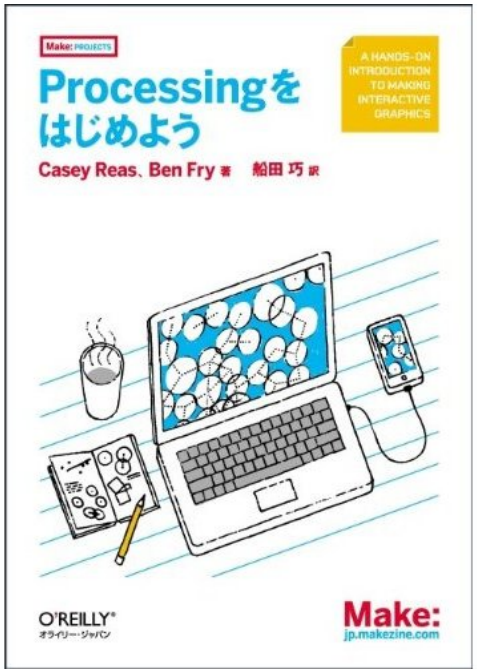
Practice #4

アニメーション（時間的な処理）

演習４C　関数定義・三角関数



Processingをはじめよう
第二版
(Make: PROJECTS)

オンデマンド授業　6.28

p.64 – 71

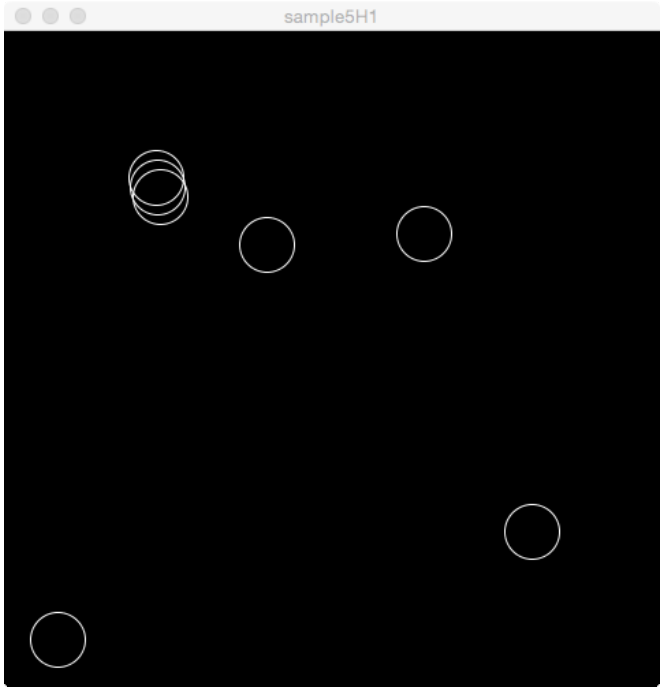
p.236 – 239

準備

クリックした位置を中心とした円を描画します.

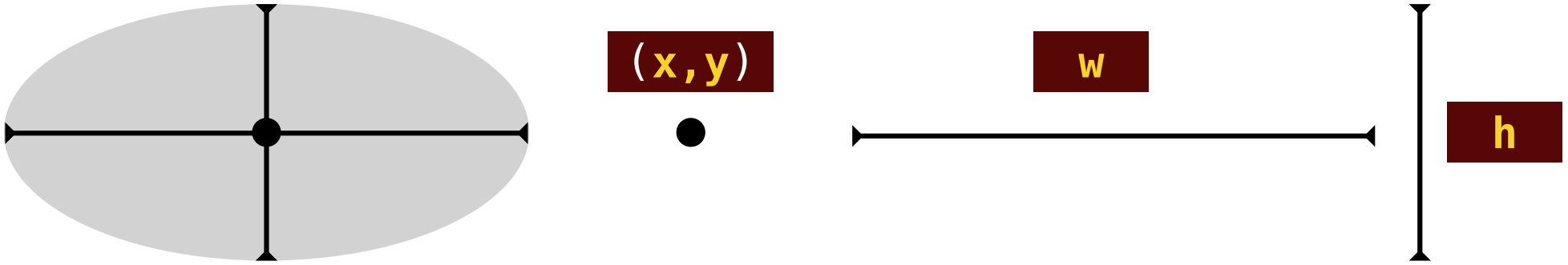
```
1
2  void setup(){
3    size(480,480);
4    background(0);
5  }
6
7  void draw(){
8  }
9
10 void mousePressed(){
11   float d = 40;
12   noFill(); stroke(255);
13   ellipse(mouseX, mouseY, d, d);
14 }
```

sample4C_1.pde



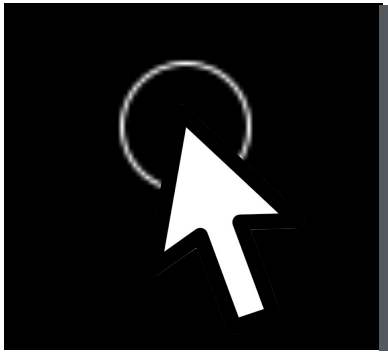
あらかじめ定義された関数

```
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);
}
```

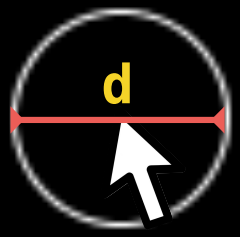
⇔

```
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);
```

