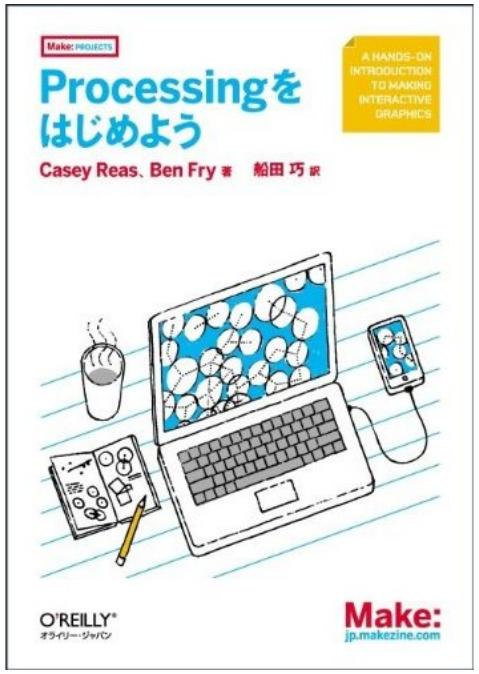
Practice #4

アニメーション（時間的な処理）

演習４A　時間処理・イベント処理・変数のスコープ



Processingをはじめよう
第二版
（Make: PROJECTS）

オンデマンド授業　6.07

課題学習　6.14

p.64 – 71

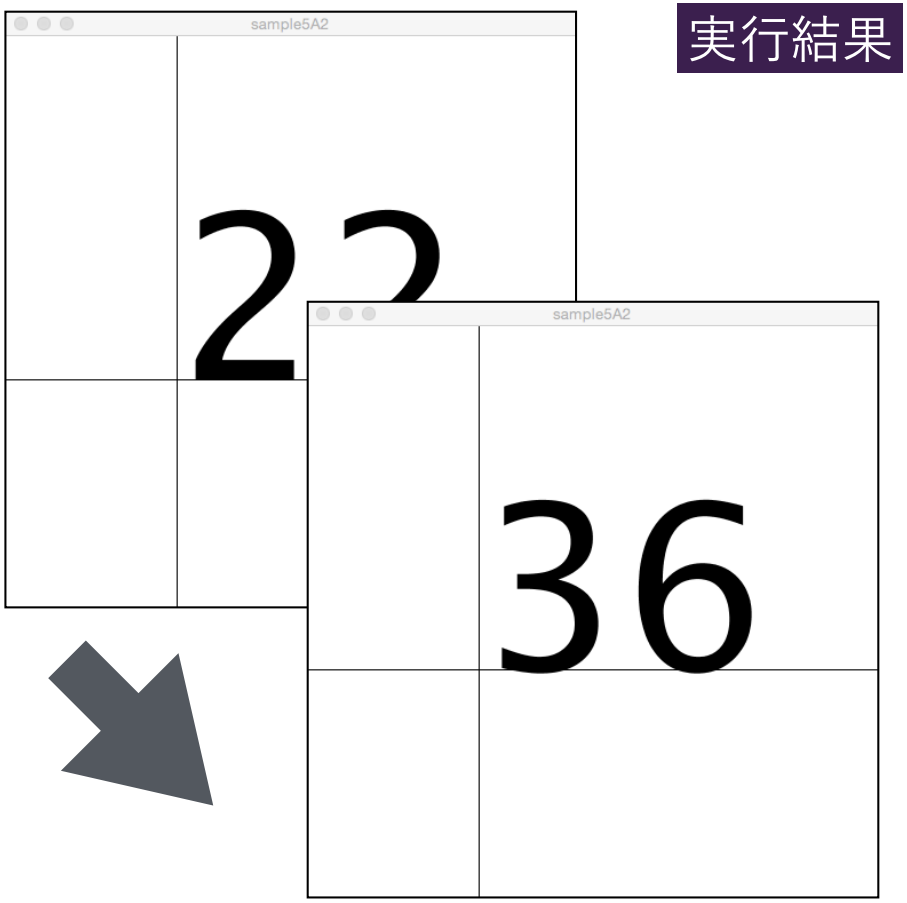
p.236 – 239

アニメーションの基礎：setup()とdraw()の関係

```
int count; //カウンター

//初回の処理
void setup(){
  size(500,500);    //サイズ
  frameRate(10);   //フレームレート

  textSize(200);   //テキストサイズ
  fill(0); stroke(0); //線・文字は黒

  count = 0;        //カウンター初期化
}

//定期的な処理
void draw(){
  background(255);   //白く塗りつぶす

  //カウンターを書く
  text(count,150,300);
  //罫線
  line(150,0,150,height);
  line(0,300,width,300);

  count++;   //カウンターを更新
}
```

sample4A_1.pde

実行結果



void text(text, x, y);

text（文字列、または数字）を，（**x, y**）に表示.

