

Practice #3

アニメーション（時間的な処理）

演習3A 時間処理・イベント処理・変数のスコープ

課題学習3A1

課題学習3A2

課題学習3A3

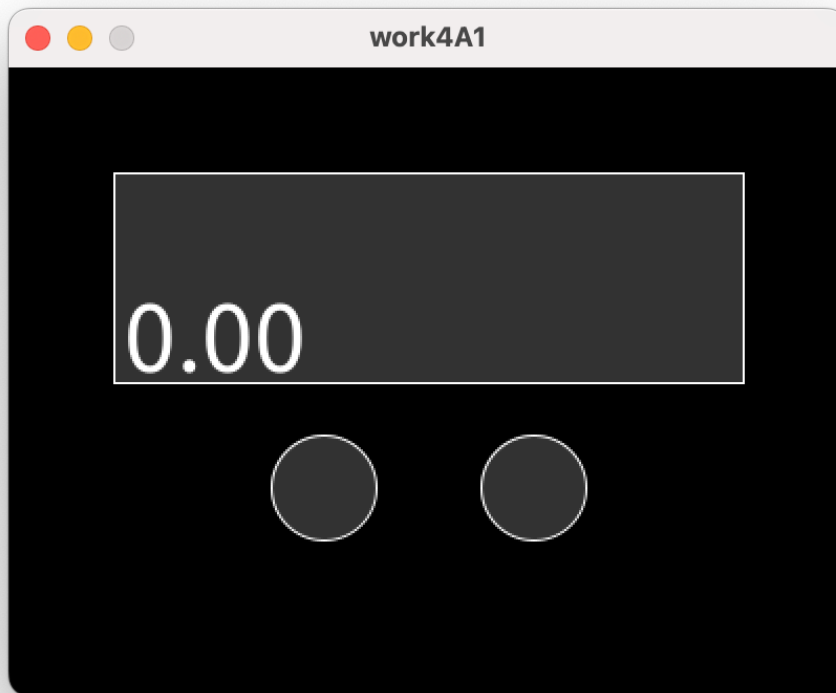
**締め切り
06.19 [木]**

課題 3A1

2つのボタンを有するストップウォッチを作ってください。左ボタンで計測の開始と停止、右ボタンでリセット（0に戻る）ができるようにします。ボタンの上には、横長のメータが描画され、1秒間を単位に、左から右へと移動していきます。メータの左下に、時間を、小数点第2位まで描画してください。細かな仕様は映像を確認すること。

work3A1.pde

(初期状態)



(前半部分)

```
1  /* カウンター (100で一秒経過) */
2  int count = 0;
3
4  /* メータのレイアウト */
5  float x_meter = 50; float y_meter = 50;
6  float w_meter = 300; float h_meter = 100;
7
8  /* ボタン1のレイアウト */
9  float b1_x = 150; float b1_y = 200; float b1_rad = 25;
10 /* ボタン2のレイアウト */
11 float b2_x = 250; float b2_y = 200; float b2_rad = 25;
12
13 /* ストップウォッチの状態 */
14 /* 初期状態は停止 */
15 boolean run = false;
16
17 void setup(){
18
19     size(400,300); background(0);
20     frameRate(100); //フレームレートを100fpsとする
21
22     /* メータの描画 */
23     stroke(255); fill(50); rect(x_meter,y_meter,w_meter,h_meter);
24     /* ボタン1の描画 */
25     stroke(255); fill(50); ellipse(b1_x,b1_y,2*b1_rad,2*b1_rad);
26     /* ボタン2の描画 */
27     stroke(255); fill(50); ellipse(b2_x,b2_y,2*b2_rad,2*b2_rad);
28     //テキストを描画 (白) (時間は0とする)
29     fill(255); textSize(50); text("0.00",x_meter+5,y_meter+h_meter-5);
30
31 }
```

