



プログラミング演習(8)

配列

中村, 高橋
小林, 橋本



- Processing で配列に挑戦！
 - 最大値・最小値を求めるにはどうするか？
 - ゲームで沢山の敵を扱うにはどうするか？
 - 不均等に並べられた沢山の丸をどんどん移動するにはどうしたらよいか？
 - ソーシャルゲームでコンプリートするためにかかる回数はどれくらいだろうか？
- 課題：
 - 避けゲームを作ってみる
 - パラパラアニメを作ってみよう
 - 占いアプリを作ってみよう

沢山の情報を扱うには？



- 多くの人の情報を管理するにはどうする？
- ゲームの沢山のマス目をどう管理するか？
- トランプのカードや色々なスコアの管理
- 沢山の敵を同時に動かしたい
 - 沢山の敵の場所を管理するにはどうするか？
 - 沢山の円が一度に動き、その円をマウスで避けるような簡単なゲームを作ってみよう！
- ソーシャルゲームのカードのコンプリート状況をどう管理するか？

カードの偏り



(Q) 0番から9番までのカードがランダムに選択されるとする。100枚引いたときのカードの枚数分布はどうなっているか結果を表示せよ

0	番目	12	枚
1	番目	12	枚
2	番目	7	枚
3	番目	7	枚
4	番目	10	枚
5	番目	14	枚
6	番目	13	枚
7	番目	12	枚
8	番目	6	枚
9	番目	7	枚

カードの偏り



```
int card0 = 0;  
int card1 = 0;  
int card2 = 0;  
int card3 = 0;  
int card4 = 0;  
int card5 = 0;  
int card6 = 0;  
int card7 = 0;  
int card8 = 0;  
int card9 = 0;
```

```
for(int i=0; i<100; i++)  
{  
    int id = (int)random(0, 100);  
    if(id == 0)  
    {  
        card0++;  
    }  
    else if(id == 1)  
    {  
        card1++;  
    }  
    else if(id == 2)  
    {  
        card2++;  
    }  
    (たくさん略)  
    }  
    else if(id == 9)  
    {  
        card9++;  
    }  
}
```

やっぺられん(怒)

```
println("0番目は"+card0+"枚");  
println("1番目は"+card1+"枚");  
println("2番目は"+card2+"枚");  
println("3番目は"+card3+"枚");  
println("4番目は"+card4+"枚");  
println("5番目は"+card5+"枚");  
println("6番目は"+card6+"枚");  
println("7番目は"+card7+"枚");  
println("8番目は"+card8+"枚");  
println("9番目は"+card9+"枚");
```



- 配列とは, ある同じ役割を持つ変数の固まり
- 変数名ではなく, **変数名 + 何番目**かで値を代入したり取り出したりすることが可能
- 配列の定義

```
int[] card = new int[10];
```

```
変数の型[] 変数名 = new 変数の型[要素数];
```

- 変数への代入 `card[5] = 1;`
- 変数の値の出力 `println(card[5]);`

配列の注意点



- 配列の要素数が n の場合、**配列は0番目から始まり、 $n-1$ 番目まで**となる

(例) `int[] value = new int[10];` の場合
`value[0]`, `value[1]`, `value[2]`, ..., `value[9]`
までとなる

- 配列の何番目かを指定する際に、指定できない数を指定するとエラーが出る

(例) `int[] value = new int[10];` の場合
`value[-1]`, `value[10]` などは存在しないのでお
かしい

カードの偏り



```
int[] card = new int[10];  
for(int i = 0; i < 10; i++)  
{  
    card[i] = 0;  
}
```

配列の定義と初期化

```
for(int i = 0; i < 100; i++)  
{  
    int id = (int)random(0, 10);  
    card[id]++;  
}
```

0 - 9 の値をランダムに生成
カードの数を増やす

カードの枚数を出力

```
for(int i = 0; i < 10; i++)  
{  
    println(i + "番目は" + card[i] + "枚");  
}
```

こんなに短く！

カードの偏り



```
int[] card = new int[10];  
// カードの枚数を初期化  
for(int i = 0; i < 10; i++)  
{  
    card[i] = 0;  
}  
  
// 100回カードをランダムに引く  
for(int i = 0; i < 100; i++)  
{  
    int id = (int)random(0, 10);  
    card[id]++;  
}  
  
// 結果を1枚ずつ出力する  
for(int i = 0; i < 10; i++)  
{  
    println(i + "番目は" + card[i] + "枚");  
}
```

こんなに短く！

カードのコンプリート



(Q) 0番から9番までの10種類のカードがランダムに選択されるとする。何枚引いたらコンプリートすることができるだろうか？

考え方

- カードの偏りのプログラムを活かしてカードを順に引いていく
- 全部コンプリートしたというのは、全部が1枚以上になったということ！

```
int num = 0;
int flagComplete = 0;
int[] card = new int[10];

for(int i = 0; i < 10; i++)
{
    card[i] = 0;
}

while(flagComplete == 0)
{
    int id = (int)random(0, 10);
    num++;
    card[id]++;
    flagComplete = 1;
    for(int i = 0; i < 10; i++)
    {
        if(card[i] == 0)
        {
            flagComplete = 0;
        }
    }
}

println(num + "枚でコンプリート!");
```

1つでも0枚のものが
あったらコンプリート
していない!



100種のコンプリート



0番から99番までの100種類のカードがランダムに選択されるとする。何枚引いたらコンプリートすることができるだろうか？

考え方

- 10種類のもをそのまま応用するだけ！

力

```
int[] card = new int[100];
int num = 0;
int flagComplete = 0;

for(int i = 0; i < 100; i++)
{
    card[i] = 0;
}

while(flagComplete == 0)
{
    int id = (int)random(0, 100);
    card[id]++;
    num++;
    flagComplete = 1;
    for(int i = 0; i < 100; i++)
    {
        if(card[i] == 0)
        {
            flagComplete = 0;
        }
    }
}

println(num + "枚でコンプリート!");
```

一つでも0枚のものが
あったらコンプリート
していない！



アイテムガチャ



(Q) クリックする度にランダムに取得できる10個のアイテムについて、取得したアイテムの数をカウントし、その数を表示するプログラムを作成せよ

- 考え方
 - 要素数が10個の配列を作成する
 - クリックの度にランダムに値を生成する
 - 生成された値に対応する要素の配列の値を増やす
 - それぞれの数をprintlnで表示する

```
item0: 3  
item1: 1  
item2: 1  
item3: 3  
item4: 2  
item5: 3  
item6: 2  
item7: 0  
item8: 5  
item9: 1
```



```
int[] items = new int[10];

void setup()
{
  int i = 0;
  while(i < 10)
  {
    items[i] = 0;
    i++;
  }
}

void draw(){
}

void mousePressed()
{
  int num = (int)random(10);
  items[num]++;
  int i = 0;
  while(i < 10)
  {
    println("item" + i + ": " + items[i]);
    i++;
  }
}
```

アイテムガチャ



(Q) クリックする度にランダムに取得できる30個のアイテムについて、取得したアイテムの数をカウントし、その数を表示するプログラムを棒グラフとして作成せよ

• 考え方

- アイテム数を30個に変更
- rectを使って表示しよう



コンプリートガチャ



- 先述のプログラムを改良し, 30個のアイテム全て1つ以上揃った時に「Complete」と表示するプログラムを作成せよ. また, その時に何回クリックしたかを表示せよ



