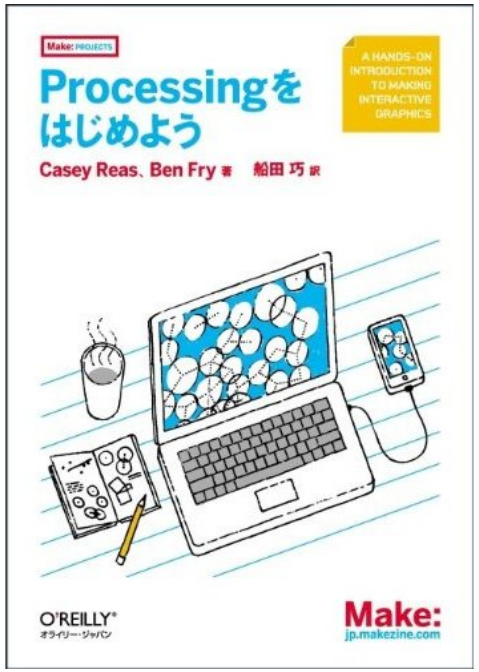
Practice #4
アニメーション（時間的な処理）

演習４BC　ランダムドローイング｜関数・三角関数



Processingをはじめよう
第二版
（Make: PROJECTS）

p.64 – 71

p.236 – 239

情報環境デザイン学科

対面授業　7.31

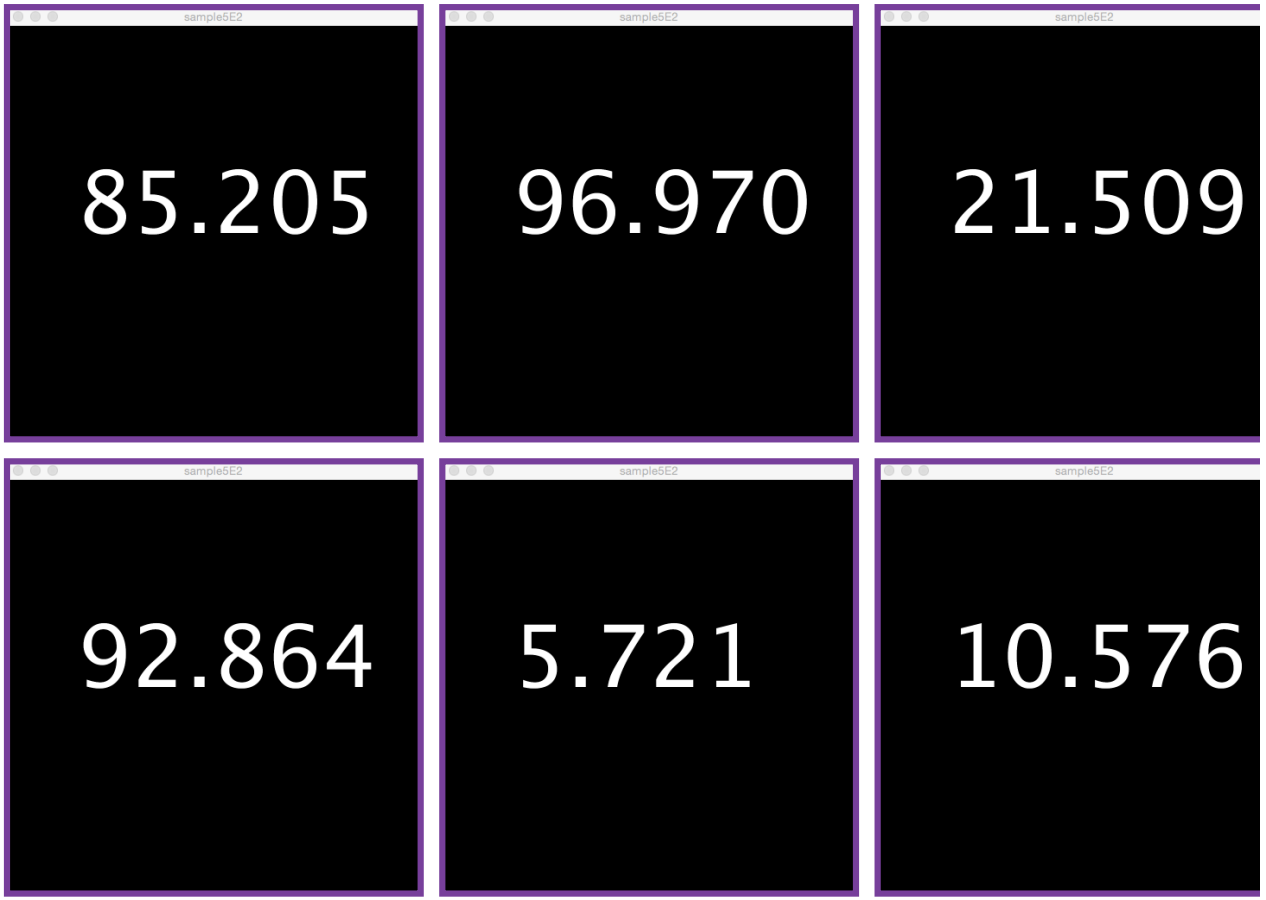
**産業イノベーション
デザイン学科**

課題学習（映像）　8.7

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 }
```

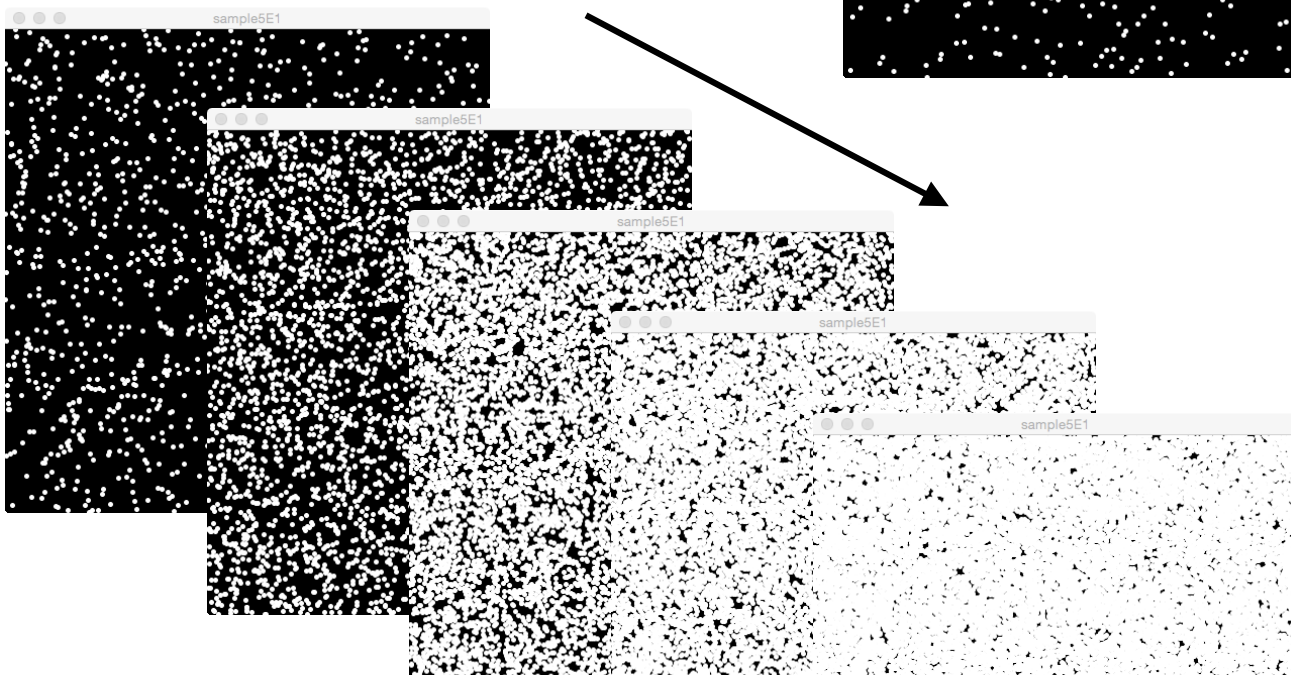
int int(f);

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
13   point(x,random(height));
14
15   x++;
16
17   if(x==width){
18     x = 0;
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

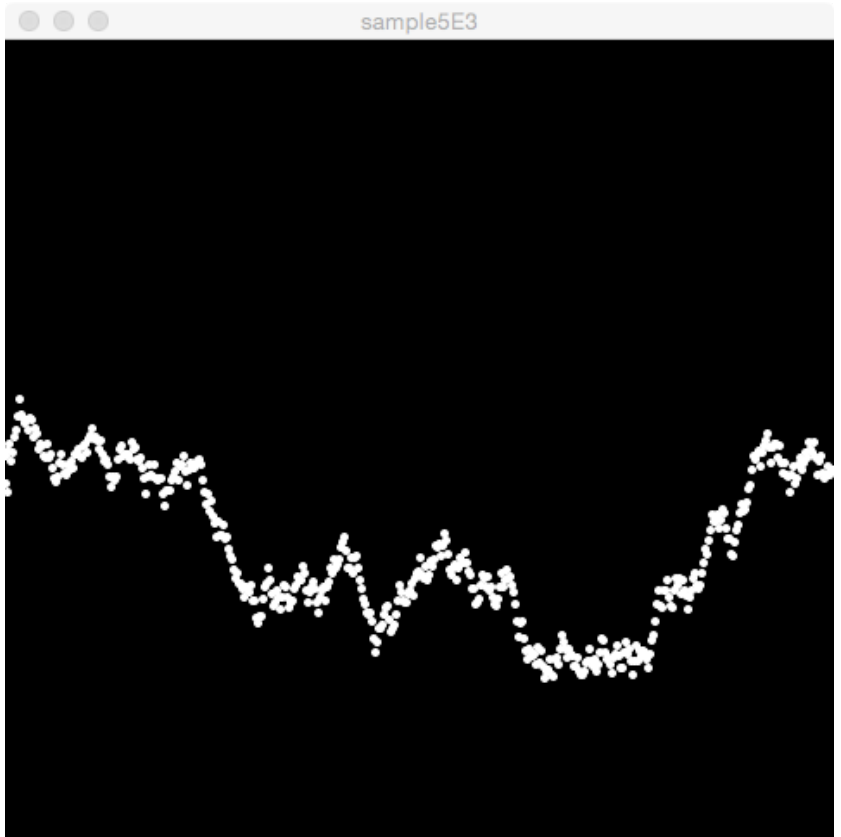
```
1  int x; float y;
2
3  void setup(){
4    size(480,480);  background(0);
5    stroke(255);  strokeWeight(5);
6    x = 0; y = 0.5 * height;
7
8    frameRate(100);
9  }
10
11 void draw(){
12   float n = 10;
13   y = y + random(2*n) - n;
14   y = constrain(y, 0, height);
15
16   point(x,y);
17   x++;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

```
-n< random(2*n) - n; <n
```

```
(y-n)< y + 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;
3
4  void setup(){
5    size(480,480);  background(0);
6    stroke(255);  strokeWeight(1);
7
8    px = 0.5 * width; py = 0.5 * height;
9    x = 0.5 * width; y = 0.5 * height;
10
11   frameRate(100);
12 }
```

(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.