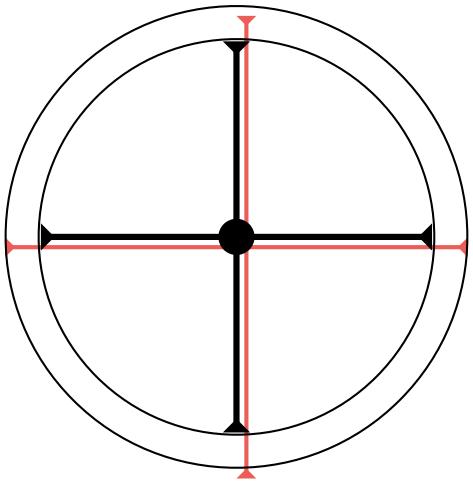
```
void ellipseMouse(float d){

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

}
```

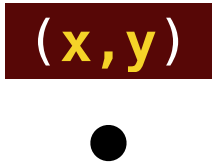
sample4C_3.pde

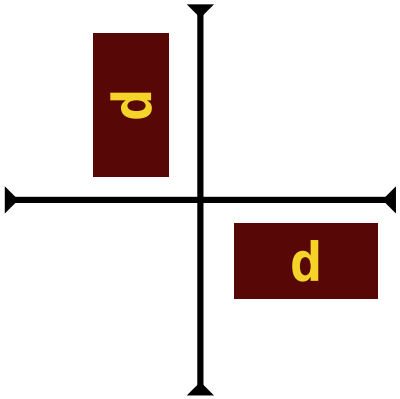
（引数のある）新しい関数をつくる　2

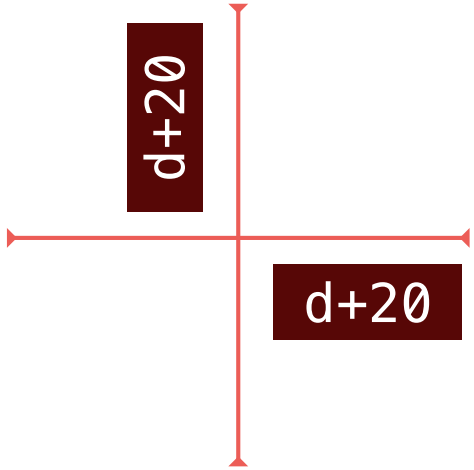
```
void dEllipse(float x,float y,float d);
```

二重丸を描画する関数









sample4C_4.pde

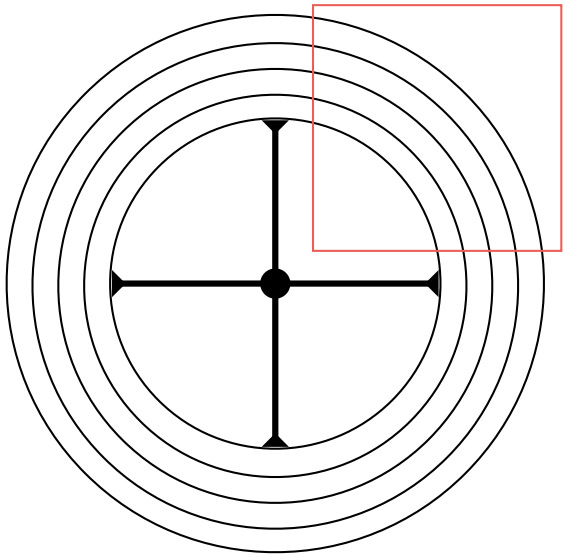
setup, drawは省略

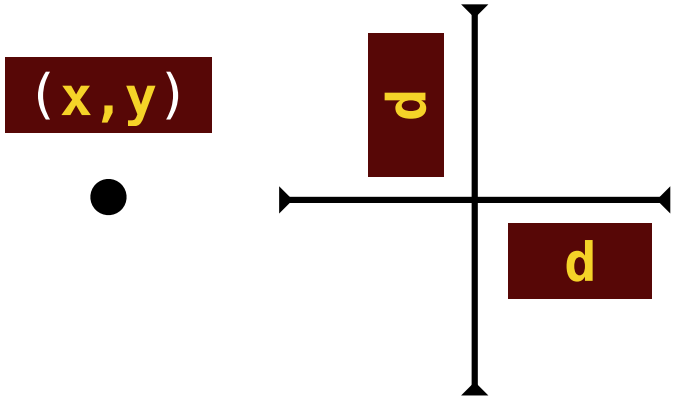
```
void mousePressed(){
  dEllipse(mouseX, mouseY, 80);
}
void mouseReleased(){
  dEllipse(mouseX, mouseY, 10);
}


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

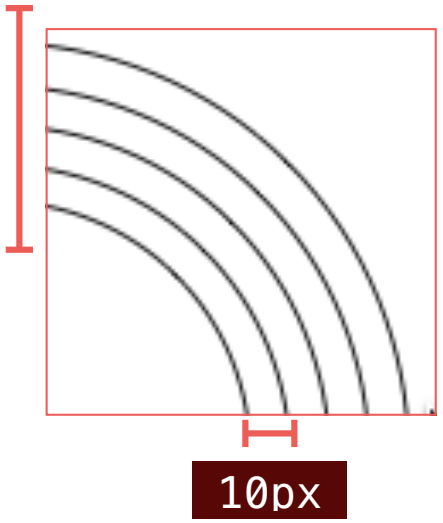
（引数のある）新しい関数をつくる　3

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

n 重丸を描画する関