左ボタンの機能

一時停止状態

再生状態

一時停止状態

再生状態

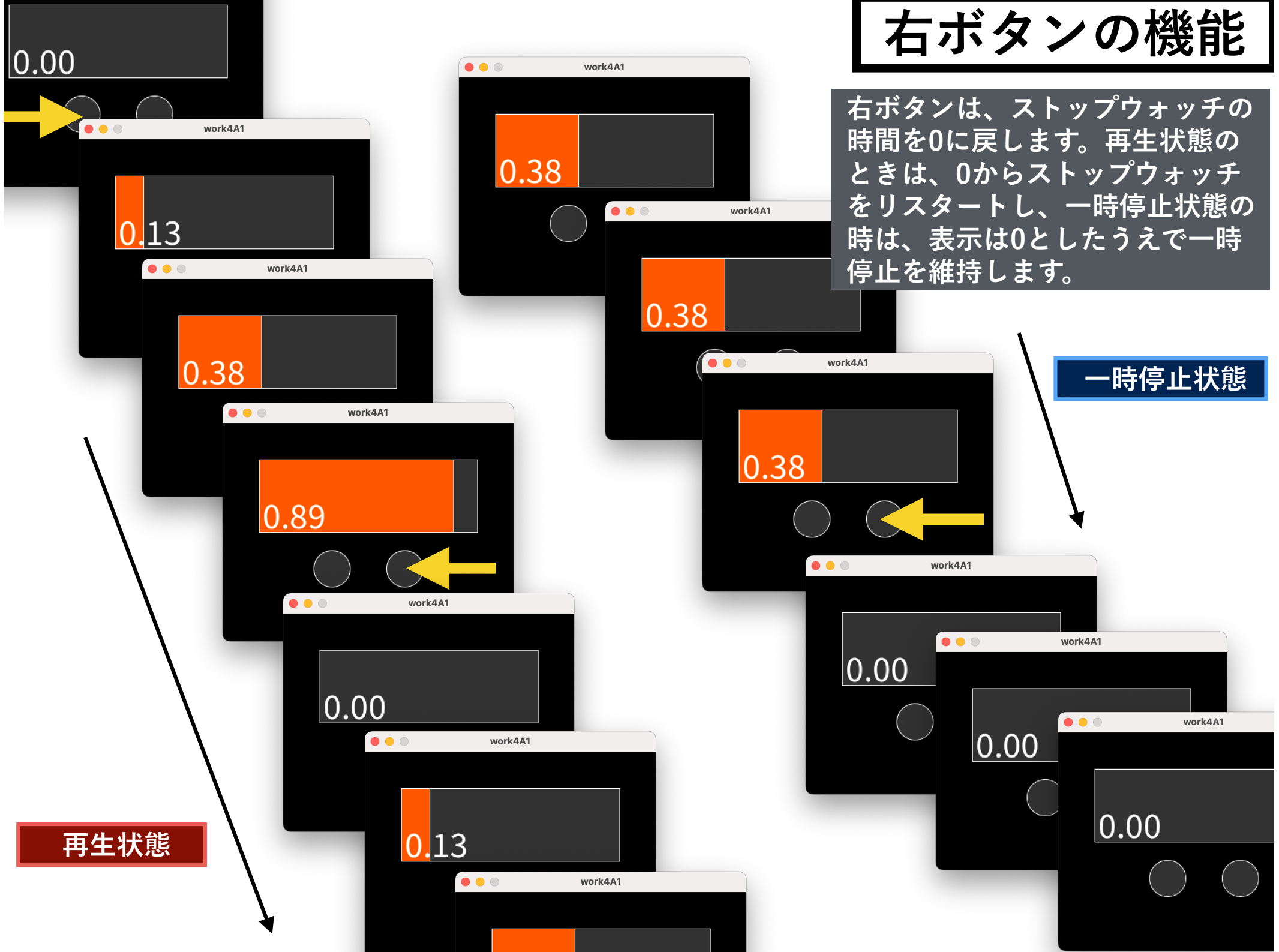
左ボタンを押すと計測が始まり、再び押すと計測を停止。さらに押すと、現在の状態から引き続き計測を再開します。

