバリエーション

- いろいろなバリエーションの guruguru を描いてみる!

引数による関数のバリエーション

- guruguruのバリエーション

```
function setup() {
    createCanvas(windowWidth, windowHeight);
    frameRate(60);
}

function draw() {
    background(0);
    guruguru(createVector(600, 300), 20, 400, color(255, 127, 31));
    guruguru(createVector(500, 400), 5, 200, color(0, 255, 127));
    guruguru(createVector(750, 250), 30, 250, color(0, 31, 255));
}
```

引数による関数のバリエーション

- 大量に guruguru を生成
- draw()の中でくりかえしguruguru()関数を呼び出してみる!

引数による関数のバリエーション

- guruguruを大量生成!

```
function setup() {
  createCanvas(windowWidth, windowHeight);
  colorMode(HSB, 360, 100, 100, 100);
  background(0);
}

function draw(){
    let x = random(width);
    let y = random(height);
    let h = random(80, 240);
    let s = 80;
    let b = 100;
    let a = 80;
    guruguru(createVector(x, y), random(10), random(width/20), color(h, s, b, a));
}

function guruguru(circleLocation, numCircles, radius, circleColor){
  noFill();
  stroke(circleColor);
  strokeWeight(2.0);
  for(let i = 0; i < numCircles; i++){
    let currentRadius = map(i, 0, numCircles, 10, radius);
    circle(circleLocation.x, circleLocation.y, currentRadius);
  }
}
```

引数による関数のバリエーション

- ちょっと応用編
- たくさんの花を描いてみる
- それぞれの花は微妙に違うように

引数による関数のバリエーション

- たくさんの花を描いてみる – 解答例

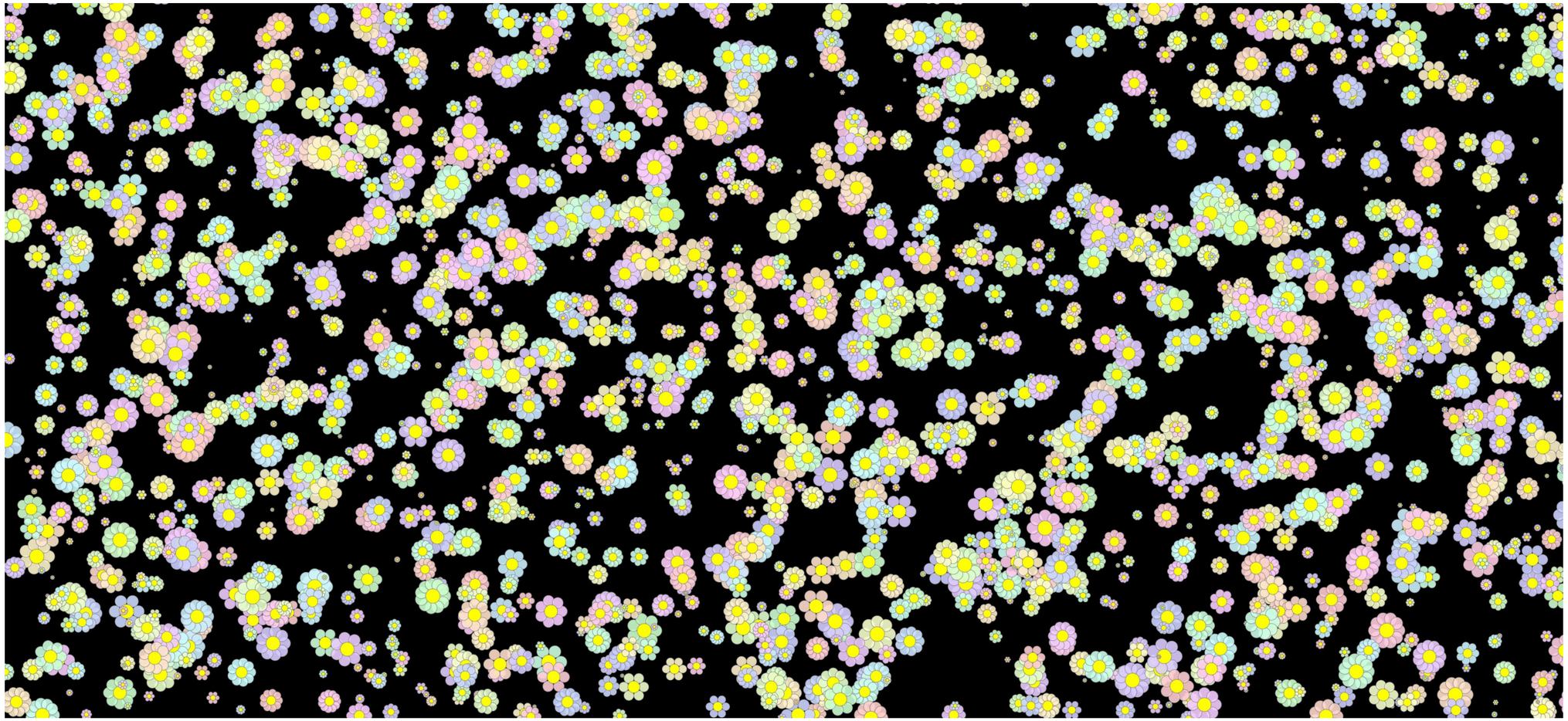
```
function setup() {
    createCanvas(windowWidth, windowHeight);
    colorMode(HSB, 360, 100, 100, 100);
    background(0);
}

function draw(){
let center = createVector(random(width), random(height));
let numPetal = int(random(4, 18));
let radius = random(width/120, width/80);
flower(center, numPetal, radius);
}

function flower(center, numPetal, radius){
    stroke(0, 0, 50, 70);
    fill(random(360), 20, 100, 90);
    push();
    translate(center.x, center.y);
    for(let i = 0; i < numPetal; i++){
        push();
        rotate(2 * PI/numPetal * i);
        circle(radius, 0, radius);
        pop();
    }
    fill(60, 1000, 100);
    circle(0, 0, radius * 1.2);
    pop();
}
```

引数による関数のバリエーション

- たくさんの花を描いてみる – 完成!!
- https://www.openprocessing.org/sketch/726406



本日の課題: 関数をつかって形を増殖

本日の課題: 関数をつかって形を増殖

- 課題: 関数をつかって、大量に形を増殖させる

- 何か形を描く関数を作成
  - 引数でバリエーションをつけられるように
  - 位置、色、形、数など

- 作成した関数を呼出して、大量に形を描いてみる
- 最後に紹介した、たくさんの花を描く関数を参考に

- 完成した作品は、OpenProcessingに投稿して投稿フォームから提出