メディアアート・プログラミングI
第4回: 反復と乱数

2020年6月5日

東京藝術大学芸術情報センター（AMC）
田所 淳

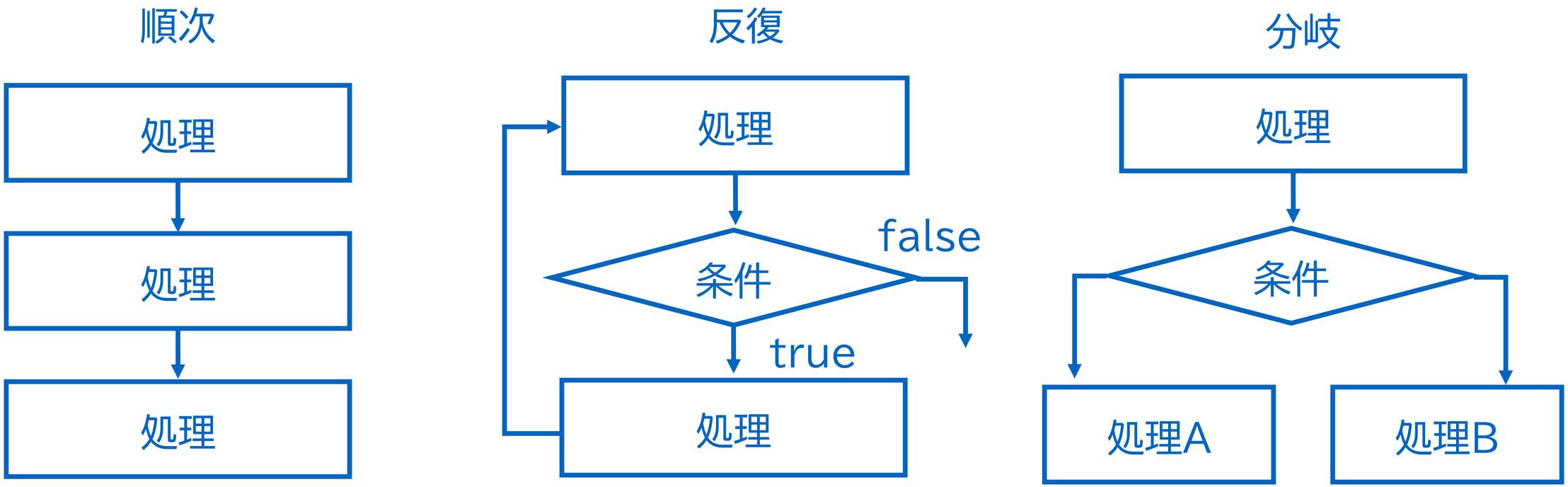
反復（Iteration）

反復（Iteration）

- コンピューターは、「くりかえし」の天才!
- 最近のマシンであれば、1秒間に2億回（2GHz）以上のクロックで、正確に計算をくりかえすことができる
- 反復は、プログラムの基本的で重要な構造のひとつ

反復（Iteration）

- 構造化プログラミング、3つの構成要素
- 順次（Sequence）、反復（Iteration）、分岐（Selection）



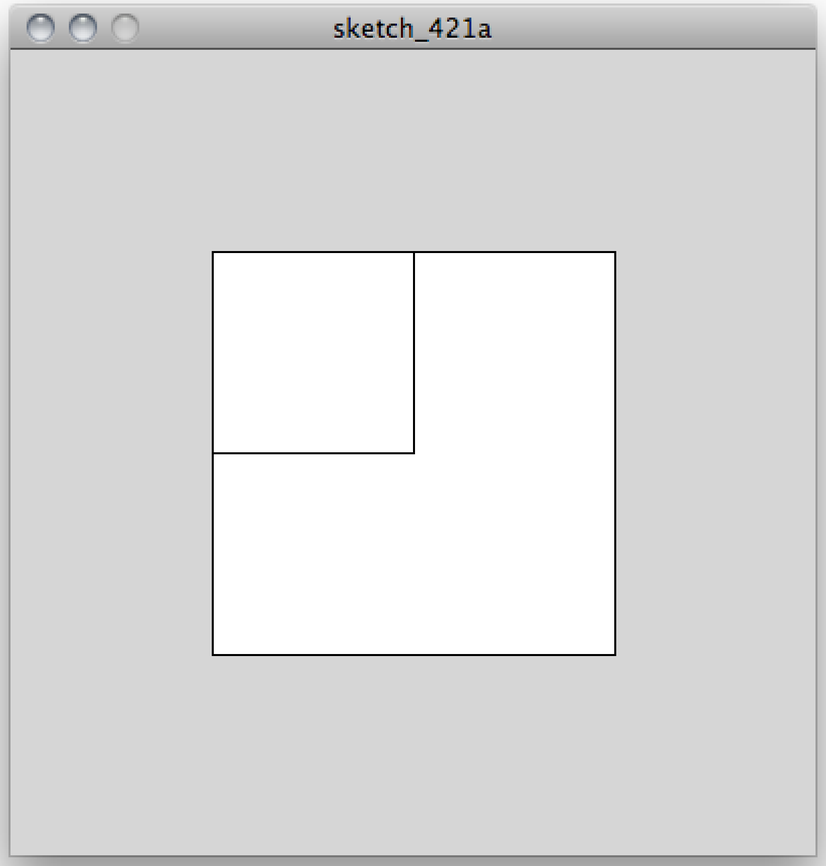
反復（Iteration）

- では、p5.jsをつかって、くりかえしの実験をしてみましょう!

変数

変数

- クイズ：
- 正方形の中に、ちょうど半分の幅と高さの正方形を描く



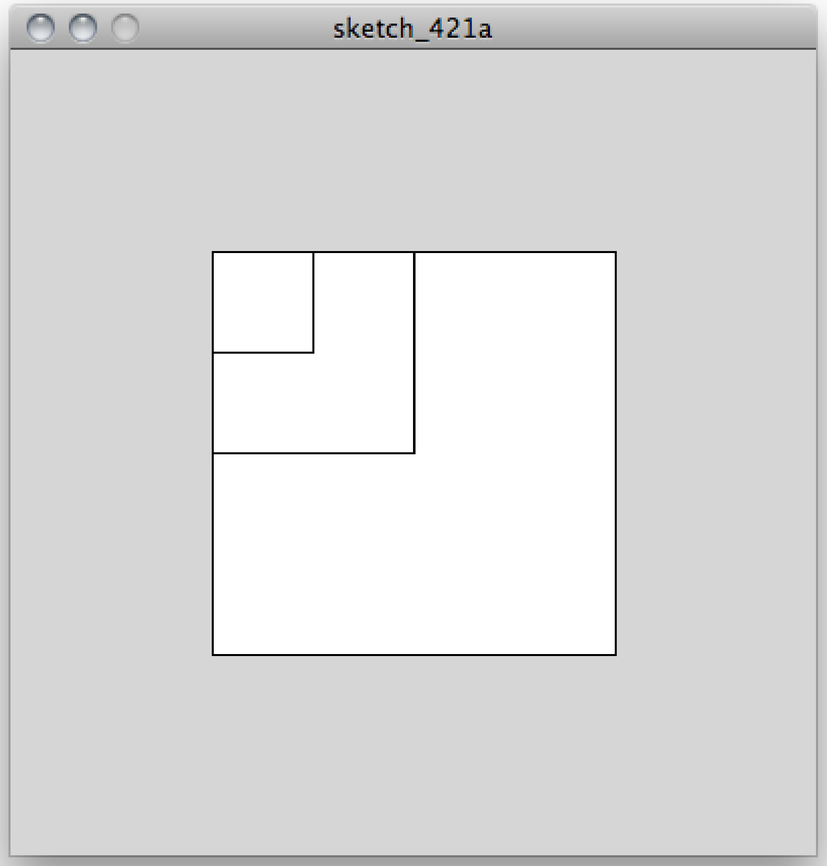
変数

▸ 答え

```
function setup() {
  createCanvas(400, 400);
}

function draw() {
  background(220);
  rect(100, 100, 200, 200);
  rect(100, 100, 100, 100);
}
```