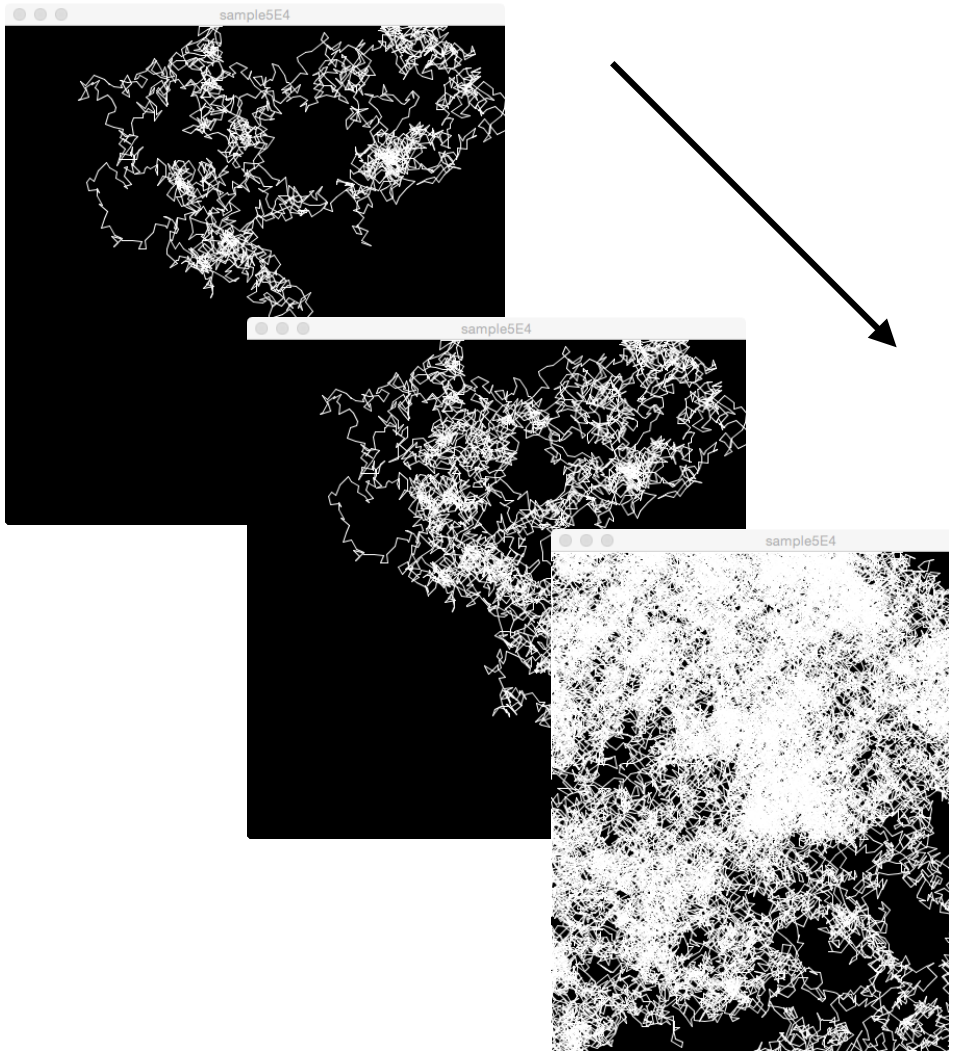
(px, py), (x, y) の初期点をウィンドウの中点とする.

```
14 void draw(){
15
16   float n = 10;
17   x = x + random(2*n) - n;
18   x = constrain(x, 0, width);
19   y = y + random(2*n) - n;
20   y = constrain(y, 0, height);
21
22   line(px,py,x,y);
23   px = x; py = y;
24
25 }
```

x座標のランダムウォーク

y座標のランダムウォーク

(px,py)と(x,y)に線を引き, (px,py)を更新する.



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 }
```

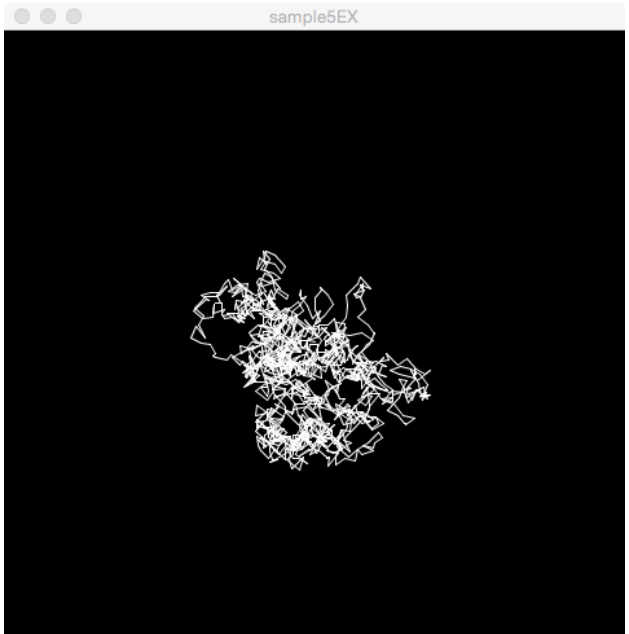
(cx, cy)はウィンドウの中点

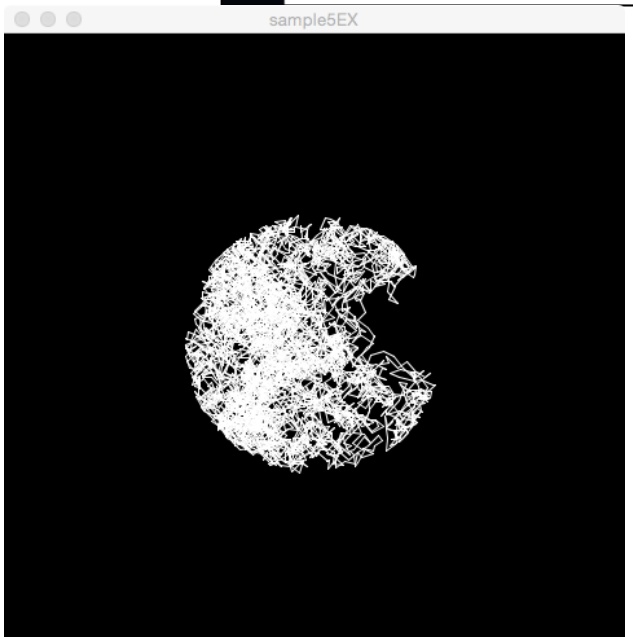
出発点が中点！！

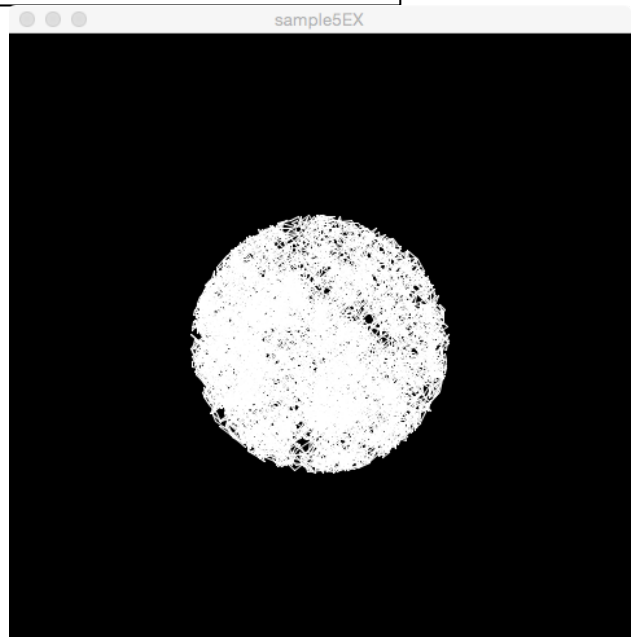
```
15 void draw(){
16
17   float n = 10;
18
19   x = x + random(2*n) - n;
20   y = y + random(2*n) - n;
21
22   if(dist(x,y,cx,cy)>100.){
23     x = px;   y = py;
24   }
25
26   line(px,py,x,y);
27   px = x; py = y;
28
29 }
```

（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

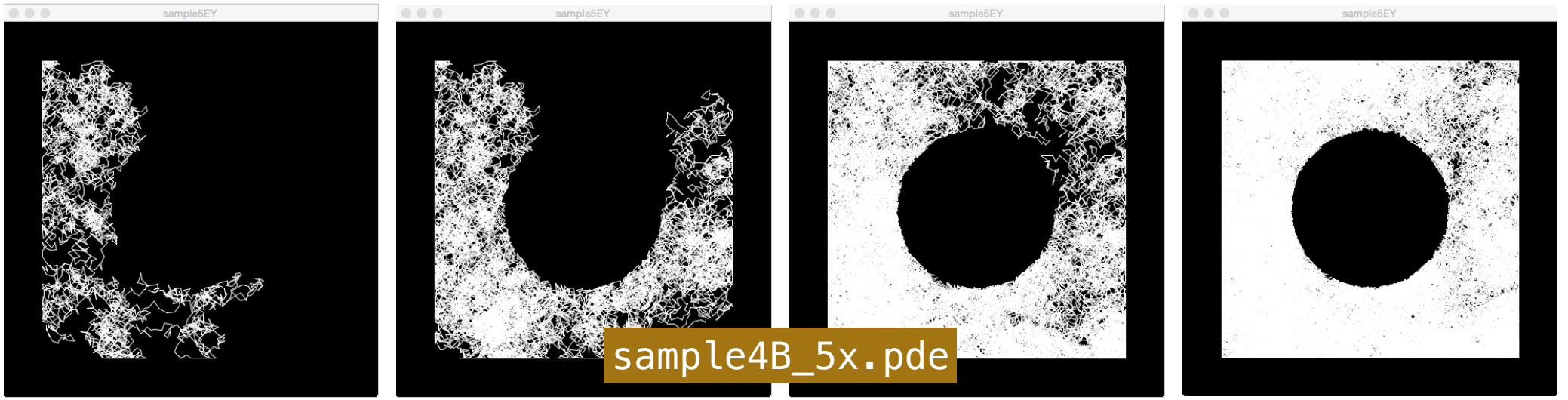


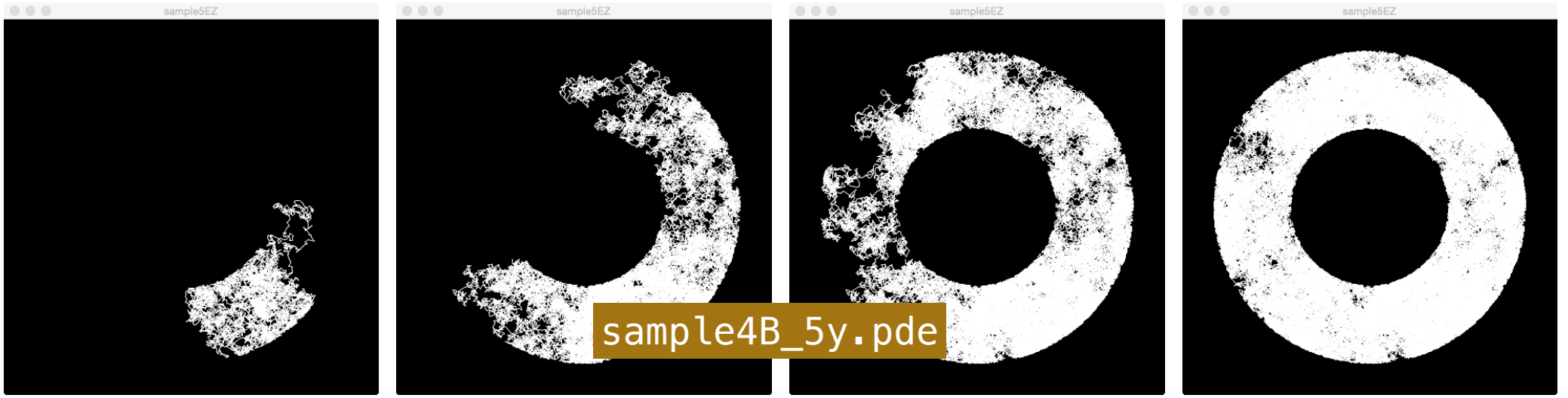




練習

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.





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 }
```