void textSize(n);

text関数実行時のサイズを **n** に指定

関数間で共有すべき変数の宣言

```
int count; //カウンター
```

グローバル変数の宣言

初期化処理（初回に一回だけ実行）

```
void setup(){
  size(500,500);    //サイズ
  frameRate(10);    //フレームレ
  textSize(200);    //テキストサ
  fill(0); stroke(0); //線・ス
  count = 0;
}
```

setup関数

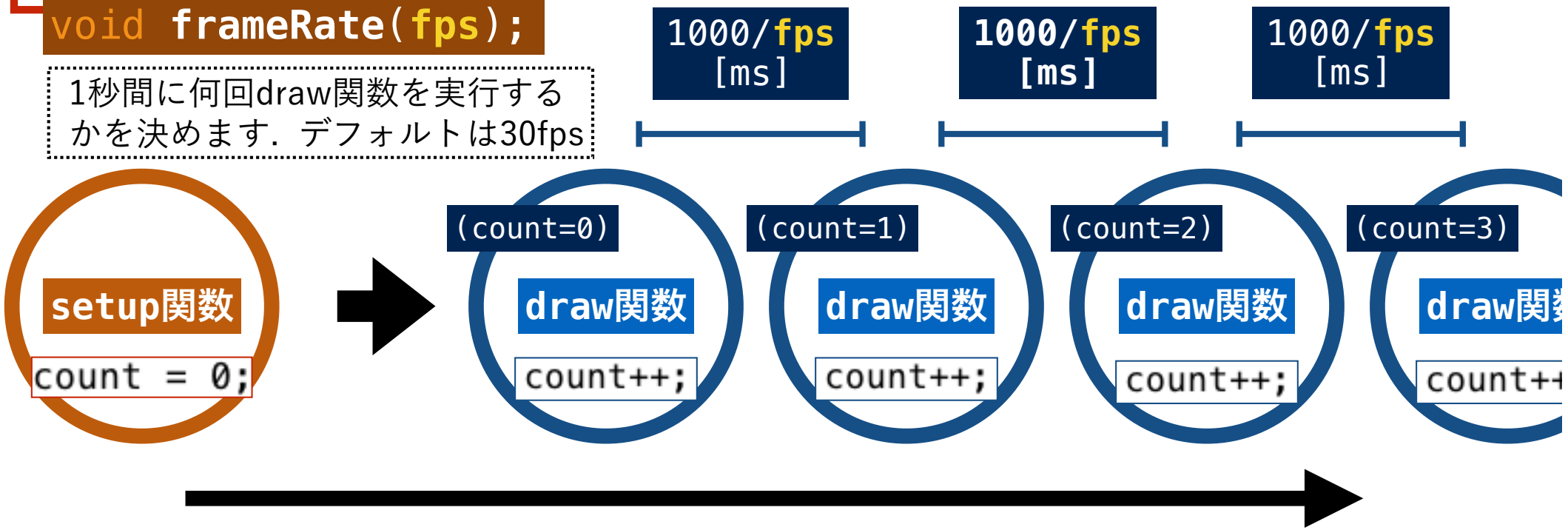
```
void frameRate(fps);