```
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

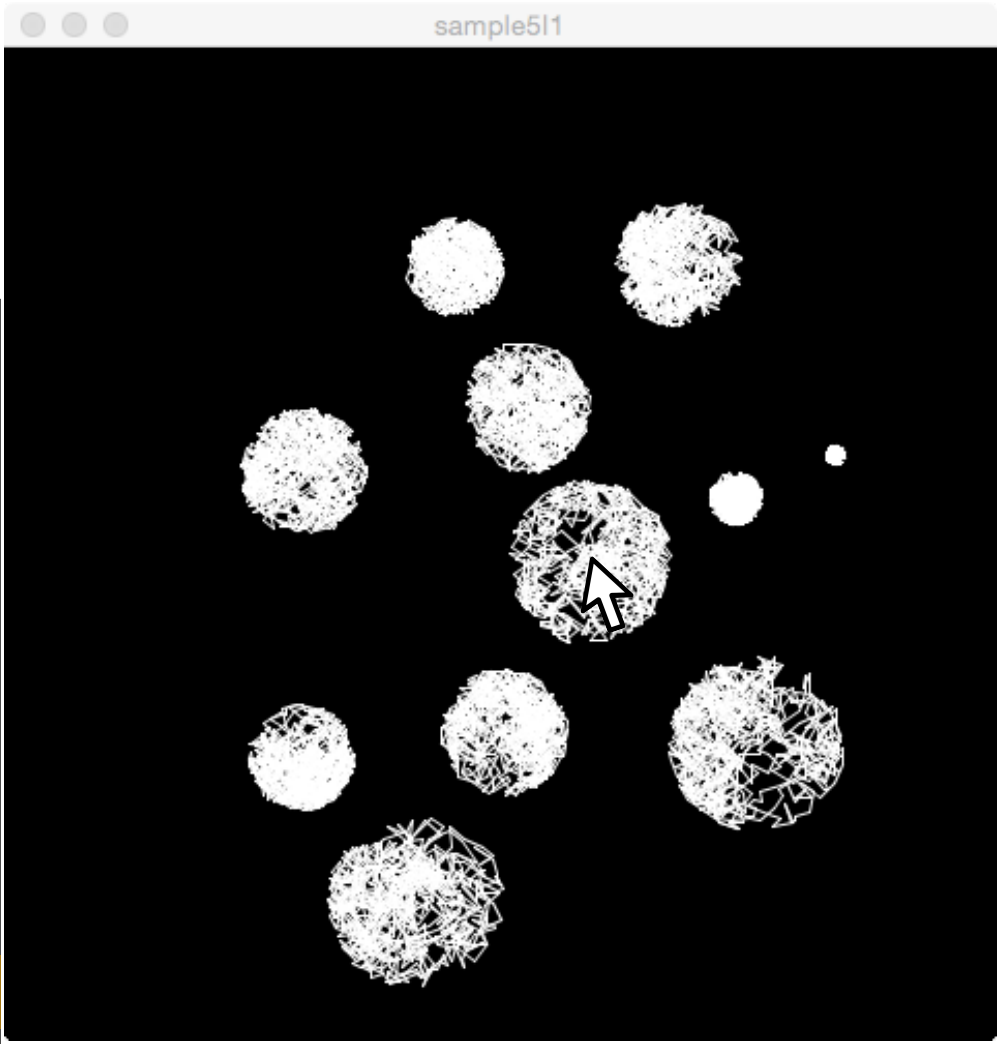
```
1  void setup(){
2    size(480,480);  background(0);
3  }
4
5  void draw(){
6  }
7
8  void mousePressed(){
9    stroke(255);
10   int r = int(random(50));
11   myEllipse(mouseX, mouseY, 1+r);
12 }
```

```
14 void myEllipse(float cx,float cy,int r){
15
16   float px = cx;  float py = cy;
17   for(int i=0;i<1000;i++){
18
19     float n = 10;
20     float x = px + random(2*n) - n;
21     float y = py + random(2*n) - n;
22
23     if(dist(x,y,cx,cy)>r){
24       x = px;  y = py;
25     }
26     line(px,py,x,y);
27     px = x; py = y;
28   }
29 }
```

