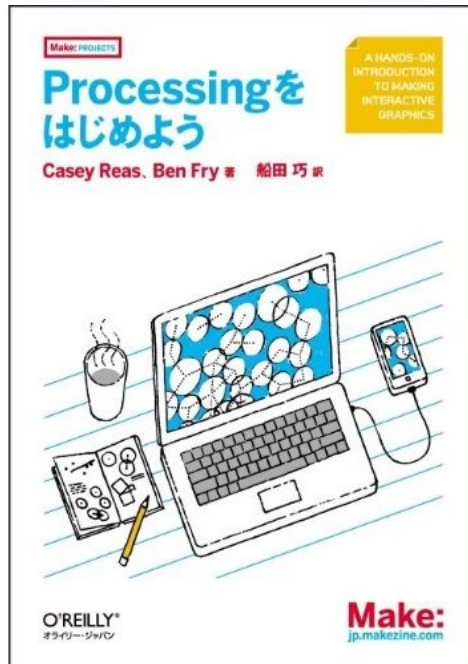


Practice #4

アニメーション (時間的な処理)

演習 4B ランダムドロ잉



Processingをはじめよう
第二版
(Make: PROJECTS)

オンデマンド授業 6.18

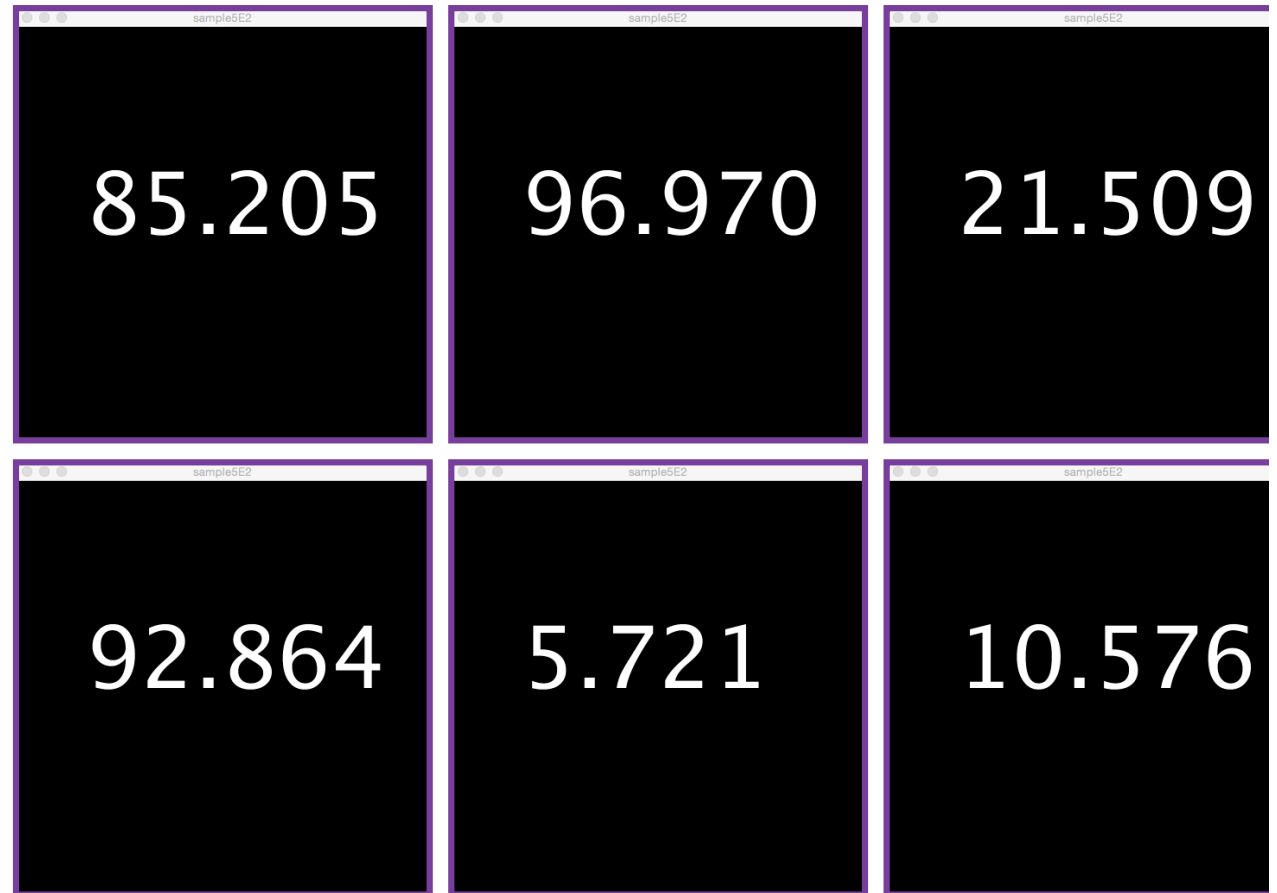
p. 64 - 71

p. 236 - 239

ランダム関数

sample4B_1.pde

```
1
2 void setup(){
3   size(480,480);
4   background(0);
5   fill(255);
6   textSize(100);
7 }
8
9 void draw(){
10 }
11
12 void mousePressed(){
13   background(0);
14   float r = random(100);
15   text(r,50,240);
16 }
```



