メディアアート・プログラミング Ⅰ
第9回: p5.js オブジェクト指向プログラミング2
コンストラクターと、オブジェクトのバリエーション

2020年7月10日

東京藝術大学芸術情報センター（AMC）
田所淳

本日の内容

- p5.jsでOOP（オブジェクト指向プログラミング）その2
  - OOP復習
  - プロパティーを変化させる
  - コンストラクターと引数
  - インスタス化で引数を渡して、オブジェクトにバリエーションをつける
  - オブジェクトとイタラクション

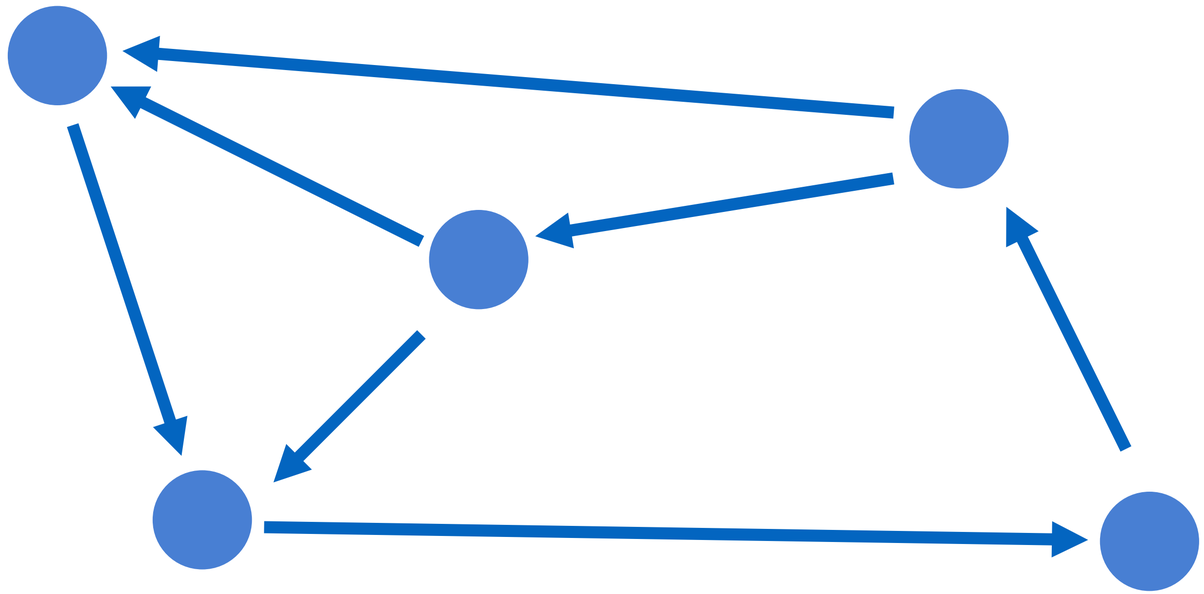
OOPについて（復習）

OOPについて（復習）

- プログラムの中の独立した機能の単位 →「オブジェクト(Object)」と呼ぶ

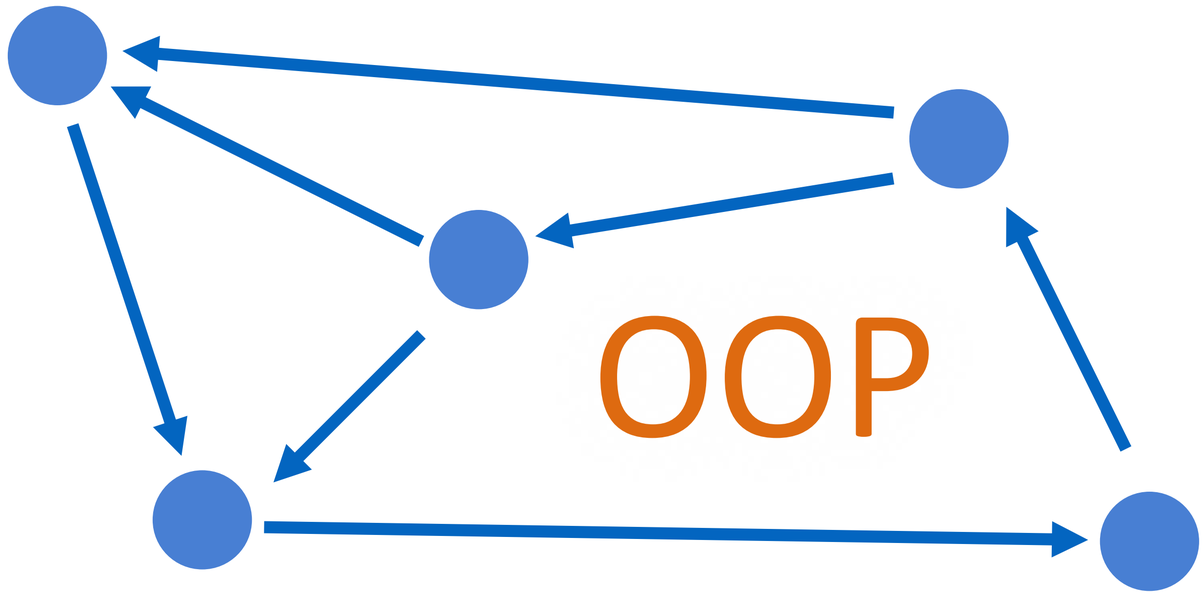
OOPについて（復習）

- オブジェクトは、それぞれ自律していて、相互にメッセージを送りあう



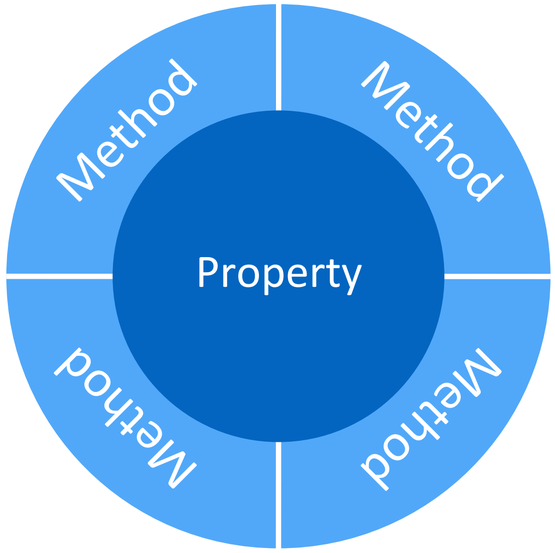
OOPについて（復習）

- こうした、オブジェクトをプログラムの構成単位とするプログラム手法
- → オブジェクト指向プログラミング（Object Oriented Programming）と呼ぶ



OOPについて（復習）

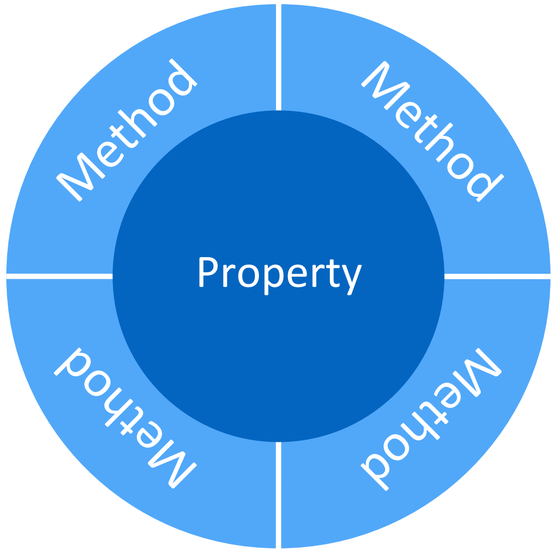
