Practice # D
アニメーション（時間的な処理）

Processingをはじめよう　第二版
(Make: PROJECTS)

p.64 – 71

p.236 – 239

演習 1　時間処理・変数のスコープ
(第1回) 7.06 (4限)

演習 2　ランダム・ドローイング
(第2回) 7.13 (4限)

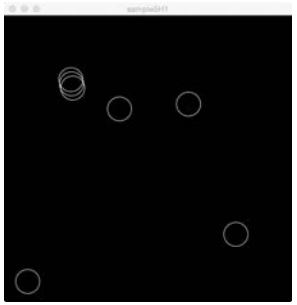
演習 3　自作関数・三角関数
(第3回) 7.20 (4限 - 5限)

準備

クリックした位置を中心とした円を描画します.

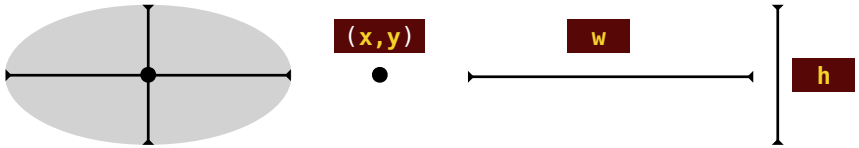
```
void setup(){
  size(480,480);
  background(0);
}

void draw(){
}

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

sampleD_H1.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

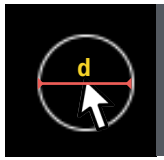
```
void mousePressed(){
  float d = 40;
  noFill(); stroke(255);
  ellipse(mouseX, mouseY, d, d);
}
```

```
void mousePressed(){
  ellipseMouse();
}
```

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

sampleD_H2.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);
}
```

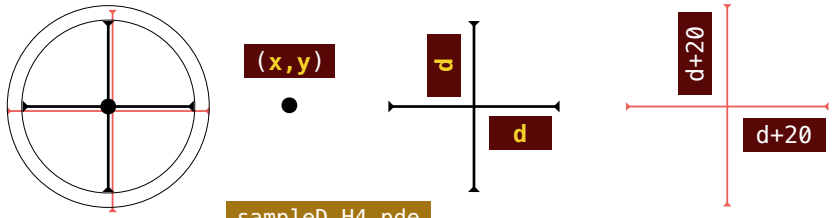
```
void mousePressed(){
  ellipseMouse(40);
}
```

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

sampleD_H3.pde

（引数のある）新しい関数をつくる　2

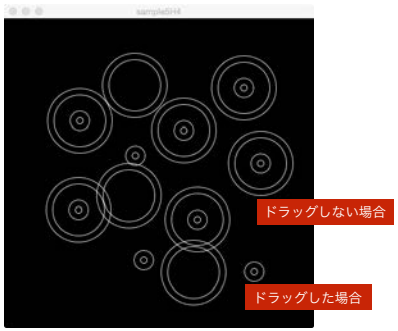
`void dEllipse(float x,float y,float d);`　二重丸を描画する関数



sampleD_H4.pde

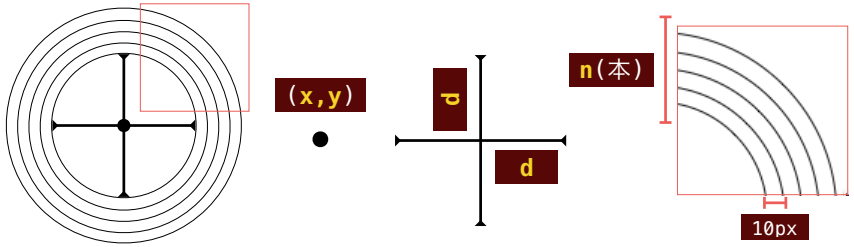
setup, drawは省略

```
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

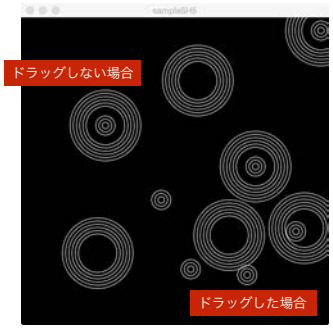
`void mEllipse(float x,float y,float d,int n);`　n 重丸を描画する関数