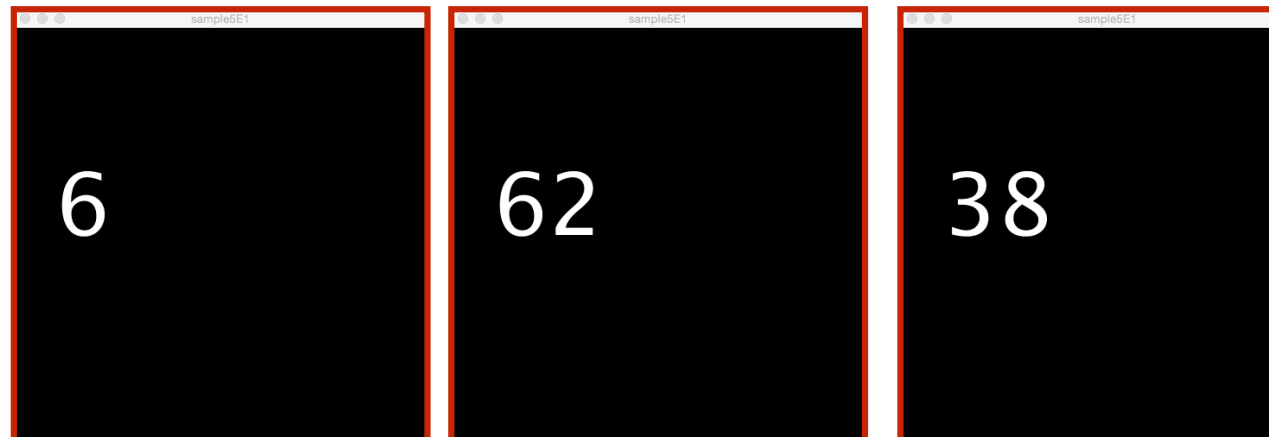
float random(n);

0からnまでの乱数を発生させる。

```
12 void mousePressed(){
13   background(0);
14   //float r = random(100);
15   int r = int(random(100));
16   text(r,50,240);
17 }
```

int int(f);

fを整数に変換する。



ランダム関数のビジュアライゼーション

sample4B_2.pde

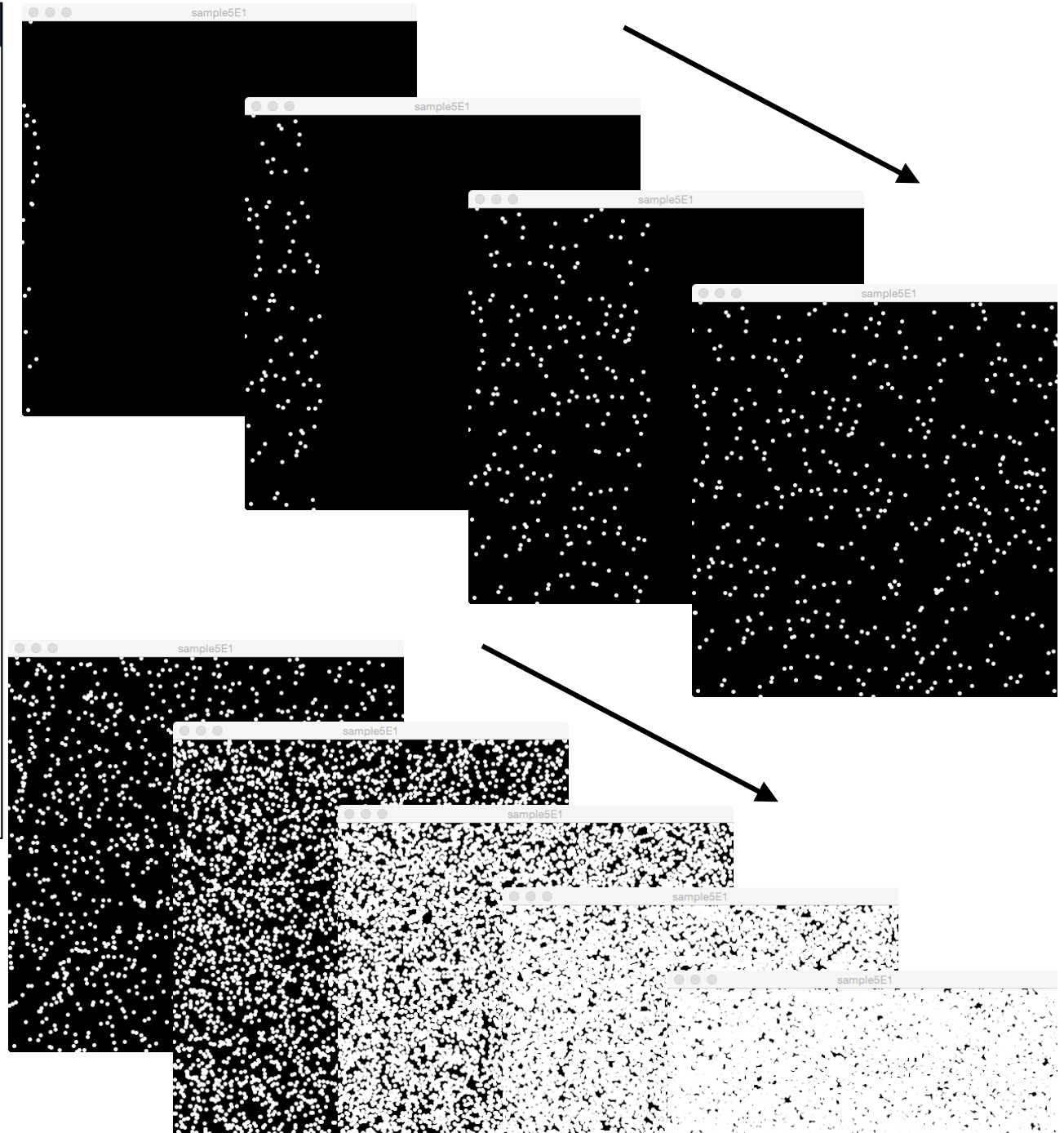
```
1 int x = 0;
2
3 void setup(){
4   size(480,480);
5   background(0);
6   stroke(255); strokeWeight(5);
7
8   frameRate(100);
9 }
10
11 void draw(){
12   point(x,random(height));
13
14   x++;
15
16   if(x==width){
17     x = 0;
18   }
19 }
20
21 }
```

