

基礎PLAY

PROCESSING FRIENDS

配列その1



中間試験のお知らせ

試験日：7月2日(火)2限

試験範囲：6月28日（金）までの授業範囲

主な内容：図形・文字等の描画、変数、分岐処理、
繰り返し処理、座標変換、関数、1次元配列

形式：マークシートと記述

持ち込み：プリント、ノート、ノートPC、本は可。人は不可。

注意事項：持ち込んだノートPCをネットに接続することは認めません。カンニング、相談などもダメです。

最終課題制作

Processingでプログラムをつくる

発表会を実施（最後の金曜日、7月26日）

前回の内容

再び関数定義

コールバック関数

マウス関連のイベント

押されたマウスボタンの種類：mouseButton変数でわかる
mouseButton変数がとる値：LEFT,CENTER,RIGHT

関数名	役割・呼び出されるタイミング
mousePressed	マウスボタンが押されたら
mouseClicked	クリックされたら
mouseMoved	マウスが移動したら
mouseDragged	マウスをドラッグしたら
mouseReleased	マウスボタンが離された
mouseWheel	マウスのホイールが回されたら

mouseWheelだけちょっと特別

キーボード関連のイベント

押されたキーの種類： key変数とkeyCode変数でわかる

key変数：どのキーが押されたかを保持している

key == 'k' ← 小文字のkキーが押されたか？

key == '1' ← 1キーが押されたか？

key変数の値がCODEDの時は、特殊キーが押されたら、
どのキーが押されたかの情報がkeyCode変数に入っている

関数名	役割・呼び出されるタイミング
keyPressed	キーが押された(ShiftキーなどでもOK)
keyReleased	キーが押されなくなったら
keyTyped	キーが押されたら(Shiftキーなどはダメ)

プログラミング言語を 学ぶときの基本要素

逐次処理

変数

分岐処理

関数

繰り返し処理

配列

データ構造

1個の円を移動させるためには、
3個の変数を使っている



10個の円を移動させるためには、
30個の変数が必要

```
float y0; // 円の中心のY座標  
float x0; // 円の中心のX座標  
float v0; // 円の縦方向の移動速度  
int radius;  
void setup() {  
    size(300, 400);  
    smooth();  
    radius = 10;  
    v0 = random(1, 2);  
    y0 = -random(radius, 2*radius);  
    x0 = random(radius, width-radius);  
}
```

```
void draw() {  
    background(255);  
    // 中心(x0,y0)の円の処理  
    y0 = y0+v0;  
    if (y0 -radius> height) {  
        x0 =random(radius, width-radius);  
        y0 = -random(radius, 2*radius);  
    }  
    stroke(255, 10, 10);  
    fill(255, 10, 10);  
    ellipse(x0, y0, 2*radius, 2*radius);  
}
```

```
float y0, y1,y2; // 円の中心のY座標  
float x0, x1,x2; // 円の中心のX座標  
float v0, v1,v2; // 円の縦方向の移動速度  
int radius;  
void setup() {  
    size(300, 400);  
    smooth();  
    radius = 10;  
    v0 = random(1, 2);  
    y0 = -random(radius, 2*radius);  
    x0 = random(radius, width-radius);  
    v1 = random(1, 2);  
    y1 = -random(radius, 2*radius);  
    x1 = random(radius, width-radius); }  
v2 = random(1, 2);  
y2 = -random(radius, 2*radius);  
x2 = random(radius, width-radius); }
```

```
void draw() {  
    background(255);  
    y0 = y0+v0;  
    if (y0 -radius> height) {  
        x0 =random(radius, width-radius);  
        y0 = -random(radius, 2*radius);  
    }  
    stroke(255, 10, 10);  
    fill(255, 10, 10);  
    ellipse(x0, y0, 2*radius, 2*radius);  
    y1 = y1+v1;  
    if (y1 -radius> height) {  
        x1 = random(radius, width-radius);  
        y1 = -random(radius, 2*radius);  
    }  
    stroke(255, 10, 10);  
    fill(255, 10, 10);  
    ellipse(x1, y1, 2*radius, 2*radius);  
    y2 = y2+v2;  
    if (y2 -radius> height) {  
        x2 =random(radius, width-radius);  
        y2 = -random(radius, 2*radius);  
    }  
    stroke(255, 10, 10);  
    fill(255, 10, 10);  
    ellipse(x2, y2, 2*radius, 2*radius); } }
```