(Q) 5人の身長から平均を求めたい

- 考え方

- 合計を計算する変数 `total` を置く
- `total` に配列の値を足し合わせていく
- `total` の値を人数で割って平均を求める

合計と平均



```
// 事前に与えられる身長の情報
// こんな感じで初期化することが可能
float[] values = {170.5, 200.3, 145.3, 156.3, 182.5};
// 全員分まず表示してみよう
println(values);

// こちらで計算する
float total = 0.0;
for(int i = 0; i < 5; i++)
{
    total += values[i];
}
println("平均は" + (total / 5) + "cm");
```

最大（と最小）



(Q) 10人の身長から最大値を求める

• 考え方

- 現在の最大値を `max_value` として保存する
- i 番目の人と, `max_value` とでどちらが大きいかを比較し, i 番目の人が大きければ `max_value` を i 番目の人と同じにしていく
- (ちなみに, 最小値も似たような感じで処理できる)

最大

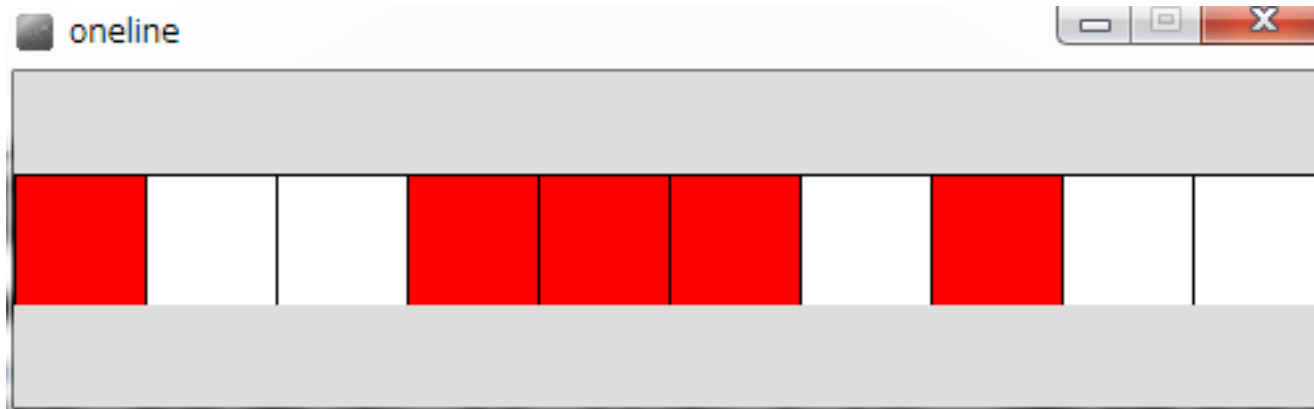
```
// 事前に与えられる身長の情報
float[] values = new float[10];
for(int i = 0; i < 10; i++)
{
    values[i] = (float)random(145, 200);
}
// 全員分まず表示してみよう
println(values);
// 0番目の人のを最大値とする
float max_value = value[0];
for(int i = 1; i < 10; i++)
{
    if(max_value < values[i])
    {
        max_value = values[i];
    }
}
println("最大値は" + max_value + "cm");
```



1次元の掲示板



(Q) 500x50のウィンドウ上に、縦横50ピクセルの正方形を横に10個並べたボードを作成し、正方形の内部をクリックする度にその図形の色が【白→赤→白→赤】と変化させるようにする



1次元の掲示板



• 考え方

- 信号のプログラムの応用で，10個の正方形の状態記憶する整数（int型）の変数を10個用意
 - `int status0, status1, status2, status3, ..., status8, status9;`
- クリック時のマウス座標（`mouseX, mouseY`）から，どの図形内でクリックされたかを判定し，その`statusX`の値を`0→1→0→1`と書き換える
- `draw()`内で，`statusX`の情報に基づき0なら白色で，1なら赤色で正方形を描画する



```
int status0 = 0;
int status1 = 0;
int status2 = 0;
int status3 = 0;
int status4 = 0;
int status5 = 0;
int status6 = 0;
int status7 = 0;
int status8 = 0;
int status9 = 0;
```

```
void setup()
{
  size(500, 50);
}
```

```
void draw()
{
  background(255);
  if(status0 == 0)
  {
    fill(255);
  }
  else
  {
    fill(255, 0, 0);
  }
  rect(0, 0, 50, 50);

  (たくさん略)

  if(status9 == 0)
  {
    fill(255);
  }
  else
  {
    fill(255, 0, 0);
  }
  rect(450, 0, 50, 50);
}
```

```
void mousePressed()
{
  // どこをクリックしたか
  int tx = mouseX / 50;

  if(tx == 0)
  {
    status0 ++;
    if(status0 > 1)
    {
      status0 = 0;
    }
  }
  else if(tx == 1)
  {
    (たくさん略)
  }
  else if(tx == 9)
  {
    status9 ++;
    if(status9 > 1)
    {
      status9 = 0;
    }
  }
}
```

やっつけられん (怒)



配列を使ってみる

```
int[] status = new int[10];
```

```
void setup()
{
  size(500, 50);
  status[0] = 0;
  status[1] = 0;
  status[2] = 0;
  status[3] = 0;
  status[4] = 0;
  status[5] = 0;
  status[6] = 0;
  status[7] = 0;
  status[8] = 0;
  status[9] = 0;
}
```

```
void draw()
{
  background(255);
  if(status[0] == 0)
  {
    fill(255);
  }
  else
  {
    fill(255, 0, 0);
  }
  rect(0, 0, 50, 50);

  (たくさん略)
```

```
if(status[9] == 0)
{
  fill(255);
}
else
{
  fill(255, 0, 0);
}
rect(450, 0, 50, 50);
}
```

```
void mousePressed()
{
  // どこをクリックしたか
  int tx = mouseX / 50;
  status[tx]++;
  if(status[tx] > 1)
  {
    status[tx] = 0;
  }
}
```

**status[数字]と
することで楽！**



配列を使ってみる

```
int[] status = new int[10];
```

```
void setup()
{
  size(500, 50);
  status[0] = 0;
  status[1] = 0;
  status[2] = 0;
  status[3] = 0;
  status[4] = 0;
  status[5] = 0;
  status[6] = 0;
  status[7] = 0;
  status[8] = 0;
  status[9] = 0;
}
```

```
void draw()
{
  background(255);
  if(status[0] == 0)
  {
    fill(255);
  }
  else
```

```
void mousePressed()
{
  // 0ボタンが押されたら、status[0]を1にする
}
```

ただ、drawは相変わらず長いし
わかりにくいのでは？

```
if(status[9] == 0)
{
  fill(255);
}
else
{
  fill(255, 0, 0);
}
rect(450, 0, 50, 50);
}
```

status[数字]と
することで楽！

配列 + 繰り返し



- 配列は繰り返しとの組合せにより能力発揮

```
int[] status = new int[10];
```

```
void setup()
{
  size(500, 50);
  int i = 0;
  while(i < 10)
  {
    status[i] = 0;
    i++;
  }
}
```

```
void draw()
{
  background(255);
  int i = 0;
  while(i < 10)
  {
    if(status[i] == 0)
    {
      fill(255);
    }
    else
    {
      fill(255, 0, 0);
    }
    rect(i * 50, 0, 50, 50);
    i++;
  }
}
```

配列 + 繰り返し最強！

```
void mousePressed()
{
  // どこをクリックしたか
  int tx = mouseX / 50;
  status[tx]++;
  if(status[tx] > 1)
  {
    status[tx] = 0;
  }
}
```

配列 + 繰り返し (for)



- 配列は繰り返しとの組合せにより能力発揮！
 - 繰り返しで増えていく数字を利用し，配列の何番目を指すかを指定する
 - for は配列との組み合わせでかなり有効！