- OOPを構成する単位「Object」に注目
- Objectは、「状態（プロパティー）」と「動作（メソッド）」で特徴を記録する



Object

OOPについて（復習）

- 状態 → 値を記録する ＝ 変数(Variables)
- 動作 → 一連の処理をまとめる ＝ 関数(Functions)

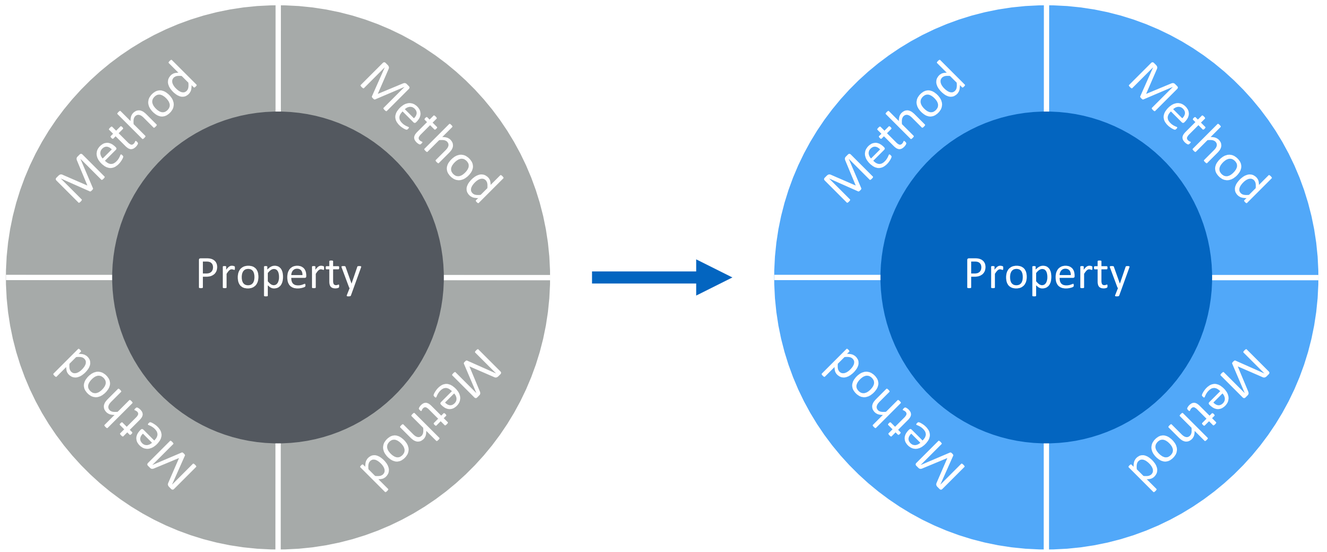


Object

- Property → Variable
- Method → Function

OOPについて（復習）

- OOPでは、オブジェクトの設計図をまず書く
- オブジェクトの設計図のことをクラス（Class）と呼ぶ



OOPについて（復習）

- P5.jsのクラスの基本構造

```
class MyClass {
    //コンストラクター
    constructor(){
      <クラスの初期化の処理>
    }
    //関数を宣言
    function myFunction(){
      ...
    }
    ...
}
```

OOPについて（復習）

- コンストラクター（Constructor）とは？
- クラスが初期化される際に呼びだされる特別な関数
- 関数名は、constructor()

OOPについて（復習）

- 今回は、このコンストラクターについて、さらに掘り下げていきます!