右ボタンの機能

右ボタンは、ストップウォッチの時間を0に戻します。再生状態のときは、0からストップウォッチをリスタートし、一時停止状態の時は、表示は0としたうえで一時停止を維持します。

一時停止状態

再生状態



課題 3A1 (ヒント：小数点第一位までの描画)

以下を参考に、小数点第二位までの描画の方法を考えてください。

frameRateを10とすることで、countが10で一秒（1.0）と対応します。

```
work4A1_decimalpoint
1 int count = 0;
2
3 void setup() {
4   size(400, 400);
5   background(0);
6   frameRate(10);
7
8   noStroke();
9   textSize(100);
10  fill(0);
11 }
12
13
14 void draw() {
15
16   background(255);
17
18   int a = floor(count / 10);
19   int b = count % 10;
20   String s = a+"."+b;
21
22   text(s, 100, 220);
23   count++;
24 }
```



小数点の描画方法の例です。aが小数点の整数部分、bが小数第一の数字と対応します。

`float floor(a);`

小数点以下を切り捨てた値を返す

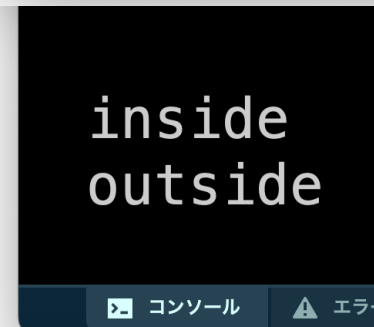
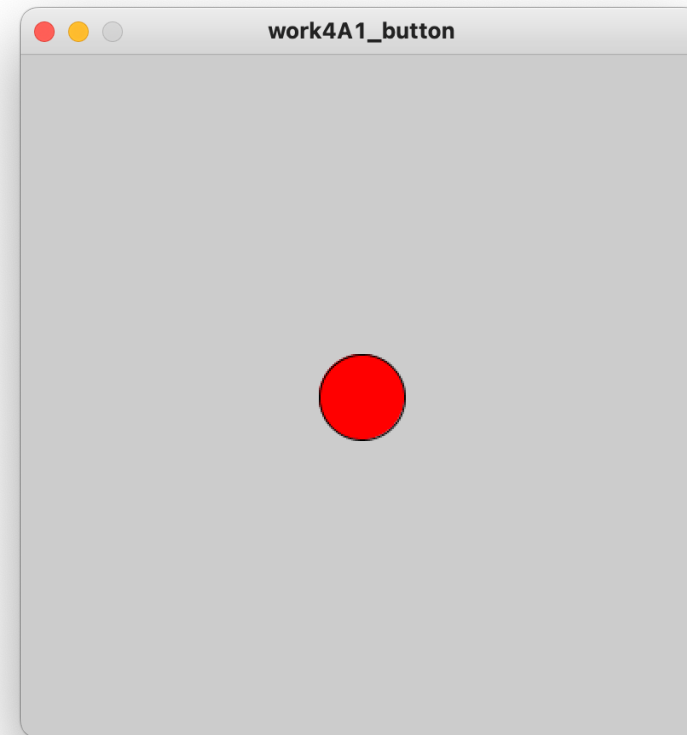
課題 3A1 (ヒント: ボタンの実装)

work4A1 button

```
1 int count = 0;
2
3 int bx = 200;
4 int by = 200;
5 int rad = 25;
6
7 void setup() {
8     size(400, 400);
9     fill(255,0,0);
10    ellipse(bx,by,2*rad,2*rad);
11 }
12
13 void draw() {
14 }
15
16 void mousePressed(){
17
18     if(dist(mouseX,mouseY,bx,by)<rad){
19         println("inside");
20     }else{
21         println("outside");
22     }
23 }
24 }
```