変数

- クイズ2：
- さらに半分の正方形を中に



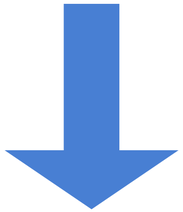
変数

▸ 答え

```
function setup() {
  createCanvas(400, 400);
}

function draw() {
  background(220);
  rect(100, 100, 200, 200);
  rect(100, 100, 100, 100);
  rect(100, 100, 50, 50);
}
```

変数

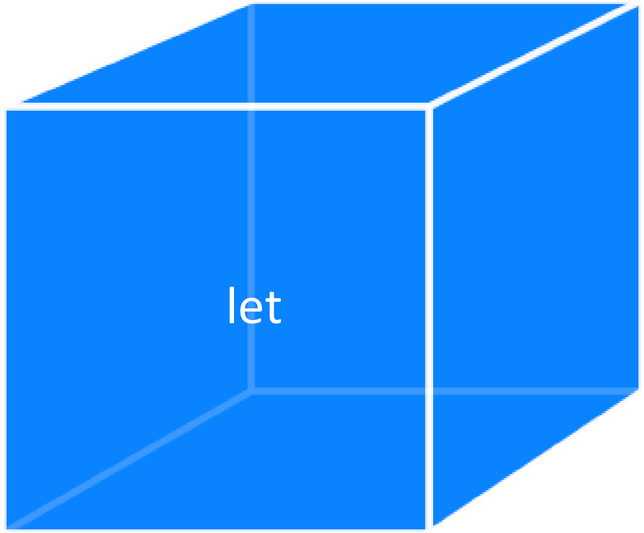
- 手で計算するのではなく、もっとかしこくできないか
- 描いた四角形の常に半分のサイズで次の四角形を描く
- 現在のサイズの値を記録できると便利なのに…



- 「値を記録する」→ 変数を利用する

変数

- 変数とは？
- 一時的に値（文字、文字列、数字など）を記憶しておく場所
- データを入れておく「箱」のようなもの
- JavaScriptでは let を用いる



変数

- 宣言:使用する変数の名前の箱を準備する

```
let hoo;
```

- 代入:変数の箱に値を入れる

```
let = 0;
```

- 演算:変数の値を計算する

```
hoo = hoo + 1;
```

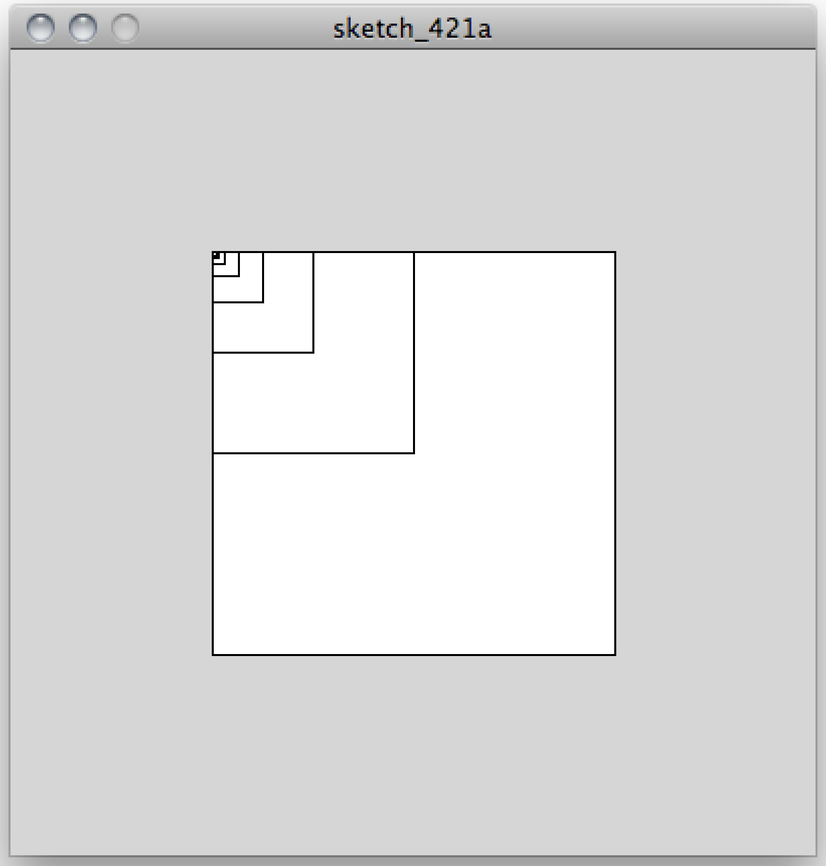
変数

- 先程の例は、変数を利用すると下記のようになる

```
function setup() {
  createCanvas(400, 400);
}

function draw() {
  background(220);

  let rectSize = 200;

  rect(100, 100, rectSize, rectSize);
  rectSize = rectSize / 2;
  rect(100, 100, rectSize, rectSize);
  rectSize = rectSize / 2;
  rect(100, 100, rectSize, rectSize);
  rectSize = rectSize / 2;
  rect(100, 100, rectSize, rectSize);
  rectSize = rectSize / 2;
  rect(100, 100, rectSize, rectSize);
  rectSize = rectSize / 2;
}
```