開始サンプル（非OOP）

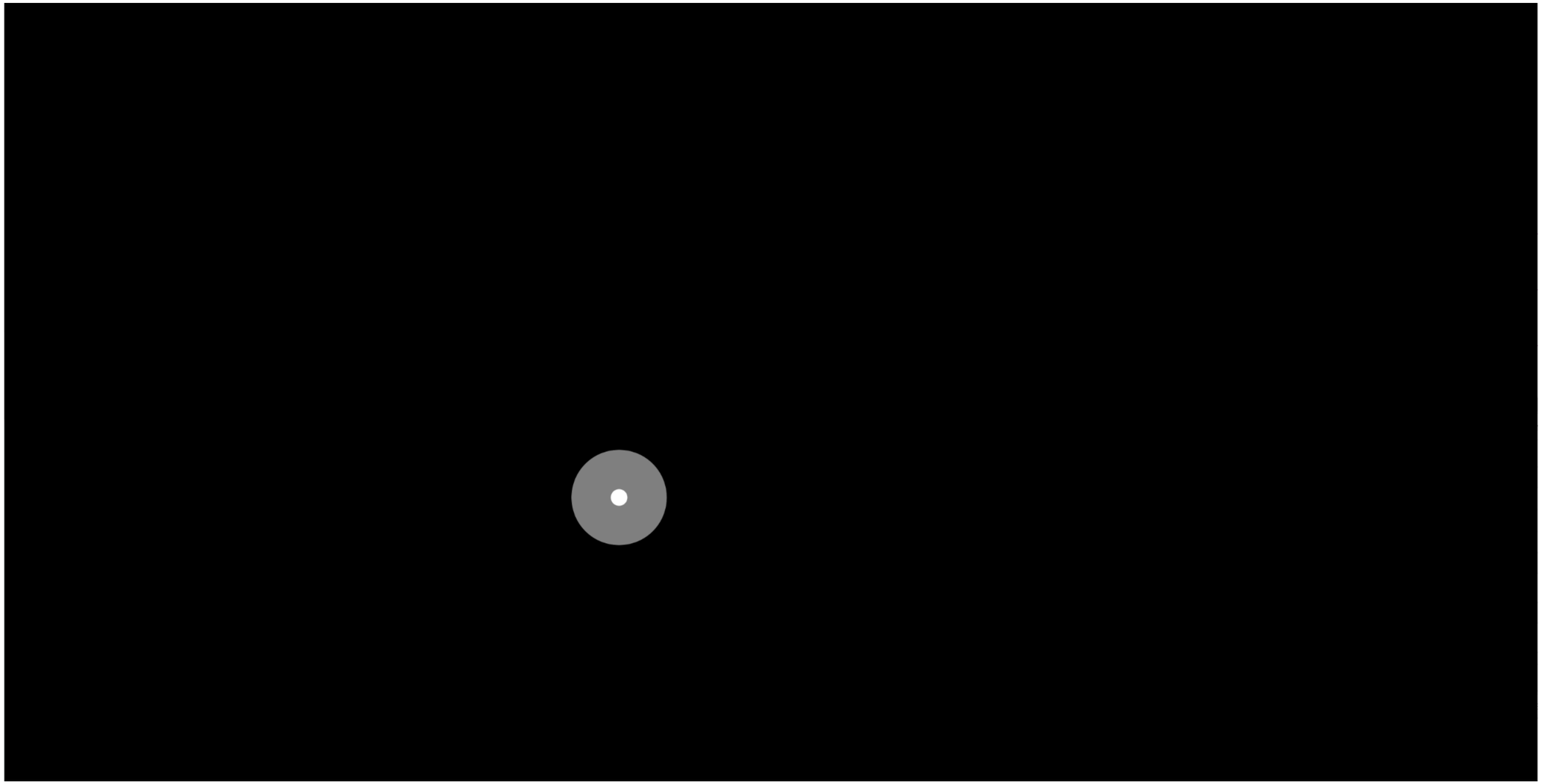
開始サンプル（非OOP）

- まず、ここからスタート https://www.openprocessing.org/sketch/730517

```
let diameter; //直径
let position; //位置
let velocity; //速度

function setup() {
    //初期設定
    createCanvas(windowWidth, windowHeight);
    diameter = 100.0;
    position = createVector(width / 2, height / 2);
    velocity
        = createVector(random(-2, 2), random(2, 2));
}

function draw() {
    background(0);
    position.add(velocity); //位置を更新
    //壁でバウンド
    if (position.x < diameter / 2
        || position.x > width - diameter / 2) {
        velocity.x *= -1;
    }
    if (position.y < diameter / 2
        || position.y > height - diameter / 2) {
        velocity.y *= -1;
    }
    //色の設定
    noStroke();
    fill(255, 127);
    //プヨプヨする動きを計算
    currentDiameter
        = sin(frameCount * 0.1) * (diameter / 4) + diameter;
    //周囲の円を描画
    circle(position.x, position.y, currentDiameter);
    //核となる円を描画
    fill(255);
    circle(position.x, position.y, diameter / 6.0);
}
```

開始サンプル（非OOP）

- プヨプヨと伸縮しながら動きまわる物体



開始サンプル（非OOP）

- draw()内を関数に分けて機能を整理 https://www.openprocessing.org/sketch/730519

```
let diameter; //直径
let position; //位置
let velocity; //速度

function setup() {
    createCanvas(windowWidth, windowHeight);
    diameter = 100.0;
    position = createVector(width / 2, height / 2);
    velocity = createVector(random(-2, 2), random(2, 2));
}

function draw() {
    background(0);
    move(); //移動
    bounce(); //壁でバウンド
    display(); //表示
}

function move() {
    position.add(velocity);
}

function bounce() {
    if (position.x < diameter / 2 || position.x > width -
diameter / 2) {
        velocity.x *= -1;
    }
    if (position.y < diameter / 2 || position.y > height -
diameter / 2) {
        velocity.y *= -1;
    }
}

function display() {
    noStroke();
    fill(255, 127);
    currentDiameter = sin(frameCount * 0.1) * (diameter / 4) +
diameter;
    circle(position.x, position.y, currentDiameter);
    fill(255);
    circle(position.x, position.y, diameter / 6.0);
}
```