付け加えたところ

```
v2 = random(1, 2);
y2 = -random(radius, 2*radius);
x2 = random(radius, width-radius);

y2 = y2+v2;
if (y2 -radius> height) {
  x2 =random(radius, width-radius);
  y2 = -random(radius, 2*radius);
}
stroke(255, 10, 10);
fill(255, 10, 10);
ellipse(x2, y2, 2*radius, 2*radius);
```

10個の円を移動させるためには、 30個の変数が必要

```
float y0,y1,y2,...,y9; // 円の中心のY座標
float x0,x1,x2,...,x9; // 円の中心のX座標
float v0,v1,v2,...,v9; // 円の縦方向の移
動速度
int radius;
void setup() {
  size(300, 400);
  smooth();
  radius = 10;
  v0 = random(1, 2);
  y0 = -random(radius, 2*radius);
  x0 = random(radius, width-radius);
  v1 = random(1, 2);
  y1 = -random(radius, 2*radius);
  x1 = random(radius, width-radius);
// 途中省略
  v9 = random(1, 2);
  y9 = -random(radius, 2*radius);
  x9 = random(radius, width-radius);
}
```

```
void draw() {
  background(255);
  // 中心(x0,y0)の円の処理
  y0 = y0+v0;
  if (y0 -radius> height) {
    x0 = random(radius, width-radius);
    y0 = -random(radius, 2*radius);
  }
  stroke(255, 10, 10);
  fill(255, 10, 10);
  ellipse(x0, y0, 2*radius, 2*radius);
  // 途中省略
  // 中心(x9,y9)の円の処理
  y9 = y9+v9;
  if (y9 -radius> height) {
    x9 = random(radius, width-radius);
    y9 = -random(radius, 2*radius);
  }
  stroke(255, 10, 10);
  fill(255, 10, 10);
  ellipse(x9, y9, 2*radius, 2*radius);
}
```

やっていること やりたいこと

変数名と数字のペアで変数を表す



配列

数学の授業で使われているような添え字 a_2, a_n のような機能の欲しい

配列（1次元配列）

変数名と数字のペアで変数を表す

数字のことを添え字

変数名[数字]で1つの変数を表す

数字は0から数え始める、

先頭からどれだけ移動したか

例えば、

$x[0], x[1], x[2]$ など

配列x

$x[0]$	$x[1]$	$x[2]$	$x[3]$	$x[4]$	$x[5]$	$x[6]$
--------	--------	--------	--------	--------	--------	--------

配列（1次元配列）を使うためには

変数宣言が必要

データ型[] 変数;
例えば、
float[] y;

何個のデータを使うか
のかを指定する

データ型[] 変数 = new データ型[必要な個数];
例えば、
float[] y = new float[100]; y[0],y[1],...,y[99]
y = new float[10]; y[0],y[1],...,y[9]

Plmage[] faces;
int[] x;
String[] names;

データを入れる
場所を確保

配列を表す
マーク

配列（1次元配列）を 使うためには

```
Plmage[] faces;  
int[] x;  
String[] names;
```

```
Plmage[] faces =new Plimage[5];  
int[] x = new int[1000];  
String[] names = new String[5];
```