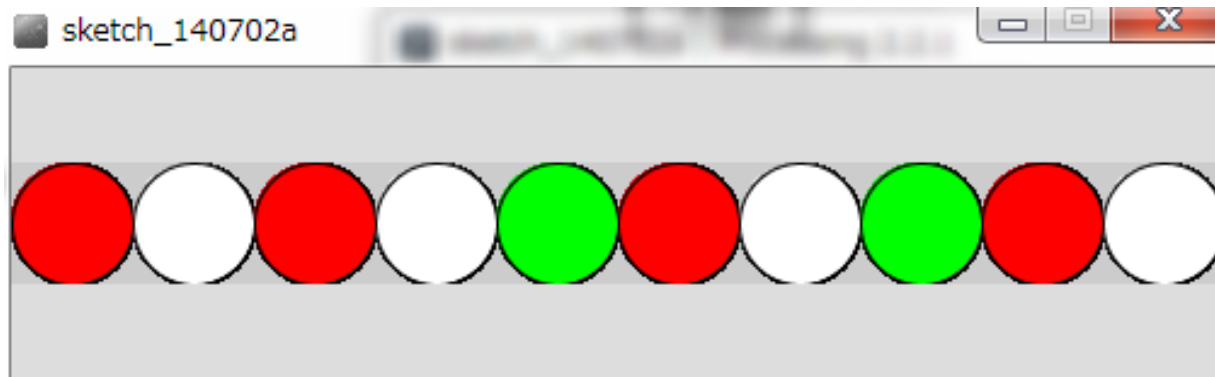
```
int[] status = new int[10];  
void setup()  
{  
  size(500, 50);  
  for(int i = 0; i < 10; i++)  
  {  
    status[i] = 0;  
  }  
}
```

```
void draw()  
{  
  background(255);  
  for(int i = 0; i < 10; i++)  
  {  
    if(status[i] == 0)  
    {  
      fill(255);  
    }  
    else  
    {  
      fill(255, 0, 0);  
    }  
    rect(i * 50, 0, 50, 50);  
  }  
}
```

```
void mousePressed()  
{  
  // どこをクリックしたか  
  int tx = mouseX / 50;  
  status[tx]++;  
  if(status[tx] > 1)  
  {  
    status[tx] = 0;  
  }  
}
```



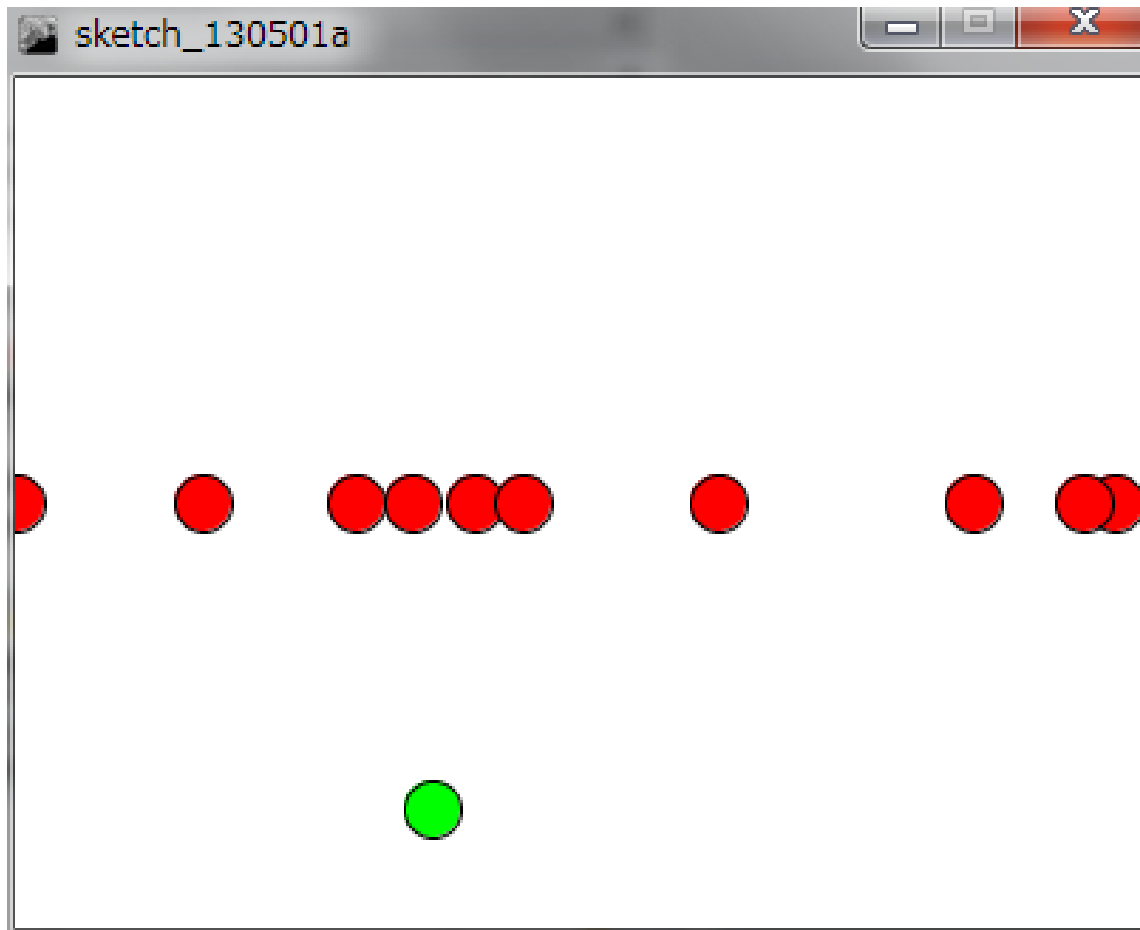
- 直径50の円を横に10個並べたボードを作成せよ。円の内部をクリックする度に、その円の色が【白→赤→黄→緑→白】と変化させるようにすること。



円を沢山描画する



(Q) 400x300の画面上の、Y座標150の場所に円を10個、カーソルの下に円を描画（直径20）



円を沢山描画する



• 考え方

- カーソルの下に描画するには `mouseX`, `mouseY`
 - `ellipse(mouseX, mouseY, 20, 20);`
- 10個の円のX座標を記憶する実数 (`float`型) の変数を10個用意
 - `float posX0; float posX1; float posX2; ... ,`
`float posX9;`
- `setup()` 内で10個の変数に `random` を利用してx座標の値を代入
 - 横幅が400なので, `posX0 = random(400); ...`
- `draw()` 内で10個の変数の値を利用して円を描画
 - `ellipse(posX0, 150, 20, 20); ...`

円を沢山描画する



100個ならどうする？

```
float posX0;  
float posX1;  
float posX2;  
float posX3;  
float posX4;  
float posX5;  
float posX6;  
float posX7;  
float posX8;  
float posX9;  
  
void setup()  
{  
  size(400, 300);  
  posX0 = random(400);  
  posX1 = random(400);  
  posX2 = random(400);  
  posX3 = random(400);  
  posX4 = random(400);  
  posX5 = random(400);  
  posX6 = random(400);  
  posX7 = random(400);  
  posX8 = random(400);  
  posX9 = random(400);  
}
```

```
void draw()  
{  
  background(255);  
  fill(0, 255, 0);  
  ellipse(mouseX, mouseY, 20, 20);  
  fill(255, 0, 0);  
  ellipse(posX0, 150, 20, 20);  
  ellipse(posX1, 150, 20, 20);  
  ellipse(posX2, 150, 20, 20);  
  ellipse(posX3, 150, 20, 20);  
  ellipse(posX4, 150, 20, 20);  
  ellipse(posX5, 150, 20, 20);  
  ellipse(posX6, 150, 20, 20);  
  ellipse(posX7, 150, 20, 20);  
  ellipse(posX8, 150, 20, 20);  
  ellipse(posX9, 150, 20, 20);  
}
```