クラスにまとめる（OOP）

クラスにまとめる（OOP）

- プヨプヨと動く物体（Blob）をクラスにまとめる → Blobクラス

- プロパティ（状態、変数）
  - 直径（diameter）
  - 位置（position）
  - 速度（velocity）

- メソッド（動作、関数）
  - constractor() ※現状はsetup()
  - move()
  - bounce()
  - display()

クラスにまとめる（OOP）

- まずBlobクラスの骨組を書いてみる

```
class Blob {
    constructor(){
        this.diameter;
        this.position;
        this.velocity;
    }
    move(){

    }
    bounce(){

    }
    display(){

    }
}
```

クラスにまとめる（OOP）

- それぞれのメソッドを本体から移植していく
  - constractor() ※現状はsetup()
  - move()
  - bounce()
  - display()

- クラス全体で使用される変数には “this.” が先頭に追加されることに注意!!
  - this.diameter
  - this.position
  - this.velocity

クラスにまとめる（OOP）

- Blobクラス

```
class Blob {
    constructor(){
        this.diameter = 100.0;
        this.position = createVector(width / 2, height / 2);
        this.velocity = createVector(random(-2, 2), random(2, 2));
    }
    move(){
        this.position.add(this.velocity);
    }
    bounce(){
        if (this.position.x < this.diameter / 2 || this.position.x > width - this.diameter / 2) {
            this.velocity.x *= -1;
        }
        if (this.position.y < this.diameter / 2 || this.position.y > height - this.diameter / 2) {
            this.velocity.y *= -1;
        }
    }
    display(){
        noStroke();
        fill(255, 127);
        let currentDiameter = sin(frameCount * 0.1) * (this.diameter / 4) + this.diameter;
        circle(this.position.x, this.position.y, currentDiameter);
        fill(255);
        circle(this.position.x, this.position.y, this.diameter / 6.0);
    }
}
```

クラスにまとめる（OOP）

- 作成したBlobクラスをインスタンス化(実体化)して、動かしてみる

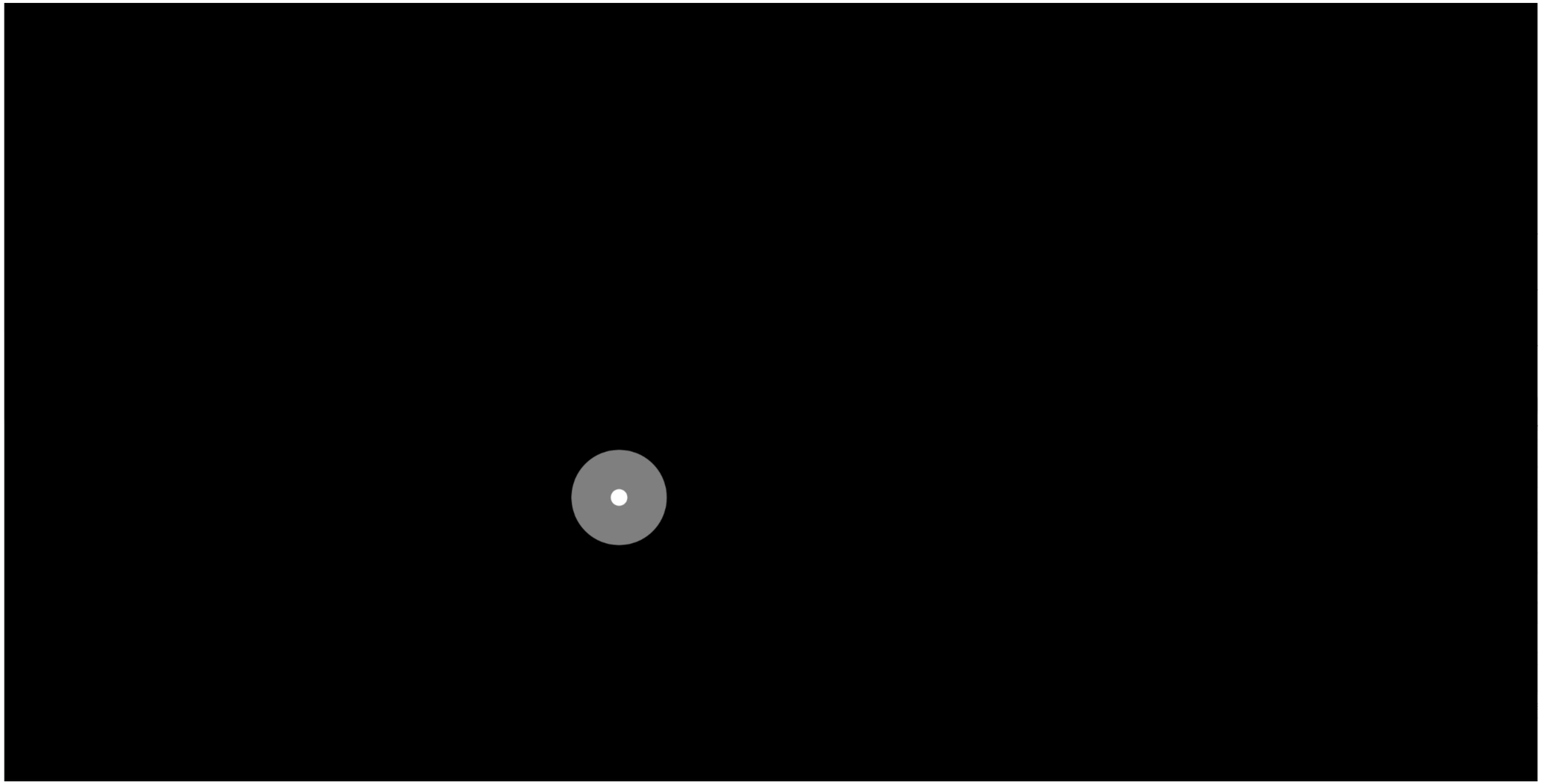
クラスにまとめる（OOP）

- Blobクラスを呼び出す

```
let blob;

function setup() {
    createCanvas(windowWidth, windowHeight);
    blob = new Blob(); //インスタンス化
}

function draw() {
    background(0);
    //Blobクラスのメソッドを実行
    blob.move();
    blob.bounce();
    blob.display();
}
```

クラスにまとめる（OOP）

- オブジェクト版のBlobが動いた https://www.openprocessing.org/sketch/730520



オブジェクトのプロパティーを変更する

オブジェクトのプロパティーを変更する

- クラスをインスタンス化(初期化)した後で、プロパティーを変更することも可能
- やってみよう!