```

1秒間に何回draw関数を実行するかを決めます．デフォルトは30fps

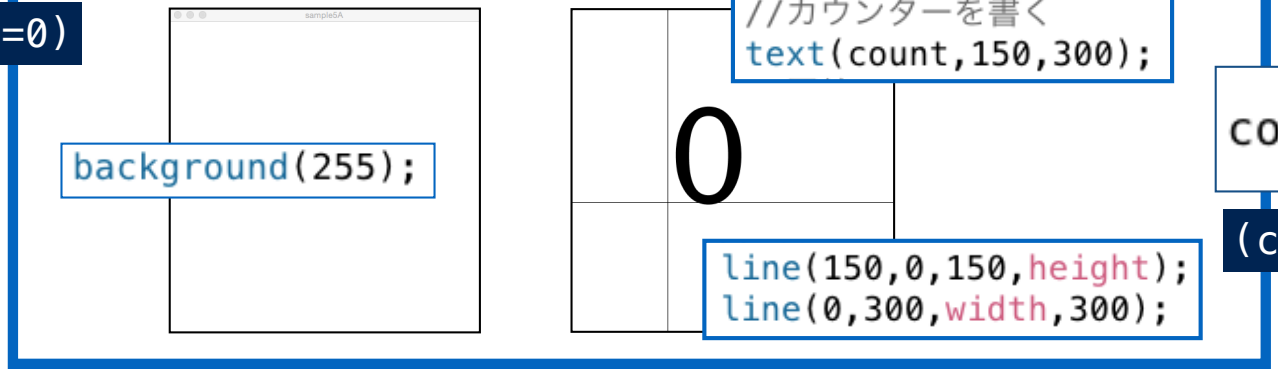
setup関数を実行後，定期的に行う処理

```
void draw(){
  background(255);  //白く塗りつぶす

  //カウンターを書く
  text(count,150,300);
  //罫線
  line(150,0,150,height);
  line(0,300,width,300);

  count++;  //カウンター
}
```

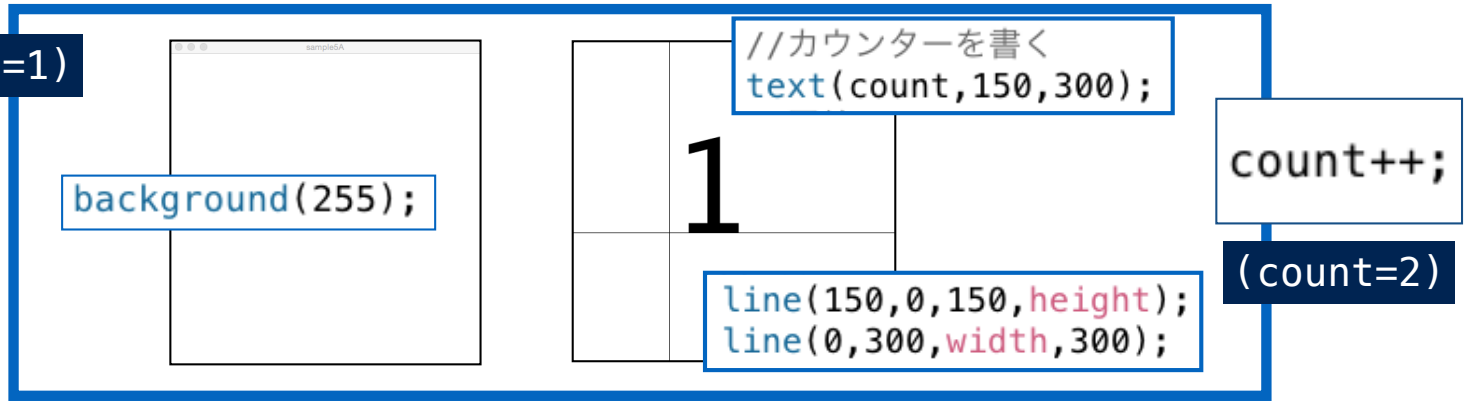
draw関数



(count=0)