何個のデータを使うか
のかを指定する

```
faces[0],faces[1],...,faces[4];  
x[0],x[1],x[2],...,x[999]  
names[0],names[1],...,names[4]
```

```
float[] x;
float[] y;
float[] v;
int radius;
void setup() {
    size(300, 400);
    x = new float[5];
    y = new float[5];
    v = new float[5];
    radius = 10;
    v[0] = random(1, 2);
    y[0] = -random(radius, 2*radius);
    x[0] = random(radius, width-radius);
    v[1] = random(1, 2);
    y[1] = -random(radius, 2*radius);
    x[1] = random(radius, width-radius);
    v[2] = random(1, 2);
    y[2] = -random(radius, 2*radius);
    x[2] = random(radius, width-radius);
    v[3] = random(1, 2);
    y[3] = -random(radius, 2*radius);
    x[3] = random(radius, width-radius);
    v[4] = random(1, 2);
    y[4] = -random(radius, 2*radius);
    x[4] = random(radius, width-radius);
}
```

```
void draw() {
    background(255);
    y[0] = y[0]+v[0];
    if (y[0] -radius> height) {
        x[0] =random(radius, width-radius);
        y[0] = -random(radius, 2*radius);
    }
    stroke(255, 10, 10);
    fill(255, 10, 10);
    ellipse(x[0], y[0], 2*radius, 2*radius);
    y[1] = y[1]+v[1];
    if (y[1] -radius> height) {
        x[1] =random(radius, width-radius);
        y[1] = -random(radius, 2*radius);
    }
    stroke(255, 10, 10);
    fill(255, 10, 10);
    ellipse(x[1], y[1], 2*radius, 2*radius);
// [1]を[2],[3]と変えた物がくる
    y[2] = y[2]+v[2];
    if (y[2] -radius> height) {
        x[2] =random(radius, width-radius);
        y[2] = -random(radius, 2*radius);
    }
    stroke(255, 10, 10);
    fill(255, 10, 10);
    ellipse(x[2], y[2], 2*radius, 2*radius);
    y[3] = y[3]+v[3];
    if (y[3] -radius> height) {
        x[3] =random(radius, width-radius);
        y[3] = -random(radius, 2*radius);
    }
    stroke(255, 10, 10);
    fill(255, 10, 10);
    ellipse(x[3], y[3], 2*radius, 2*radius);
    y[4] = y[4]+v[4];
    if (y[4] -radius> height) {
        x[4] =random(radius, width-radius);
        y[4] = -random(radius, 2*radius);
    }
    stroke(255, 10, 10);
    fill(255, 10, 10);
    ellipse(x[4], y[4], 2*radius, 2*radius);
}
```

```
x = new float[5];
```



float型のデータを入れ
ることができる変数

```
y = new float[5];
```



float型のデータを入れ
ができる変数

```
v = new float[5];
```



float型のデータを入れ
ができる変数

規則的に変わっている部分があるときには

繰り返し処理を使う

```
v[0] = random(1, 2);
y[0] = -random(radius, 2*radius);
x[0] = random(radius, width-radius);
v[1] = random(1, 2);
y[1] = -random(radius, 2*radius);
x[1] = random(radius, width-radius);
v[2] = random(1, 2);
y[2] = -random(radius, 2*radius);
x[2] = random(radius, width-radius);
v[3] = random(1, 2);
y[3] = -random(radius, 2*radius);
x[3] = random(radius, width-radius);
v[4] = random(1, 2);
y[4] = -random(radius, 2*radius);
x[4] = random(radius, width-radius);
```

```
for(int i=0;i<5;i++){
    v[i] = random(1, 2);
    y[i] = -random(radius, 2*radius);
    x[i] = random(radius, width-radius);
}
```