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．

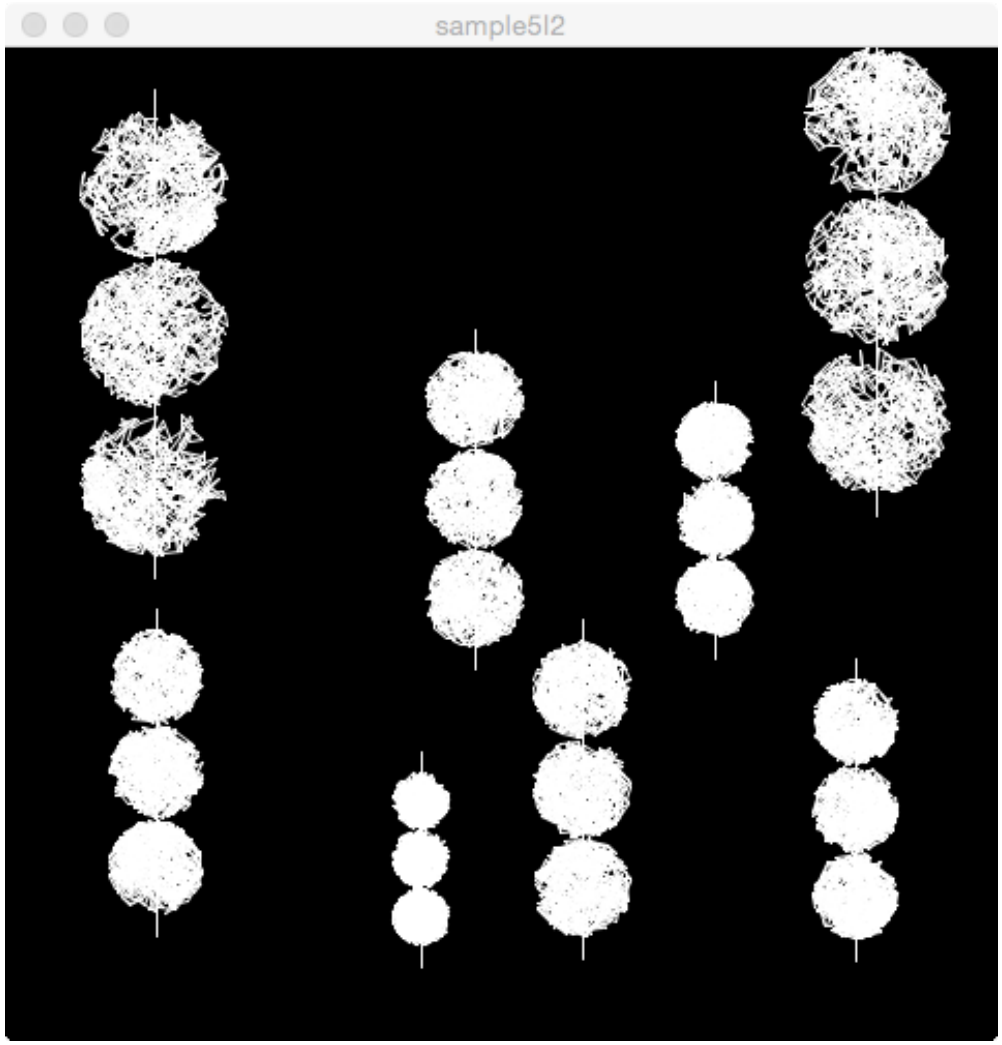


ランダムウォーク・スタンプ

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);

}
```

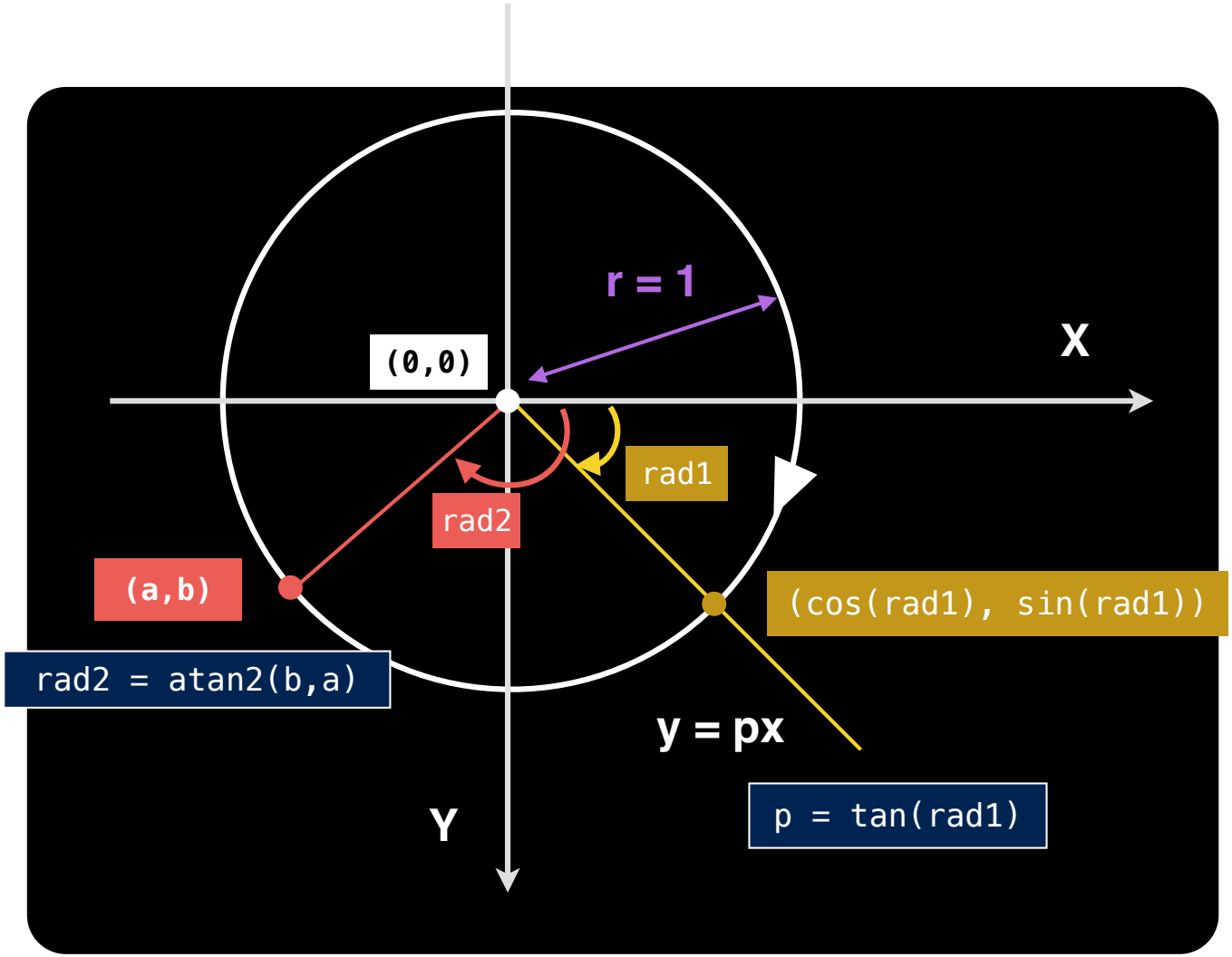


sample4_C6x.pde

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

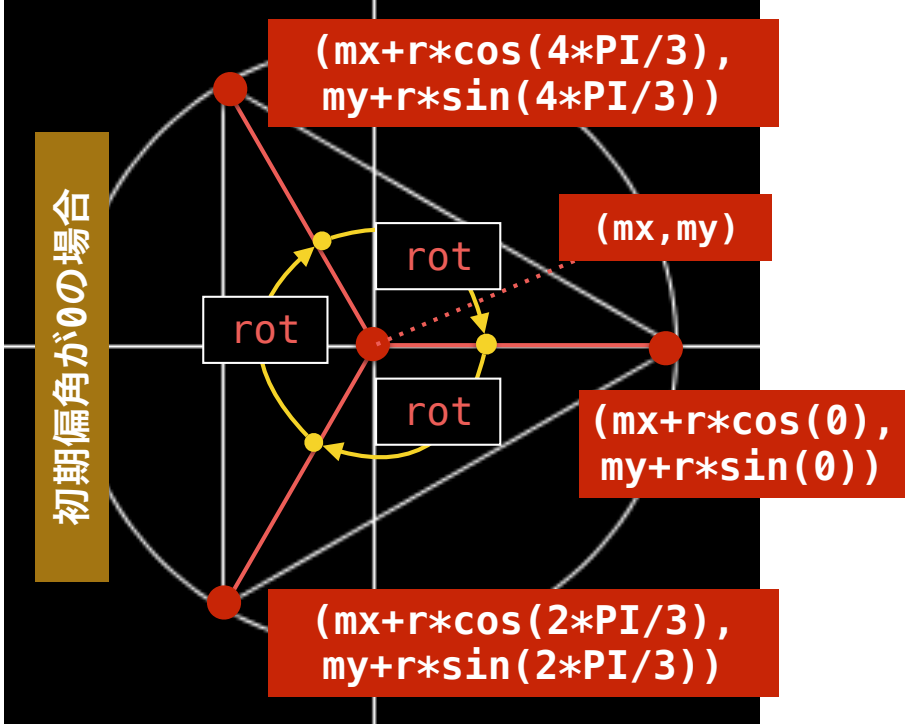
float radians(deg);

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(){
}
```