円を沢山描画する



- 配列で書き直すとこんな感じ

```
float[] posX = new float[10];  
void setup()  
{  
  size(400, 300);  
  posX[0] = random(400);  
  posX[1] = random(400);  
  posX[2] = random(400);  
  posX[3] = random(400);  
  posX[4] = random(400);  
  posX[5] = random(400);  
  posX[6] = random(400);  
  posX[7] = random(400);  
  posX[8] = random(400);  
  posX[9] = random(400);  
}
```

```
void draw()  
{  
  background(255);  
  fill(0, 255, 0);  
  ellipse(mouseX, mouseY, 20, 20);  
  fill(255, 0, 0);  
  ellipse(posX[0], 150, 20, 20);  
  ellipse(posX[1], 150, 20, 20);  
  ellipse(posX[2], 150, 20, 20);  
  ellipse(posX[3], 150, 20, 20);  
  ellipse(posX[4], 150, 20, 20);  
  ellipse(posX[5], 150, 20, 20);  
  ellipse(posX[6], 150, 20, 20);  
  ellipse(posX[7], 150, 20, 20);  
  ellipse(posX[8], 150, 20, 20);  
  ellipse(posX[9], 150, 20, 20);  
}
```

配列 + 繰り返し



- 配列は繰り返しとの組合せにより能力発揮
 - 繰り返しで増えていく数字を利用し，配列の何番目を指すかを指定する

```
float[] posX = new float[10];  
void setup()  
{  
  size(400, 300);  
  int i=0;  
  while(i<10)  
  {  
    posX[i] = random(400);  
    i++;  
  }  
}
```

```
void draw()  
{  
  background(255);  
  fill(0, 255, 0);  
  ellipse(mouseX, mouseY, 20, 20);  
  fill(255, 0, 0);  
  int j = 0;  
  while(j<10)  
  {  
    ellipse(posX[j], 150, 20, 20);  
    j++;  
  }  
}
```

配列 + 繰り返し (for)



- 配列は繰り返しとの組合せにより能力発揮
 - 繰り返しで増えていく数字を利用し，配列の何番目を指すかを指定する
 - for は配列との組み合わせでかなり有効！

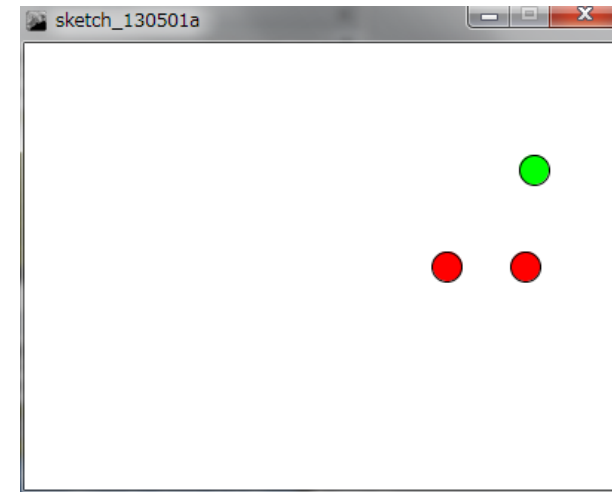
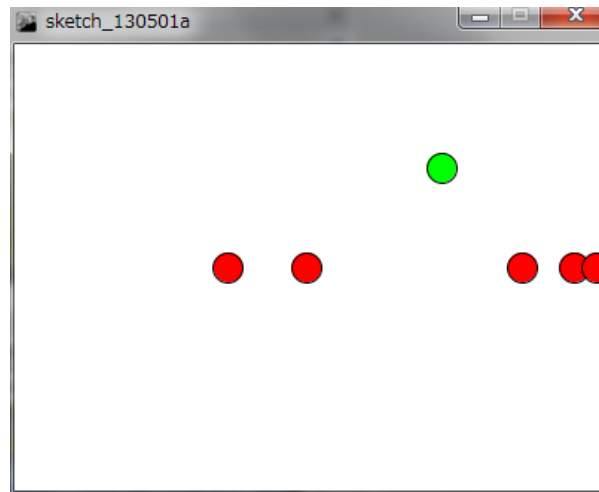
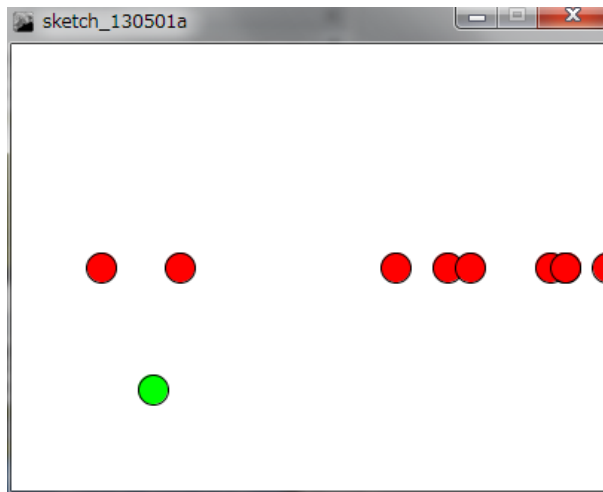
```
float[] posX = new float[10];  
void setup()  
{  
  size(400, 300);  
  for(int i = 0; i < 10; i++)  
  {  
    posX[i] = random(400);  
  }  
}
```

```
void draw()  
{  
  background(255);  
  fill(0, 255, 0);  
  ellipse(mouseX, mouseY, 20, 20);  
  fill(255, 0, 0);  
  for(int j = 0; j < 10; j++)  
  {  
    ellipse(posX[j], 150, 20, 20);  
  }  
}
```

円を一斉に動かす



(Q) 先ほど作成した400x300の画面に配置された10個の円を右方向に一斉に動かしてみよう



円を一斉に動かす



- 考え方

- 10個のX座標を格納する float 型の配列を用意
- 10個の配列に random で適当な値を代入
- 10個の配列の値に応じて円を描画
- 1回描画する度に, 配列のX座標を1つ右へ移動

円を一斉に動かす



(A) 先ほど作成した400x300の画面に配置された10個の円を右方向に一斉に動かしてみよう

```
float[] posX = new float[10];

void setup()
{
  size(400, 300);
  int i = 0;
  while(i < 10)
  {
    posX[i] = random(400);
    i++;
  }
}
```