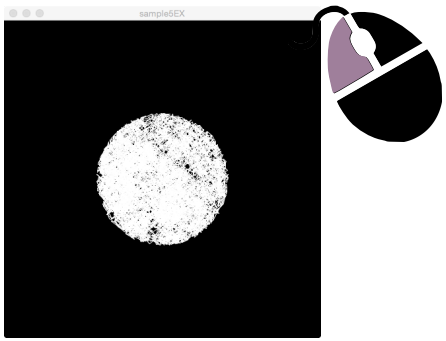
```
17 void mousePressed(){
18   for(int i=0;i<1000;i++){
19
20     float n = 10;
21     x = x + random(2*n) - n;
22     y = y + random(2*n) - n;
23
24     if(dist(x,y,cx,cy)>100.){
25       x = px;  y = py;
26     }
27     line(px,py,x,y);
28     px = x; py = y;
29
30   }
31 }
32
33 void mouseReleased(){
34   background(0);
35 }
```

ランダムウォーク

x 1000

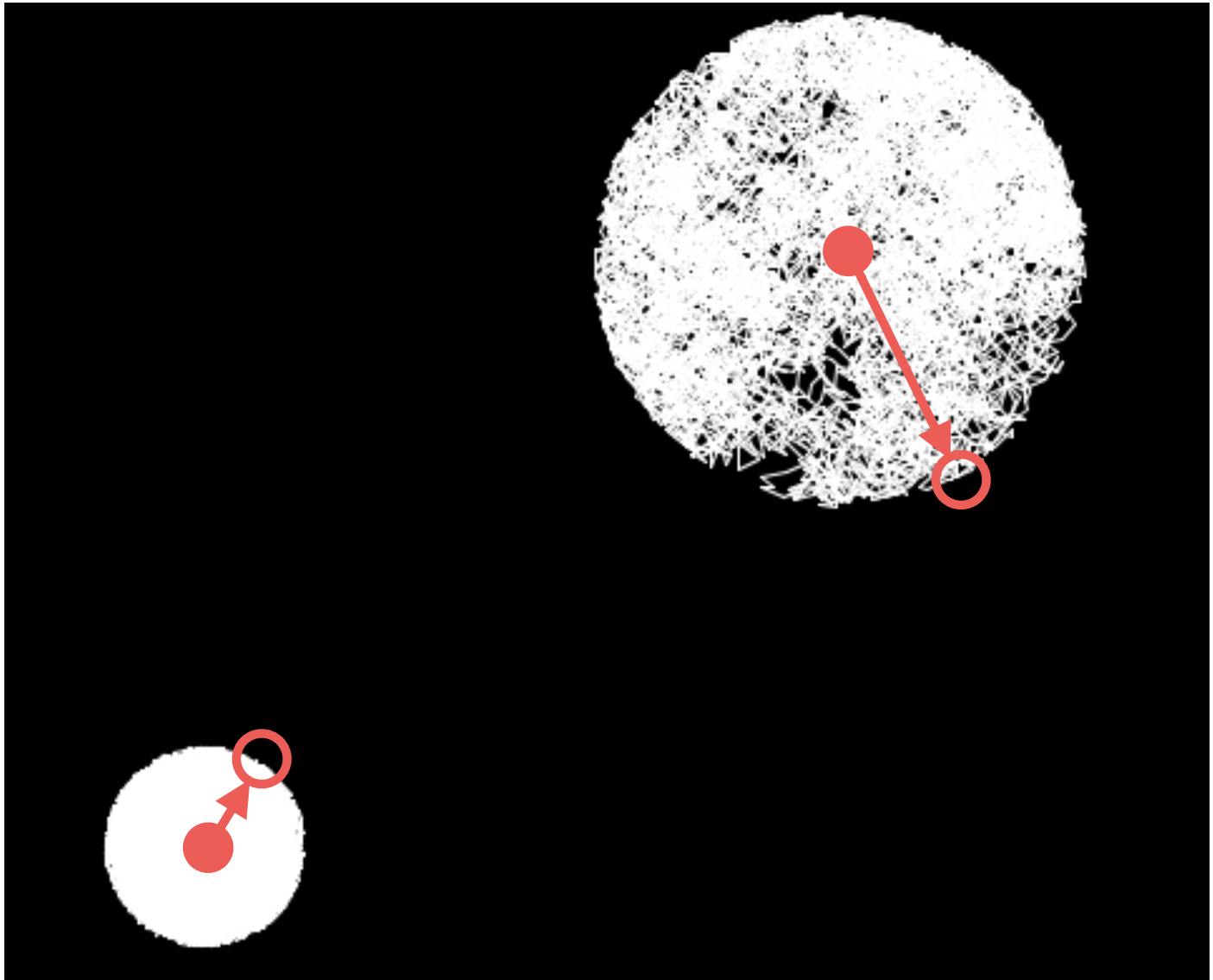
マウスを押した時の処理

マウスを離したときの処理：画面をクリアする



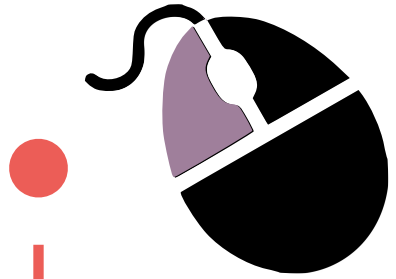
練習

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, マウスの離し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.



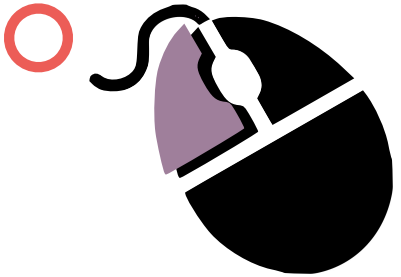
sample4_B6x.pde

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

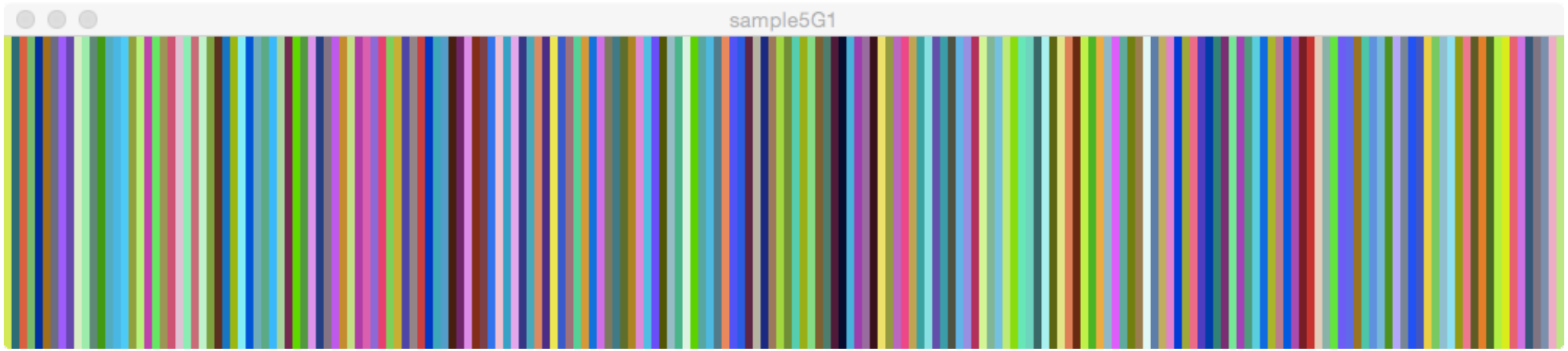
sample4B_7.pde

```
int x = 0;

void setup(){
  size(1000,200);  background(0);
  noStroke();
  frameRate(100);
}
```

```
void draw(){

  float red = random(255);
  float green = random(255);
  float blue = random(255);

  color col = color(red,green,blue);

  fill(col);
  rect(x,0,5,height);
  x = (x+5) % width;

}
```

RGB値のそれぞれをランダムに決定しています.