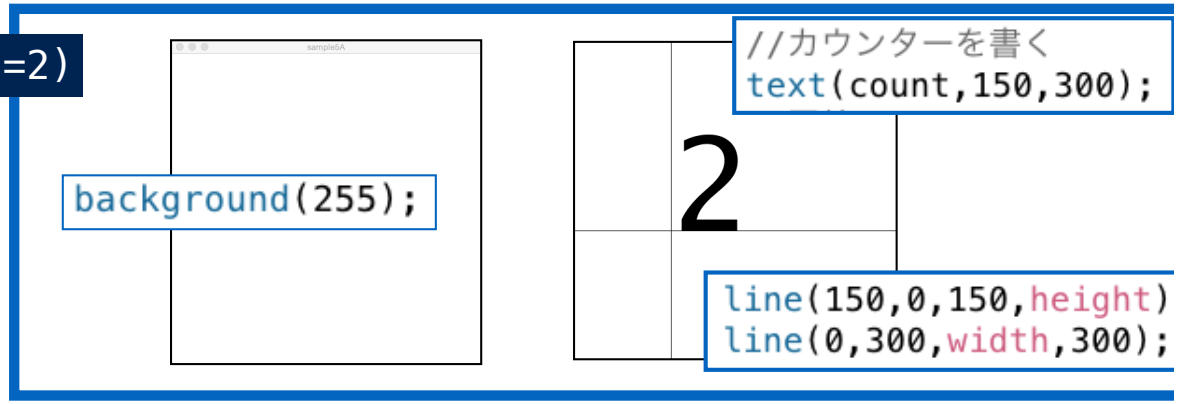


(count=1)




```
//定期的な処理
void draw(){
  background(255);   //白く塗りつぶす

  //カウンターを書く
  text(count,150,300);
  //罫線
  line(150,0,150,height);
  line(0,300,width,300);

  count++;   //カウンターを更新
}
```

(count=2)



Processingにあらかじめ用意されている関数

変更できない関数の例

返り値のない（計算結果のない）関数

```
void rect(x,y,w,h);
```

```
void line(x1,y1,x2,y2);
```

```
void fill(red,green,blue);
```

```
void stroke(red,green,blue);
```

返り値のある（計算結果のある）関数

```
float dist(x1,y1,x2,y2);
```

(x1,y1) と (x2,y2) の距離

```
float pow(x,n);
```

xのn乗

```
float sqrt(x);
```

xの平方根

変更することが前提とされている関数の例

以下の関数は, 初期状態では空白ですが, 独自に定義することによって, アニメーション・イベント処理を埋め込むことができます.
また, 「void」は, 返り値が無いことを意味します.

```
void setup();
```

プログラム起動時の処理

```
void draw();
```

setup関数終了後, 定期的に行う処理

```
void mousePressed();
```

マウスが押された時の処理

```
void mouseReleased();
```

マウスが離された時の処理

変更することが前提とされている関数の例

`void setup();`	`void draw();`
プログラム起動時の処理	setup関数終了後, 定期的に行う処理

`void mousePressed();`	`void mouseReleased();`
マウスが押された時の処理	マウスが離された時の処理

これらの関数は, 初期状態では空白ですが, 独自に定義することによって, アニメーション・イベント処理を埋め込むことができます.
また, 「void」は, 返り値が無いことを意味します.

関数を定義するための記法

p.256

```
void setup(){
}
```

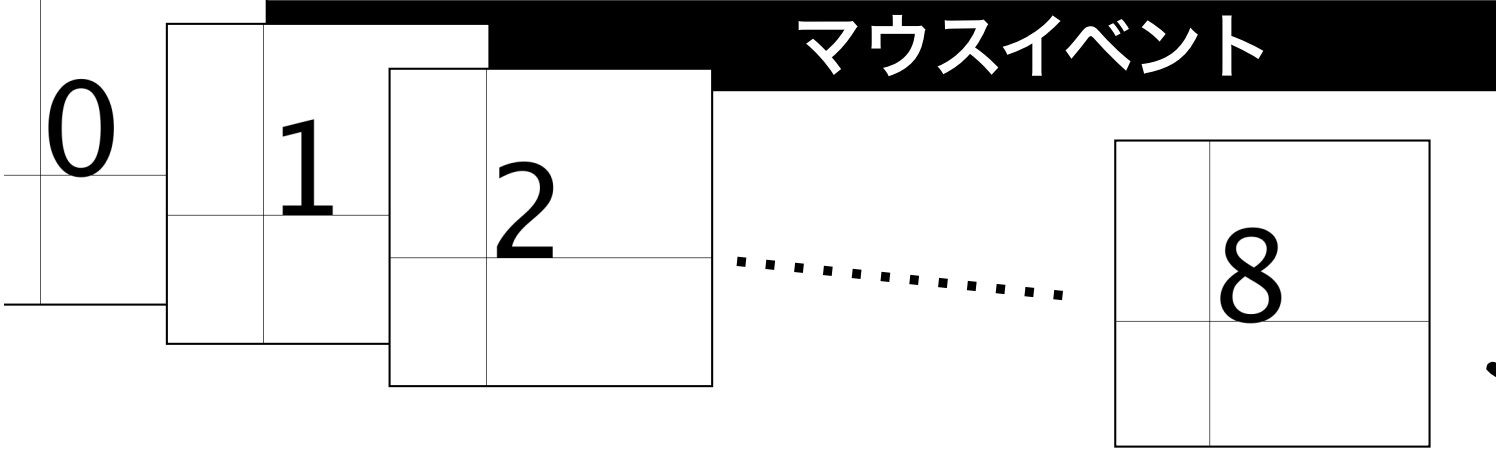
```
void draw(){
}
```

```
void mousePressed(){
}
```