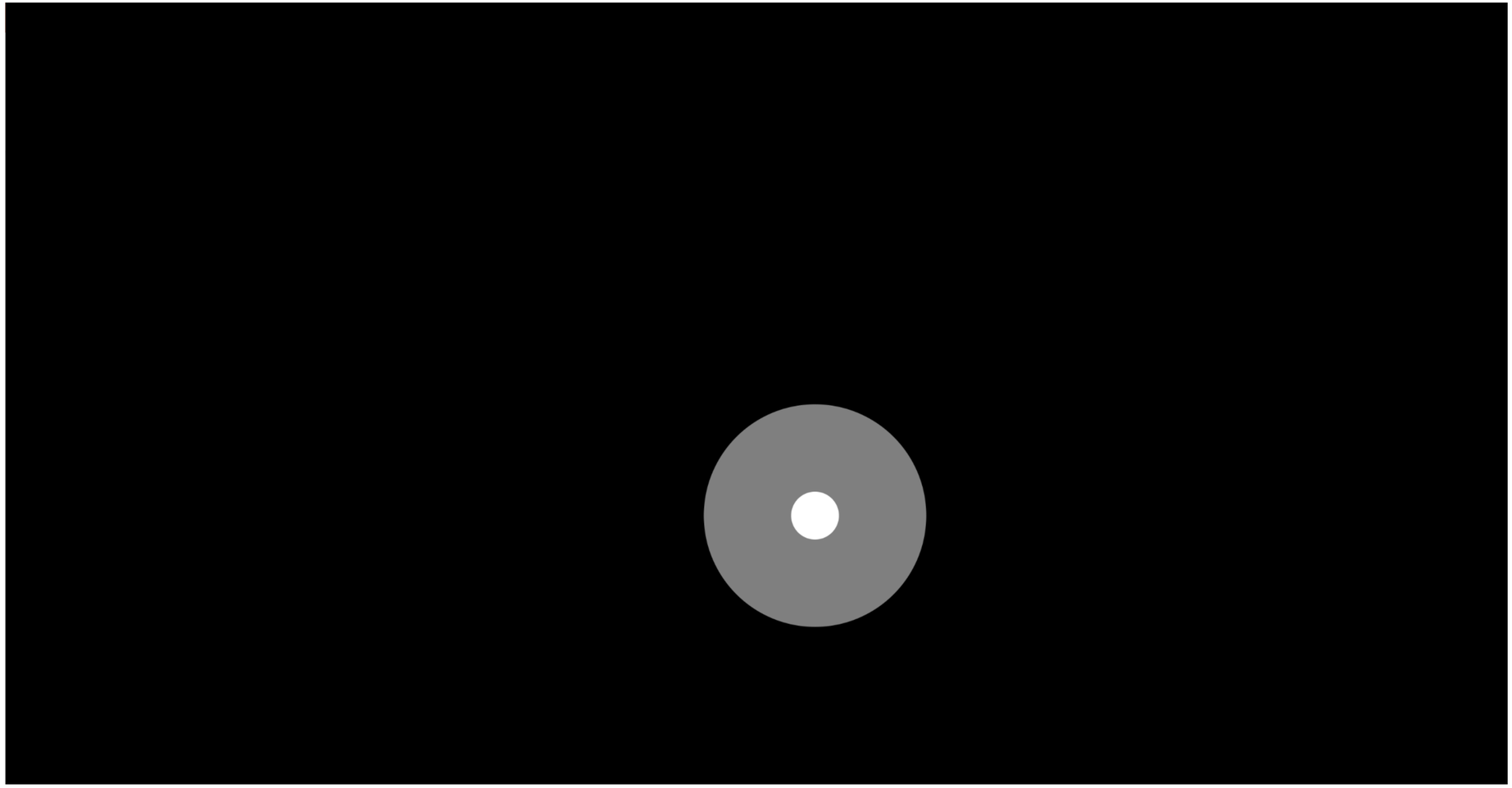
オブジェクトのプロパティーを変更する

- Blobクラスを呼び出す

```
let blob;

function setup() {
    createCanvas(windowWidth, windowHeight);
    blob = new Blob(); //インスタンス化
    blob.diameter = 300; //サイズを変更
    blob.velocity = createVector(10, 15); //速度を変更
}

function draw() {
    background(0);
    //Blobクラスのメソッドを実行
    blob.move();
    blob.bounce();
    blob.display();
}
```