繰り返し処理と配列は相性が良い

```
void draw() {
    background(255);
    y[0] = y[0]+v[0];
    if (y[0] -radius> height) {
        x[0] =random(radius, width-radius);
        y[0] = -random(radius, 2*radius);
    }
    stroke(255, 10, 10);
    fill(255, 10, 10);
    ellipse(x[0], y[0], 2*radius, 2*radius);
    y[1] = y[1]+v[1];
    if (y[1] -radius> height) {
        x[1] =random(radius, width-radius);
        y[1] = -random(radius, 2*radius);
    }
    stroke(255, 10, 10);
    fill(255, 10, 10);
    ellipse(x[1], y[1], 2*radius, 2*radius);
// [1]を[2],[3]と変えた物がくる
    y[4] = y[4]+v[4];
    if (y[4] -radius> height) {
        x[4] =random(radius, width-radius);
        y[4] = -random(radius, 2*radius);
    }
    stroke(255, 10, 10);
    fill(255, 10, 10);
    ellipse(x[4], y[4], 2*radius, 2*radius);
}
```

```
void draw() {
    background(255);
    for (int i=0; i<5; i++) {
        y[i] = y[i]+v[i];
        if (y[i] -radius> height) {
            x[i] =random(radius, width-radius);
            y[i] = -random(radius, 2*radius);
        }
        stroke(255, 10, 10);
        fill(255, 10, 10);
        ellipse(x[i], y[i], 2*radius, 2*radius);
    }
}
```

```
void draw() {  
    background(255);  
    for (int i=0; i<5; i++) {  
        y[i] = y[i]+v[i];  
        if (y[i] -radius> height) {  
            x[i] =random(radius, width-radius);  
            y[i] = -random(radius, 2*radius);  
        }  
        stroke(255, 10, 10);  
        fill(255, 10, 10);  
        ellipse(x[i], y[i], 2*radius, 2*radius);  
    }  
}
```

配列yの要素の個数

```
void draw() {  
    background(255);  
    for (int i=0; i<y.length; i++) {  
        y[i] = y[i]+v[i];  
        if (y[i] -radius> height) {  
            x[i] =random(radius, width-radius);  
            y[i] = -random(radius, 2*radius);  
        }  
        stroke(255, 10, 10);  
        fill(255, 10, 10);  
        ellipse(x[i], y[i], 2*radius, 2*radius);  
    }  
}
```

配列に入るデータの個数はy.lengthとすると
取り出すことが出来る

もう一つの配列宣言方法

データ型[] 変数名={データ0,データ1,…};

String[] name={"Akagi","Kaga","Souryu","Hiryu","Shokaku","Zuikaku"};

name[0] -> "Akagi"

name[1] -> "Kaga"

name[2] -> "Souryu"

name[3] -> "Hiryu"

name[4] -> "Shokaku"

name[5] -> "Zuikaku"

newは不要

繰り返し処理を 途中で止める

break命令を利用する

break命令が含まれている一番内側の
繰り返し処理から抜け出す

関数の引数としても 利用できる

配列を関数の引数として
利用することが出来る

配列を関数の戻り値とし
て利用することが出来る

関数の引数の例

drawCircleAtBalance(x,y);

```
void drawCircleAtBalance(float[] x,  
                         float[] y){  
    float gx=0;  
    float gy=0;  
    for(int i=0;i < xPos.length;i++){  
        gx = gx + x[i];  
    }  
    for(int i=0;i < yPos.length;i++){  
        gy = gy + y[i];  
    }  
    gx = gx / x.length;  
    gy = gy / y.length;  
    fill(100);  
    ellipse(gx,gy,2*radius,2*radius);  
}
```

drawCircleAtBalance(x,y);

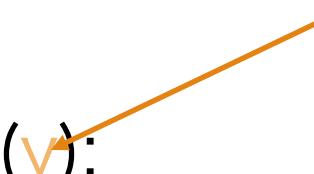
```
void drawCircleAtBalance(float[] x, float[] y){  
    float gx=0;  
    float gy=0;  
    for(int i=0;i < x.length;i++){  
        gx = gx + x[i];  
    }  
    for(int i=0;i < y.length;i++){  
        gy = gy + y[i];  
    }  
    gx = gx / x.length;  
    gy = gy / y.length;  
    fill(100);  
    ellipse(gx,gy,2*radius,2*radius);  
}
```

関数の引数として配列を使うときの注意

関数内で引数の配列の値を変えると、
実引数の配列変数の値も変わってしまう。

```
void down(int[] h){  
    for(int i=0;i < h.length;i++){  
        h[i] = h[i] + i % 3;  
    }  
}  
  
down(y);
```

この配列に入っている値が
変わってしまう



今日やった内容

中間試験の告知

配列



授業時に配布した資料

<http://www.sato-lab.jp/imfu/index.html>

においてあります。