円いボタンの中心座標を(bx,by)、半径をradとしています。

マウスを押した位置が、(bx,by)からrad以内の距離にあるかでボタンの内外判定をしています。

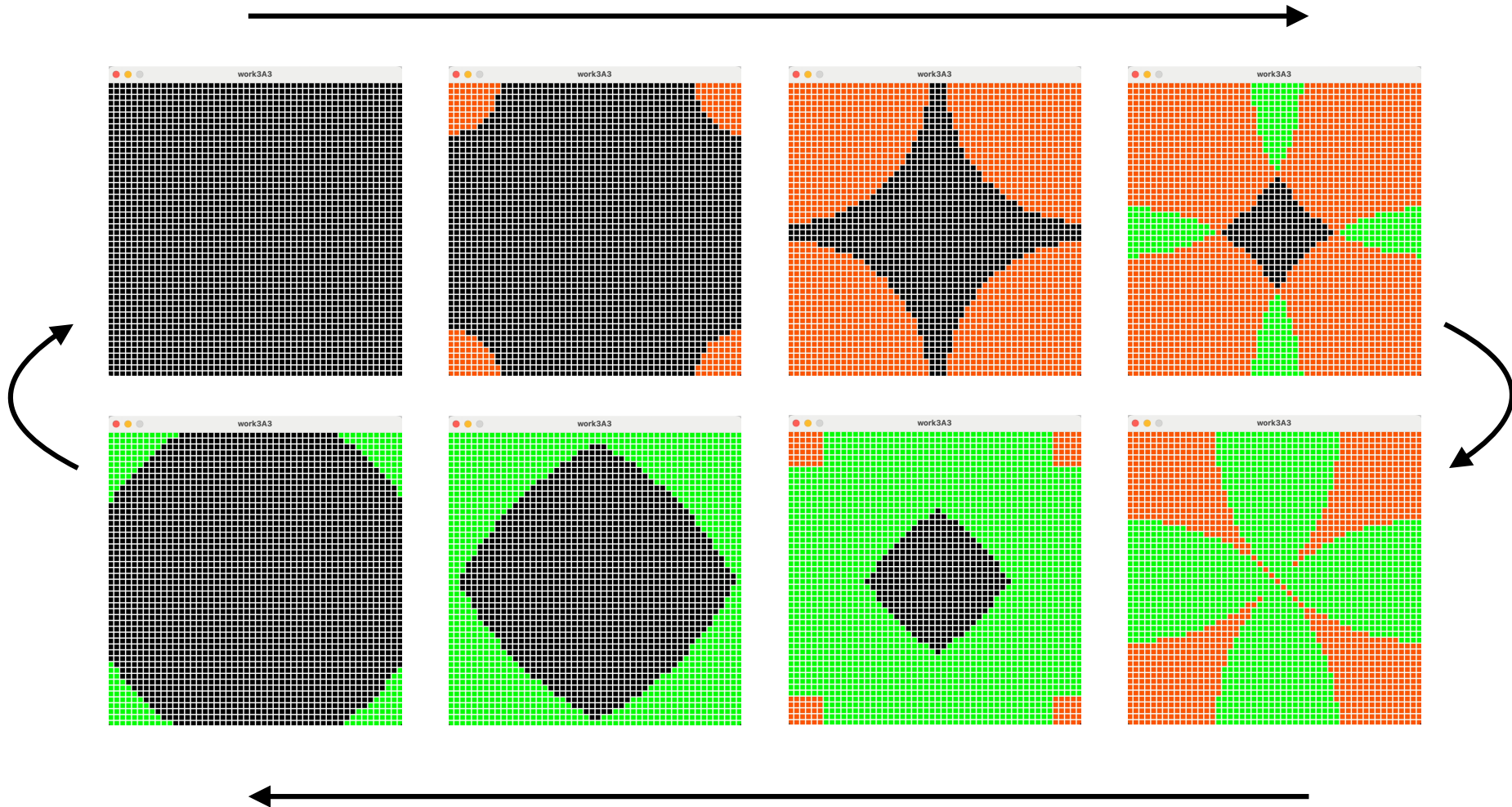


赤ボタンを押した時に「inside」、外側を押すと「outside」がコンソールに出力されます。

課題 3A2

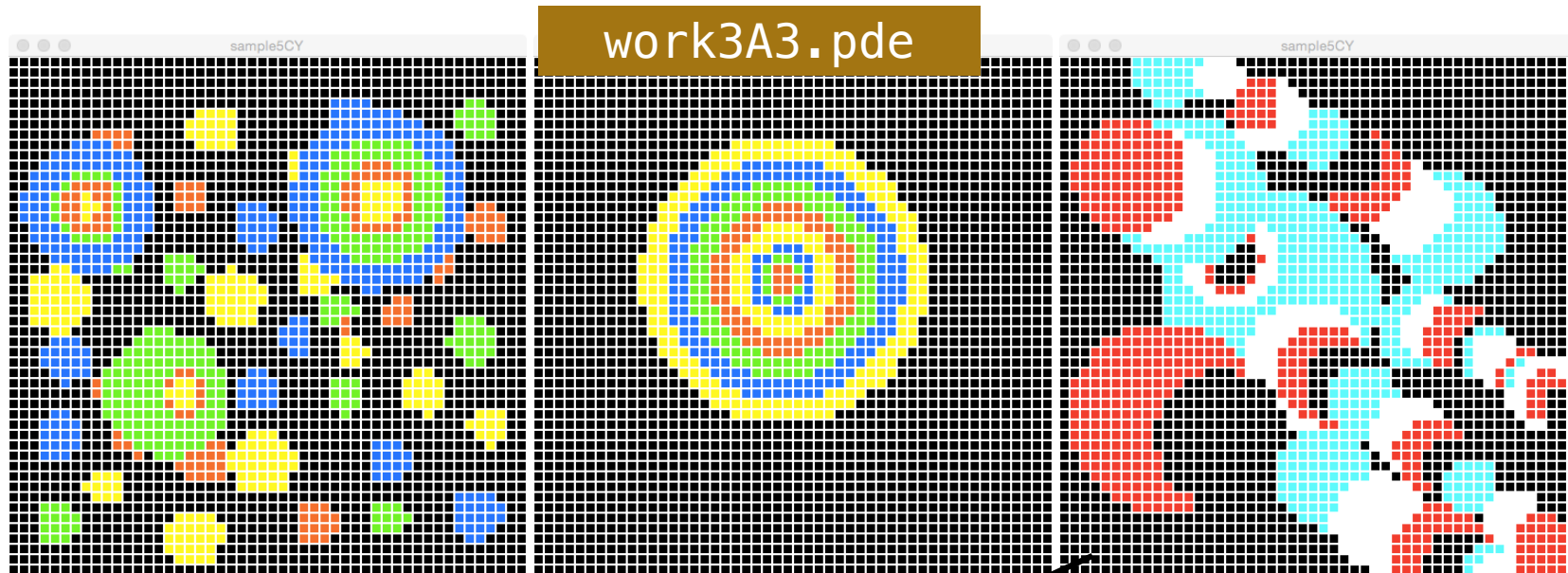