```
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

sampleD_H5.pde



練習 1

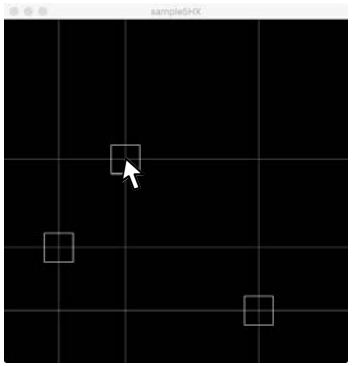
`void square(float x,float y,float d);`

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

void draw(){
}

void mousePressed(){
  stroke(255,100);
  line(mouseX,0,mouseX,height);
  line(0,mouseY,width,mouseY);
  square(mouseX, mouseY, 40);
}
```

引数の（x, y）座標を中心とする正方形の枠線を描くsquare関数を新たに作成してください.

```
void square(float x, float y, float d){
  noFill(); stroke(255);

  float x2 = x - 0.5 * d;
  float y2 = y - 0.5 * d;

  rect(x2, y2, d, d);
}
```

sampleD_HX.pde



練習 2

`void mSquare(float x,float y,float d, float n);`

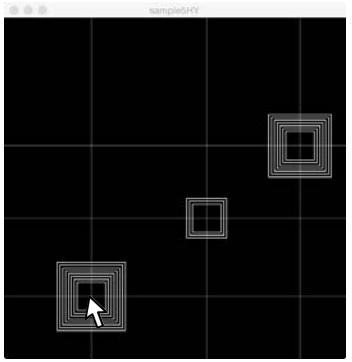
```
void mousePressed(){
  stroke(255,100);
  line(mouseX,0,mouseX,height);
  line(0,mouseY,width,mouseY);

  int n = 3 + int(random(7));
  mSquare(mouseX, mouseY, 40, n);
}
```

```
void mSquare(float x, float y, float d, int n){

  float d2 = d;

  for(int i=0;i<n;i++){

    square(x, y, d2);

    d2 += 5;
  }
}
```

sampleD_HY.pde

setup, draw, squareは省略

練習1のsquare関数を再利用して, 新たに, n重の枠線を描画するmSquare関数を完成させてください.



練習３

`void eSquare(float x,float y,float d, float n);`

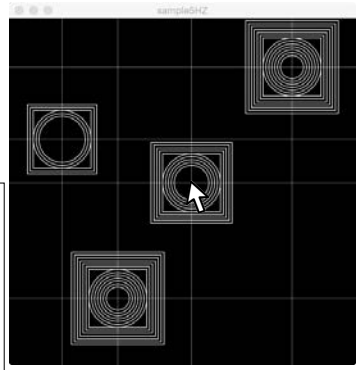
```
void setup(){
  size(480,480);
  background(0);
}
void draw(){
}

void mousePressed(){
  stroke(255,100);
  line(mouseX,0,mouseX,height);
  line(0,mouseY,width,mouseY);

  int n = 3 + int(random(7));
  eSquare(mouseX, mouseY, 80, n);
}
```

sampleD_HZ.pde

mSquareは省略

練習２のmSquare関数を再利用して，新たに，n重の円周を内部に描画するeSquare関数を完成させてください．

```
void eSquare(float x, float y, float d, int n){

  mSquare(x, y, d, n);
  float d2 = d;
  for(int i=0;i<n;i++){

    ellipse(x, y, d2, d2);
    d2 -= 5;
  }

}
```



ランダムウォーク・スタンプ

sample5I1

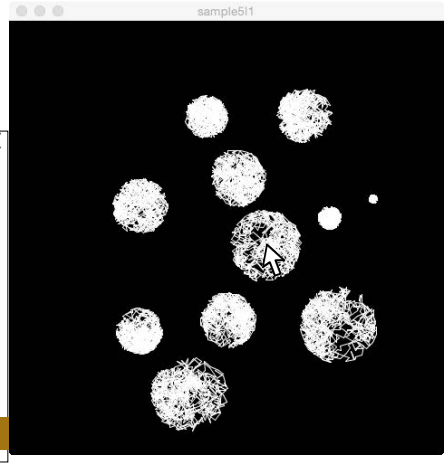
```
void setup(){
  size(480,480);  background(0);
}

void draw(){
}

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

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

（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．

```
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;
  }
}
```