変数

- どんどん小さな正方形を描いていくことができる



変数

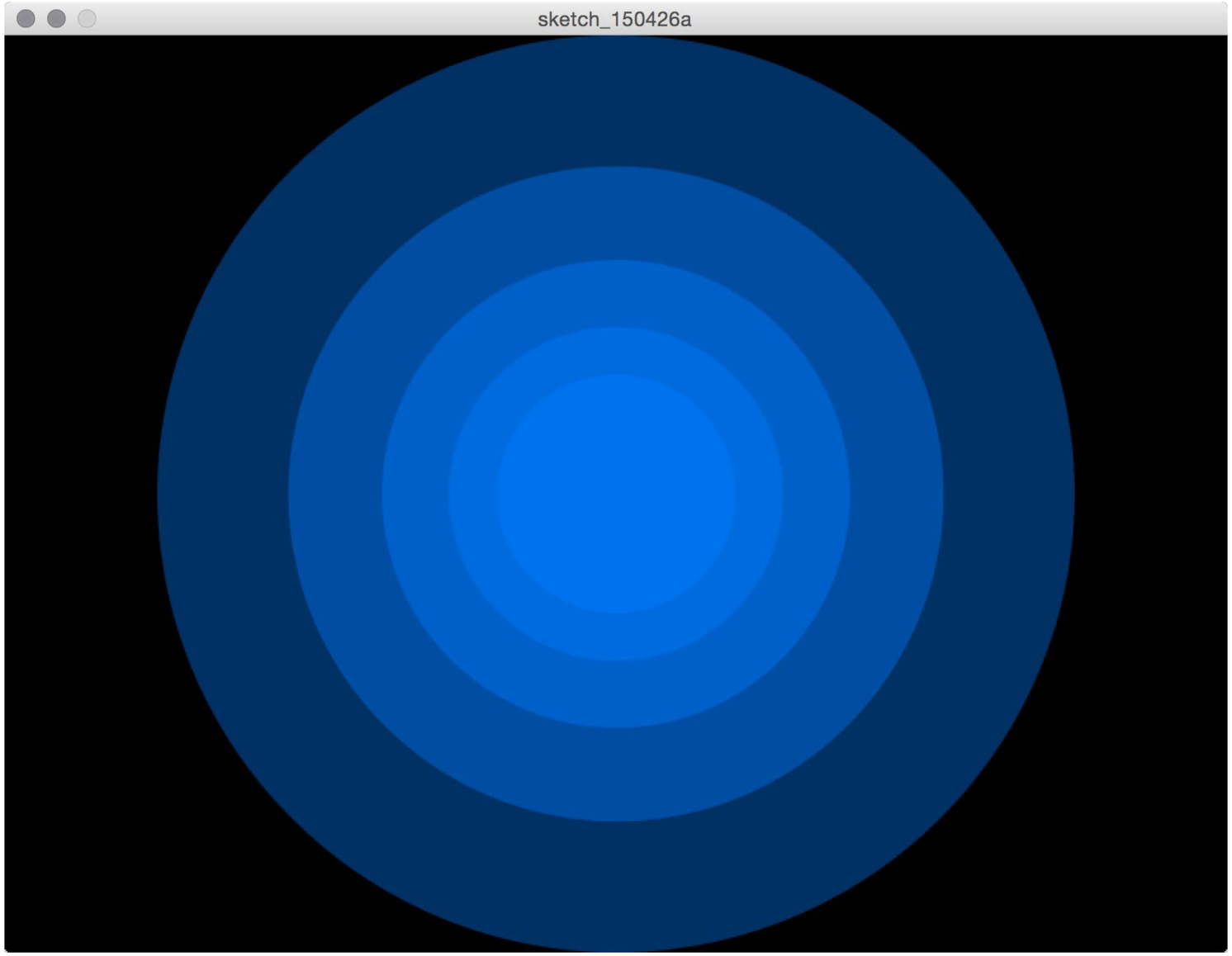
- 変数を利用して、いろいろ実験してみましょう

変数

- 変数の使用例

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

function draw() {
  background(0);
  noStroke();
  fill(0, 127, 255, 127);
  let x = width / 2;
  let y = height / 2;
  let diameter = height;
  ellipse(x, y, diameter, diameter);
  diameter = diameter / 1.4;
  ellipse(x, y, diameter, diameter);
  diameter = diameter / 1.4;
  ellipse(x, y, diameter, diameter);
  diameter = diameter / 1.4;
  ellipse(x, y, diameter, diameter);
  diameter = diameter / 1.4;
  ellipse(x, y, diameter, diameter);
}
```

変数



繰り返し

繰り返し

- さっきのプログラムに注目
- 途中から同じ命令のくりかえしになっていないだろうか？
- たとえば最初の例

```
ellipse(x, y, diameter, diameter);
diameter = diameter / 1.4;

ellipse(x, y, diameter, diameter);
diameter = diameter / 1.4;

ellipse(x, y, diameter, diameter);
diameter = diameter / 1.4;

ellipse(x, y, diameter, diameter);
diameter = diameter / 1.4;

ellipse(x, y, diameter, diameter);
diameter = diameter / 1.4;
...
```

繰り返し

- 繰り返しの部分を何度も書かずに一気に指定したい

```
ellipse(x, y, diameter, diameter);
diameter = diameter / 1.4;
```

これを10回くりかえし

繰り返し

- for文の書きかた

```
for（初期化式; 継続条件式; 再初期化式）{
  文;
}
```

- 初期化式：初期化の際の条件式
- 継続条件式：繰り返しを継続する条件式
- 再初期化式：繰り返されるたびに実行される式

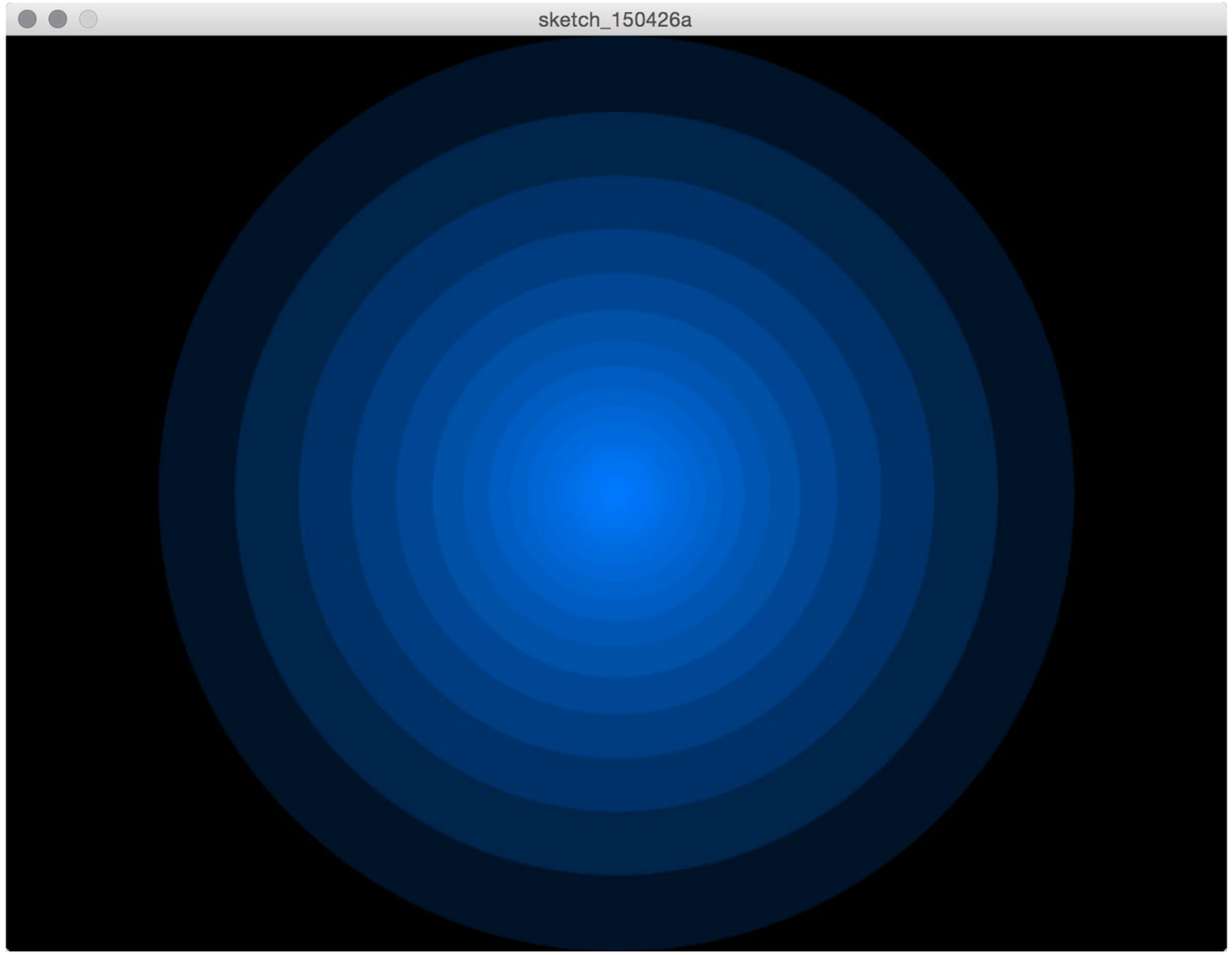
繰り返し

- 同心円をくりかえしを用いて描画

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

function draw() {
  background(0);
  noStroke();
  fill(0, 127, 255, 63);

  let x = width / 2;
  let y = height / 2;
  let diameter = height;

  for (let i = 0; i < 100; i++) {
    ellipse(x, y, diameter, diameter);
    diameter = diameter / 1.4;
  }
}
```