クラスにまとめる（OOP）

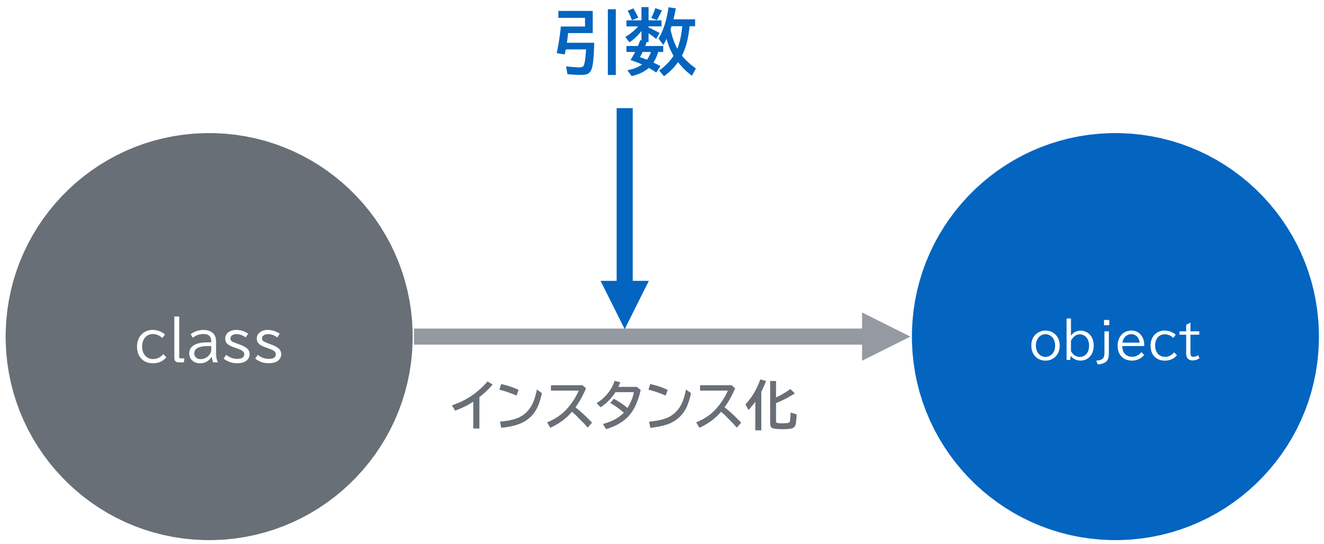
- 大きさと速度が変化した



コンストラクターの引数からパラメーターを渡す

コンストラクターの引数からパラメーターを渡す

- クラスがインスタンス化するタイミングでパラメータを設定することが可能
  - クラスをインスタンス化してからプロパティーを指定するのではない
  - クラスのコンストラクターの引数として指定する



コンストラクターの引数からパラメーターを渡す

- コンストラクターで値を受けとれるようにするには
  - コンストラクターの定義に引数を追加
  - 受け取った値を、クラス全体の変数（this.xxx）に適用する

コンストラクターの引数からパラメーターを渡す

- Blobクラスのconstructor()に引数を追加

```
class Blob {
    constructor(diameter, position, velocity){
        this.diameter = diameter;
        this.position = position;
        this.velocity = velocity;
    }

...(後略)...
```