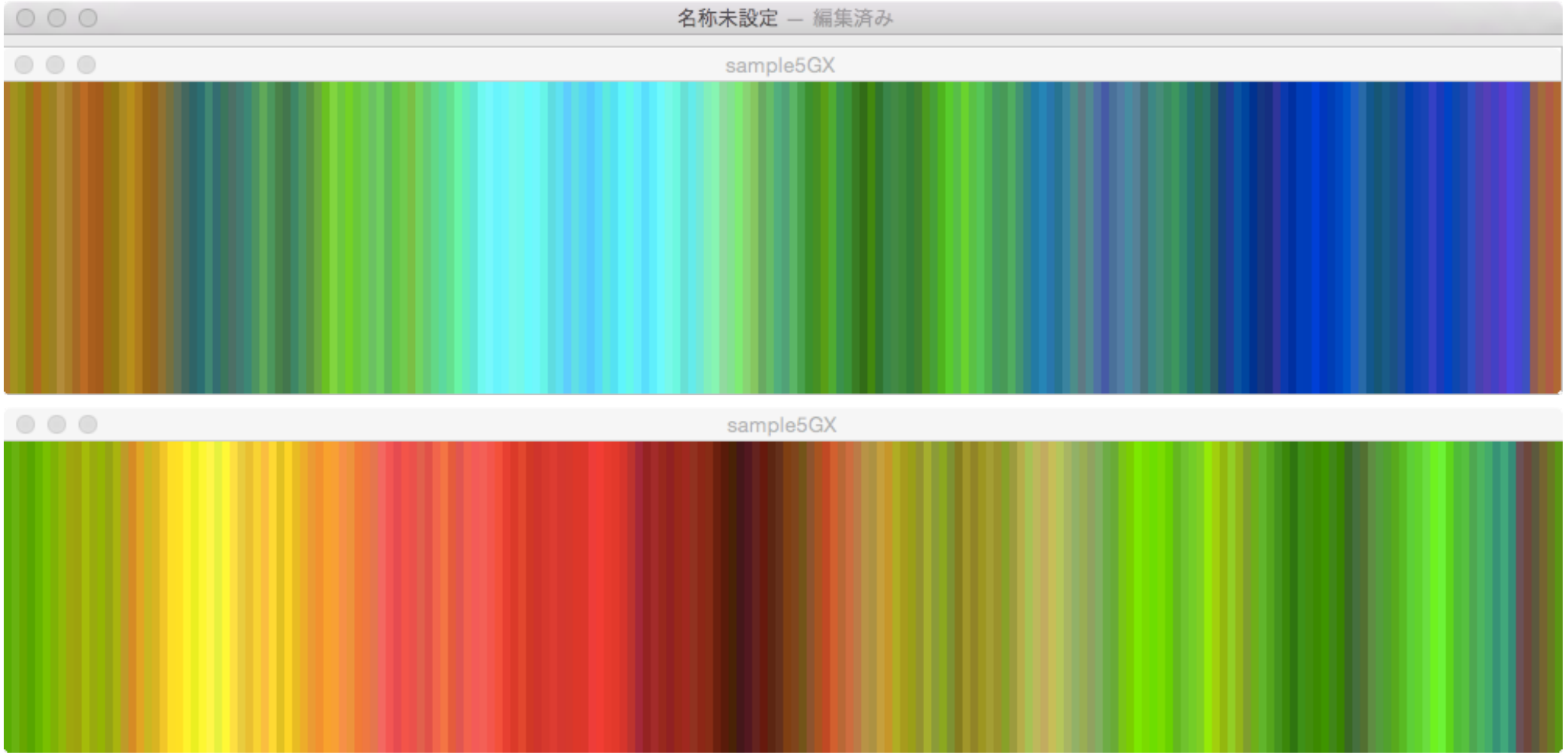
幅5x高さheightの長方形を,
5pxずつ右にずらしていきます.



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.

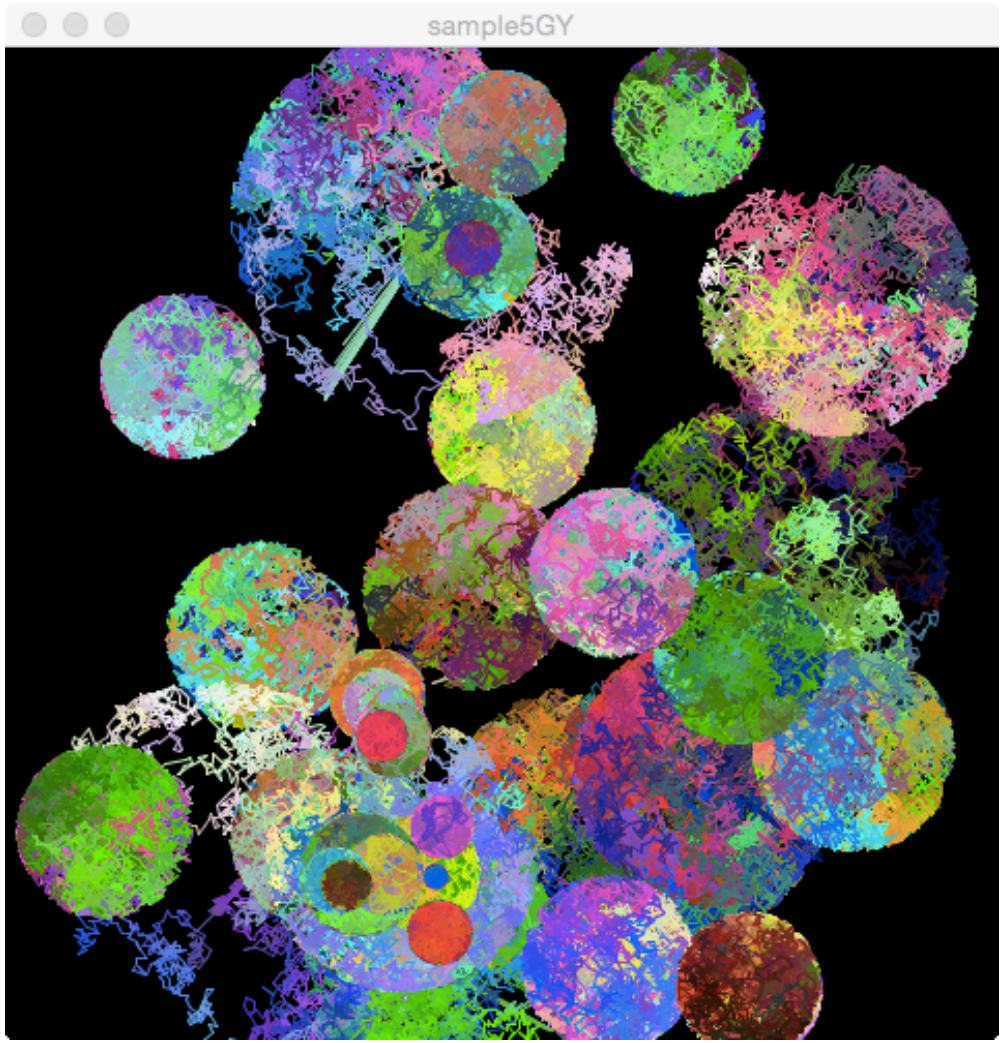
`sample4B_7x.pde`



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`

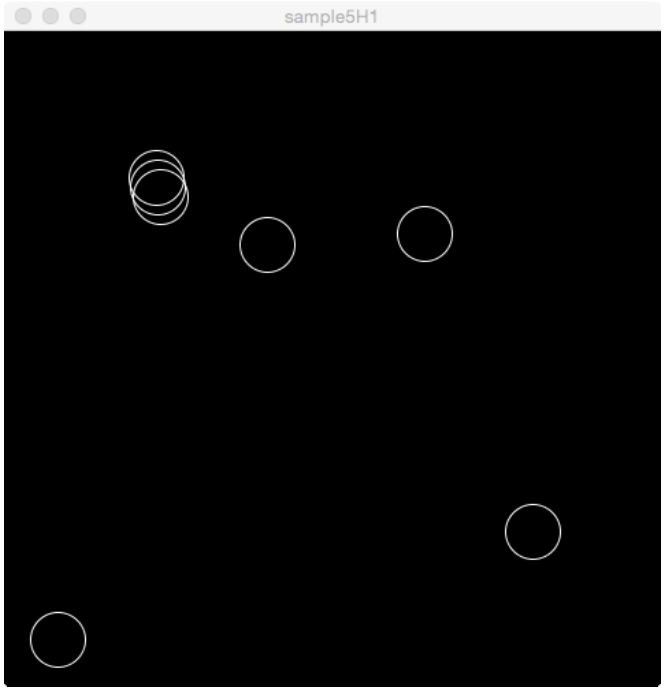


準備

クリックした位置を中心とした円を描画します.

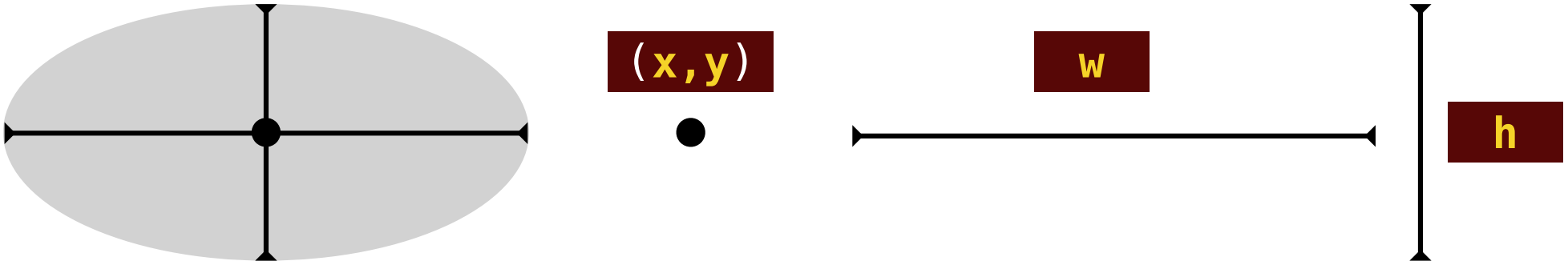
```
1
2  void setup(){
3    size(480,480);
4    background(0);
5  }
6
7  void draw(){
8  }
9
10 void mousePressed(){
11   float d = 40;
12   noFill(); stroke(255);
13   ellipse(mouseX, mouseY, d, d);
14 }
```

sample4C_1.pde



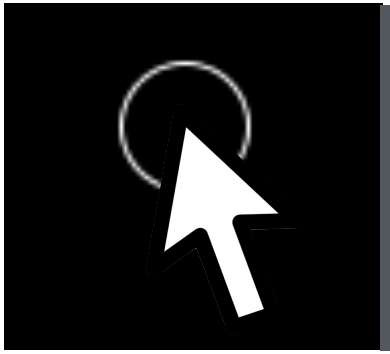
あらかじめ定義された関数

```
void ellipse(float x,float y,float w,float h);
```



（引数のない）新しい関数をつくる

一連の処理を，ellipseMouseという名前の関数にコピーします．



新しい関数の仕様

```
void ellipseMouse();
```

マウスの位置を中心に直径40pxの白い円を描く

```
void mousePressed(){
  float d = 40;
  noFill(); stroke(255);
  ellipse(mouseX, mouseY, d, d);
}
```