繰り返し

- 同心円をくりかえしを用いて描画

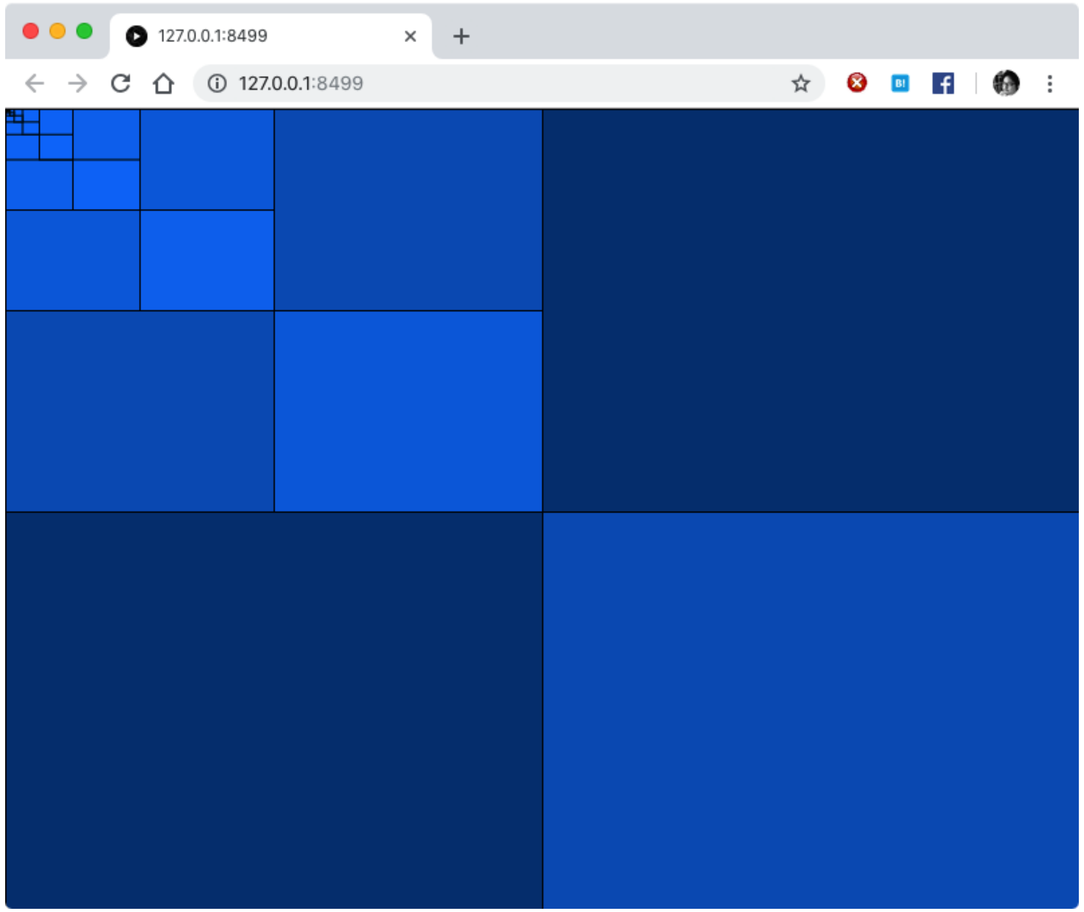


クイズ!!

- for分を使用したくりかえしを用いた表現
- 入れ子状になった形を生成してみる!

繰り返し

- 第1問：入れ子状の矩形



繰り返し

- ヒント：下記のコードから開始

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

function draw() {
  background(0);
  stroke(0);
  fill(0, 127, 255, 127);

  let x = 0;
  let y = 0;
  let w = width;
  let h = height;
}
```

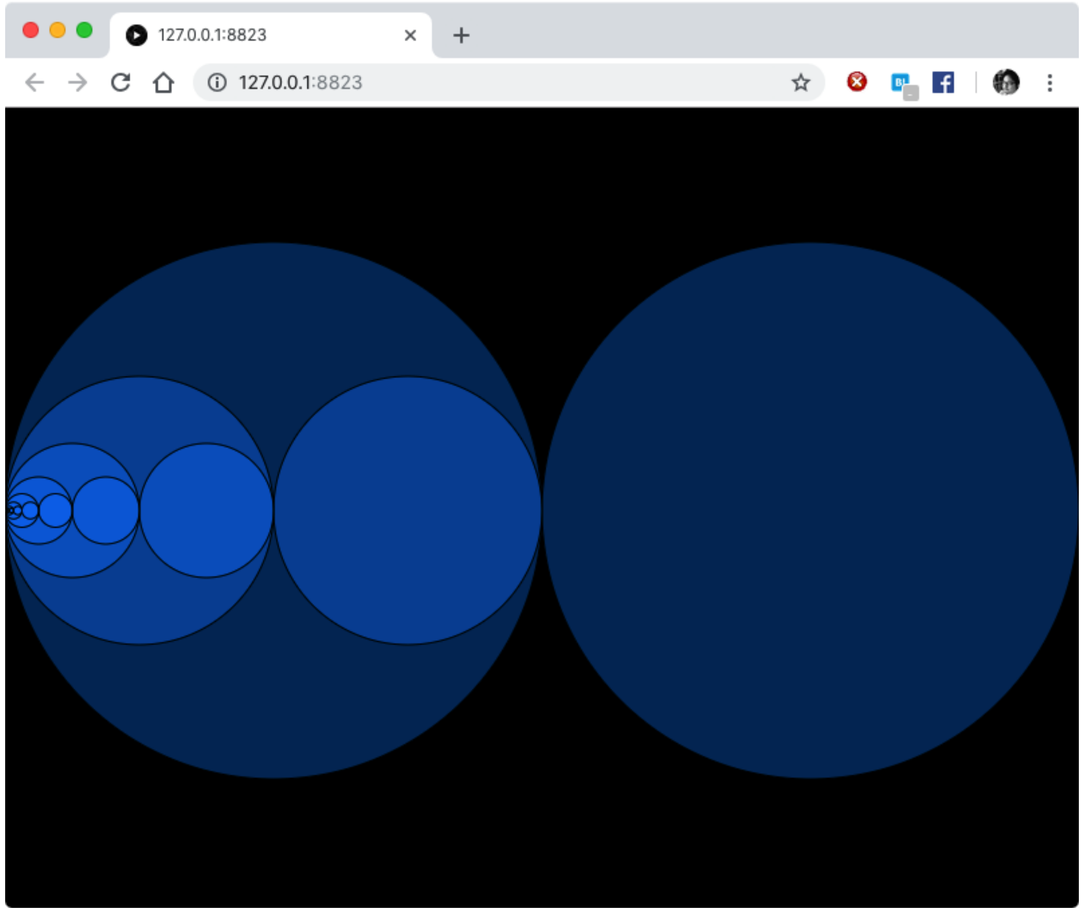
繰り返し

▸ 解答

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

function draw() {
  background(0);
  stroke(0);
  fill(0, 127, 255, 127);

  let x = 0;
  let y = 0;
  let w = width;
  let h = height;

  for (let i = 0; i < 32; i++) {
    rect(x, y, w, h);
    rect(x + w, y + h, w, h);
    w = w / 2;
    h = h / 2;
  }
}
```