sampleD_I1.pde



ランダムウォーク・スタンプ

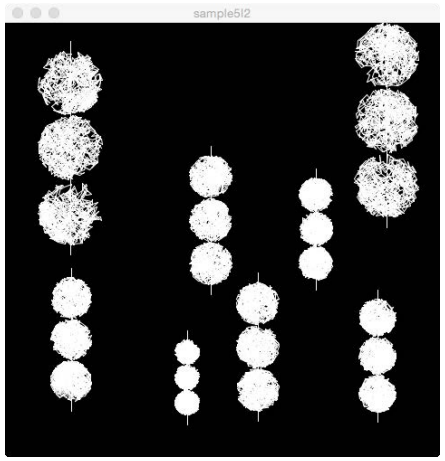
myEllipse(cx,cy,r) を再利用して，(cx,cy)を真ん中の団子の中点として，半径 r の団子が縦に3つ並ぶ myEllipse2(cx,cy,r)を作成してください．

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);

}
```

sampleD_I2.pde

myEllipseは省略



ランダムウォーク・スタンプ

`void myEllipse3(float cx,float cy,int r1, int r2);`

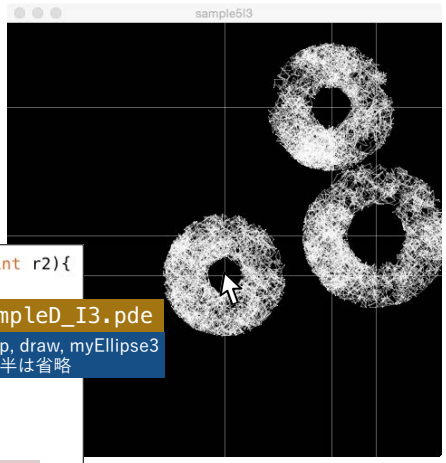
（cx, cy）を中心とした半径（r1）の円と（r2）の円で挟まれた領域に，最大5000のランダムウォークの線を描く．

```
void mousePressed(){
  stroke(255,100);
  line(mouseX, 0, mouseX, height);
  line(0,mouseY, width, mouseY);

  int r1 = 10 + int(random(30));
  int r2 = r1 + 50;
  myEllipse3(mouseX, mouseY, r1, r2);
}
```

```
void myEllipse3(float cx,float cy, int r1, int r2){

  float px = cx + r1+ (r2-r1)*0.5;
  float py = cy;

  for(int i=0;i<5000;i++){

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

    if(dist(x,y,cx,cy)<r1 || dist(x,y,cx,cy)>r2){
      x = px;  y = py;
```

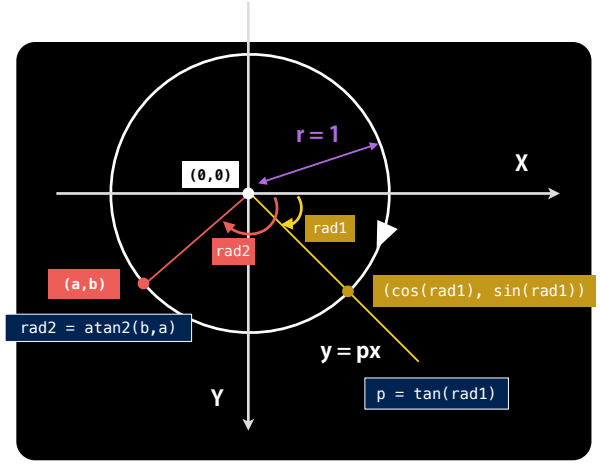
sampleD_I3.pde

setup, draw, myEllipse3の後半は省略



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

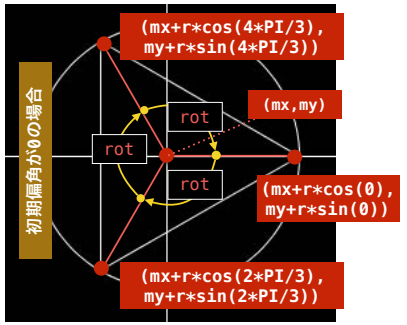
`float atan2( b,a );`
座標（a, b）の角度（偏角）を返す

`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
  }
}
```