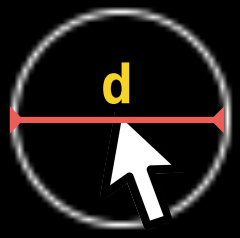
⇔

```
ellipseMouse();
```

```
void ellipseMouse(){
  float d = 40;
  noFill(); stroke(255);
  ellipse(mouseX, mouseY, d, d);
}
```

sample4C_2.pde

（引数のある）新しい関数をつくる　1



新しい関数の仕様

```
void ellipseMouse(float d);
```

マウスの位置を中心に直径（d）pxの白い円周を描く.

```
void mousePressed(){

  float d = 40;
  noFill(); stroke(255);
  ellipse(mouseX, mouseY, d, d);

}
```

```
ellipseMouse(40);
```

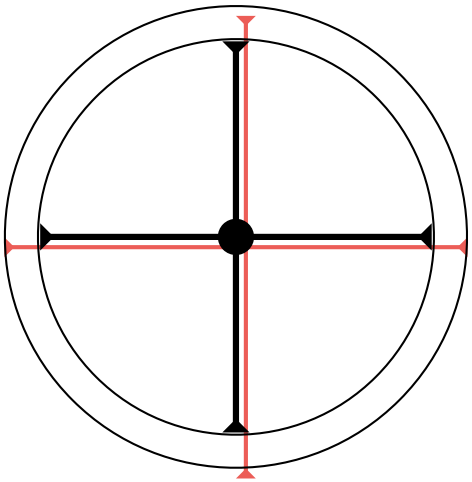
```
void ellipseMouse(float d){

  noFill(); stroke(255);
  ellipse(mouseX, mouseY, d, d);

}
```

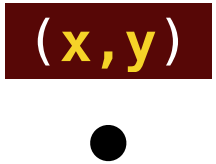
sample4C_3.pde

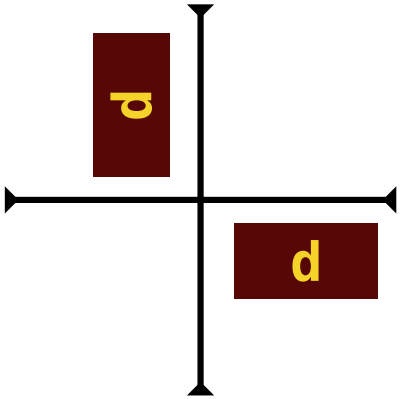
（引数のある）新しい関数をつくる　2

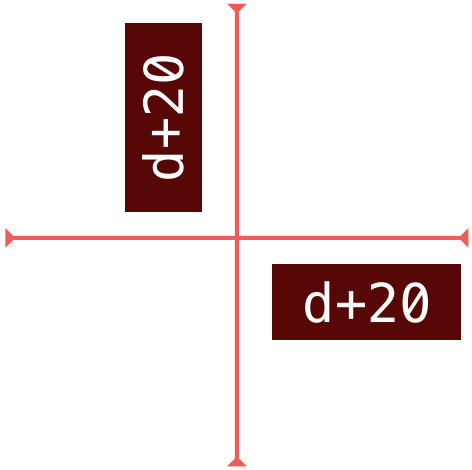
```
void dEllipse(float x,float y,float d);
```

二重丸を描画する関数









sample4C_4.pde

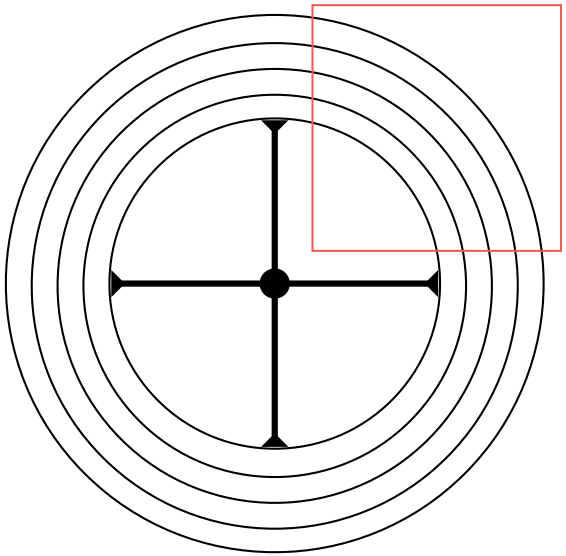
setup, drawは省略

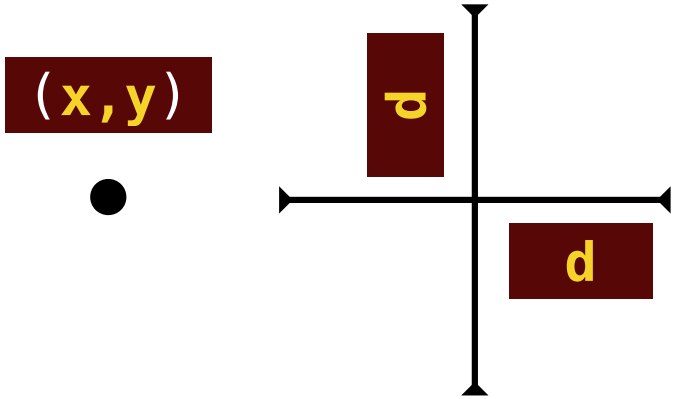
```
void mousePressed(){
  dEllipse(mouseX, mouseY, 80);
}
void mouseReleased(){
  dEllipse(mouseX, mouseY, 10);
}


void dEllipse(float x, float y, float d){
  noFill(); stroke(255);
  ellipse(x, y, d, d);
  ellipse(x, y, d+20, d+20);
}
```

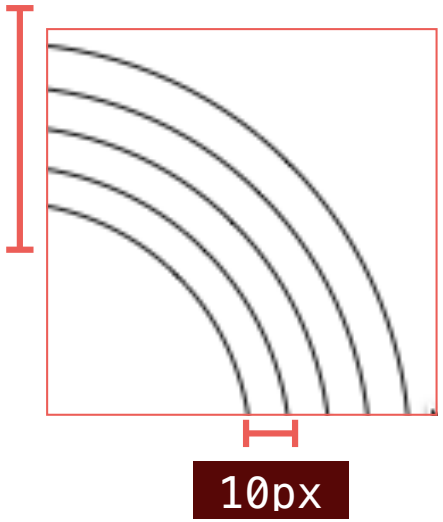
（引数のある）新しい関数をつくる　3

```
void mEllipse(float x,float y,float d,int n);
```

n 重丸を描画する関数








```
void mousePressed(){
  mEllipse(mouseX, mouseY, 60,6);
}
void mouseReleased(){
  mEllipse(mouseX, mouseY, 10,3);
}

void mEllipse(float x, float y, float d, int n){

  noFill(); stroke(255);

  for(int i=0;i<n;i++){
    float d2 = d + 10*i;
    ellipse(x,y,d2,d2);
  }
}
```

setup, drawは省略

sample4C_5.pde

ランダムウォーク・スタンプ

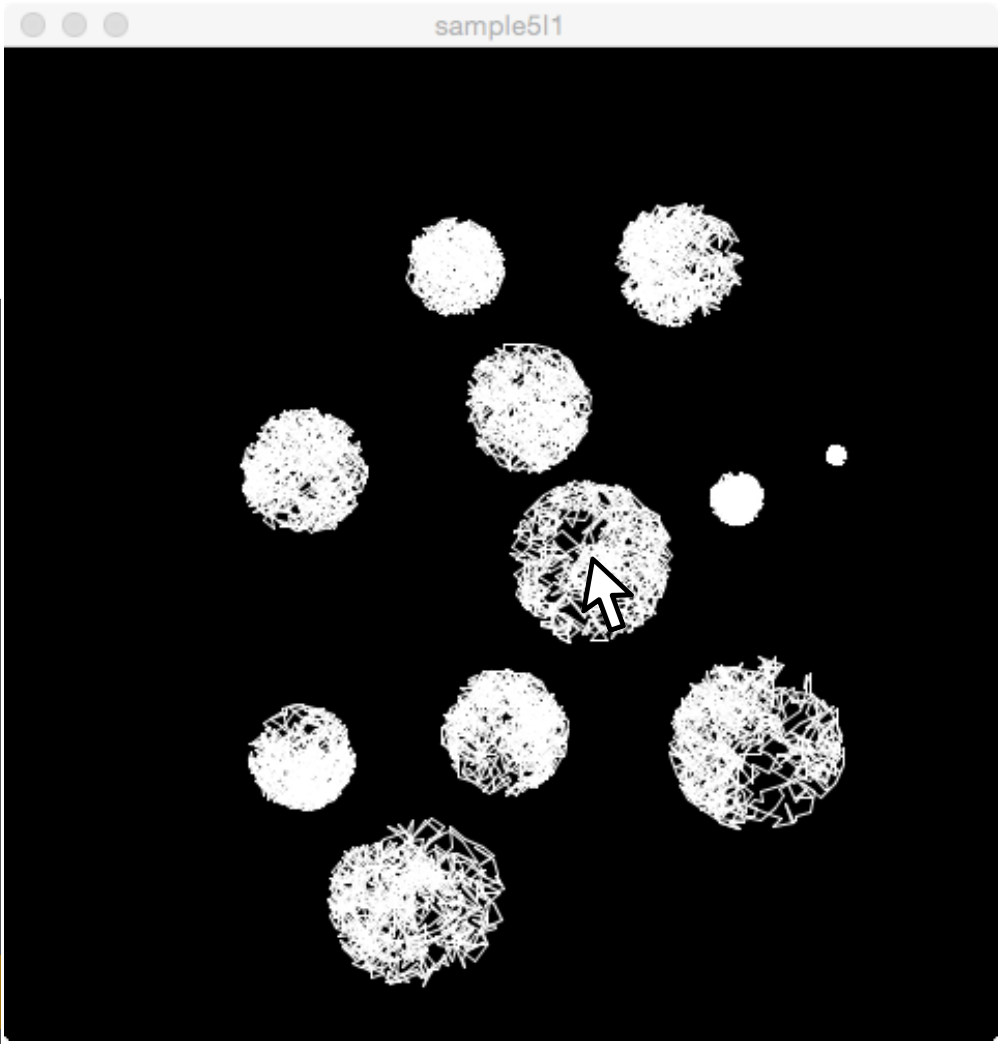
```
void setup(){
  size(480,480);  background(0);
}

void draw(){
}

void mousePressed(){
  stroke(255);
  int r = int(random(50));
  myEllipse(mouseX, mouseY, 1+r);
}
```

```
void myEllipse(float cx,float cy,int r){

  float px = cx;  float py = cy;
  for(int i=0;i<1000;i++){

    float n = 10;
    float x = px + random(2*n) - n;
    float y = py + random(2*n) - n;

    if(dist(x,y,cx,cy)>r){
      x = px;  y = py;
    }
    line(px,py,x,y);
    px = x; py = y;
  }
}
```