void strokeWeight(n);

線の幅をnピクセルとする

void point(x, y);

(x,y) に点を打つ。



ランダムウォーク

sample4B_3.pde

```
1 int x; float y;
2
3 void setup(){
4   size(480,480); background(0);
5   stroke(255); strokeWidth(5);
6   x = 0; y = 0.5 * height;
7
8   frameRate(100);
9 }
10
11 void draw(){
12   float n = 10;
13   y = y + random(2*n) - n;
14   y = constrain(y, 0, height);
15
16   point(x,y);
17   x++;
18
19   if(x==width){
20     x = 0;
21   }
22 }
```

以下の計算により、現在のY座標を中心に、最大移動距離 (n) でランダムウォークできます。

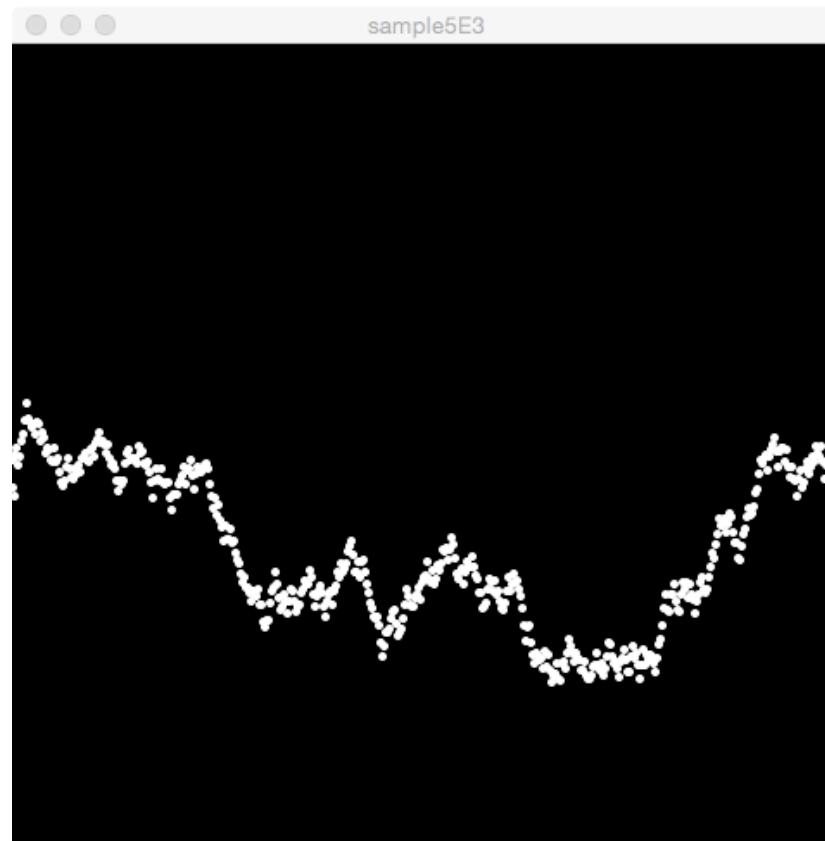
$$y = y + \text{random}(2*n) - n;$$

$$-n < \text{random}(2*n) - n < n$$

$$(y-n) < y + \text{random}(2*n) - n < (y+n)$$

`float constrain(a,min,max);`

aをminとmaxの間に収める。
minより小さい場合はminに、
maxより大きい場合はmax、
それ以外はそのままaを返す。



二次元ランダムウォーク

sample4B_4.pde

```
1 float px; float py;  
2 float x; float y;
```

(px, py)を1フレーム前の
ポイント, (x, y)を現フレ
ームのポイントとする。

```
4 void setup(){  
5   size(480,480); background(0);  
6   stroke(255); strokeWidth(1);
```

```
8   px = 0.5 * width; py = 0.5 * height;  
9   x = 0.5 * width; y = 0.5 * height;
```

```
11  frameRate(100);  
12 }
```