繰り返し

- 第2問!! 入れ子状の円



繰り返し

- ヒント - 入れ子状の円

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

function draw() {
  background(0);
  stroke(0);
  fill(0, 127, 255, 100);

  let x = width / 4;
  let y = height / 2;
  let diameter = width / 2;
}
```

繰り返し

▸ 解答

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

function draw() {
  background(0);
  stroke(0);
  fill(0, 127, 255, 100);

  let x = width / 4;
  let y = height / 2;
  let diameter = width / 2;

  for (let i = 0; i < 32; i++) {
    ellipse(x, y, diameter, diameter);
    ellipse(x + diameter, y, diameter, diameter);
    diameter = diameter / 2;
    x = x / 2;
  }
}
```

乱数

乱数

- プログラムは、設定された手続きを何度でも繰り返す
- そのままでは、予想外の挙動はしない

- 完全に全てをコントロールするのではなく、偶然性、意外性をとり入れたい
- 実行する度に毎回異なる値を出力する仕組み

- 乱数
  - 何ら法則性、規則性のない、でたらめな値を出力
  - ランダム

乱数

- Processingで乱数を生成 - 乱数の範囲を指定する
- 例:0から100の乱数を生成

```
random(100);
```

- 例:100から1000の乱数を生成

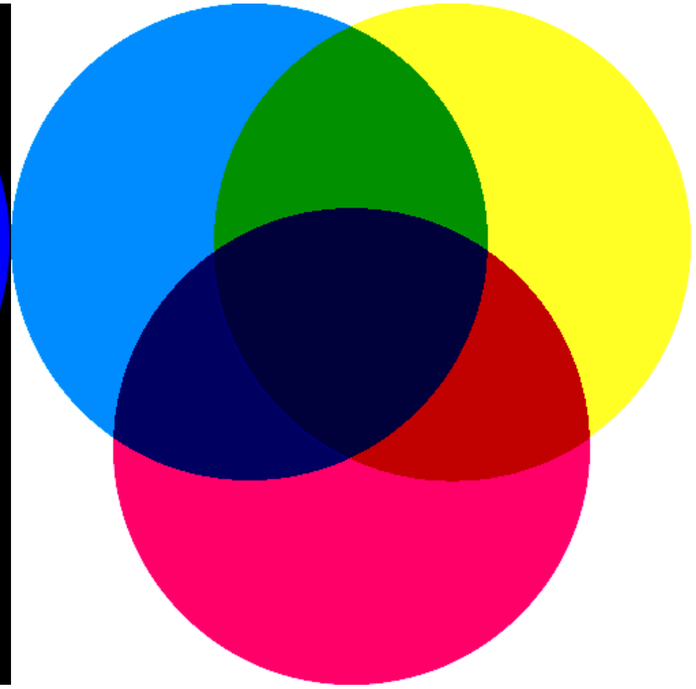
```
random(100, 1000);
```

色について - RGBモデルとHSBモデル

色について - 補足

- 色を指定するには？
  - R(赤) G(緑) B(青)の三原色で指定する
  - 加法混色（光の三原色であることに注意）←→ 色料の三原色

光の三原色



色料の三原色

色について - 補足

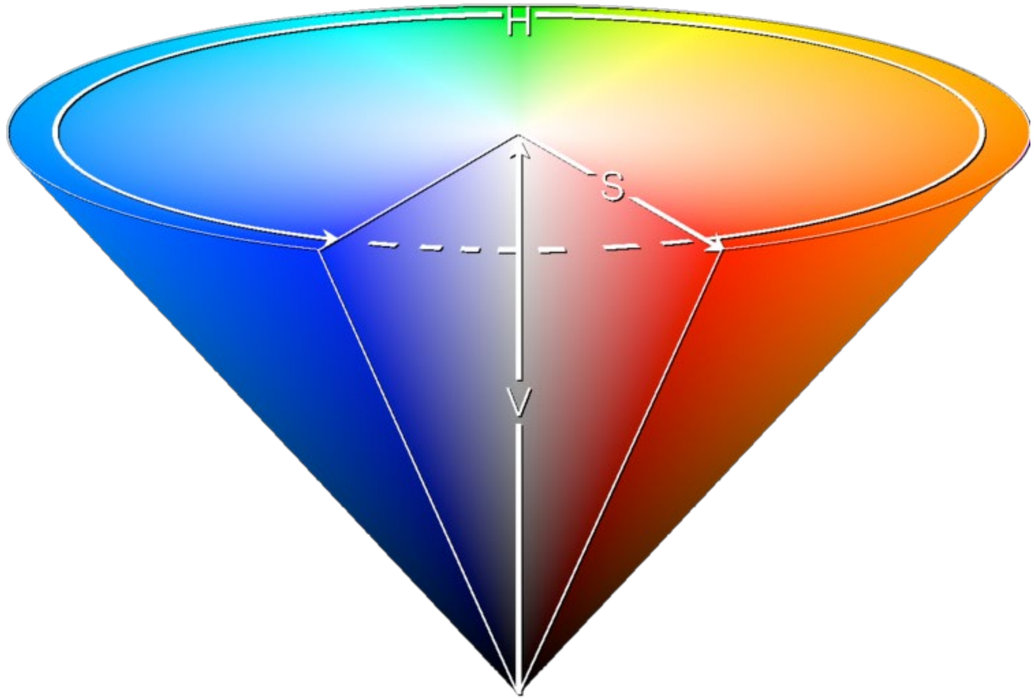
- 乱数で色を指定してみたい!
  - RGB以外の方法での色の指定したほうがやり易い

- HSB（HSV)による指定
  - 色相（Hue)：色の種類
  - 彩度（Saturation)：色の鮮かさ
  - 明度（Brightness)：色の明かるさ

- プログラミングしながら色を指定する場合は、HSBの方が直感的に望みの色を指定し易いことも多い

色について - 補足

- HSB（HSV）色空間の視覚イメージ
- 円錐のイメージ
  - 色相：外環の角度
  - 彩度：中心点からの距離
  - 明度：高さ



色について - 補足

- HSB（HSV）色空間への切替方法
- colorMode関数を使用する、それぞれのパラメータの範囲も同時に指定
- 例:
  - 色相:360階調(°)
  - 彩度:100階調(%)
  - 明度:100階調(%)

```
colorMode(HSB, 360, 100, 100);
```

乱数と繰り返しによる色彩と形態

乱数と繰り返しによる色彩と形態

- さらに色だけでなく、形や位置も乱数で制御してみる
- ランダムな形態を大量に生成すると、どのように見えるか
- やってみよう!