Sample3A_6を拡張するかたちで、windowの四隅から中心に向かって波が現れ（オレンジ）、2つの波が重なった領域で別の色に転じ（黄緑）、4つの波が重なった領域で再び「黒」に戻るアニメーションを作成せよ。詳しくは映像を確認してください。

work3A2.pde

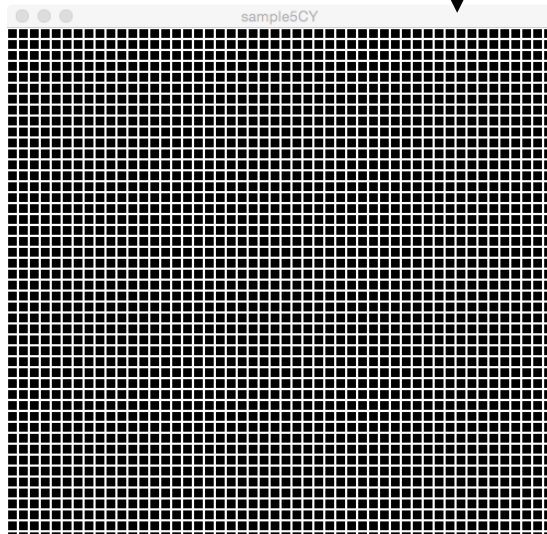


課題 3A3

「sample3A_7.pde」を修正して、マウスをクリックするたびに、円の色が変わるように（最低3種類）してください。また、スペースキーを押すと、描画がクリアされるようにしてみましょう。