(px, py), (x, y) の初期点を
ウィンドウの中点とする。

```
14 void draw(){
```

```
16   float n = 10;
```

```
17   x = x + random(2*n) - n;  
18   x = constrain(x, 0, width);
```

x座標のラン
ダムウォーク

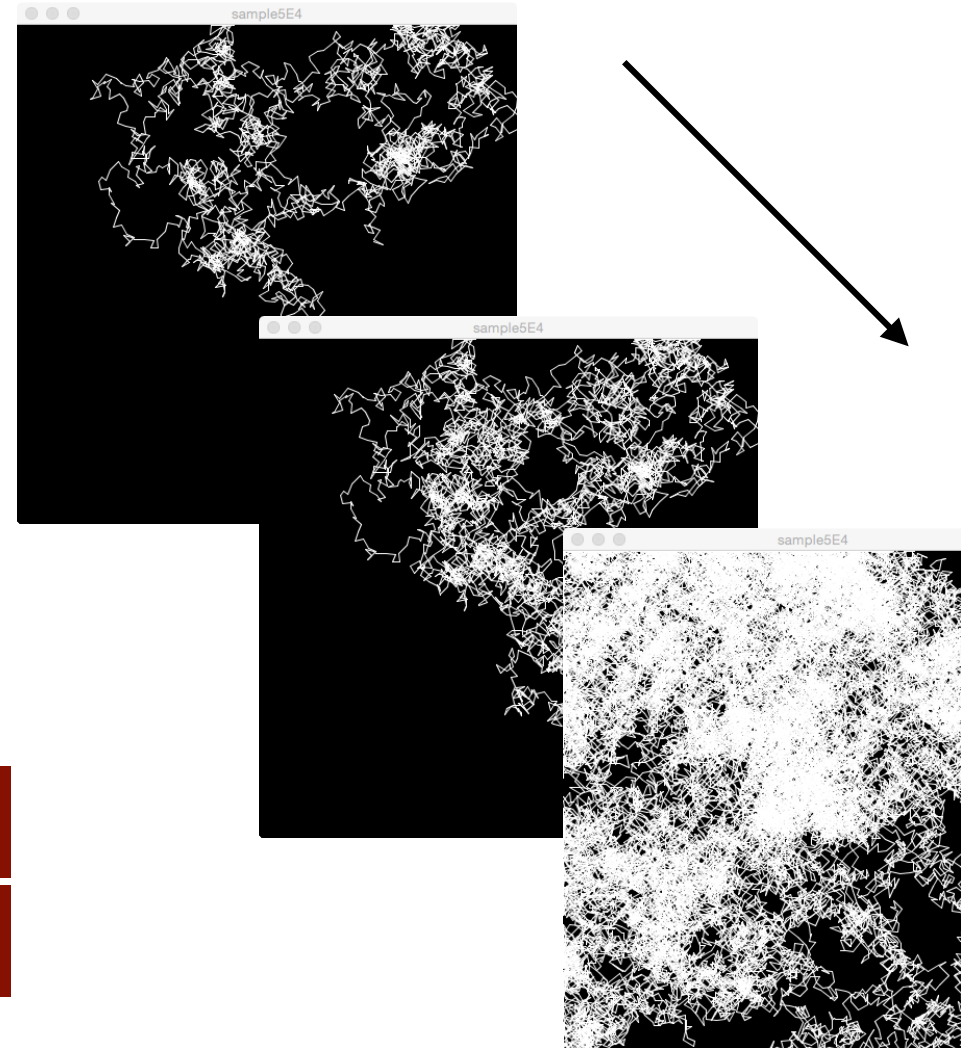
```
19   y = y + random(2*n) - n;  
20   y = constrain(y, 0, height);
```

y座標のラン
ダムウォーク

```
22   line(px,py,x,y);  
23   px = x; py = y;
```

(px,py)と(x,y)に線を引き,
(px,py)を更新する。

```
25 }
```



最大移動距離 (n) を変更することで、ランダム
ウォークの細やかさを変更することができます。

二次元ランダムウォーク インサイドサークル

sample4B_5.pde

```
1 float px; float py;
2 float x; float y;
3 float cx; float cy;
4
5 void setup(){
6   size(480,480); background(0);
7   stroke(255); strokeWeight(1);
8
9   cx = 0.5 * width; cy = 0.5 * height;
10  px = cx; py = cy; x = cx; y = cy;
11
12  frameRate(1000);
13 }
```