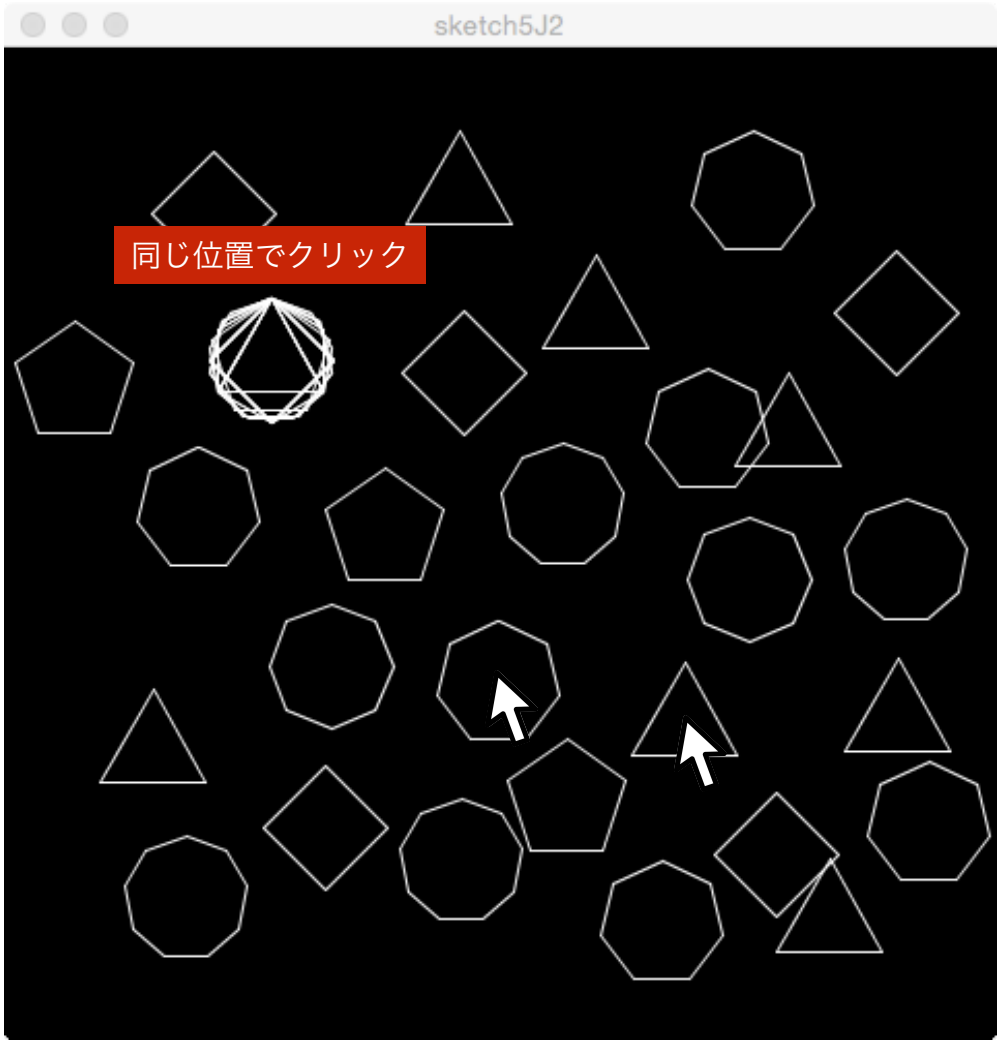
```
void mousePressed(){
  r : 半径, irad: 初期偏角, rot : 回転角
  float irad = random(2*PI);//初期偏角
  float r = 50;                //半径
  float rot = 2*PI / 3; //回転角