コンストラクターの引数からパラメーターを渡す

- インスタンス化の際に値を引数として受け渡す

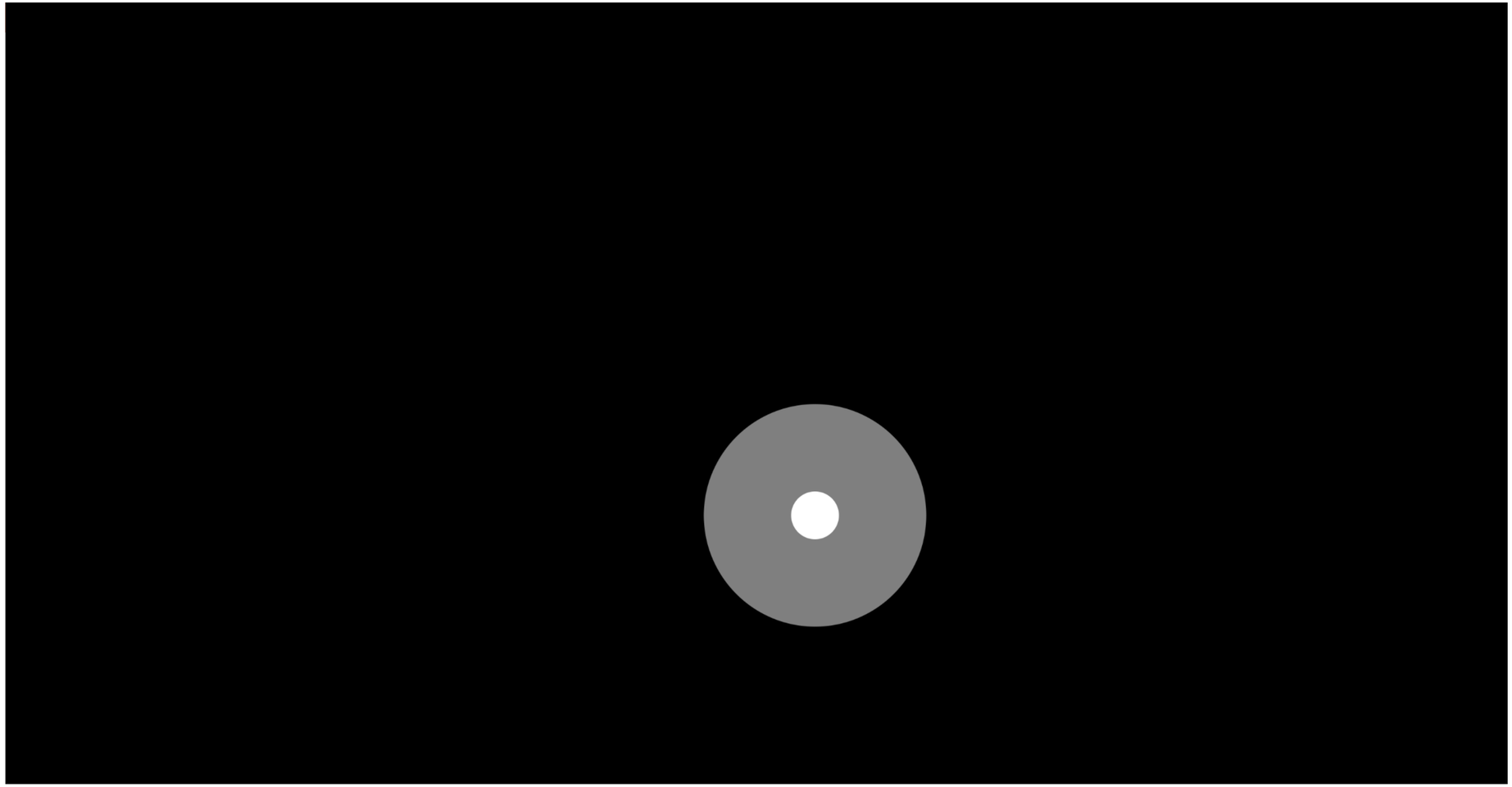
コンストラクターの引数からパラメーターを渡す

- 引数を指定してインスタンス化

```
let blob;

function setup() {
    createCanvas(windowWidth, windowHeight);
    //引数を指定してインスタンス化（直径，位置，速度）
    blob = new Blob(100, createVector(200, 200), createVector(-10, 5));
}

...(後略)...
```

コンストラクターの引数からパラメーターを渡す

- インスタンス化で指定したパラメータでオブジェクトが生成された



1つのクラスから様々なバリエーションのオブジェクトを大量生成

1つのクラスから様々なバリエーションのオブジェクトを大量生成

- クラスは工場のようなもの
- 1つの工場をつくってしまえば、大量の車を次々と生産できる
- コンストラクターに引数をつけることで、生産する車のバリエーションも増やせる!!

- Blobクラスで試してみる!

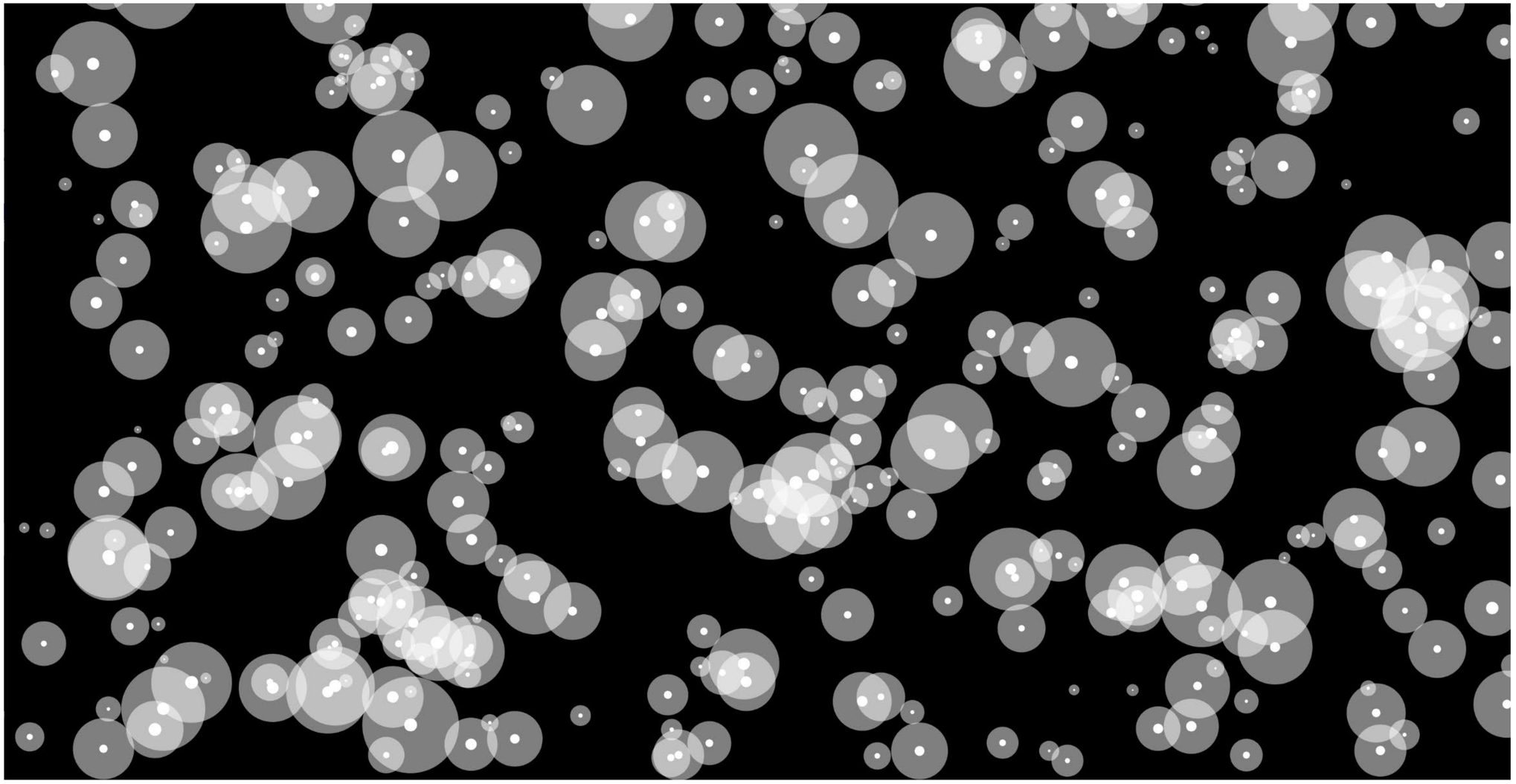
コンストラクターの引数からパラメーターを渡す

- 1つのクラスから様々なパラメータを適用したオブジェクトを生成

```
let blob = []; //オブジェクトの配列
const num = 400; //オブジェクトの数

function setup() {
    createCanvas(windowWidth, windowHeight);
    for(let i = 0; i < num; i++){
        blob.push(
            new Blob(
                random(10, 80),
                createVector(random(100, width-100), random(100, height-100)),
                createVector(random(-5, 5), random(-5, 5)))
        );
    }
}

function draw() {
    background(0);
    //Blobクラスのメソッドを実行
    for(let i = 0; i < num; i++){
        blob[i].move();
        blob[i].bounce();
        blob[i].display();
    }
}
```

1つのクラスから様々なバリエーションのオブジェクトを大量生成

- 様々なバージョンのBlobが動き回る! https://www.openprocessing.org/sketch/730534



よりインタラクティブに!