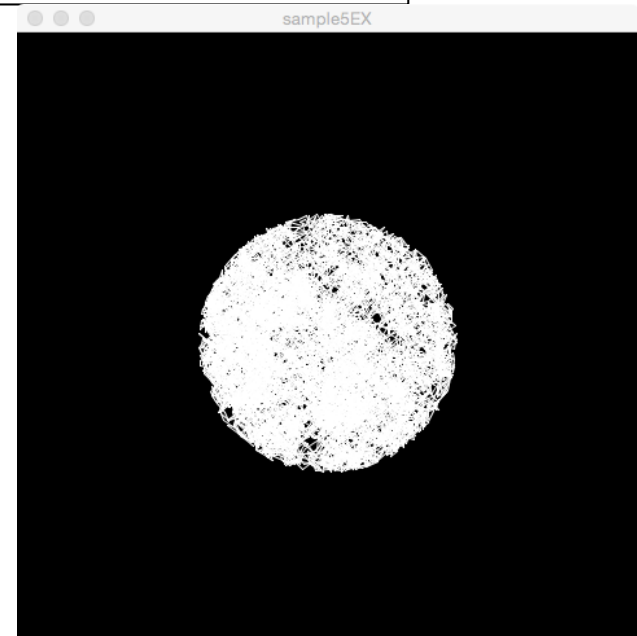
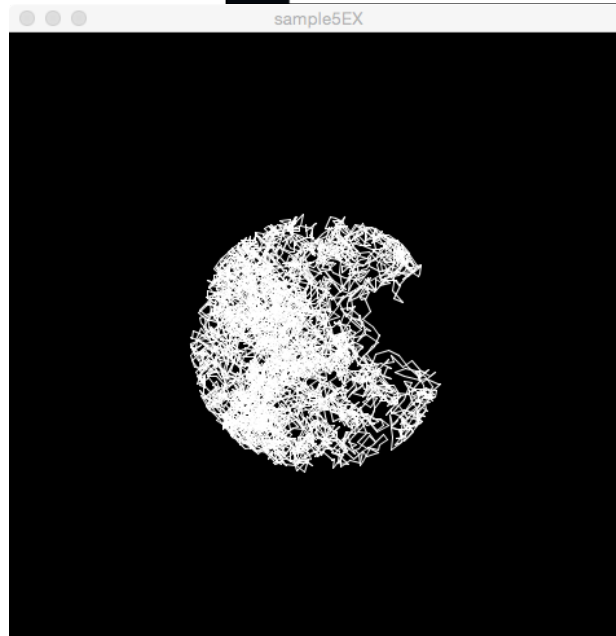
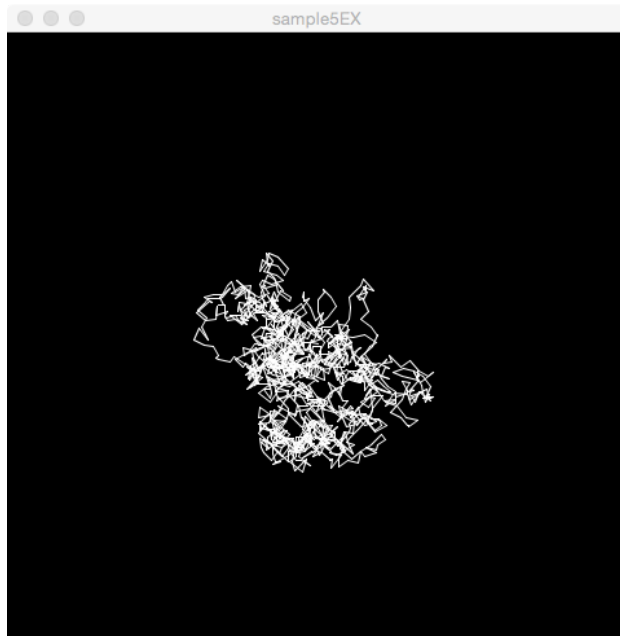
(cx, cy)はウィンドウの中心

出発点が中心！！

```
15 void draw(){
16
17   float n = 10;
18
19   x = x + random(2*n) - n;
20   y = y + random(2*n) - n;
21
22   if(dist(x,y,cx,cy)>100.){
23     x = px; y = py;
24   }
25
26   line(px,py,x,y);
27   px = x; py = y;
28
29 }
```

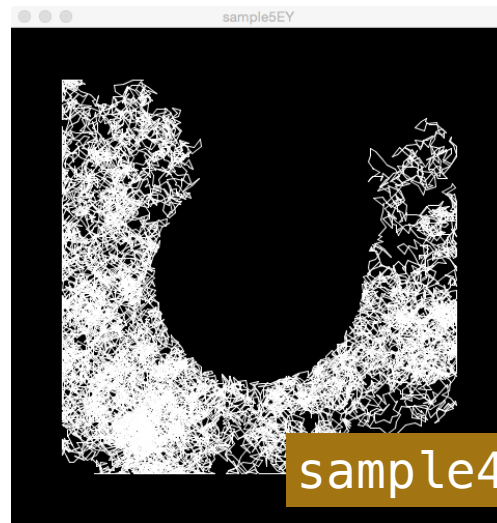
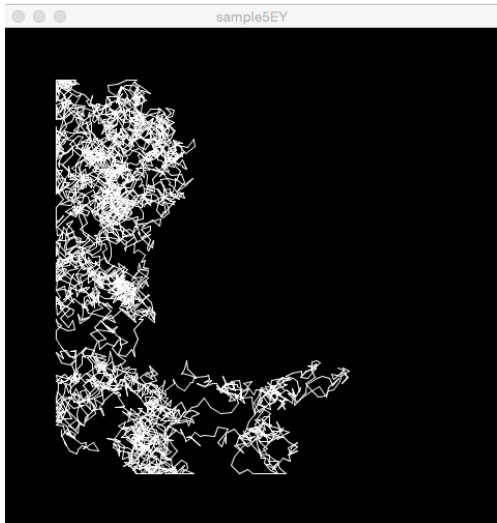
(各軸) 最大距離 n のランダムウォーク