  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

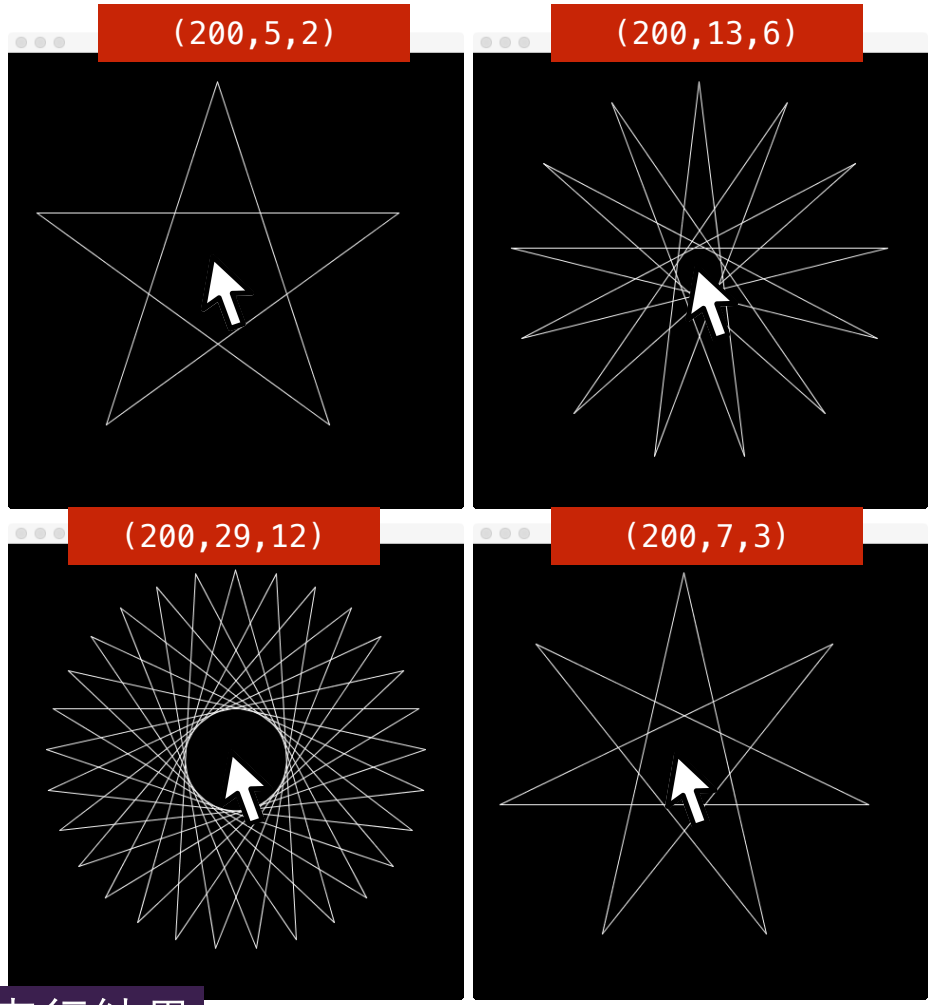
setup, drawは省略

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

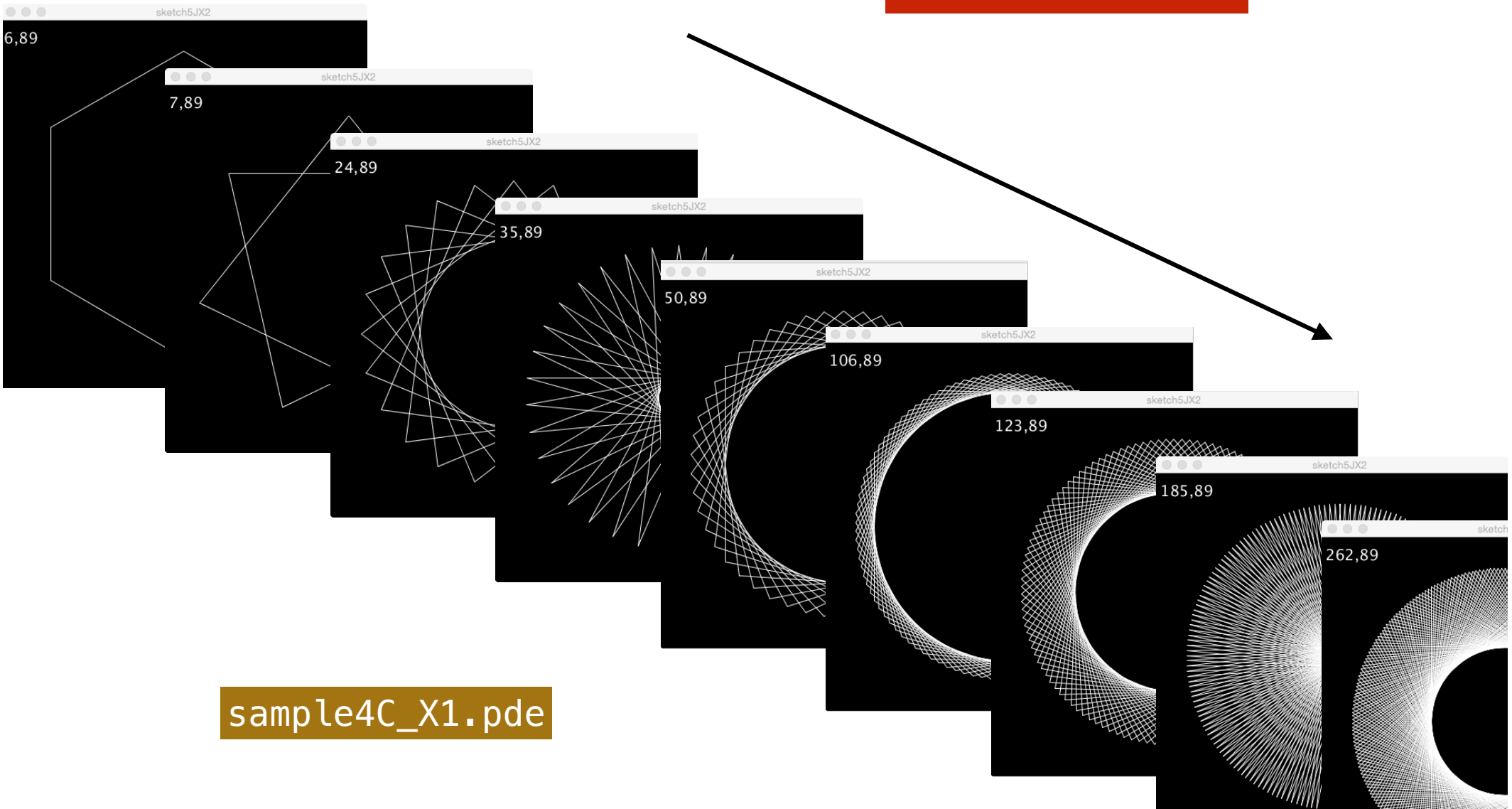
```
void mousePressed(){