sample4C_6.pde

```
void myEllipse(float cx,float cy,int r);
```

（cx, cy）を中心とした半径（r）の円の内部に，（最大）1000のランダムウォークの線を描く．

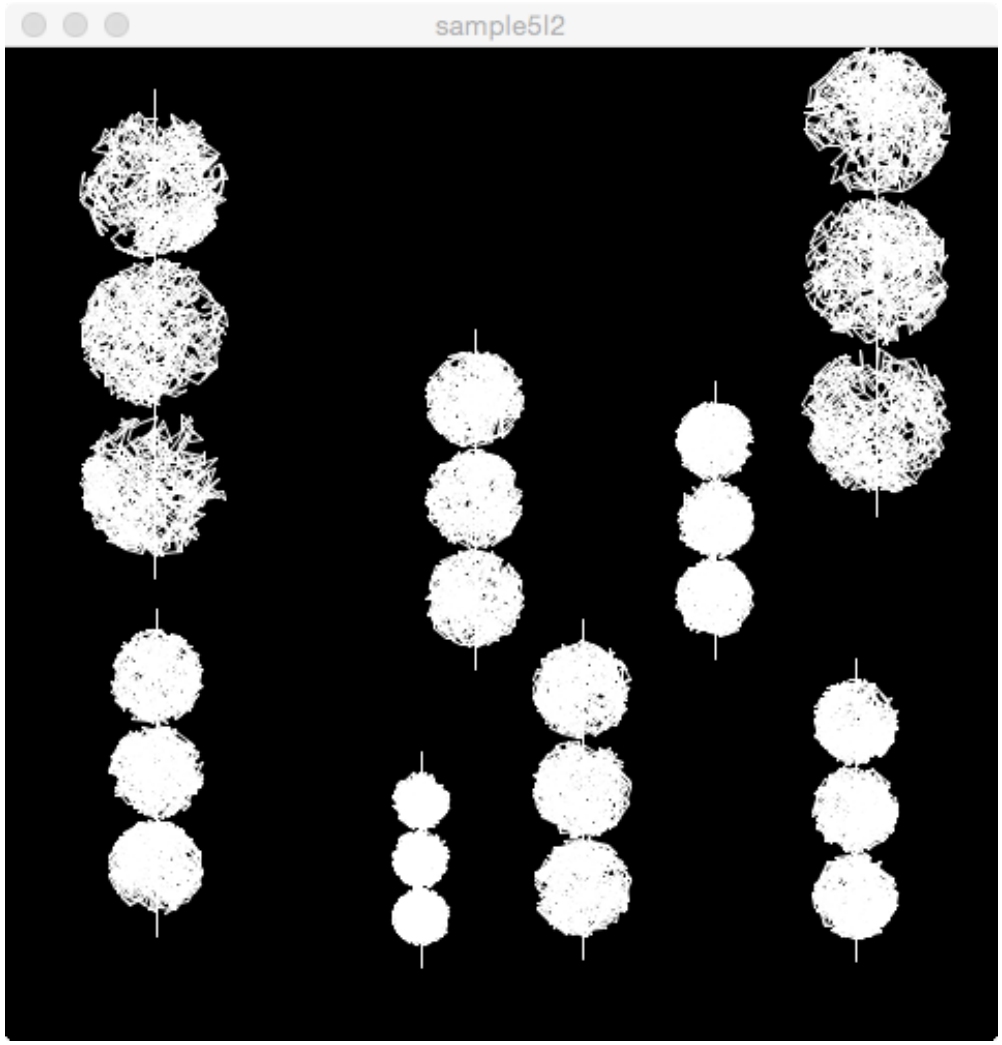


ランダムウォーク・スタンプ

myEllipse(cx,cy,r) を再利用して, (cx,cy)を真ん中の団子の中点として, 半径 r の団子が縦に3つ並ぶ myEllipse2(cx,cy,r)を作成してください.

sample5I2

```
void setup(){
  size(480,480);  background(0);
}

void draw(){
}

void mousePressed(){
  stroke(255);
  int r = int(random(30));
  myEllipse2(mouseX, mouseY, 10+r);
}

void myEllipse2(float cx,float cy,int r){

  line(cx,cy-3*r-10,cx,cy+3*r+10);
  myEllipse(cx,cy,r);
  myEllipse(cx,cy-2*r,r);
  myEllipse(cx,cy+2*r,r);

}
```

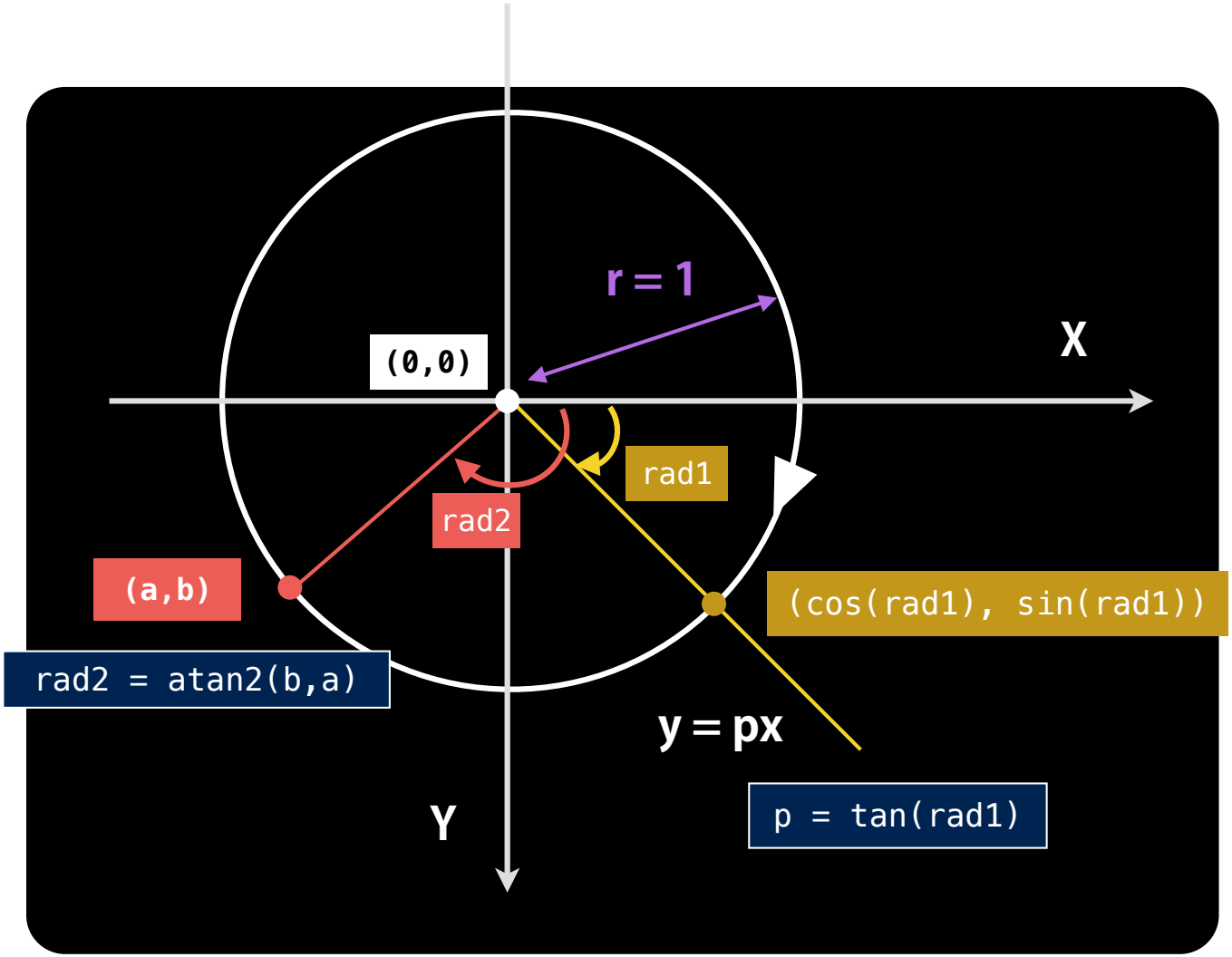


sample4_C6x.pde

myEllipseは省略

よく使う三角関数

ウィンドウ環境では, Y軸正方向が下方向になります.
このため, 偏角も時計回りが正方向となることに注意してください.



PI

3.14152156…

float sin(rad);

正弦関数

float cos(rad);

余弦関数

float tan(rad);

正接関数

float atan2(b,a);

座標（a, b）の角度（偏角）を返す

float degrees(rad);

角度の単位をラジアンから度に変換

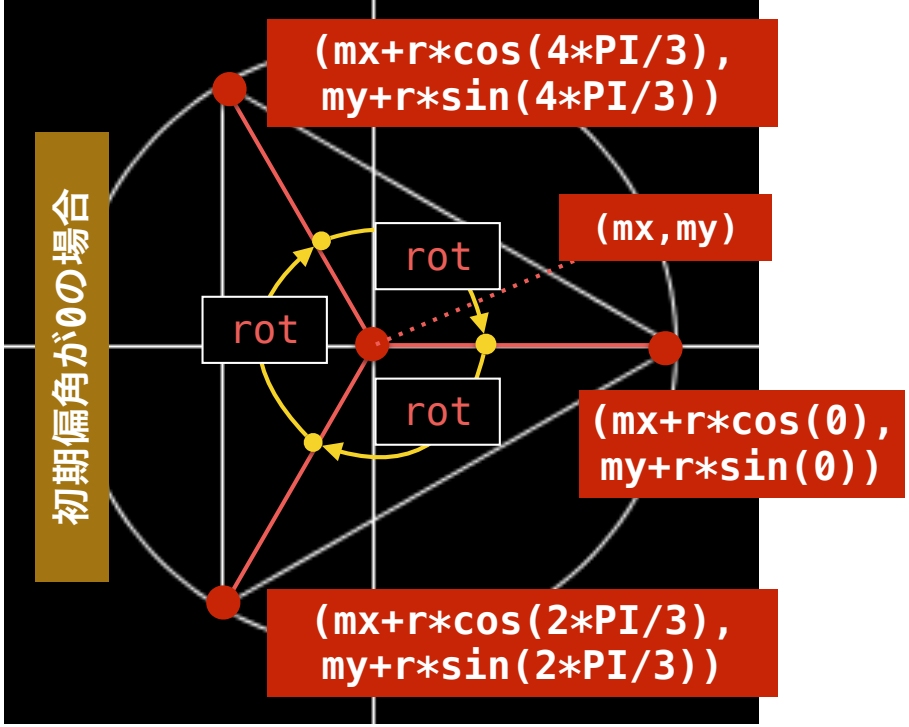
float radians(deg);