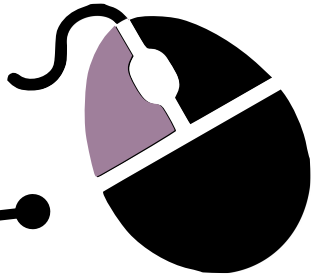
```
void mouseReleased(){
}
```

{ と } の間に, 必要な処理を書きます. 何も書かないということは, 何の処理もしないことを積極的に意味するので, エラーにはなりません.

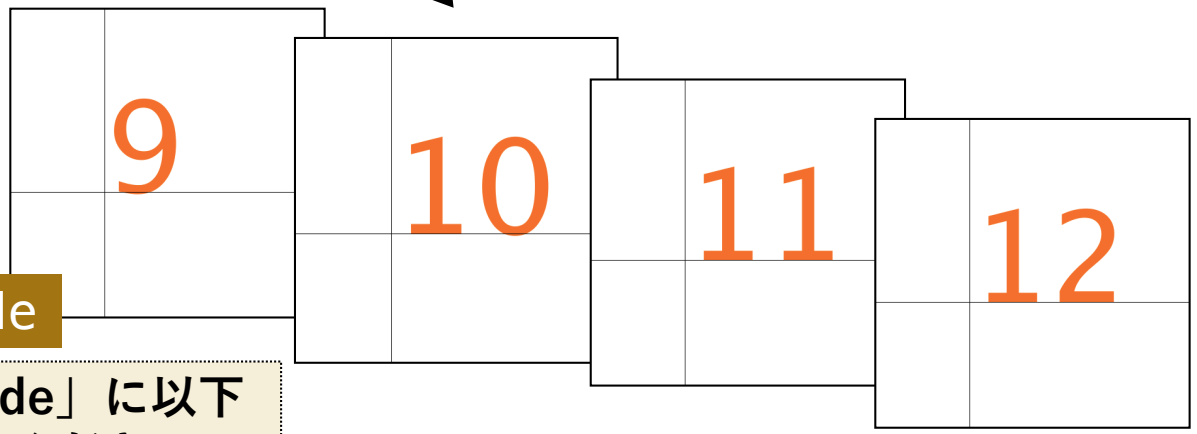
マウスイベント



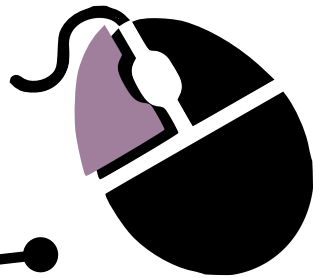
マウスをクリック



実行結果



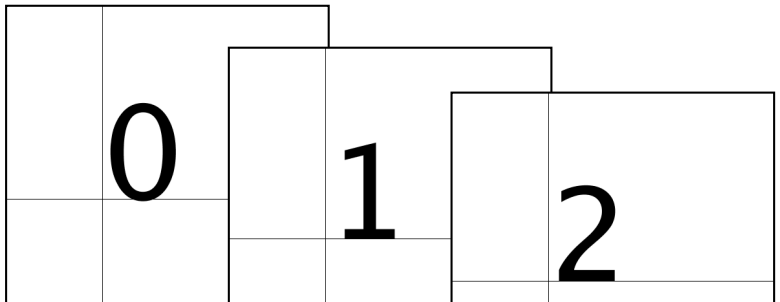
マウスをリリース



sample4A_2.pde

「sample4A_1.pde」に以下を書き加えてみてください.

```
//マウスを押された時の処理
void mousePressed(){
  fill(255,100,0); //色を変える
}

//マウスを離した時の処理
void mouseReleased(){
  fill(0);    //色を黒に戻す
  count = 0; //カウンターを戻す
}
```



