

Processing でプログラミングに挑戦！

—第4回 アニメーションとインタラクション—

杉浦 学

鎌倉女子大学

前号の宿題

前号では、カラフルな円の模様 (図-1) を描く宿題を出題しました。

● 前号の宿題



図-1 カラフルな円の模様

この宿題の解答例をスケッチ 1 に示します。7 行目から 12 行目では、前号で解説した「繰り返し」と「乱数」を利用しています。繰り返しの for ループを二重にすることで、横一列に円を描く (内側のループ) ことを、画面の上から下まで (外側のループ) 行っています。

```
1 //描画の準備
2 size(480,480);
3 background(0);
4 noStroke();
5
```

```
6 //カラフルな円を敷き詰める
7 for(int y=0; y<=height; y+=40){
8   for(int x=0; x<=width; x+=40){
9     fill(random(256),random(256),random(256));
10    ellipse(x,y,35,35);
11  }
12 }
```

スケッチ 1 カラフルな円の模様を描く (宿題の解答例)

二重の for ループの部分について、詳しく解説しておきます。内側の for ループ (8 行目から 11 行目) で 40 ずつ x を増やすことを繰り返し、さらにそれを外側の for ループ (7 行目から 12 行目) で 40 ずつ y を増やしながらか繰り返します。最初に、外側の for ループで y が 0 からスタートし、内側の for ループで x が 0 から width (今回は 480) まで 40 ずつ増えていきます。具体的には、x が 0 → 40 → 80 → 120 → (中略) → 400 → 440 → 480 と増えていくことで合計 13 個の円を描くことになります。この間は y の値は 0 なので、ウィンドウの一番上 (図-2 の上) の位置に円を描くことになります。次に y を 40 増やし、同じように x を 0 から 480 まで増やしながらか 13 個の円を描きます (図-2 の下)。このような動作を行うことで、ウィンドウの一面に円が敷き詰められていきます。

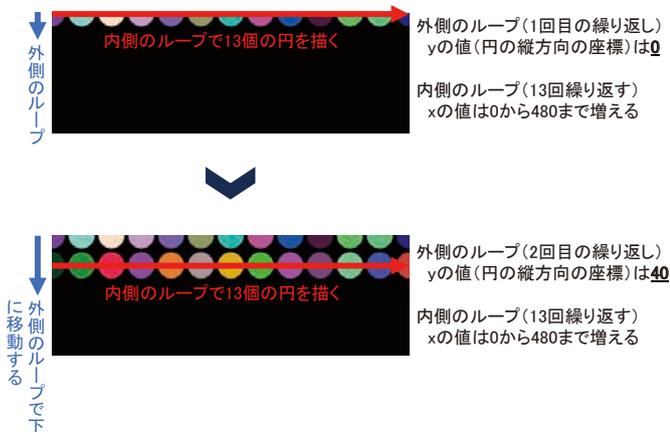


図-2 二重の for ループの動作と描画結果

```

1 int x=0;
2
3 void setup() {
4   size(480,120);
5   noStroke();
6 }
7
8 void draw() {
9   background(204); //残像を消す
10  ellipse(x,60,9,9);
11  x++; //円を右に少し動かす
12 }

```

スケッチ 2 移動する円のアニメーション

アニメーション

ここからは Processing でアニメーションを表現する方法について解説していきます。これまで作成してきたスケッチは、1行目から順番に処理が実行され、最後の行に到達すると実行が終了していました。

Processing では、スケッチ 2 のように記述することにより、アニメーションを作ることができます。スケッチ 2 を実行すると、白い円がウィンドウの左から右に移動していくアニメーションが表示されます (図-3)。スケッチ 2 では、1行目で x という変数を宣言し、0 を代入しています。これは円の横方向の位置を保存しておくための変数です。3行目から6行目の void から始まる部分と、同じく8行目から12行目の void から始まる記述は「関数」と呼ばれる部品です。void という記述以外で関数を書き始める場合もありますが、ここでは説明を省略します。void の後に半角のスペースを入れて、関数の名前を書きます。アニメーションを作る場合は「setup」と「draw」という名前にしておく決まりです。名前の後には「(」と「)」を書きます。今回は空の括弧ですが、括弧の中に何かを書く場合もあります。次に「{」と「}」で関数の範囲を指定します。



図-3 スケッチ 2 の実行結果

setup 関数と draw 関数を用意したスケッチは、図-4 に示した順序で実行されます。スケッチ 2 の場合は、最初に1行目の変数 x の宣言が実行された後に、setup 関数の「{」と「}」に囲まれた部分が1回だけ実行されます。ここではウィンドウの大きさと輪郭線なしの初期設定をしています。次に draw 関数の「{」と「}」に囲まれた部分が、スケッチの実行が終了されるまで (■のボタンが押されるまで)、1秒間につき60回繰り返して実行されます。スケッチ 2 の場合は、画面全体を一度塗りつぶし、変数 x を x 座標にした位置に円を描き、x を1増やすという3つの処理が繰り返されることとなります。結果として、円が左から右に動くアニメーションが実現できます。