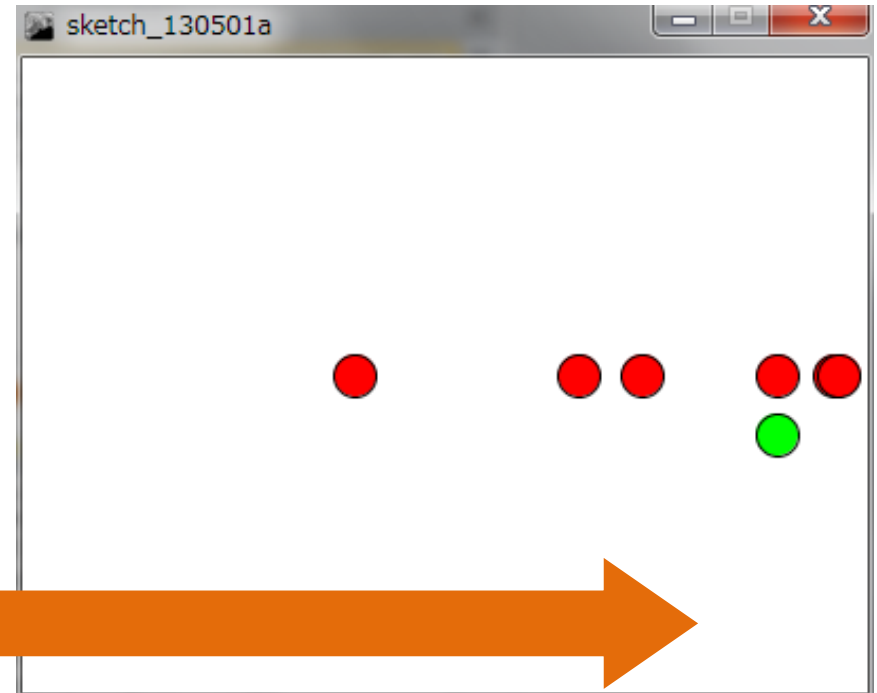
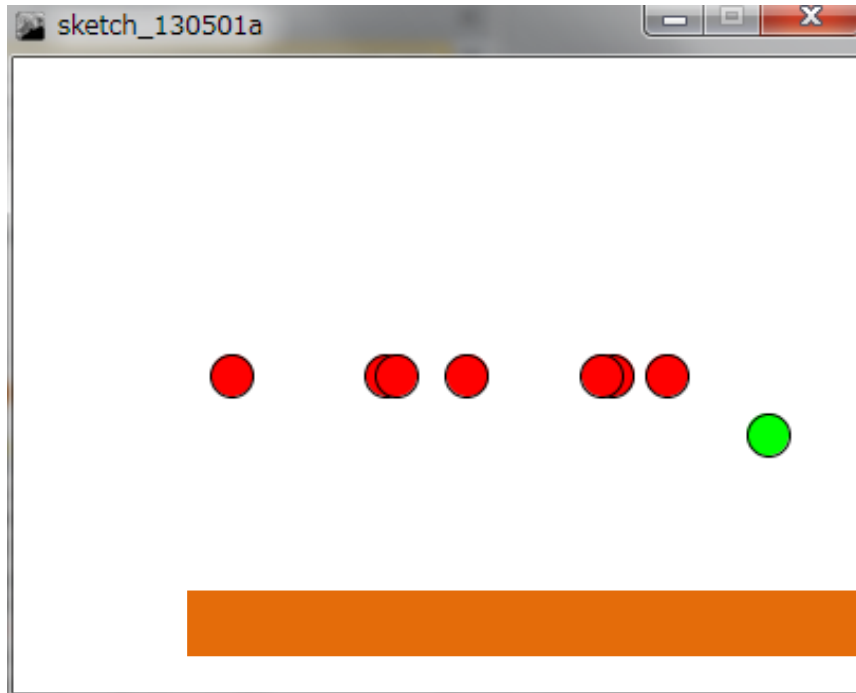
```
void draw()
{
  background(255);
  fill(0, 255, 0);
  ellipse(mouseX, mouseY, 20, 20);
  fill(255, 0, 0);
  int j = 0;
  while(j < 10)
  {
    ellipse(posX[j], 150, 20, 20);
    posX[j]++;
    j++;
  }
}
```

円を一斉に動かす ver. 2

明治大学総合数理学部
先端メディアサイエンス学科
中村研究室



(Q) 先ほど作成した400x300の画面に配置された10個の円を右方向に一斉に動かしてみよう
・ただし円ごとにスピードを変えてみよう



円を一斉に動かす ver. 2



- 考え方

- 10個のそれぞれの円についてスピードを格納する float 型の配列を用意
 - `float[] speed = new float[10];`
- スピード用の配列に random で適当な値を代入
- 10個の配列の値に応じて円を描画
- 1回描画する度に, スピードの量だけ配列のX座標を1つ右へ移動
 - `j` 番目のX座標の値に, `j` 番目のスピードを足す

円を一斉に動かす ver. 2



- それぞれのスピードで円が右に動いていく
 - random(5) の5の値を大きくすると、スピードが速くなる！

```
float[] posX = new float[10];
float[] speed = new float[10];

void setup()
{
  size(400, 300);
  int i = 0;
  while(i < 10)
  {
    posX[i] = random(400);
    speed[i] = random(5);
    i++;
  }
}
```

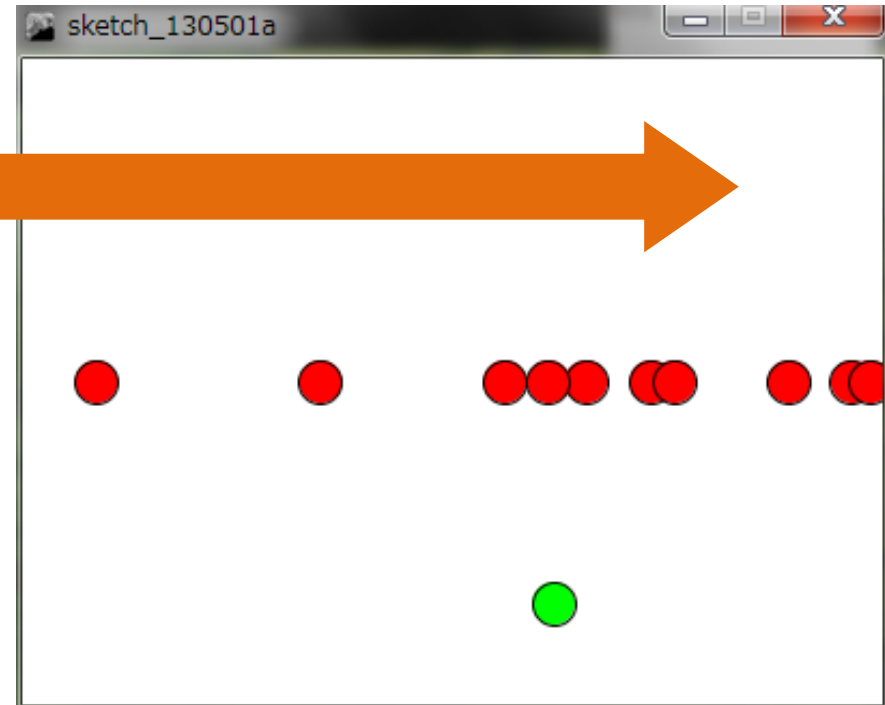
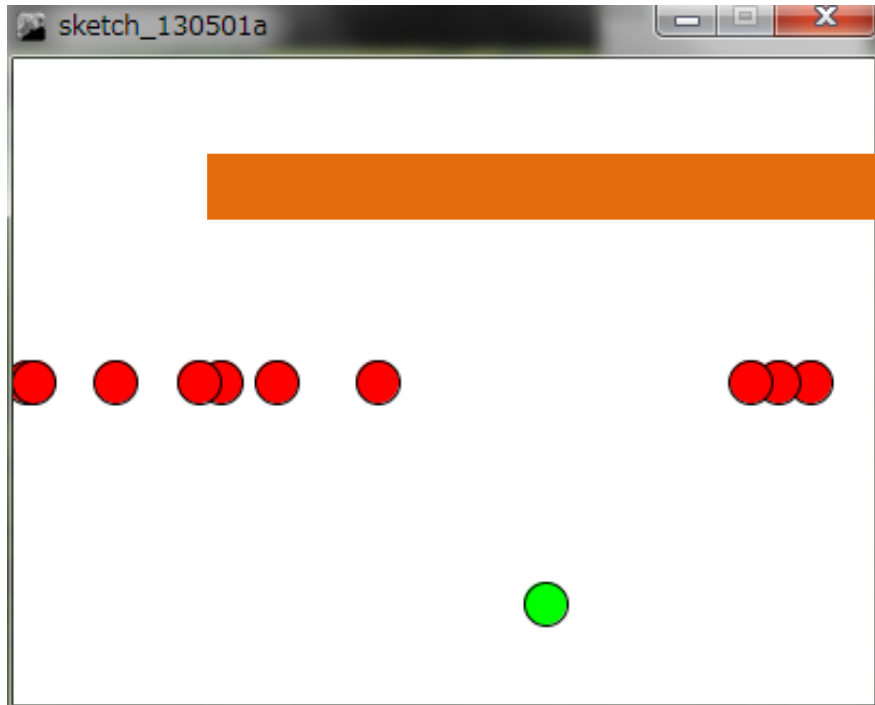
```
void draw()
{
  background(255);
  fill(0, 255, 0);
  ellipse(mouseX, mouseY, 20, 20);
  fill(255, 0, 0);
  int j = 0;
  while(j < 10)
  {
    posX[j] = posX[j] + speed[j];
    ellipse(posX[j], 150, 20, 20);
    j++;
  }
}
```