グローバル変数とローカル変数

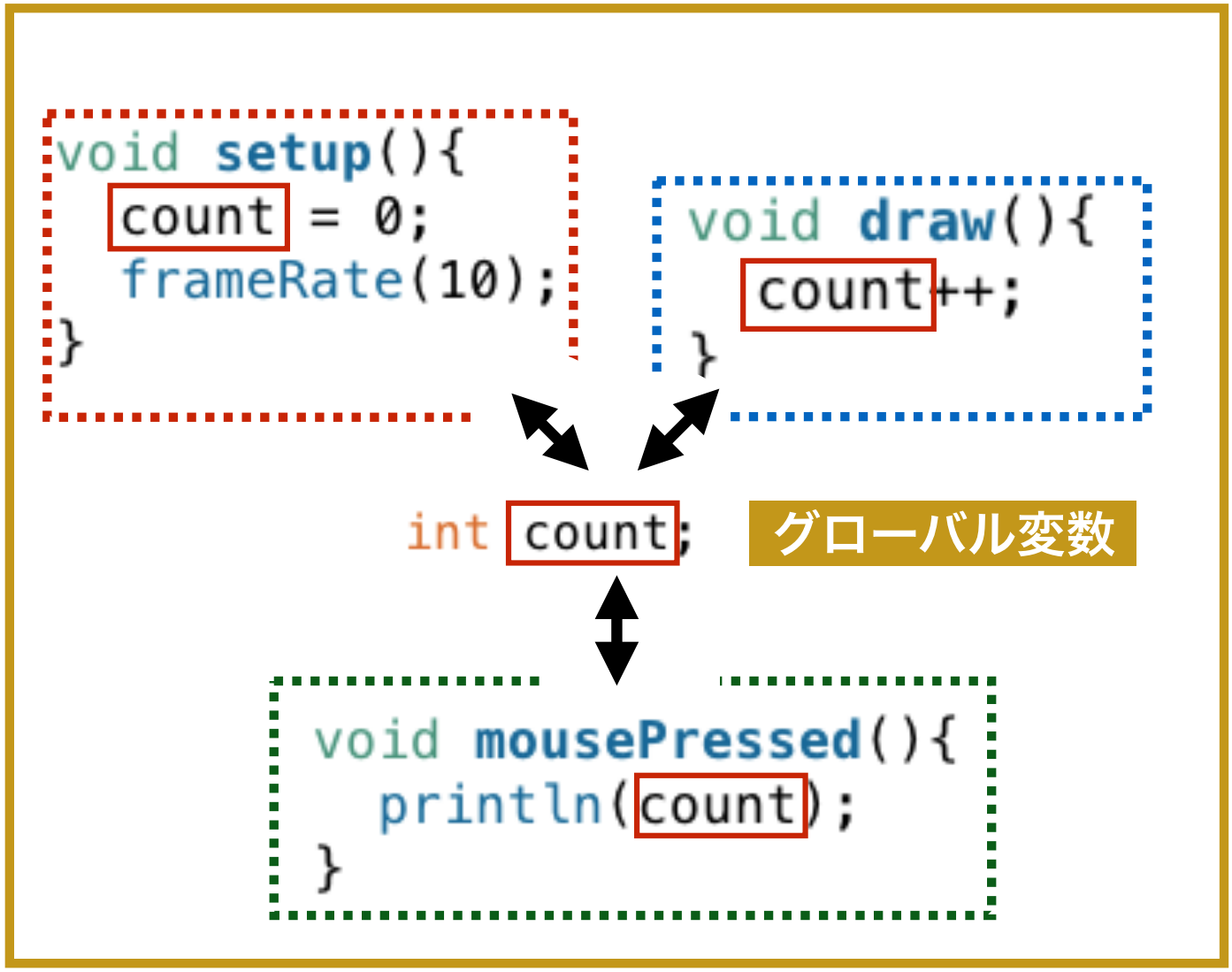
sample4A_3.pde

```
int count;

void setup(){
  count = 0;
  frameRate(10);
}

void draw(){
  count++;
}

void mousePressed(){
  println(count);
}
```

```
6
12
20
```

コンソール

COUNTは0から始まり、1秒間に10の割合で増えていきます.
マウスを押すと, 現在の数字が表示されます.



関数の定義の外側で宣言された変数は「グローバル変数」と呼ばれ, その値は, 全ての関数内で共有されます. 宣言と同時に値を代入することもできます.