角度の単位を度からラジアンに変換

p.231

cos と sin を使った描画

偏角を120度（2PI / 3）ずつ回して，
「半径 r の三角形」を描画します．




```
void setup(){
  size(480,480);  background(0);
  noFill();  stroke(255);
}
void draw(){
}
```

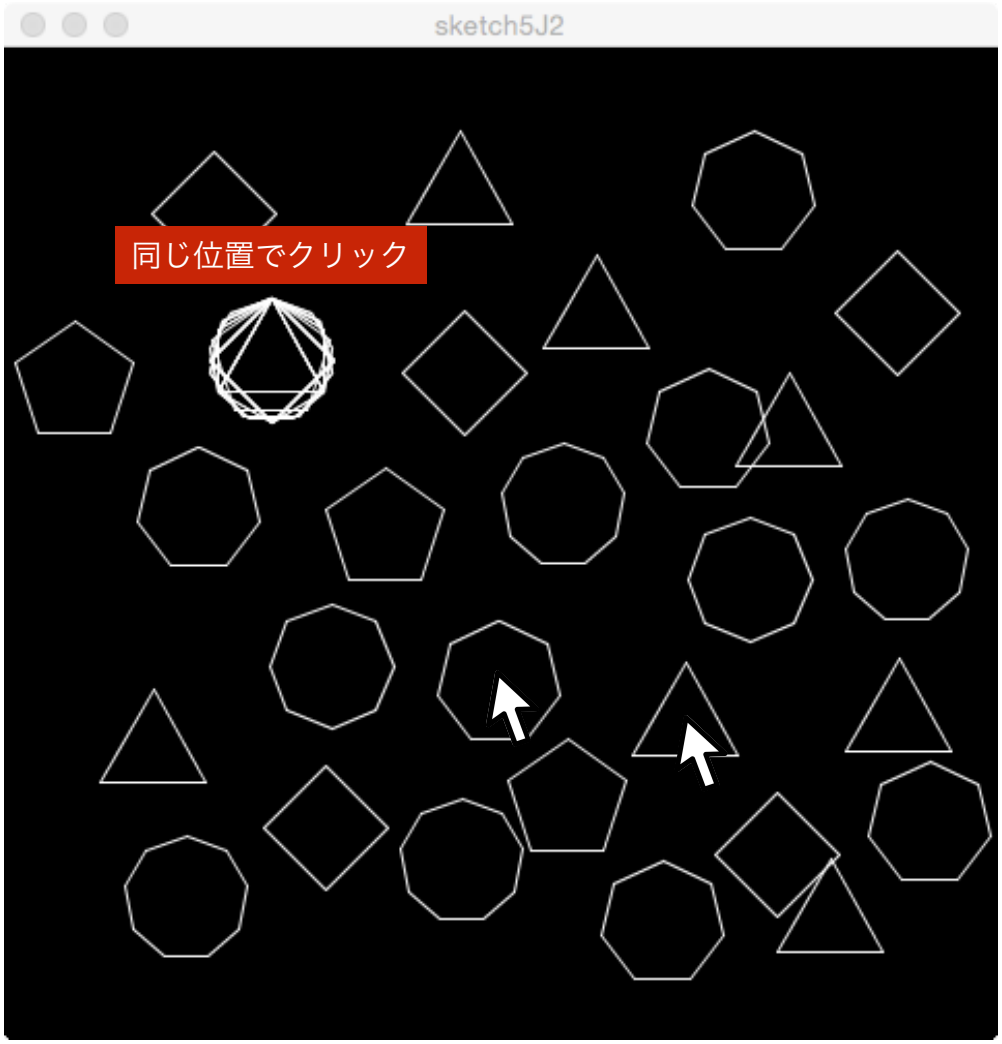
```
void mousePressed(){
  r : 半径, irad: 初期偏角, rot : 回転角
  float irad = random(2*PI);//初期偏角
  float r = 50;                //半径
  float rot = 2*PI / 3; //回転角

  float rad = irad;  //現在の偏角

  for(int i=0;i<3;i++){
    float x1 = mouseX + r * cos(rad);
    float y1 = mouseY + r * sin(rad);
    float x2 = mouseX + r * cos(rad + rot);
    float y2 = mouseY + r * sin(rad + rot);

    line(x1,y1,x2,y2);
    rad += rot; //偏角を回転
  }
}
```

sample4C_7.pde

setup, drawは省略

練習１：cos と sin を使った描画

以下のコードを参考にして, クリックした位置を中心として, 半径30の正 n 角形（n = 3, 4, 5, 6, 7）を描画するようにしてください.
それぞれの n角形は, 左右対称となっていることに注意してください.



実行結果

```
void mousePressed(){

  int n = 3 + int(random(7));
  drawRegularshape(30,n);

}
```

```
void drawRegularshape(float r, int n){

  float irad = -0.5 * PI;    //初期偏角
  float rot = 2*PI/n;        //回転角

  float rad = irad;  //現在の偏角

  for(int i=0;i<n;i++){
    float x1 = mouseX + r * cos(rad);
    float y1 = mouseY + r * sin(rad);
    float x2 = mouseX + r * cos(rad + rot);
    float y2 = mouseY + r * sin(rad + rot);
    line(x1,y1,x2,y2);
    rad += rot; //偏角を回転
  }
}
```

sample4C_8.pde

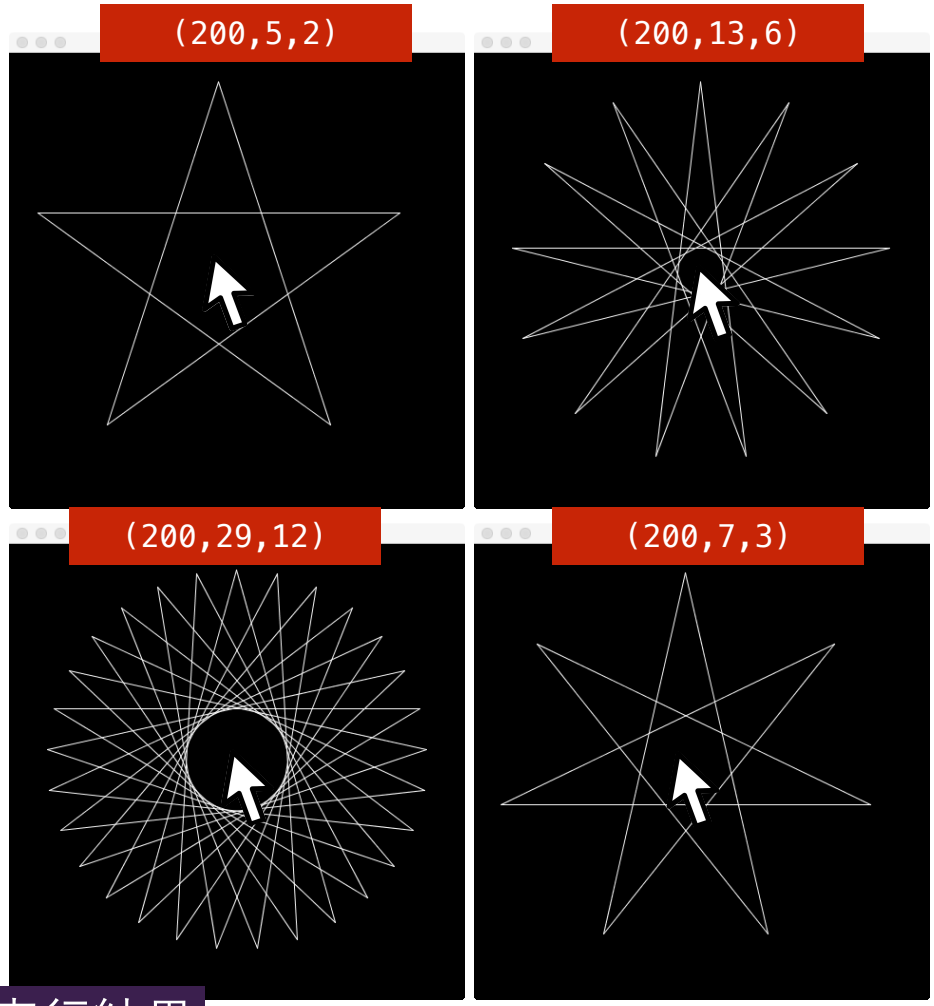
setup, drawは省略

練習２：cos と sin を使った描画

正 n 角形は, ペン先が, n 度の回転で １周（360度）することで完成する.

正 n-m 角形を, ペン先が, n 度の回転で m 周（m*360度）することで完成する図形と定義する. 例えば, 星型は, 正5-2 角形である.

以下を参考に, 正 n-m 角形を描画するdrawRegularshape2を作成せよ.