スケッチの書き方

ここにスケッチ全体で使う変数を宣言する

```

void setup() {
  ここに最初に1回だけ実行する処理を書く
}

void draw() {
  ここに繰り返す処理を書く
}

```

実行の順番

- ①変数宣言を処理
- ②準備の処理を実行
- ③停止ボタンが押されるまで繰り返す

図-4 アニメーションを表示するスケッチの仕組み



描画の結果はウィンドウに残るので、円が移動しているように見せるために、9行目でウィンドウの全面を灰色に塗りつぶしています。この塗りつぶしの処理がないと、円が移動していくたびに前回の描画の結果が残るので、[図-5](#)のように白い棒が伸びていくようなアニメーションが表示されます。

円が次々と重なって描かれる

図-5 円が重なって描画される

□ 練習問題

前回の宿題の解答例(スケッチ 1)をアニメーションに改造してください。スケッチを実行するとさまざまな色の円がウィンドウに表示されるようにしてみましょう。draw 関数が 1 秒間に実行される回数(フレームレートと呼びます)を 5 回に設定するために、setup 関数に `frameRate(5);` を追加しましょう。



<解説>

前号の宿題をアニメーションさせるスケッチの例

```
1 void setup() {
2   size(480, 480);
3   background(0);
4   noStroke();
5   frameRate(5); //1秒間にdraw関数を5回実行
6 }
7
```

```
8 void draw() {
9   background(0); //前のフレームの描画結果を消す
10  for(int y=0; y<=height; y+=40) {
11    for(int x=0; x<=width; x+=40) {
12      fill(random(256), random(256), random(256));
13      ellipse(x, y, 35, 35);
14    }
15  }
16 }
```

■ インタラクション1 マウスの座標

アニメーションの方法が分かったところで、次はマウスやキーボードなどのユーザの入力に反応する「インタラクション(対話的なやりとり)」について解説していきます。

スケッチ 3 のようにすることで、マウスの位置に円が描けるようになります。実行をしてウィンドウの中でマウスを動かした後の様子を [図-6](#) に示しました。3 行目の fill 関数は 2 つのパラメータをとっていますが、2 番目は透明度で 0 (完全に透明) から 255 (不透明) の値で指定します。これにより、少し透けた黒色の円を描くように設定しています。8 行目の `mouseX` と `mouseY` は、現在のマウスの位置の座標が格納されている変数です。これらの変数は、`height` や `width` と同じように宣言をしないで使える変数のうちの 1 つです。8 行目でマウスの位置を中心にして円を描くように指定しています。マウスを速く動かした場合は円の重なりが少なくなって薄い黒色に、ゆっくり動かすと重なる円の数が増えるため濃い黒色になります。

```

1 void setup(){
2   size(480,120);
3   fill(0, 102);
4   noStroke();
5 }
6
7 void draw(){
8   ellipse(mouseX, mouseY, 9, 9);
9 }

```

スケッチ 3 マウスの位置に円が描かれる
(文献1) p.58 より)



図-6 スケッチ 3 の実行結果

インタラクション 2 マウスクリック

マウスの位置だけでなく、マウスボタンの状態を調べることもできます。マウスのボタンが押されると、`mousePressed` という変数の値が変化します。この変数の型はブーリアン型と呼ばれており、変数の値は真 (true) か偽 (false) のどちらか一方です。マウスのボタンが押されている間は、`mousePressed` の値は真 (true) となり、押されていない場合は偽 (false) となります。

この変数の値を調べることで、マウスがクリックされたときに特定の処理を実行することができます。まずはこの部分について詳しく考えていきましょう。スケッチ 4 では、マウスのボタンがクリックされたときに線の色が変化 (図-7) します。

```

1 void setup(){
2   size(240,120);
3   strokeWeight(30);
4 }
5
6 void draw(){
7   background(204);
8   stroke(102);
9   line(40,0,70,height);
10
11 // マウスがクリックされていたら線を黒に
12 if (mousePressed==true) {
13   stroke(0);
14 }
15
16 line(0,70,width,50);
17 }

```

スケッチ 4 マウスのクリックに反応する
(文献1) p.64 より)



ウインドウの中でマウスのボタンをクリック

図-7 スケッチ 4 の実行結果

ある条件が成立しているかを調べて、成立しているときにだけ指定した処理を実行したい場合は、スケッチ 4 の 12 行目から 14 行目のように if を使った「条件分岐」を記述します (図-8)。条件分岐は繰り返しの for ループと似ています。括弧中の条件の記述は、前号で紹介した比較演算子を使って記述します。

```

if(条件){
  条件が成立したときに実行されるコード
}