SPACE



p.72-

この部分に、描画をクリアする処理を書き込めばよいです。

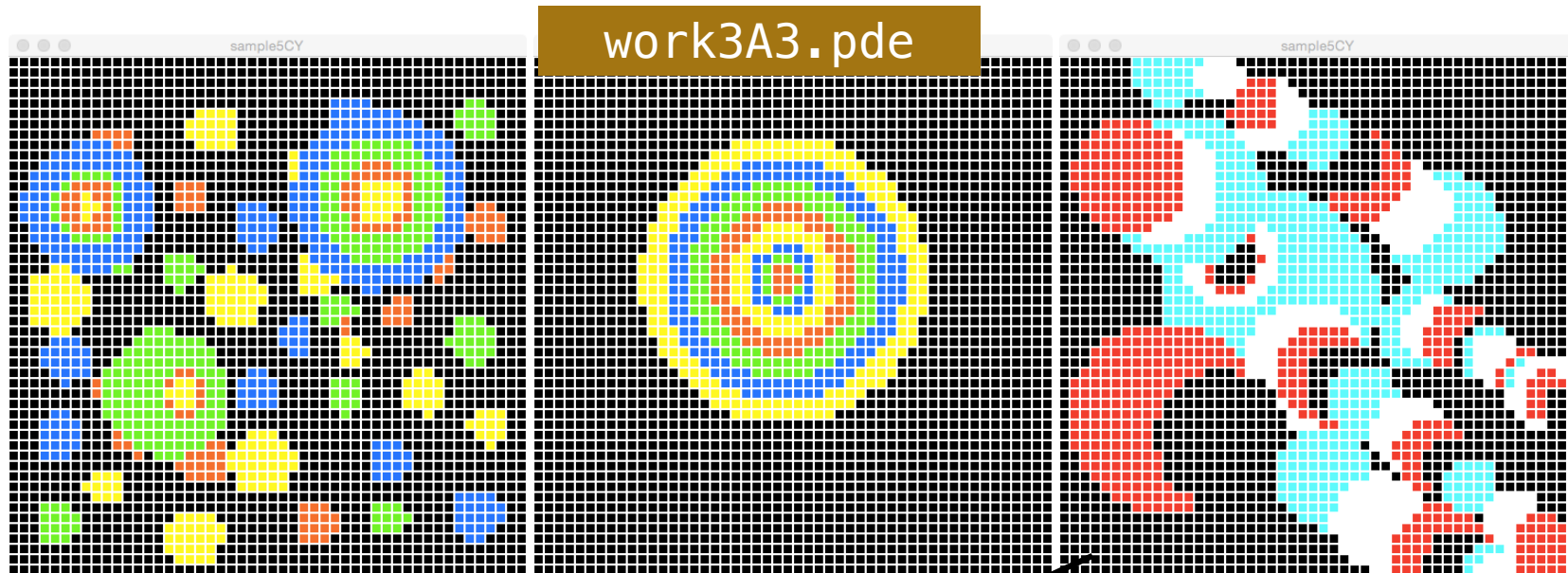
`void keyPressed();`

なんらかのキーボードのキーが押された時の処理

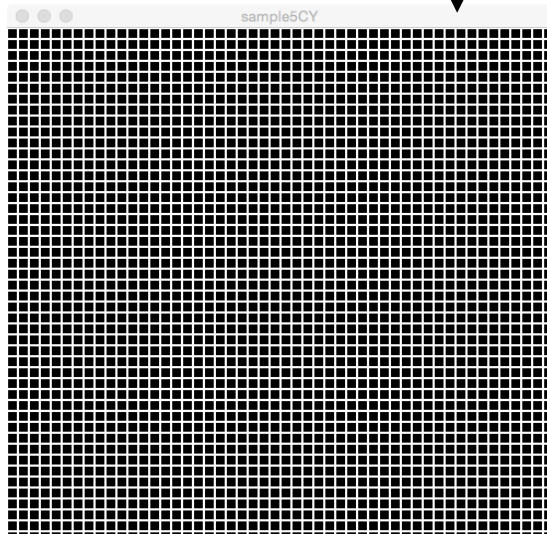
```
void keyPressed(){  
  if(key == ' '){  
    println("push space!!");  
  }  
}
```