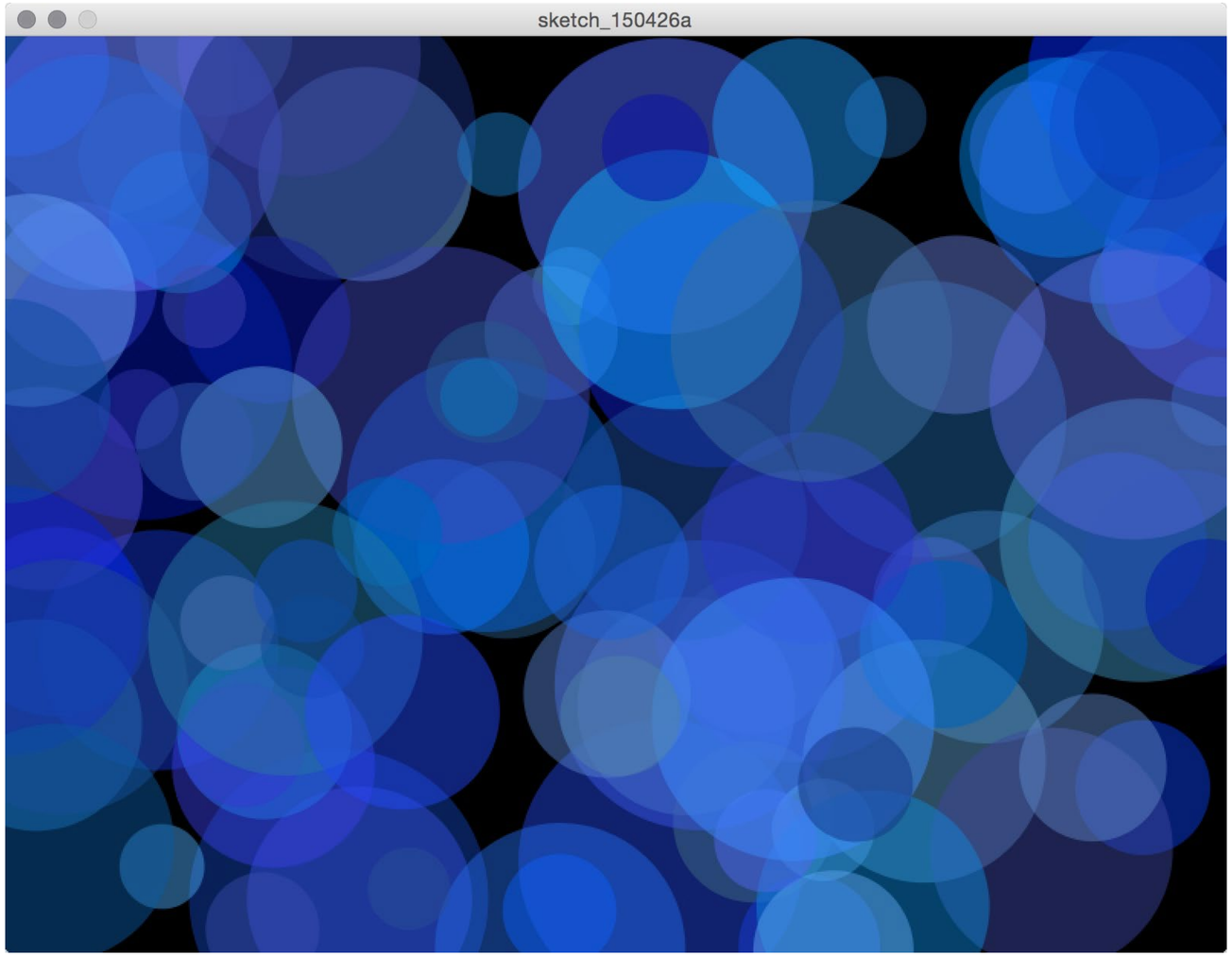
乱数と繰り返しによる色彩と形態

- たくさん円を描く

```
function setup() {
  createCanvas(windowWidth, windowHeight);
  colorMode(HSB, 360, 100, 100, 100);
  background(0);
  noStroke();
  for (let i = 0; i < 100; i++) {
    fill(random(200, 240), 90, 90, 50);
    circle(random(width), random(height), random(40, 80));
  }
}
```