よりインタラクティブに!

- ユーザーのアクションを取り込んでみる

- 例えば…
  - ユーザーが画面上をクリックすると、新たなBlobが生成される
  - マウスをクリックした場所から生成される

- やってみよう!

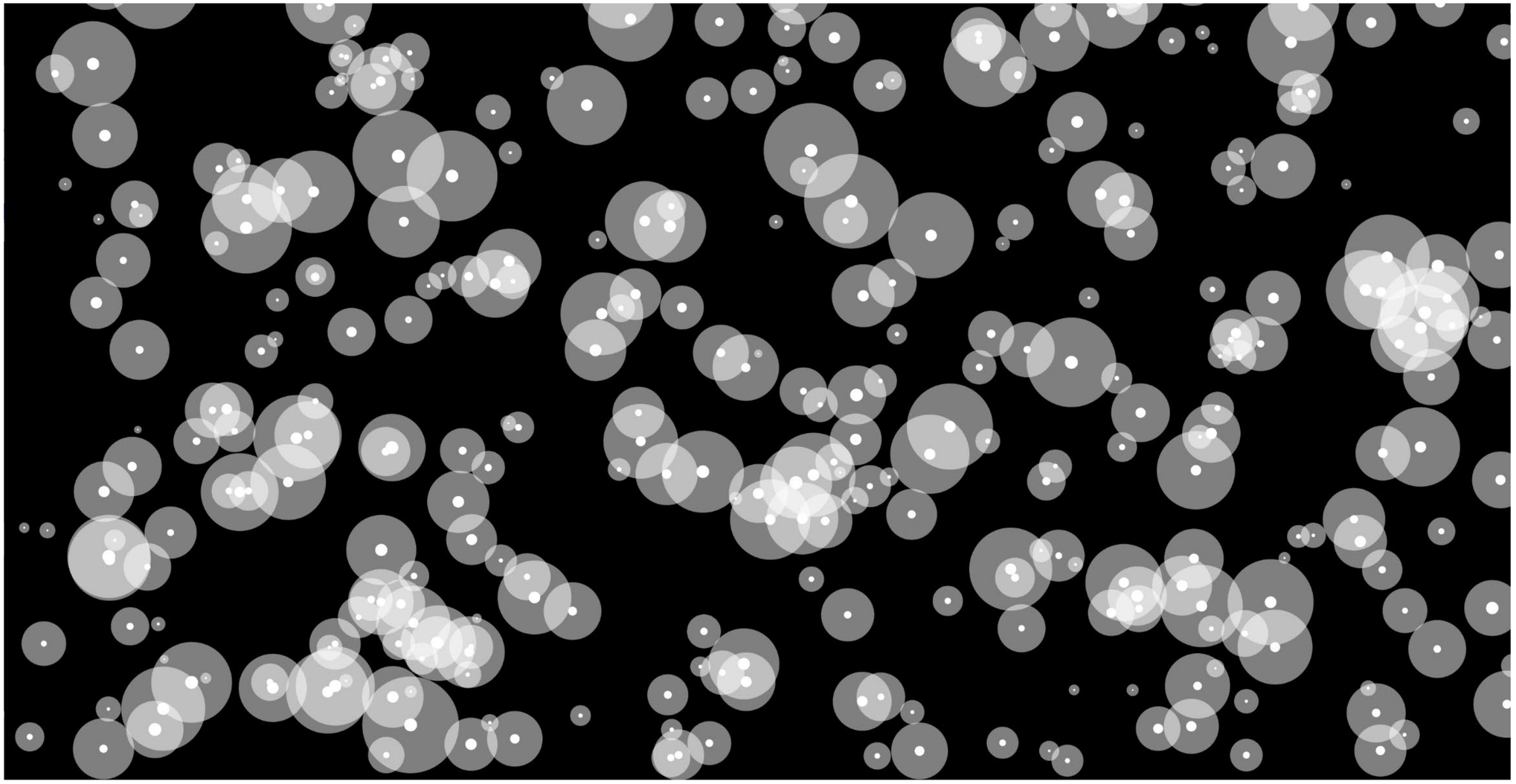
よりインタラクティブに!

- クリックした場所からBlobを生成

```
let blob = []; //オブジェクトの配列

function setup() {
    createCanvas(windowWidth, windowHeight);
}

function draw() {
    background(0);
    //Blobクラスのメソッドを実行
    for(let i = 0; i < blob.length; i++){
        blob[i].move();
        blob[i].bounce();
        blob[i].display();
    }
}

function mouseClicked(){
    blob.push(
        new Blob(
            random(10, 80),
            createVector(mouseX, mouseY),
            createVector(random(-5, 5), random(-5, 5)))
    );
}
```

1つのクラスから様々なバリエーションのオブジェクトを大量生成

- 完成!! https://www.openprocessing.org/sketch/730539



本日の課題

本日の課題

- 今回作成したクラス「Blob」を改造して新たな作品を制作
  - コンストラクターから引数を受けとって、いろいろな種類のオブジェクトを生成
  - 大量にオブジェクトを生成してみる
  - 全然違う形や動きや色にしてしまってOK

- OpenProcessingに作成して、URLをオンラインフォームから提出