(x, y)と(cx,cy)の距離が100を超えた場合、移動をキャンセル

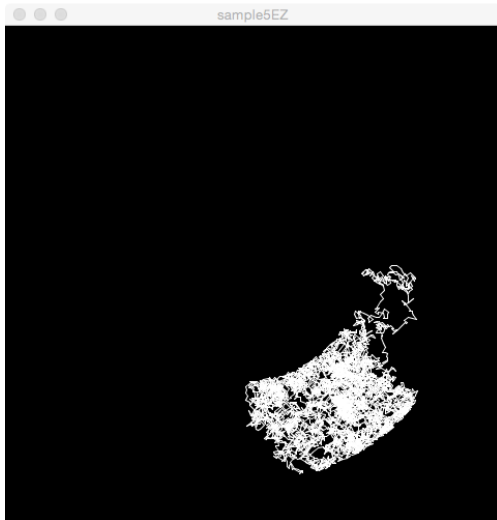
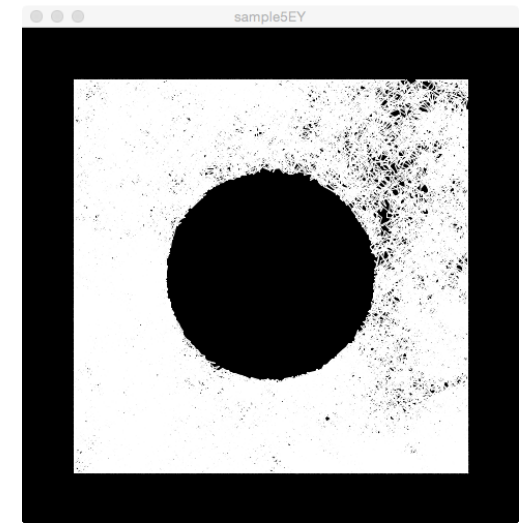
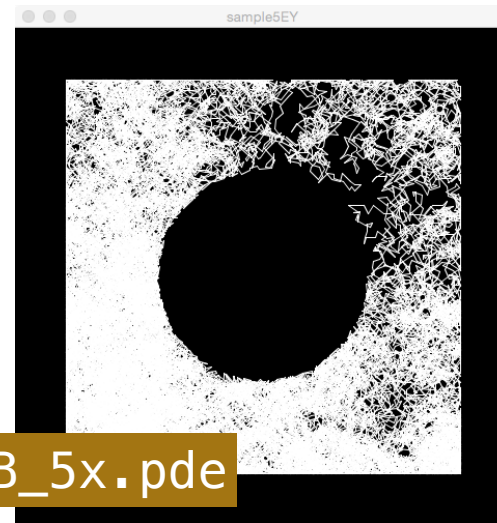


練習

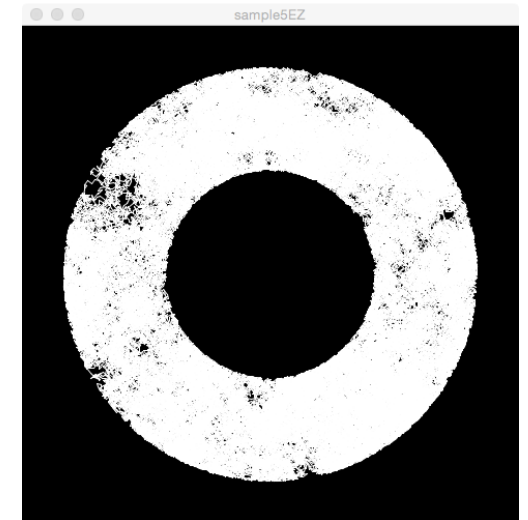
以下のような区間内でランダムウォークをする描画をプログラムしてください。



sample4B_5x.pde



sample4B_5y.pde

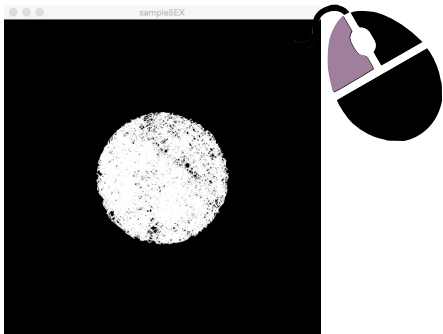


マウスイベント

sample4_B5のdraw()関数の中身を, mousePressed関数の中に移すことによって, マウスのクリックによって, ランダムウォークの描画を呼び出すことができます.

sample4B_6.pde

```
1 float px; float py;
2 float x; float y;
3 float cx; float cy;
4
5 void setup(){
6   size(480,480); background(0);
7   stroke(255); strokeWeight(1);
8
9   cx = 0.5 * width; cy = 0.5 * height;
10  px = cx; py = cy; x = cx; y = cy;
11
12 }
13
14 void draw(){
15 }
```



```
17 void mousePressed(){
18   for(int i=0;i<1000;i++){
19
20     float n = 10;
21     x = x + random(2*n) - n;
22     y = y + random(2*n) - n;
23
24     if(dist(x,y,cx,cy)>100.){
25       x = px; y = py;
26     }
27     line(px,py,x,y);
28     px = x; py = y;
29
30   }
31 }
```