乱数と繰り返しによる色彩と形態

- たくさん円を描く



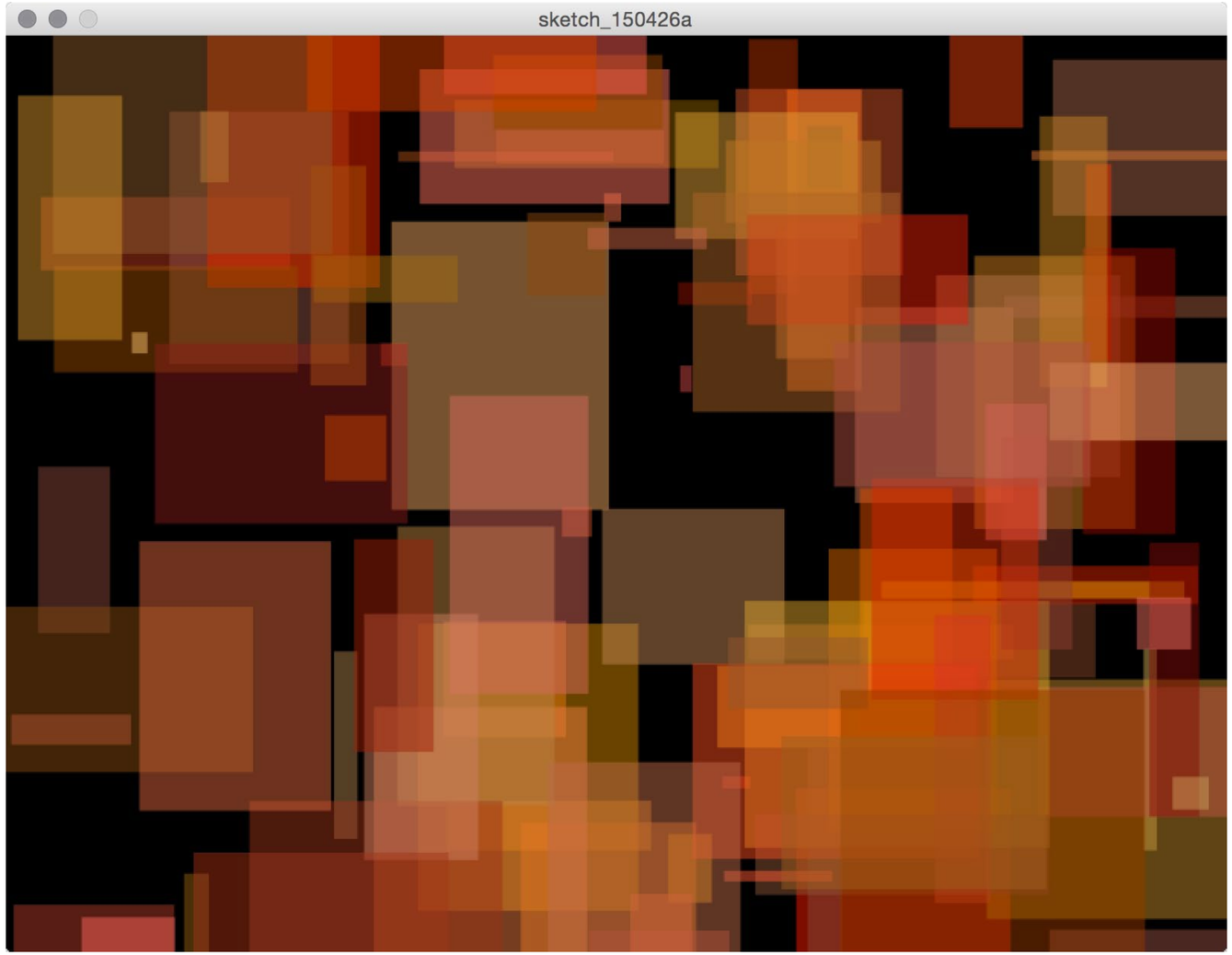
乱数と繰り返しによる色彩と形態

- たくさん四角形を描く

```
function setup() {
  createCanvas(windowWidth, windowHeight);
  colorMode(HSB, 360, 100, 100, 100);
  background(0);
  noStroke();
  rectMode(CENTER);
  for (let i = 0; i < 100; i++) {
    fill(random(0, 40), random(50, 100), random(50, 100), 50);
    rect(random(width), random(height), random(2, 200), random(2, 200));
  }
}
```

乱数と繰り返しによる色彩と形態

- たくさん四角形を描く



乱数に変化を加えてみる

乱数に変化を加えてみる

- 単に乱数をそのまま使うのではなく、少し人為的な操作を加えてみる
  - 範囲を制限
  - 徐々にランダムの幅を変化させる
  - 複数のランダムの組み合わせ
  - ..etc

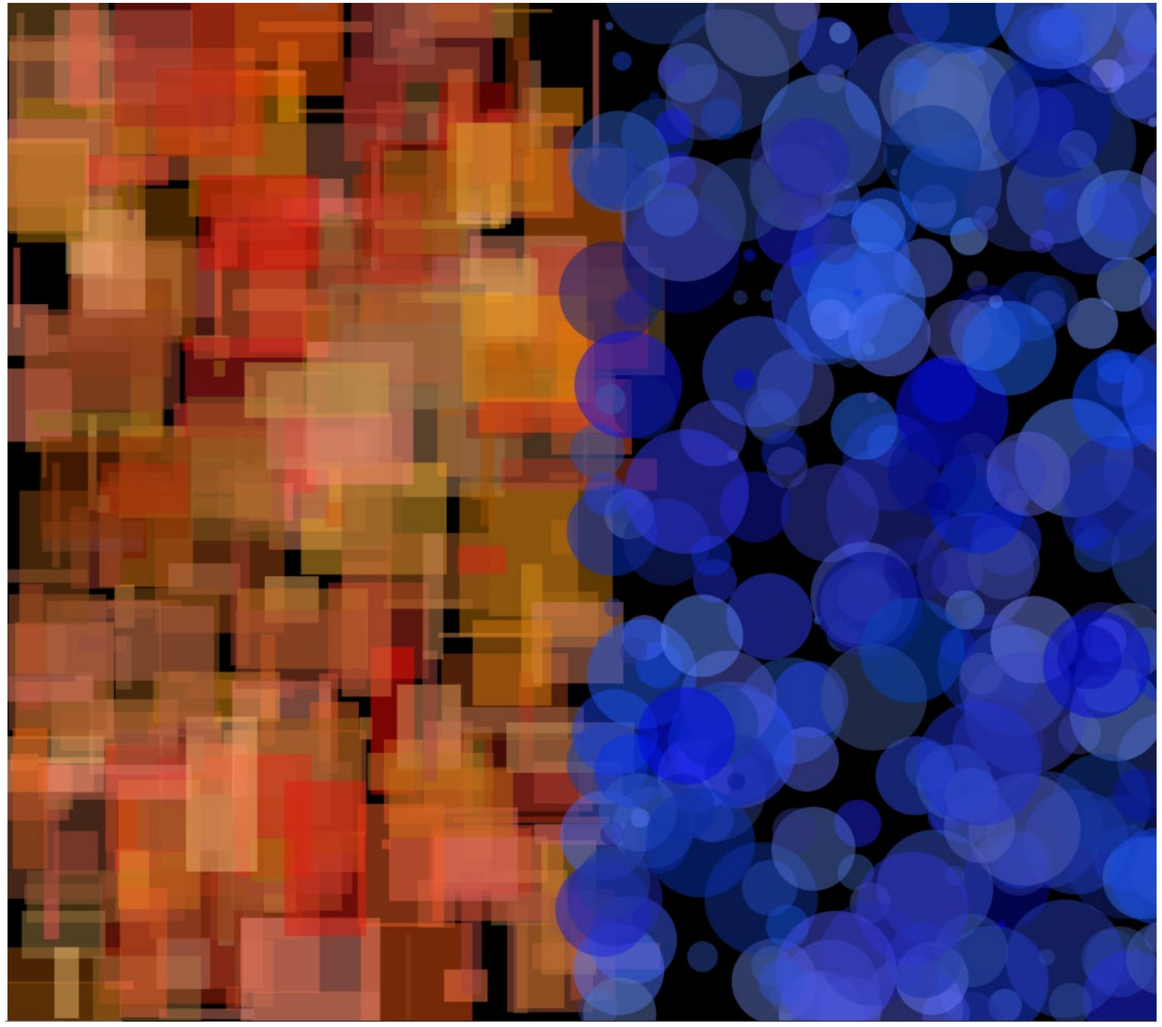
乱数に変化を加えてみる

- ランダムの幅を調整、左右で描き分ける

```
function setup() {
  createCanvas(windowWidth, windowHeight);
  colorMode(HSB, 360, 100, 100, 100);
  background(0);
  noStroke();
  rectMode(CENTER);
  for (let i = 0; i < 300; i++) {
    fill(random(0, 40), random(50, 100), random(50, 100), 50);
    rect(random(width / 2), random(height), random(2, 100), random(2, 100));
  }
  for (let i = 0; i < 300; i++) {
    fill(random(220, 240), random(50, 100), random(50, 100), 50);
    circle(random(width / 2, width), random(height), random(2, 40));
  }
}
```