sampleD_J1.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
  }
}
```

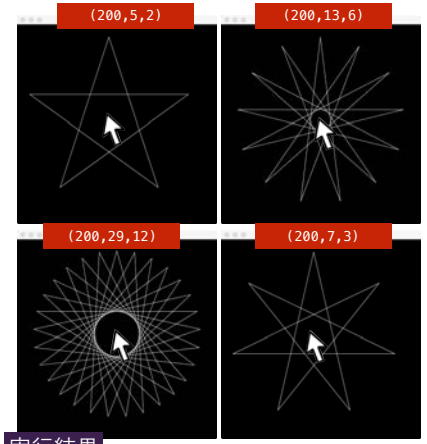
sampleD_J2.pde
setup, drawは省略

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

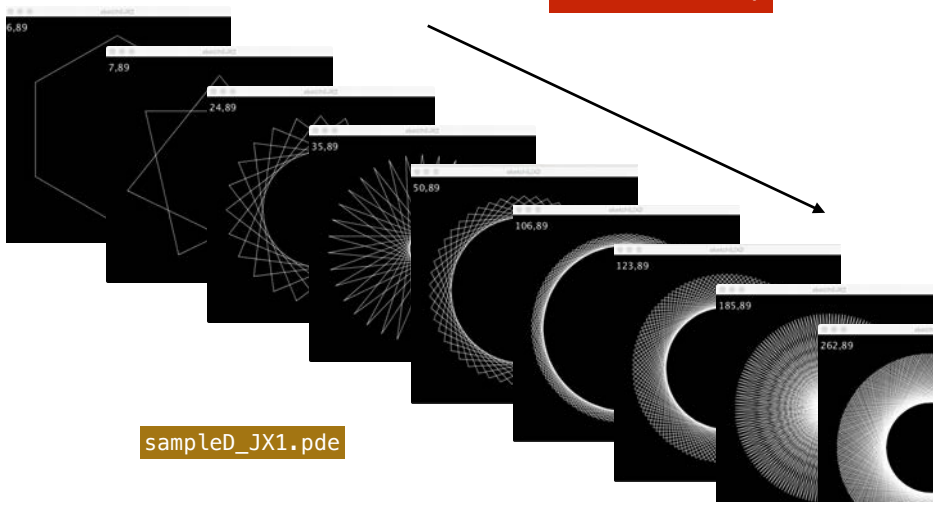
```
void mousePressed(){

  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
  }
}
```

sampleD_J3.pde

練習３：cos と sin を使った描画

マウスの位置によって，「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);
```