  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);
```



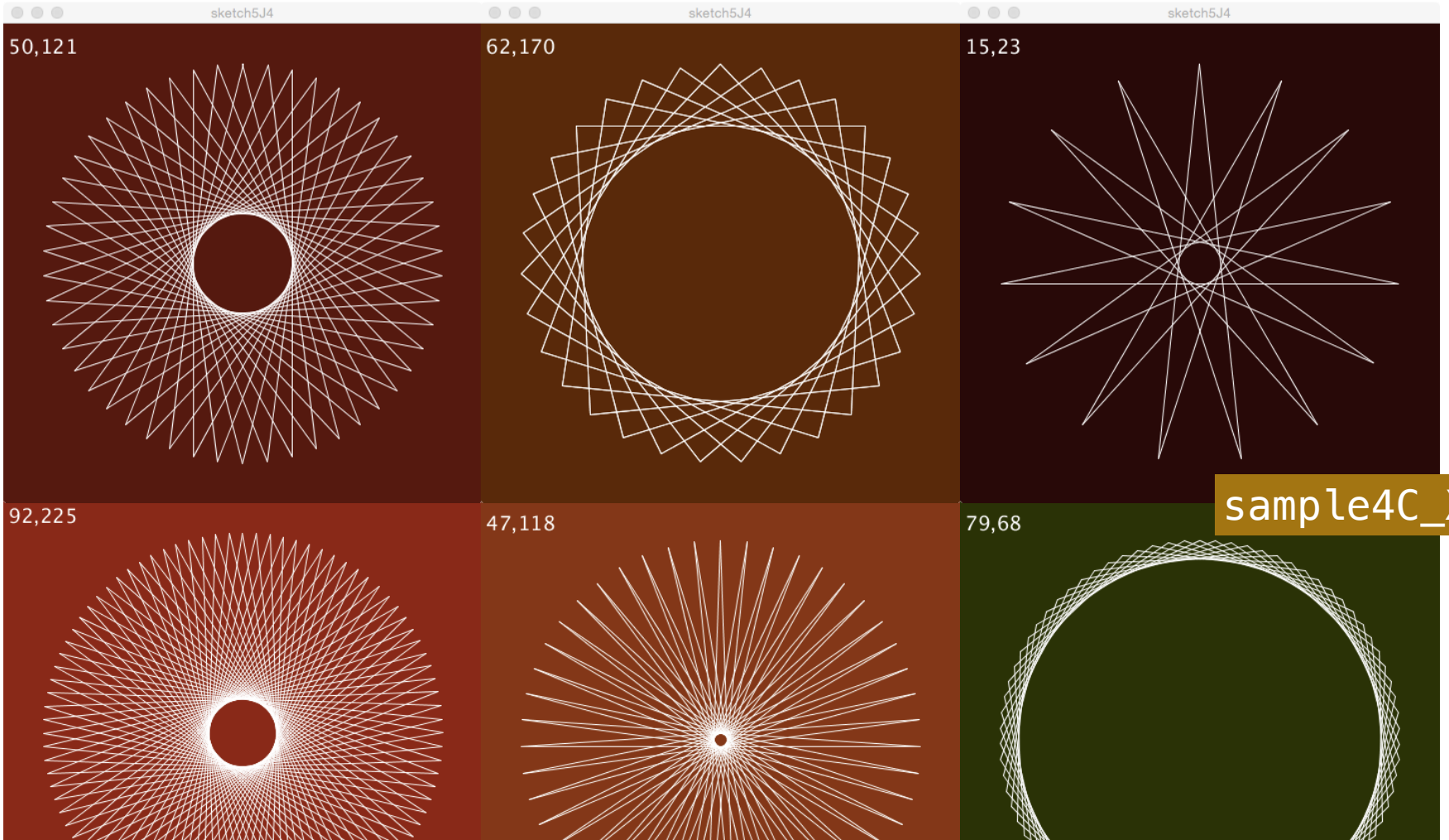
sample4C_X1.pde

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

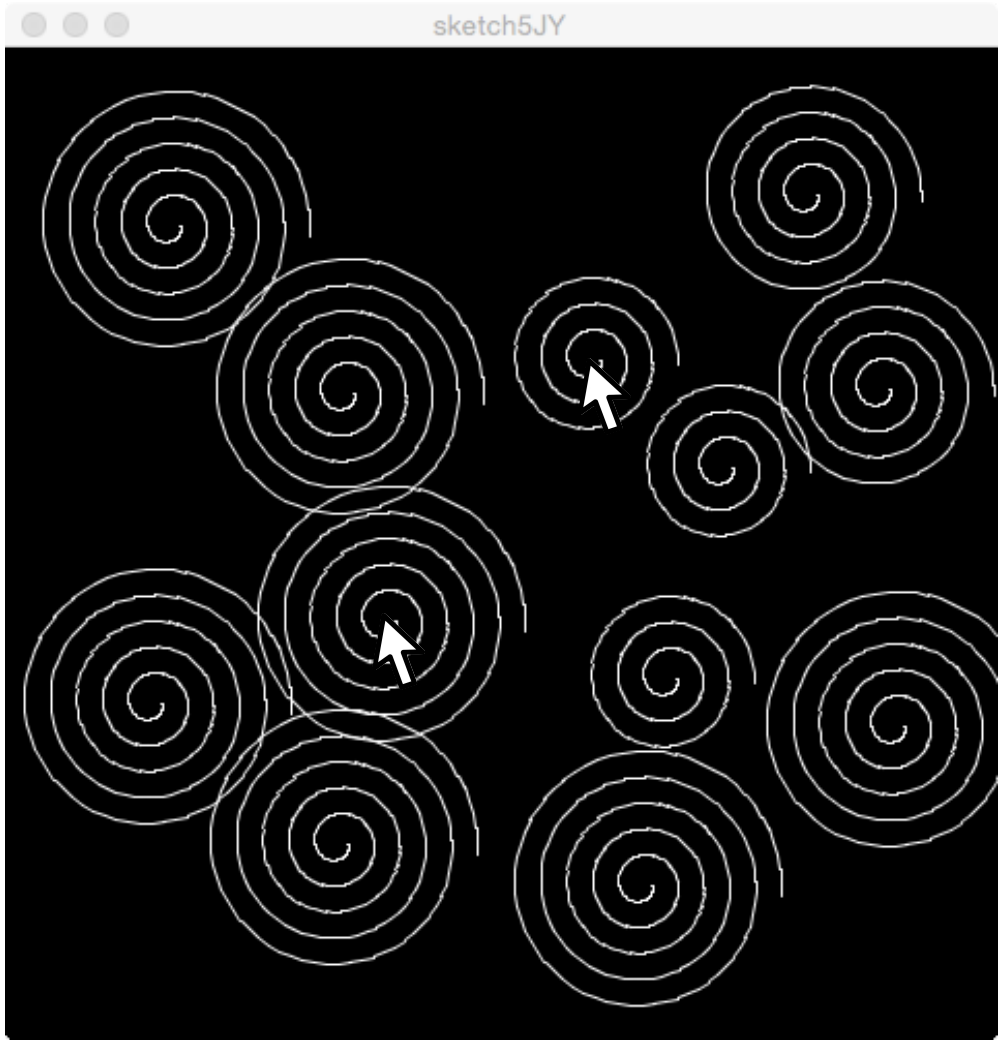
マウスが動いた時のみ, 処理を行うのがスマートです.



sample4C_X2.pde

練習４：cos と sin を使った描画

以下を参考に, マウスの座標を中心として, 引数の数だけ渦を巻く drawUzumaki関数を作成してください.