```

スケッチでの書き方

参考:ブロック型の言語では

```

if (mousePressed) {
  stroke(0);
}

```



条件に記述する変数の値が真偽の場合は == true を省略できる

図-8 条件分岐の記述方法



ある条件が成立したときと、それ以外のときに別々の処理を実行したい場合は、ifの「{」と「}」の後にelseを追加します。スケッチ4の条件分岐の部分(12行目から14行目)を、elseを使ったものに変更したものをスケッチ5に示します。実行結果と動作の解説は図-9をご覧ください。

```
if(mousePressed){
  stroke(0);
}else{
  stroke(255);
}
```

スケッチ5 elseを追加したスケッチ4の抜粋(文献1) p.65より)

ボタンのクリックなし



```
if (mousePressed) {
  stroke(0);
} else {
  stroke(255);
}
```

ボタンのクリックあり



```
if (mousePressed) {
  stroke(0);
} else {
  stroke(255);
}
```

図-9 スケッチ5の動作と解説

● 練習問題

最後の練習問題として、簡単なアート作品を作成してみましょう。以下に示したスケッチは、毎回同じ位置に、1秒間に1回ずつ、半透明の赤い円が繰り返して描かれます。

```
1 void setup() {
2   size(800,600);
3   background(0);
4   noStroke();
5   frameRate(1);
6 }
7
8 void draw() {
9   fill(255,0,0,90); //90は透明度を指定
10  ellipse(400,300,30,30);
11 }
```

このスケッチを、位置と色と大きさがランダムな円が増えていくようなアニメーションに変更してみましょう。



次のような順番で作業をしてみるとよいでしょう。

Step1: 円の描画位置を毎回異なる値に変更しましょう。x座標とy座標にrandom関数を使って乱数を指定しましょう(横は0からwidth, 縦は0からheight)。

Step2: 円の色を赤で固定ではなく、いろいろな色に変更してみましょう。fill関数のRGBのそれぞれの値をrandom関数で置き換えます(0から255)。

Step3: 円の大きさを50以上・200未満の間でランダムになるようにしてみましょう。
注意: 横幅と縦幅を一緒にしないと正円にならないので、生成した直径を変数に保存し、横幅と縦幅の両方で参照しましょう。

ランダムな直径を変数に保存するヒント:

```
float diameter = random(50,200);
```

Step4: フレームレートが少し遅いので、1秒間に10フレームが実行されるように変更しましょう。

一通り完成したら、マウスをクリックしている間だけアニメーションが進んだり、マウスの位置に円が描かれたりといったアレンジを試みるのもよいでしょう。

<解説>

Step4 まで作業をしたスケッチの例

```
1 void setup(){
2   size(800,600);
3   background(0);
4   noStroke();
5   frameRate(10);
6 }
7
8 void draw(){
9   fill(random(256),random(256),random(256),90);
10  float diameter = random(50,200);
11  ellipse(random(width),random(height),diameter,diameter);
12 }
```

投稿のすすめと発展学習

これまで4回にわたって、中高生のジュニア会員の皆様を读者に想定した連載を掲載しました。ページ数の都合から詳細の説明を省いた部分もありますが、Processingのようなプログラミング言語と、それをを用いた創作活動に興味を持つきっかけとなれば幸いです。

また、ジュニア会員の皆さんがプログラミングに挑戦した結果は、ぜひ本誌の連載「集まれ！ジュニア会員！！」のページに投稿（投稿方法は <https://www.ipsj.or.jp/magazine/jrlist.html> を参照）してみましょう。たとえば、最後の練習問題を少しアレンジしたような、シンプルなスケッチも大歓迎です。

Processingはさまざまな創造的な表現をする仕組みが整っています。最初の一冊としては、文献1)

に示した『Processingをはじめよう』がおすすめですが、より発展的な内容も含めて学習したい場合は、文献2)に示した『Processing クリエイティブ・コーディング入門』も手に取ってみてください。

参考文献

- 1) Reas, C., Fry, B. 著, 船田 巧 訳: Processingをはじめよう 第2版, オライリージャパン(2016).
- 2) 田所 淳: Processing クリエイティブ・コーディング入門—コードが生み出す創造表現, 技術評論社(2017).

(2020年1月4日受付)

杉浦 学 (正会員) manabu@kamakura-u.ac.jp

鎌倉女子大学家政学部家政保健学科准教授。慶應義塾大学大学院政策・メディア研究科後期博士課程修了。博士(政策・メディア)。プログラミング教育をはじめとした情報教育に関する研究に取り組む。中高生向けの著書に『Scratchをはじめよう! プログラミング入門 Scratch3.0版』(日経BP社)など。