課題 3A3

「sample3A_7.pde」を修正して、マウスをクリックするたびに、円の色が変わるように（最低3種類）してください。また、スペースキーを押すと、描画がクリアされるようにしてみましょう。



SPACE



p.72-

この部分に、描画をクリアする処理を書き込めばよいです。

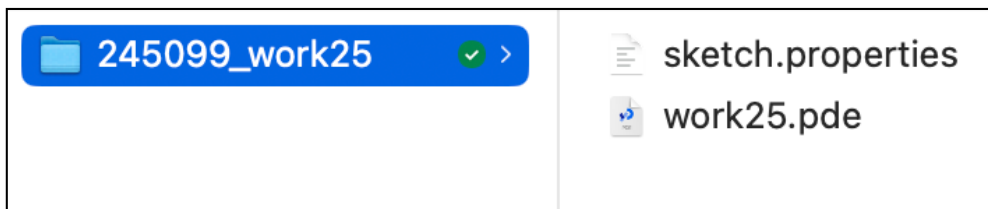
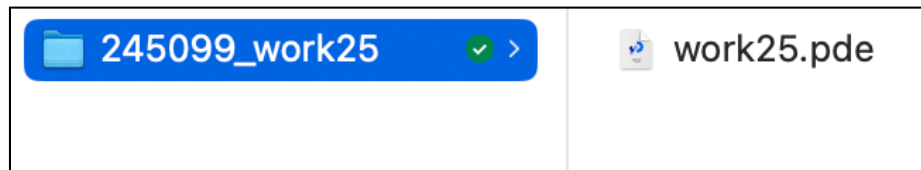
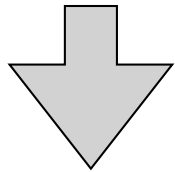
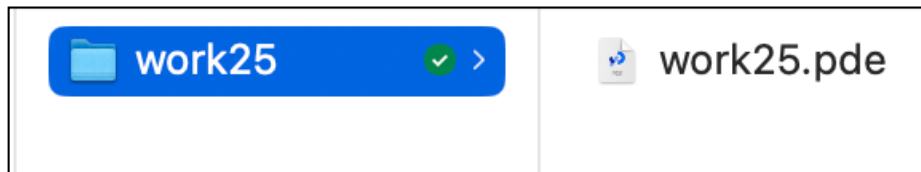
`void keyPressed();`

なんらかのキーボードのキーが押された時の処理

```
void keyPressed(){  
  if(key == ' '){  
    println("push space!!");  
  }  
}
```

提出方法

フォルダ名のみを2550xx_workXXと変更します。
pdeファイルを開くと、同じ階層に設定ファイル
ができますが気にしないで下さい。
(下図は、work25のときの例です。)



対応する課題のリンクから、フォルダを
まるごと提出してください。

[https://lab.kenrikodaka.com/
univclass/mediabasic2025/](https://lab.kenrikodaka.com/univclass/mediabasic2025/)

[課題提出 | WORK3A1]

[課題提出 | WORK3A2]

[課題提出 | WORK3A3]

提出方法

リクエスト時は、名前の先頭に学籍番号を入れるようにしてください。

kodaka kenri さんからのリクエストです
WORK25

▶  245099_work25

⊕ ファイルを追加

名前

245099_小鷹研理

メールアドレス

kenrikodaka@gmail.com

アップロード



ファイルは kodaka kenri さんの Dropbox に安全にアップロードされます。 [ファイル リクエスト](#)と Dropbox の[プライバシー ポリシー](#)についての詳細をご覧ください。