ランダム
ウォーク

x 1000

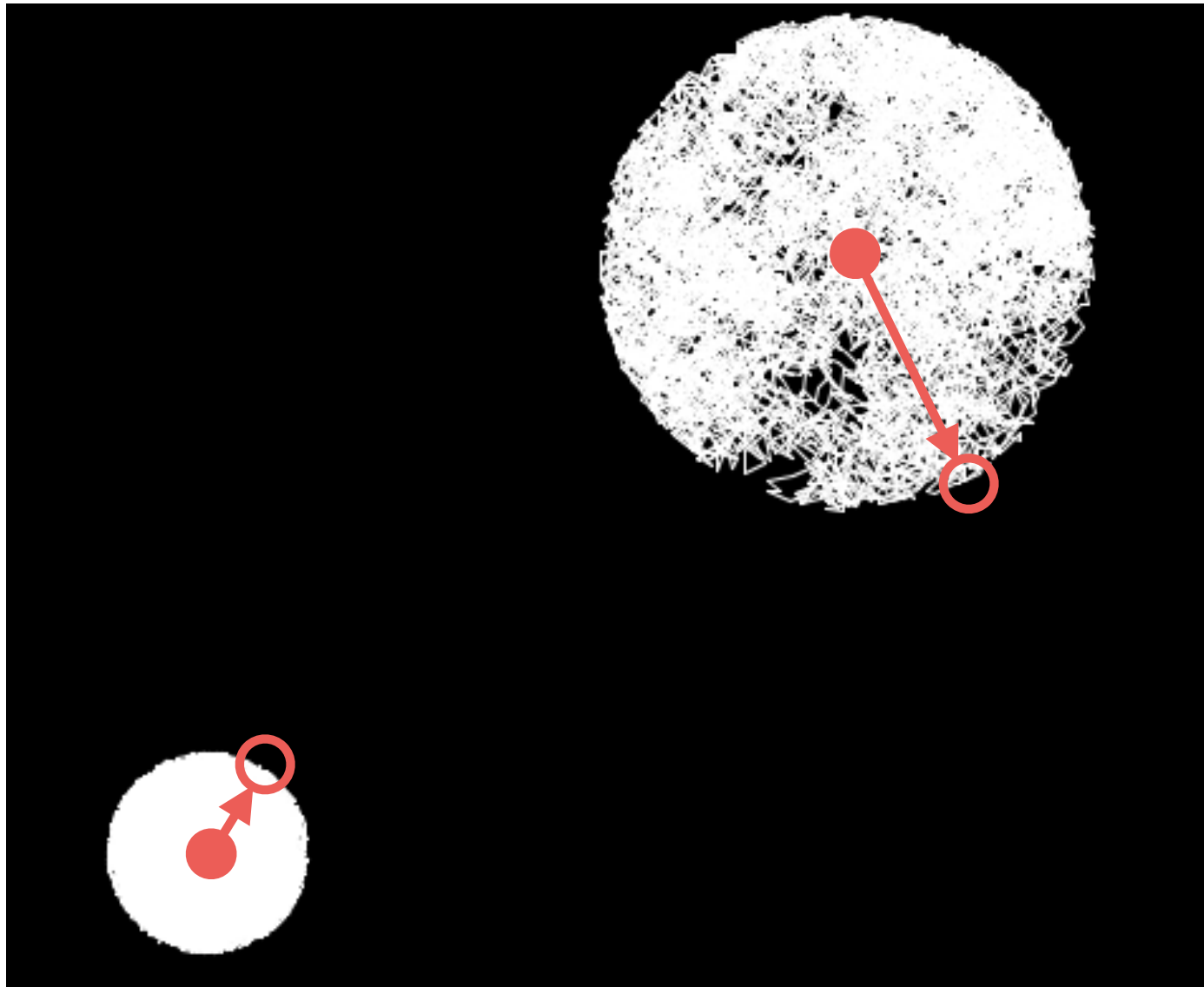
マウスを押した時の処理

```
32
33 void mouseReleased(){
34   background(0);
35 }
```

マウスを離れたときの処理：
画面をクリアする

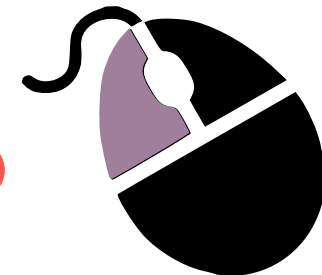
練習

sample4B_6を修正し, 下のよう^に, マウスをクリックした位置を中心とし, その中心から, マウスの離れた位置までの距離を半径とする円の内部にランダムウォークで描画を完成させてください.



sample4_B6x.pde

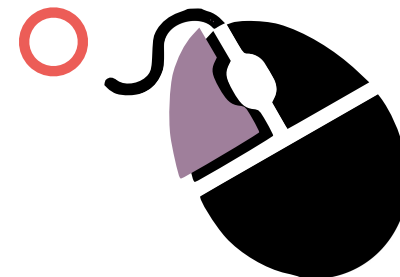
マウスをクリック



ドラック&ドロップ



マウスを離す



ランダムカラー

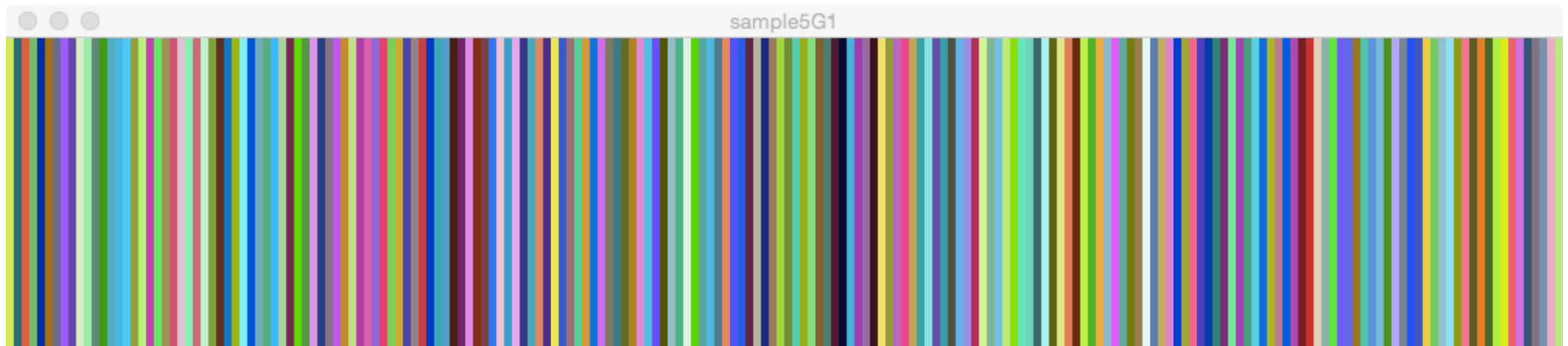
sample4B_7.pde

```
1 int x = 0;
2
3 void setup(){
4   size(1000,200); background(0);
5   noStroke();
6   frameRate(100);
7 }
```