乱数に変化を加えてみる

- ランダムの幅を調整、左右で描き分ける



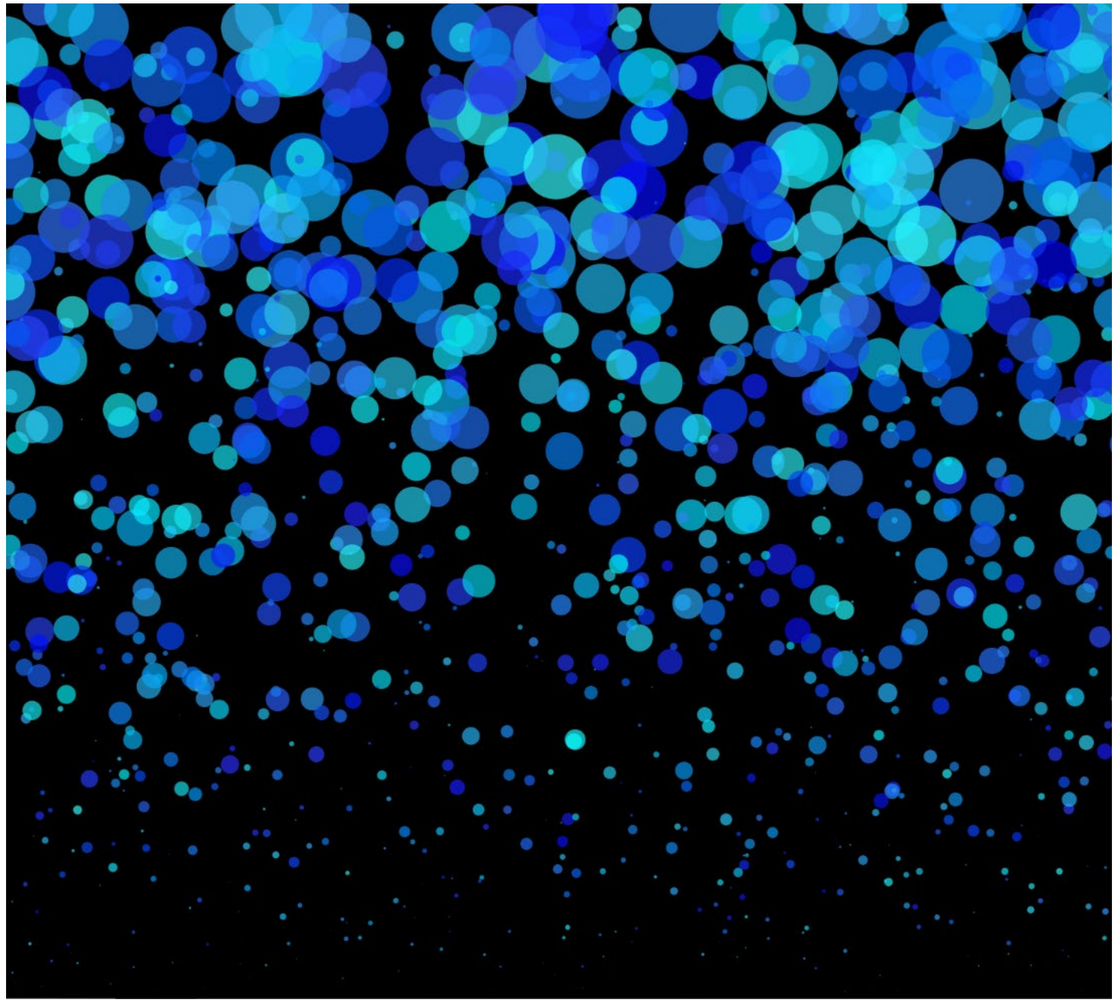
乱数に変化を加えてみる

- y座標で大きさを変化させる

```
function setup() {
  createCanvas(windowWidth, windowHeight);
  colorMode(HSB, 360, 100, 100, 100);
  background(0);
  noStroke();
  rectMode(CENTER);
  for (let i = 0; i < 800; i++) {
    let x = random(width);
    let y = random(height);
    fill(random(220, 240), random(50, 100), random(50, 100), 50);
    circle(x, y, random(0.05) * (height - y));
  }
}
```

乱数に変化を加えてみる

- y座標で大きさを変化させる



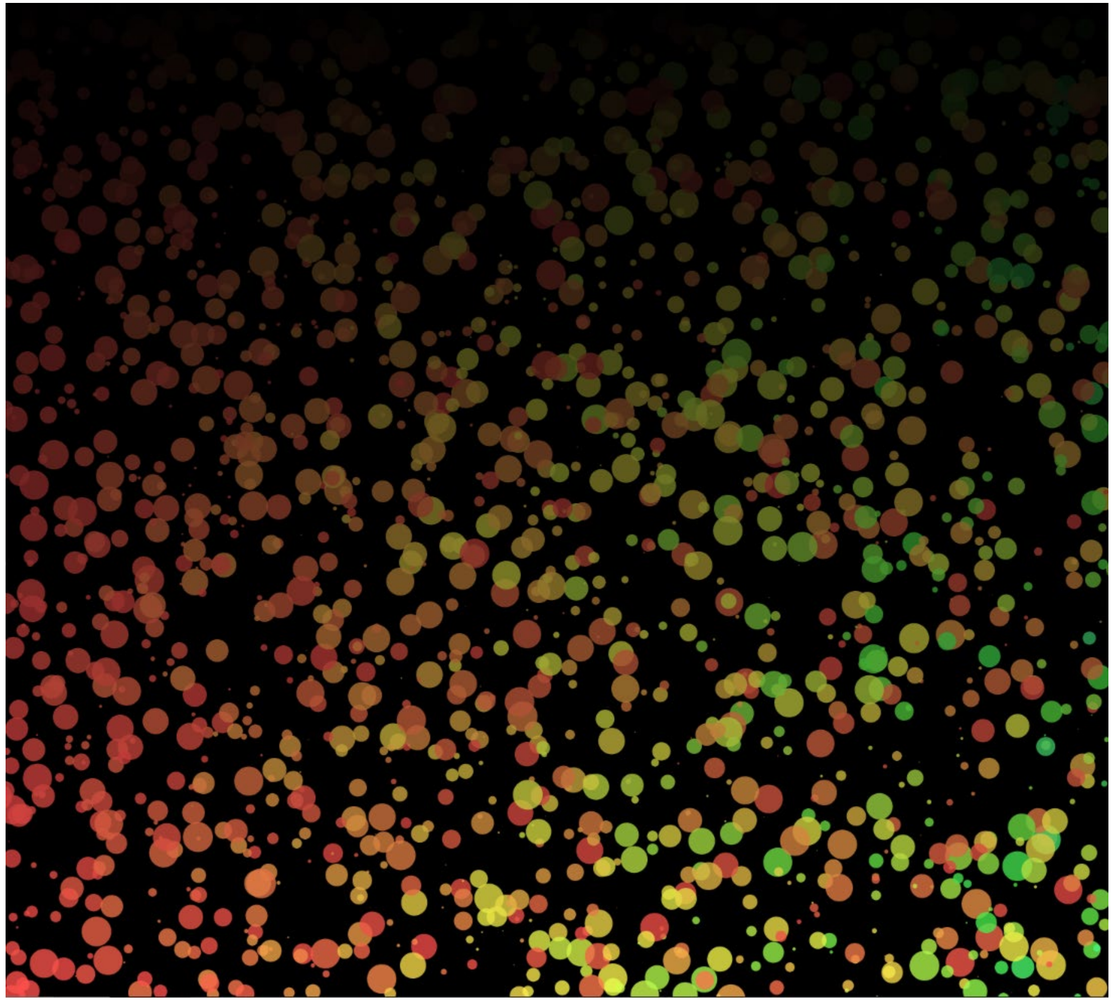
乱数に変化を加えてみる

- 色相と明るさを座標で変化

```
function setup() {
  createCanvas(windowWidth, windowHeight);
  colorMode(HSB, 360, 100, 100, 100);
  background(0);
  noStroke();
  rectMode(CENTER);
  for (let i = 0; i < 2000; i++) {
    let x = random(width);
    let y = random(height);
    fill(random(x * 0.2), 70, y / height * 100, 80);
    circle(x, y, random(10));
  }
}
```

乱数に変化を加えてみる

- 色相と明るさを座標で変化



実習!!

実習!!

- 本日のテーマ:「反復と乱数 - 色彩と形態による画面構成」

- ここまで解説したプログラミング手法を活用して画面構成をする
  - 繰り返し
  - 乱数
  - 色彩（RGB, HSB）
  - 形態、座標