実行結果

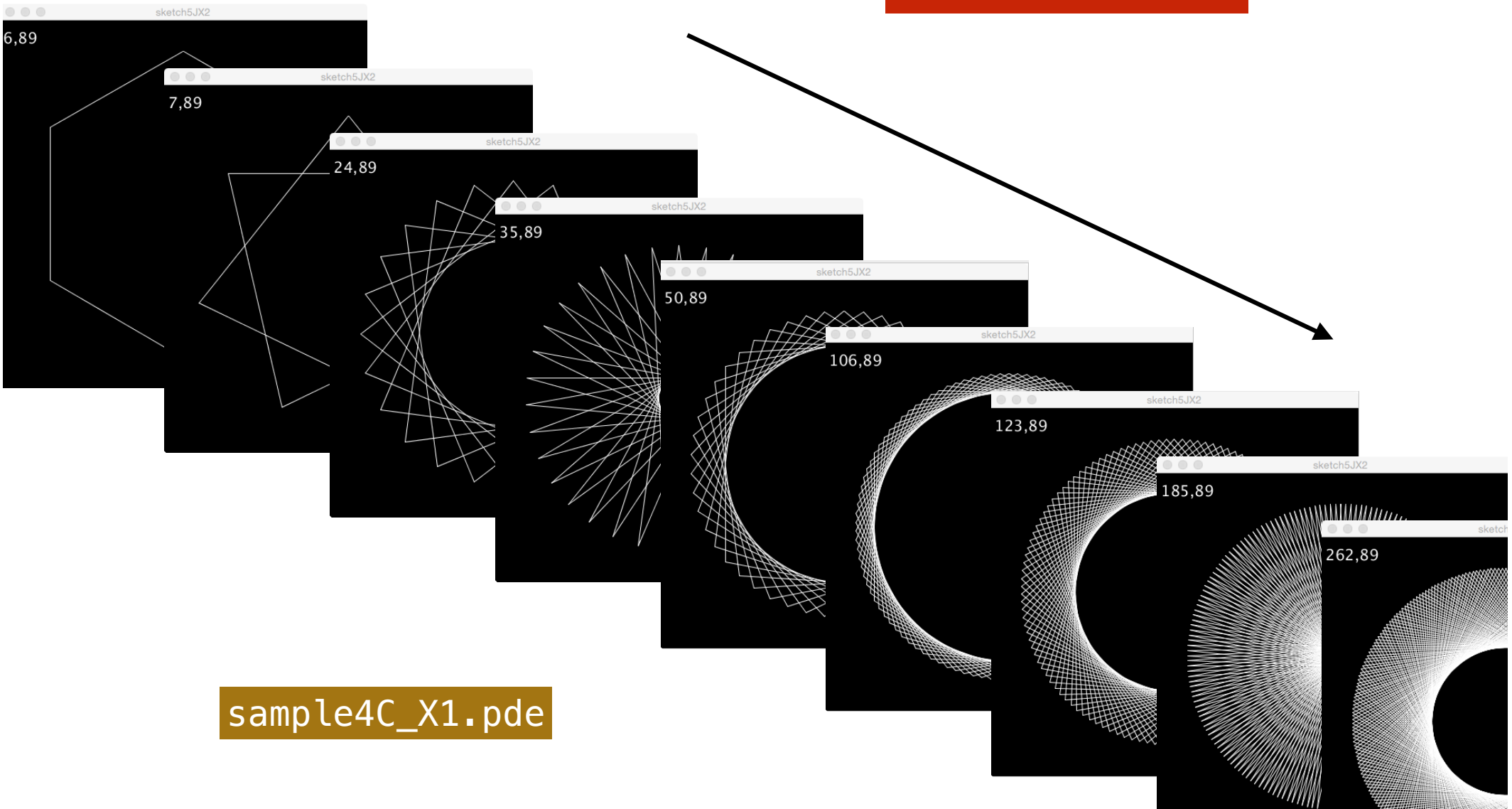
```
void mousePressed(){

  drawRegularshape2(200,5,2);

}

void drawRegularshape2(float r, int n, int m){

  float irad = -0.5 * PI;    //初期偏角
  float rot = m * 2*PI / n;          //回転角

  float rad = irad;  //現在の偏角

  for(int i=0;i<n;i++){
    float x1 = mouseX + r * cos(rad);
    float y1 = mouseY + r * sin(rad);
    float x2 = mouseX + r * cos(rad + rot);
    float y2 = mouseY + r * sin(rad + rot);
    line(x1,y1,x2,y2);
    rad += rot; //偏角を回転
  }
}
```

sample4C_9.pde

練習３：cos と sin を使った描画

「正 n-89角形」をn=1から順に増やして描画していくアニメーションを作成してください. フレームレートは1秒間に10フレームくらいに設定しましょう.

```
frameRate(10);
```



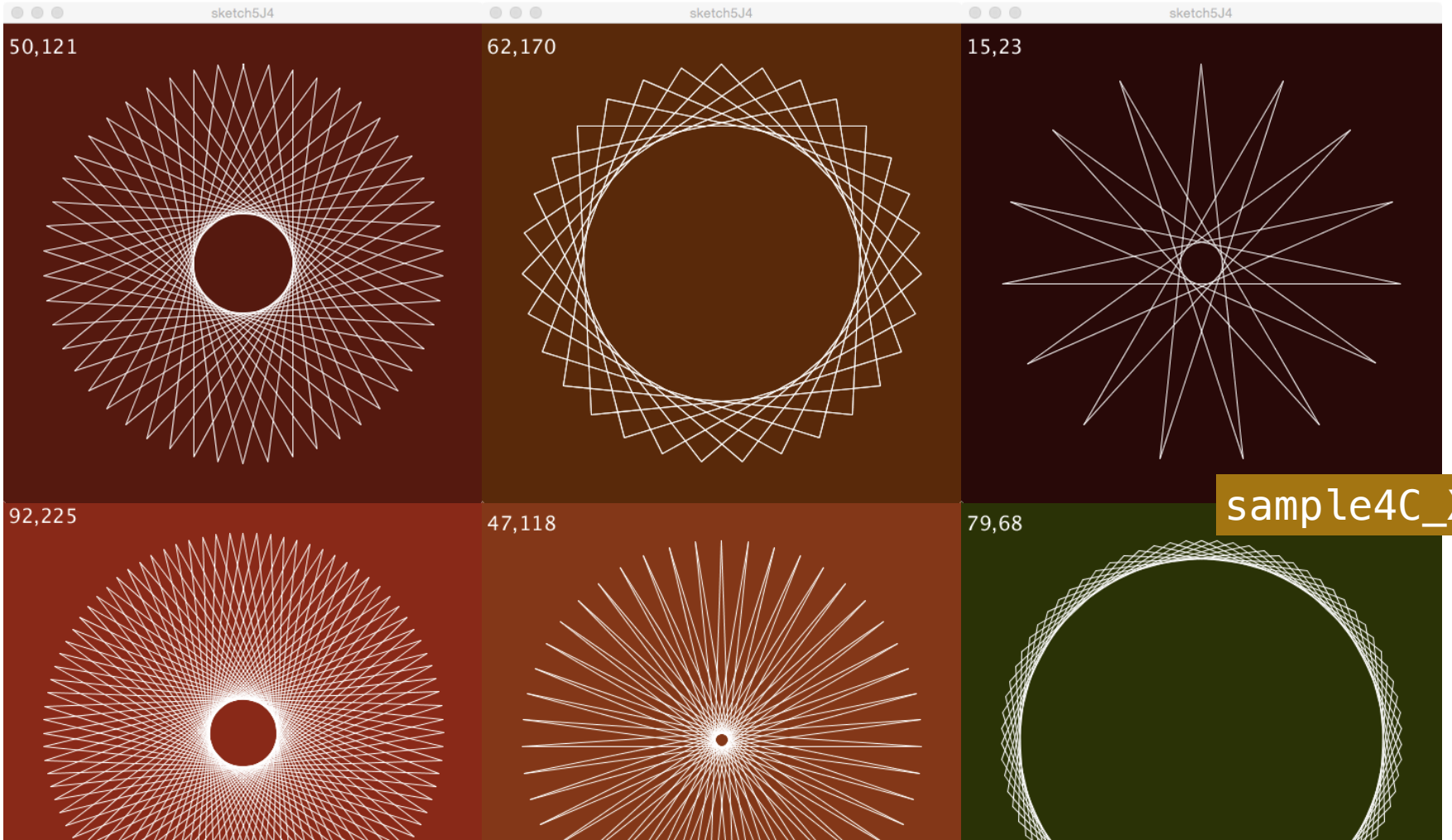
sample4C_X1.pde

練習３：cos と sin を使った描画

マウスの位置によって, nとmを変化させ, 様々な「正 n-m角形」を連続的に描画するプログラムを完成させてください.

mouseX	mouseY	マウスのX座標とY座標
pmouseX	pmouseY	1フレーム前のマウスのX座標とY座標

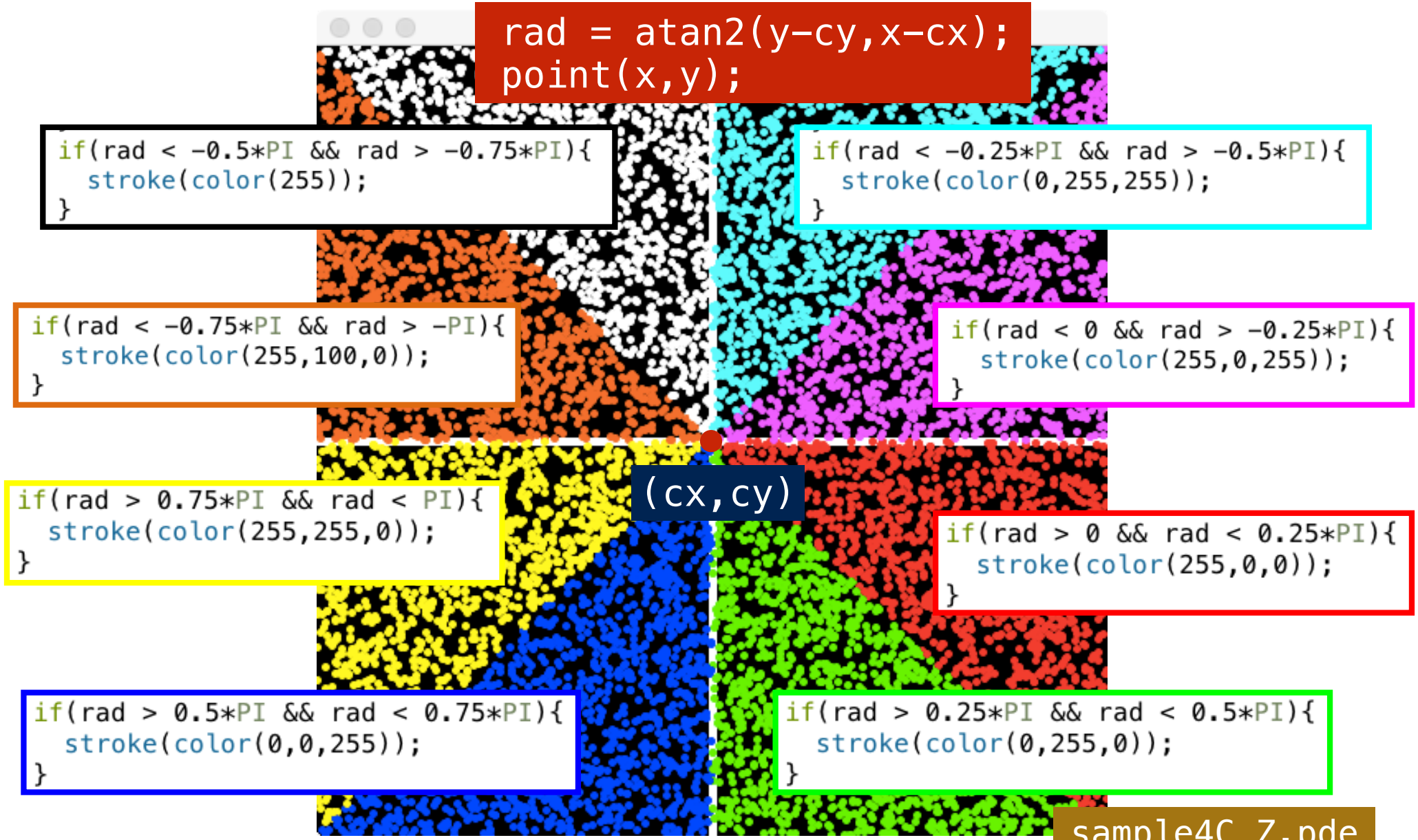
マウスが動いた時のみ, 処理を行うのがスマートです.



sample4C_X2.pde

練習４：cos と sin を使った描画

以下のように，ウィンドウ全体を8つの区画に分割し，異なる色で塗り分けてみてください．



sample4C_Z.pde