RGB値のそれぞれをランダムに決定しています。

```
9 void draw(){
10
11   float red = random(255);
12   float green = random(255);
13   float blue = random(255);
14
15   color col = color(red,green,blue);
16
17   fill(col);
18   rect(x,0,5,height);
19   x = (x+5) % width;
20
21 }
```

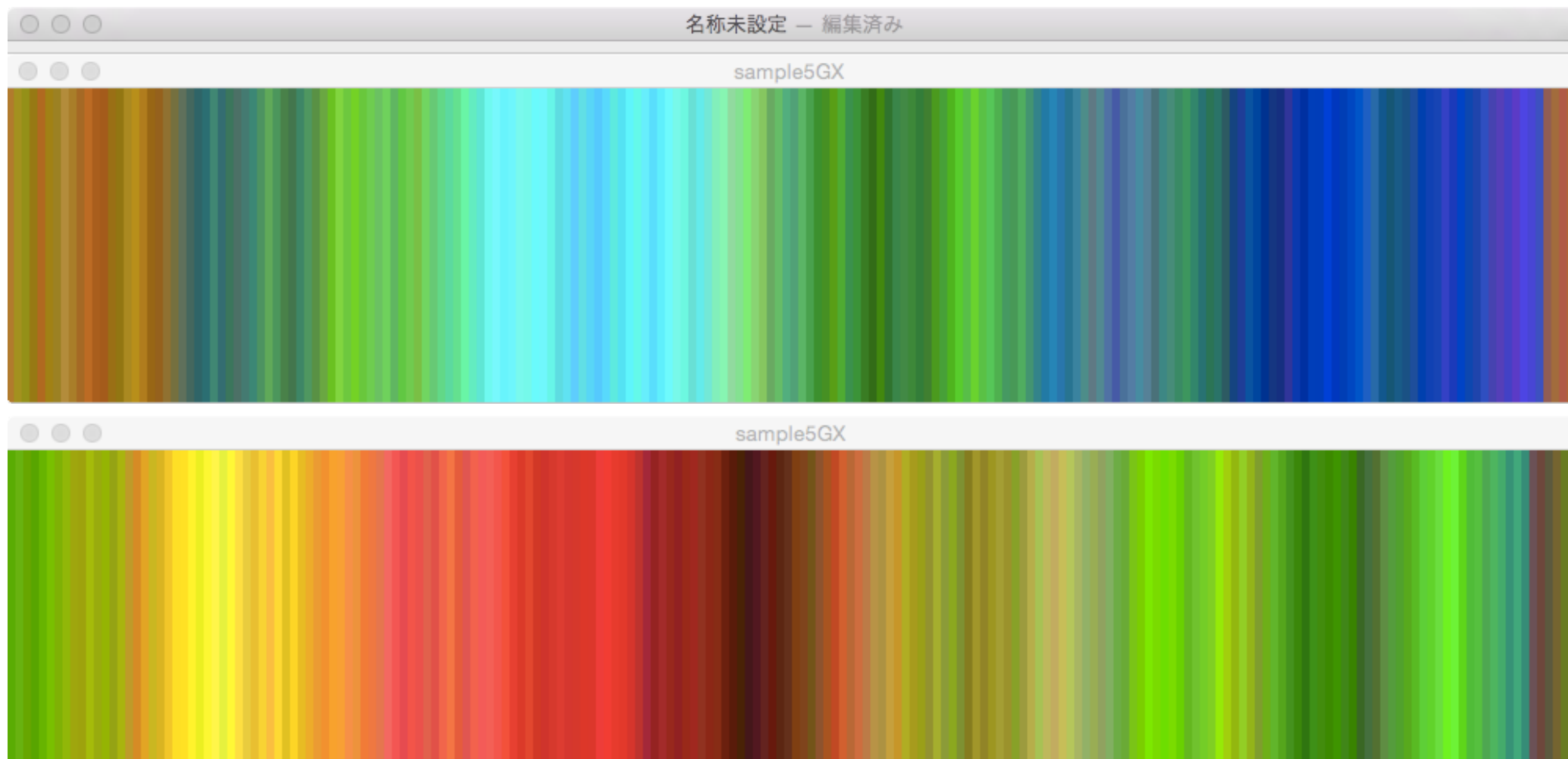
幅5x高さheightの長方形を、5pxずつ右にずらしていきます。



練習 1 (色のランダムウォーク)

sample4B_7を修正し, 四角形の色が, 徐々に変わるようなパターンに変更してください. この際, RGB値それぞれを, 0-255の区間でランダムウォークさせてください.

sample4B_7x.pde



練習 2 (色のランダムウォーク)

sample4B_6xを修正して、線の色もランダムウォークするようにしてください。

sample4B_7y.pde