円を一斉に動かす ver. 3

明治大学総合数理学部
先端メディアサイエンス学科
中村研究室



(Q) 先ほど作成した400x300の画面に配置された10個の円を右方向に一斉に動く円は、右端に来ると左端から出てくるようにしよう



円を一斉に動かす ver. 3

明治大学総合数理学部
先端メディアサイエンス学科
中村研究室



- 考え方

- 円が右端に来た時，円を左端に登場させる
- 右端のX座標は400，左端のX座標は0
- 移動後の円の座標を計算し，400を超えていたら0にする（または，座標から400引く） → if文

円を一斉に動かす ver. 3

明治大学総合数理学部
先端メディアサイエンス学科
中村研究室



- `if(posX[j] > 400)` なら `posX[j]` から 400引く！

```
float[] posX = new float[10];
float[] speed = new float[10];

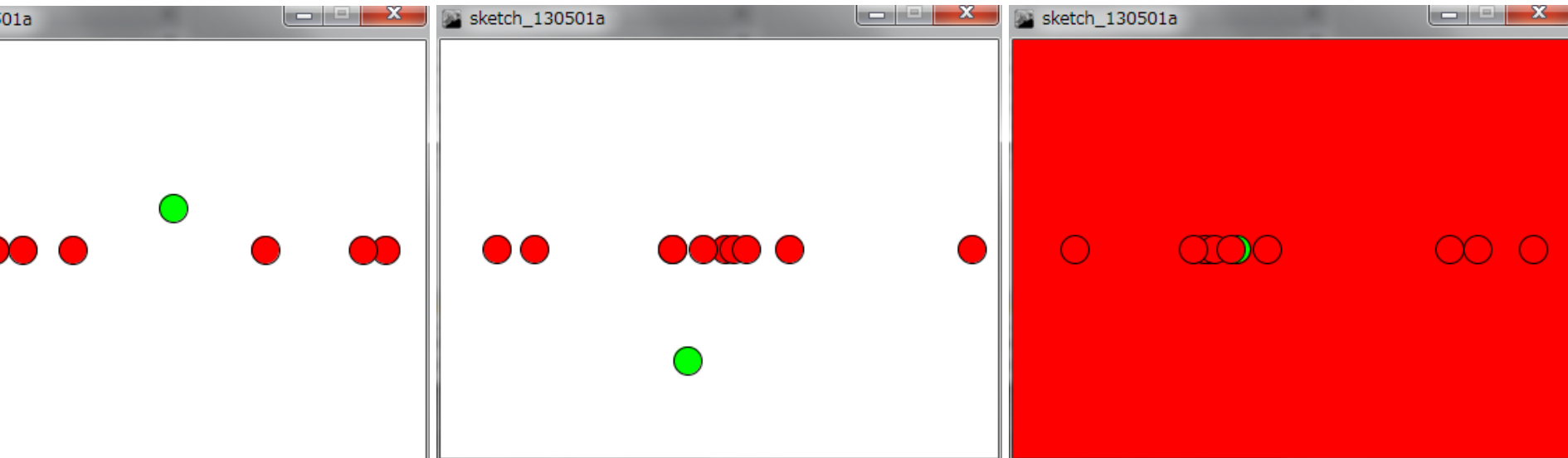
void setup()
{
  size(400, 300);
  int i = 0;
  while(i < 10)
  {
    posX[i] = random(400);
    speed[i] = random(5);
    i++;
  }
}
```

```
void draw()
{
  background(255);
  fill(0, 255, 0);
  ellipse(mouseX, mouseY, 20, 20);
  fill(255, 0, 0);
  int j = 0;
  while(j < 10)
  {
    posX[j] = posX[j] + speed[j];
    if(posX[j] > 400)
    {
      posX[j] = posX[j] - 400;
    }
    ellipse(posX[j], 150, 20, 20);
    j++;
  }
}
```

動く円群との衝突判定



(Q) 先ほど作成した400x300の画面に配置された10個の右方向に動く円と、マウスカーソルが衝突したら背景を赤色にしよう！



動く円群との衝突判定



• 考え方

- カーソルとそれぞれの円の距離は `dist` で求める
- `j` 番目の円とカーソルまでの距離は
 - `dist(posX[j], 150, mouseX, mouseY);`
- `dist` が 20 以下なら円と衝突しているので, その場合に背景の色を赤色にする
- 背景の色を変更するための変数を用意
 - `int crash;` など. 通常は 0, クラッシュすると 1 にするなど
- 背景色を変更の変数に応じて背景色を変更

動く円群との



```
float[] posX = new float[10];  
float[] speed = new float[10];
```

```
void setup()  
{  
  size(400, 300);  
  int i = 0;  
  while(i < 10)  
  {  
    posX[i] = random(400);  
    speed[i] = random(5);  
    i++;  
  }  
}
```

```
void draw()  
{  
  int clash = 0;  
  for(int j = 0; j < 10; j++)  
  {  
    posX[j] = posX[j] + speed[j];  
    if(posX[j] > 400)  
    {  
      posX[j] = posX[j] - 400;  
    }  
    if(dist(posX[j], 150, mouseX, mouseY) < 20)  
    {  
      clash = 1;  
    }  
  }  
  
  if(clash == 0){  
    background(255, 255, 255);  
  } else {  
    background(255, 0, 0);  
  }  
  
  fill(0, 255, 0);  
  ellipse(mouseX, mouseY, 20, 20);  
  fill(255, 0, 0);  
  for(int j = 0; j < 10; j++)  
  {  
    ellipse(posX[j], 150, 20, 20);  
  }  
}
```