```
void mousePressed(){

  int n = 3 + int(random(3));
  drawUzumaki(n);

}
```

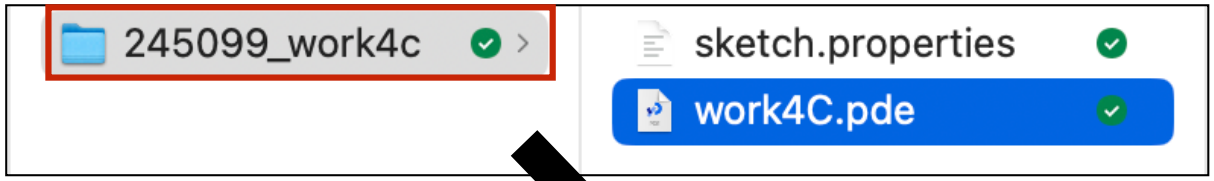
```
void drawUzumaki(int n){

  float l = 5.0;   //動径（半径）
  float rad = 0;   //偏角
  float rot = 0.1; //回転角

  while(true){
    float x1 = mouseX +
    float y1 = mouseY +
    float x2 = mouseX +
    float y2 = mouseY +
    line(x1,y1,x2,y2);
    //偏角を回転し, 動径を0.2ずつ増やす
    rad += rot; l += 0.2;

    if(rad >=          ){
      break;
    }
  }
}
```

work4C.pde

setup, drawは省略

提出方法

ファイル名を学籍番号_work4Cとします.



06/30	(演習4B-C) アニメーション2-3	オンデマンド 授業	[資料PDF] [YOUTUBE] [課題提出｜WORK4C]

https://lab.kenrikodaka.com/univclass/mediabasic2024/

締切は7月4日とします。

練習５：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.