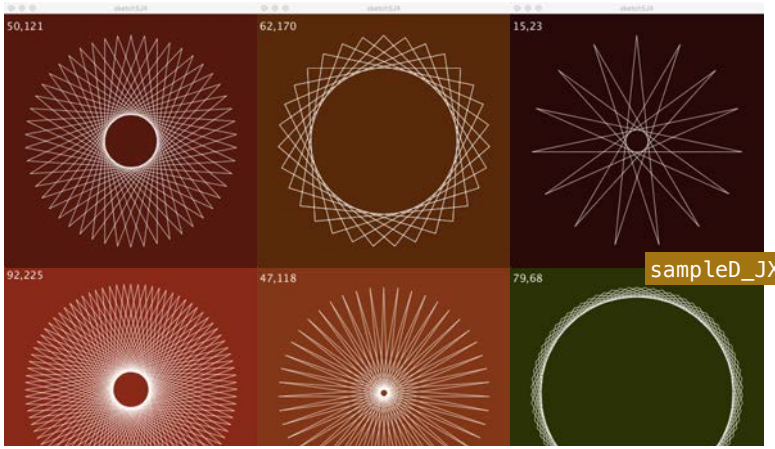
sampleD_JX1.pde

練習４：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

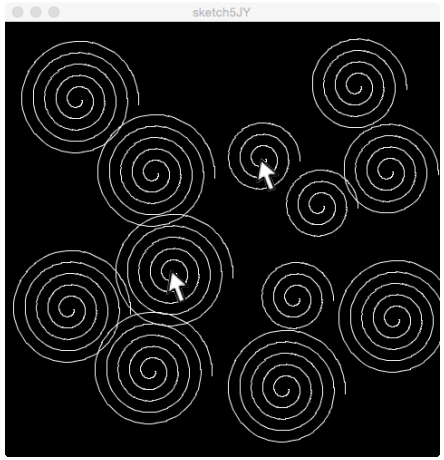
マウスが動いた時のみ，処理を行うのがスマートです．



sampleD_JX2.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．



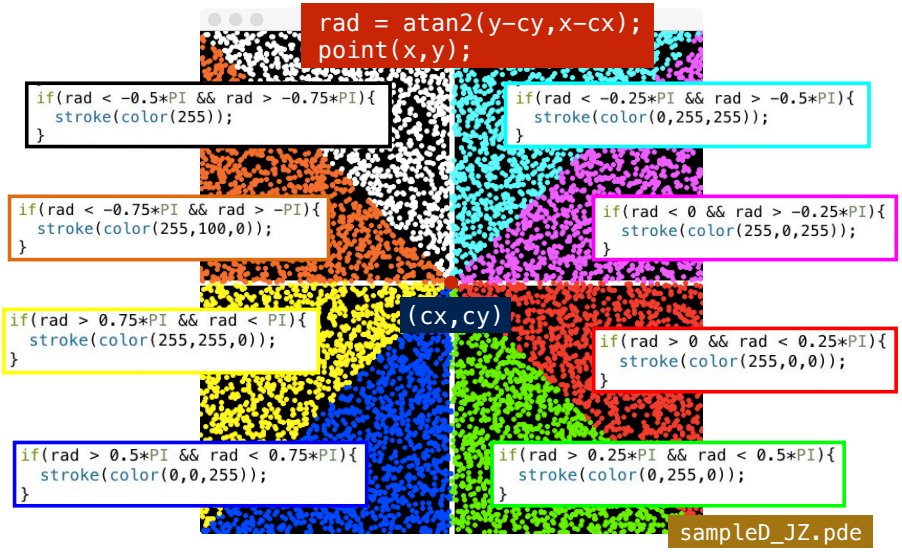

```
void mousePressed(){

  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 + l * cos(rad);
    float y1 = mouseY + l * sin(rad);
    float x2 = mouseX + l * cos(rad + rot);
    float y2 = mouseY + l * sin(rad + rot);
    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 >= n * 2 * PI){
      break;
    }
  }
}
```

sampleD_JY.pde
setup, drawは省略

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．



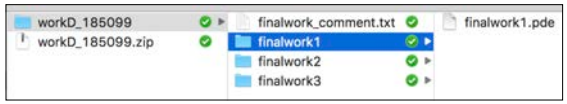
sampleD_JZ.pde

最終課題

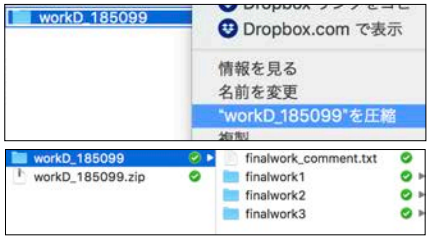
自由な発想で, ランダムなアニメーションを３つ完成させてください.

提出方法

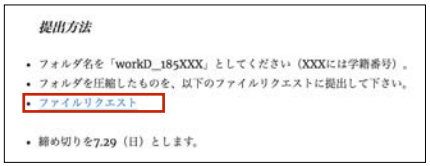
以下のようなファイル構成にします.



最上位のフォルダを圧縮します.



圧縮したzipファイルをファイルリクエストで提出します.



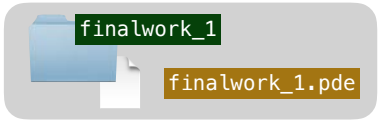
http://labrec.kenrikodaka.com/mediabasic2018/#workD

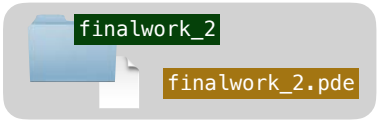
締め切りは7月29日（日曜日）とします.

最終課題（内容）

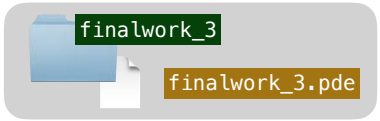
ウィンドウのサイズは自由とします.

すべて, rand関数を使ったアニメーションとしてください. つまり, 実行ごとに, 異なるアニメーションが描画されることとなります.





finalwork_1とfinalwork_2は, マウス座標・マウスイベントと無関係のアニメーションを作成してください.



finalwork_3は, マウス座標やマウスイベントによって描画が変わるアニメーションを作成してください. （実行時に僕がマウスを動かしたりクリックしてみます）

プログラム作成にあたって, 工夫したところ, 注目してほしいことを数行で書いてください. ファイル形式は自由です（rtfとかkeyとかpdfでもOK）.