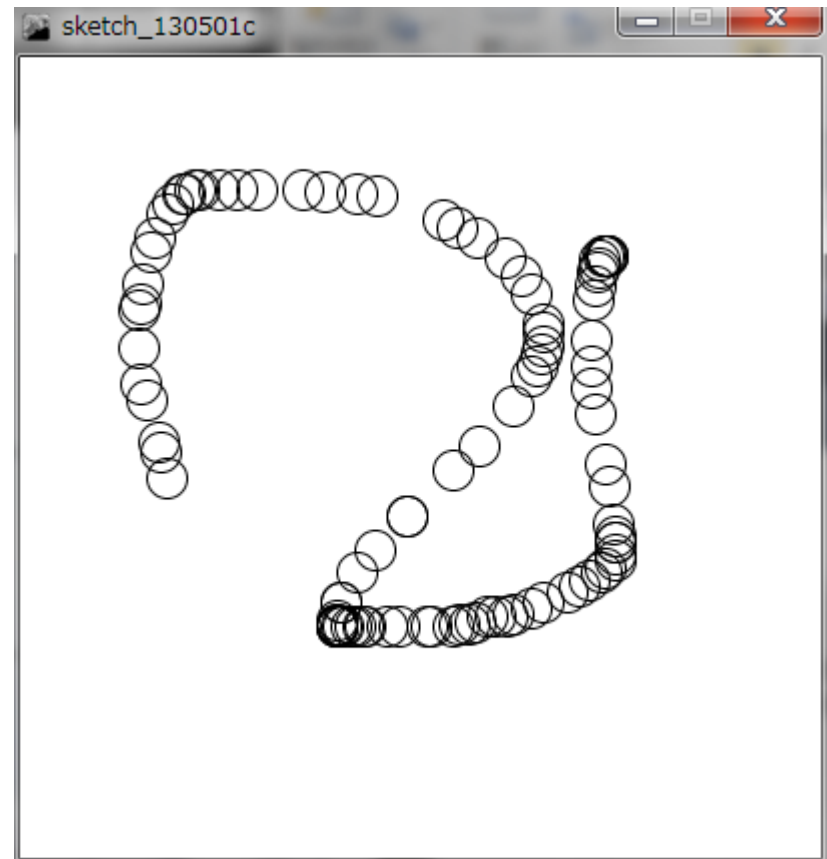
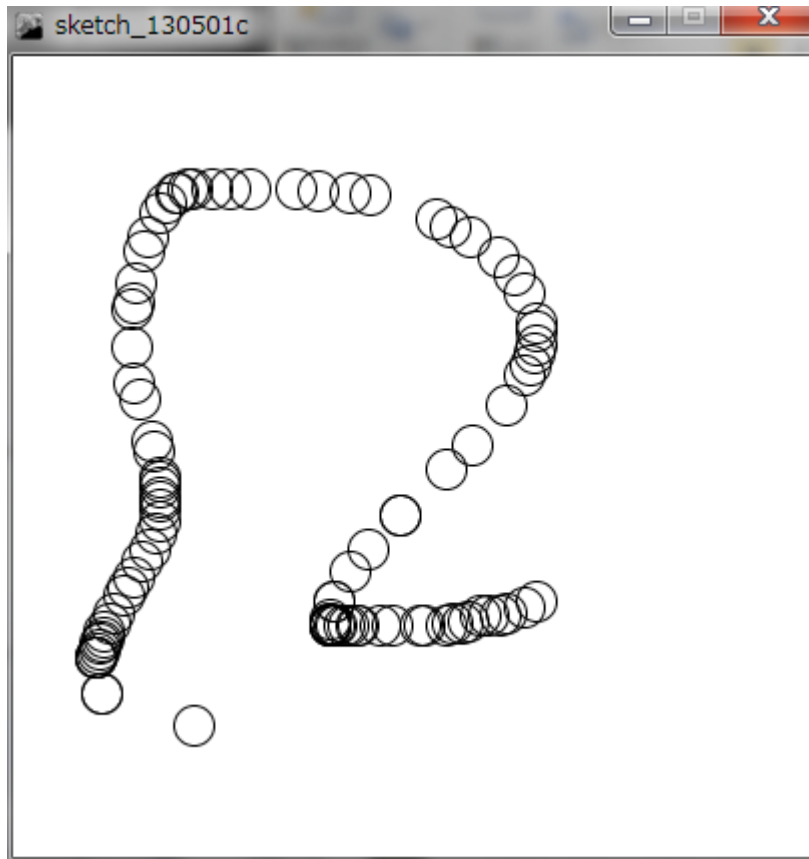


- 一斉に動く円の数を, 10個から20個に増やしてみましよう
- 一斉に動く円の数を, 10個から100個に増やしてみましよう
- 一斉に動く円の方角を左方角にしてみましよう
 - speed をマイナスにしたらOK!
- 一斉に動く円を, 右方角にも左方角にも動くようにしてみましよう
 - speed を-5から5までにしたらOK!

マウスの軌跡を表示する



(Q) マウスカーソルを追尾する100個の円を描画するにはどうするか？



マウスの軌跡を表示する



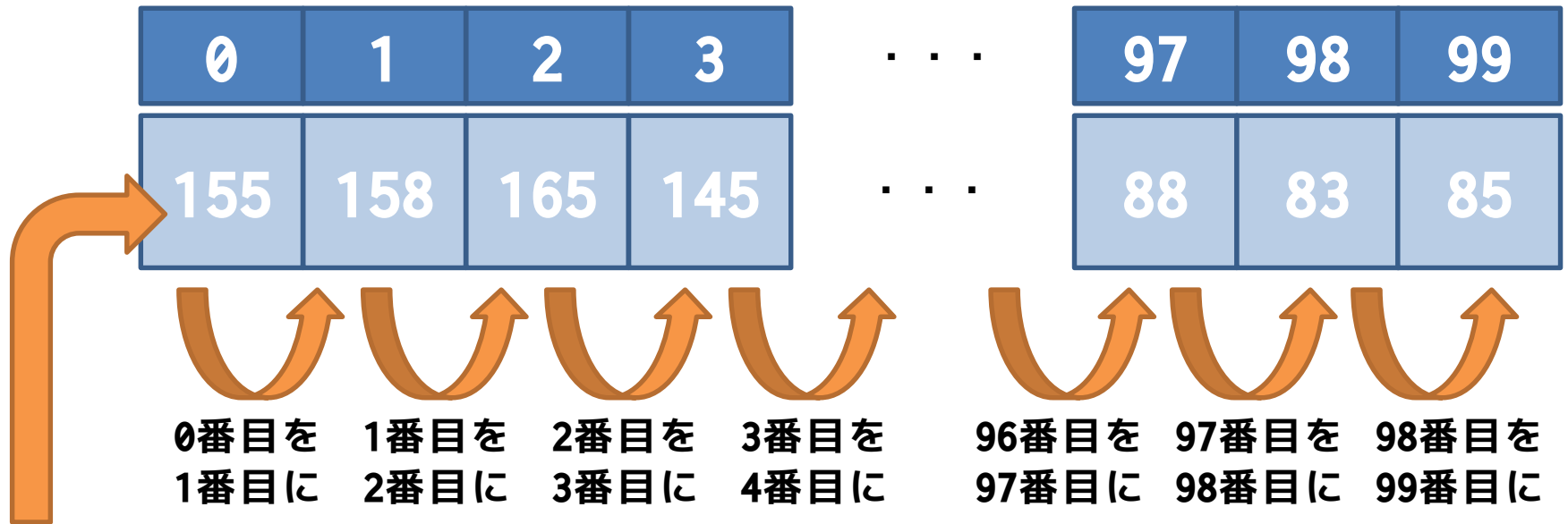
• 考え方

- マウスカーソルの座標は `mouseX`, `mouseY`
- マウスカーソルの以前の座標を記録するため, 要素数が100のXおよびYの配列用意
 - `int[] cursorX = new int[100];`
 - `int[] cursorY = new int[100];`
- 現在のフレームの座標を0番目, 1フレーム前の座標を1番目, 2フレーム前の座標を2番目 . . . とする
- 配列`cursorX`と`cursorY`について, 0から99まで変化するすべての `i` について`cursorX[i]`, `cursorY[i]`を中心とした円を描画

マウスの軌跡を表示する



cursorX



mouseX

cursorX[i] = cursorX[i-1];

cursorY[i] = cursorY[i-1];

1つずつ右へずらしていく

マウス軌跡を表示する



- ありがちなミス（setupと描画処理は省略）

```
int[] cursorX = new int[100];
int[] cursorY = new int[100];

void draw()
{
    cursorX[0] = mouseX;
    cursorY[0] = mouseY;
    int i = 0;
    while(i <= 99)
    {
        cursorX[i] = cursorX[i - 1];
        cursorY[i] = cursorY[i - 1];
        i++;
    }
    // ここに描画の処理
}
```

何が間違っているか？

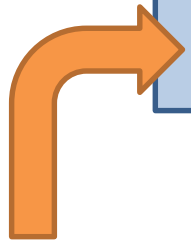
0番目を1番目に
1番目を2番目に
2番目を3番目に
3番目を4番目に
⋮

マウスの軌跡を表示する

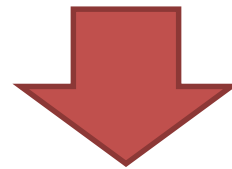


cursorX (処理前)

0	1	2	3	...	97	98	99
56	58	65	45	...	88	83	85



mouseX (55)



cursorX (処理後)

0	1	2	3	...	97	98	99
55	55	55	55	...	55	55	55

左から順に処理すると全て同じ値に

マウス軌跡を表示する



- 右から順に処理する（setupと描画は省略）

```
int[] cursorX = new int[100];
int[] cursorY = new int[100];

void draw()
{
    int i = 99;
    while(i > 0){
        cursorX[i] = cursorX[i - 1];
        cursorY[i] = cursorY[i - 1];
        i--;
    }
    cursorX[0] = mouseX;
    cursorY[0] = mouseY;
    // ここに描画の処理
}
```

98番目を99番目に
97番目を98番目に
96番目を97番目に
95番目を96番目に

⋮



```
int[] cursorX = new int[100];
int[] cursorY = new int[100];
void setup()
{
  size(400, 400);
  noFill();
}

void draw()
{
  background(255);
  int i = 99;
  while(i > 0)
  {
    cursorX[i] = cursorX[i-1];
    cursorY[i] = cursorY[i-1];
    i--;
  }
  cursorX[0] = mouseX;
  cursorY[0] = mouseY;

  i = 0;
  while(i < 100)
  {
    ellipse(cursorX[i], cursorY[i], 20, 20);
    i++;
  }
}
```

配列の内容を
一つずつずらす

cursorX[i], cursorY[i] に
円を描画

配列の値がどう動くか？



- 配列の値の動きがわかりにくい人のために
次ページにプログラムを用意したので入力
して実行し、マウスを動かしてみよう！
(マウスのX座標が順に引き継がれます)

0	1	2	3	4	5	6	7	8	9
524	524	529	552	642	715	679	560	488	446

0	1	2	3	4	5	6	7	8	9
79	94	136	160	162	300	524	524	524	524

配列の値がどう動くか？



```
void setup()
{
  size(800, 300);
  frameRate(2);
  textSize(30);
  stroke(255);
  strokeWeight(3);
}
```

配列の内容を
1つずつずらす

```
int[] arrayX = new int[10];
void draw()
{
  background(0);
  for (int i = 9; i > 0; i--)
  {
    arrayX[i] = arrayX[i - 1];
  }
  arrayX[0] = mouseX;
  for (int i = 0; i < 10; i++)
  {
    fill(0);
    rect(i * 80, 0, (i + 1) * 80, 50);
    rect(i * 80, 50, (i + 1) * 80, 70);
    fill(255);
    text(i, i * 80 + 10, 40);
    text(arrayX[i], i * 80 + 10, 90);
  }
}
```

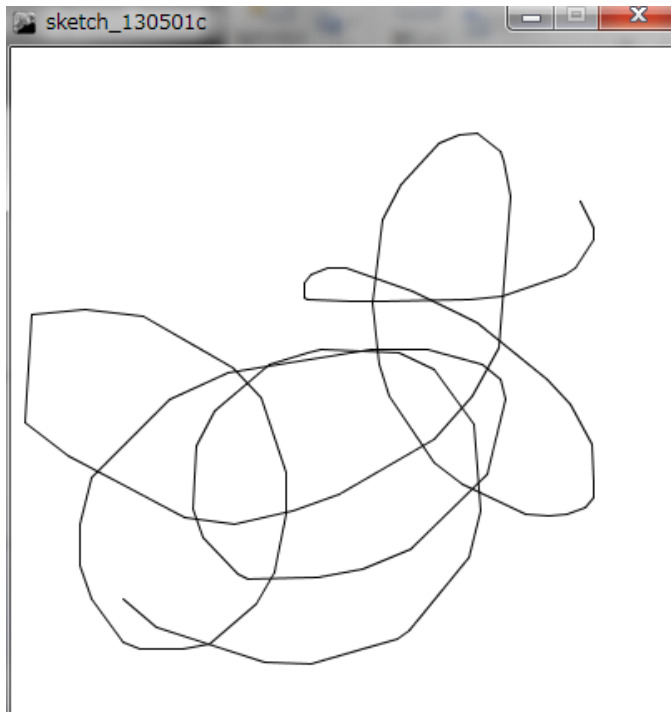
予習問題



- マウスの軌跡に線を描画するプログラムを作ってみましょう

(ヒント)

```
line(cursorX[i], cursorY[i], cursorX[i - 1], cursorY[i - 1]);
```



**配列のエラー
に注意！**

パラパラアニメーション

明治大学総合数理学部
先端メディアサイエンス学科
中村研究室



(Q) 用意した10枚の画像をパラパラ切り替えるアニメーションを作りたい

まず、画像の表示



PImage 画像用変数;

画像用変数 = loadImage("画像名"); で準備

image(画像用変数, x座標, y座標); で表示

画像はプログラムにドロップで利用可能に

(ドラッグアンドドロップしないと使えない)

```
PImage mapImage = loadImage("map.png");  
size(640, 400);  
background(255);  
image(mapImage, 0, 0);
```



- 考え方

- 画像を10枚用意（ペイントで描いても，写真を撮影してもOK．名前は適当に順番を付けましょう）
- 要素数が10の PImage 型の配列を作る
 - PImage[] parapara = new PImage[10];
- setup で画像をすべて読み込む（loadImage）
 - parapara[0] = loadImage("gazo0.jpg");
- draw で表示する画像番号を変数 i として準備
- draw の度に i 番目の画像を表示
- i が10になったら0に戻す
- frameRate(10); で draw の更新速度を設定

パラパラアニメーション

明治大学総合数理学部
先端メディアサイエンス学科
中村研究室



```
PImage[] parapara = new PImage[10];
int i = 0;

void setup()
{
  size(800, 600);
  parapara[0]=loadImage("gazo0.jpg");
  parapara[1]=loadImage("gazo1.jpg");
  parapara[2]=loadImage("gazo2.jpg");
  parapara[3]=loadImage("gazo3.jpg");
  parapara[4]=loadImage("gazo4.jpg");
  parapara[5]=loadImage("gazo5.jpg");
  parapara[6]=loadImage("gazo6.jpg");
  parapara[7]=loadImage("gazo7.jpg");
  parapara[8]=loadImage("gazo8.jpg");
  parapara[9]=loadImage("gazo9.jpg");
  frameRate(10);
}
```

```
void draw()
{
  image(parapara[i], 0, 0);
  i++;
  if(i==10)
  {
    i=0;
  }
}
```

ちなみに



```
parapara[0] = loadImage("gazo0.jpg");  
parapara[1] = loadImage("gazo1.jpg");  
parapara[2] = loadImage("gazo2.jpg");  
parapara[3] = loadImage("gazo3.jpg");  
parapara[4] = loadImage("gazo4.jpg");  
:  
parapara[9] = loadImage("gazo9.jpg");
```



```
for(int j = 0; j < 10; j++)  
{  
    parapara[j] = loadImage("gazo" + j + ".jpg");  
}
```

予習問題



- 400x300の画面内にX座標Y座標ともにランダムに配置された50個の円を描き，それぞれのX，Y方向へのスピードに応じて移動するようにせよ
 - posX， posY という座標の配列と， speedX， speedY というXおよびY方向の速度をもつ配列を導入！