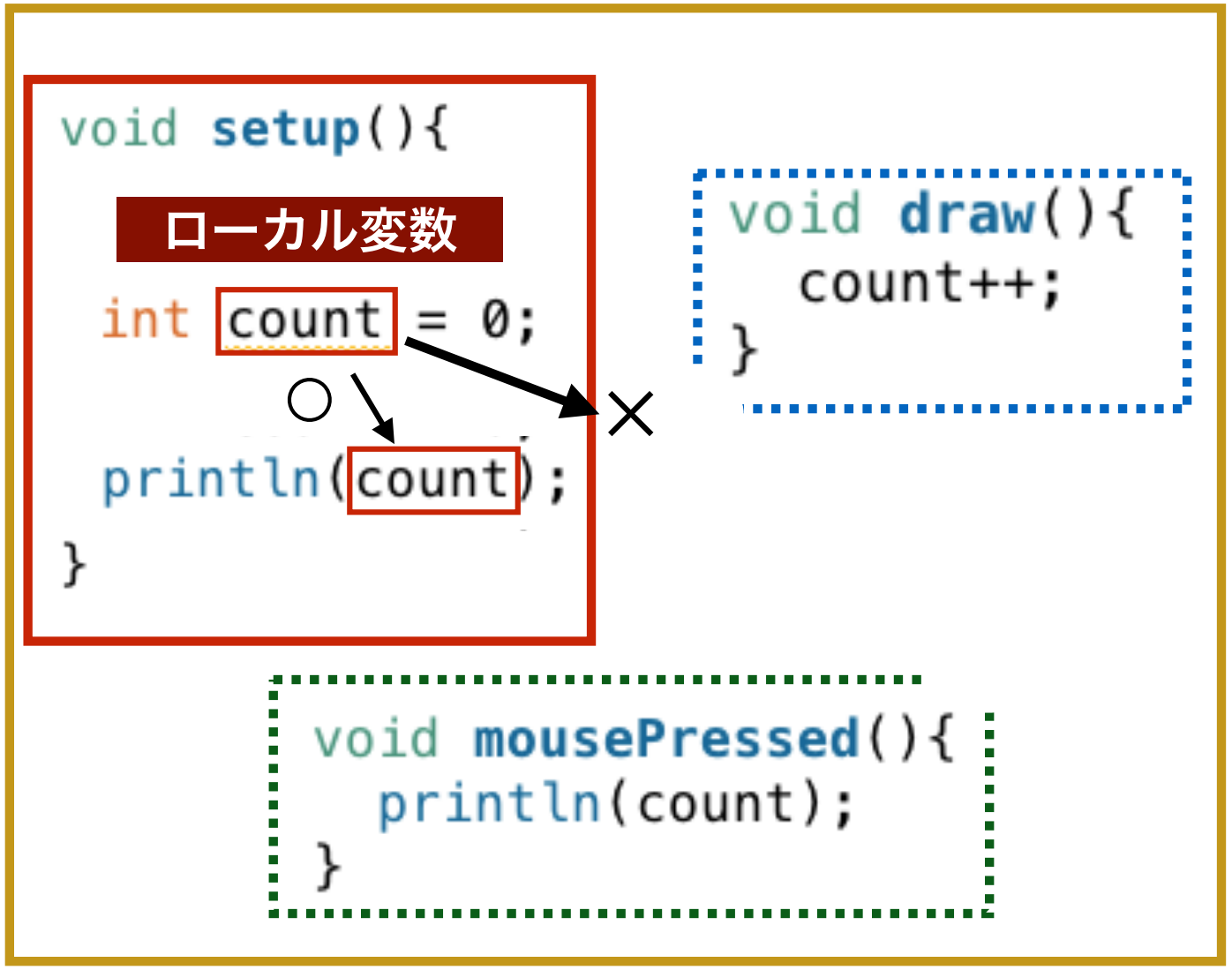
グローバル変数とローカル変数

sample4A_4.pde

```
void setup(){
  int count = 0;
  println(count);
}

void draw(){
  count++;
}

void mousePressed(){
  println(count);
}
```

文法エラー

文法エラー



特定の関数の内部で宣言された変数は，その関数の内部でのみ使用できます．（関数をまたいで）別の関数内で共有することはできません．

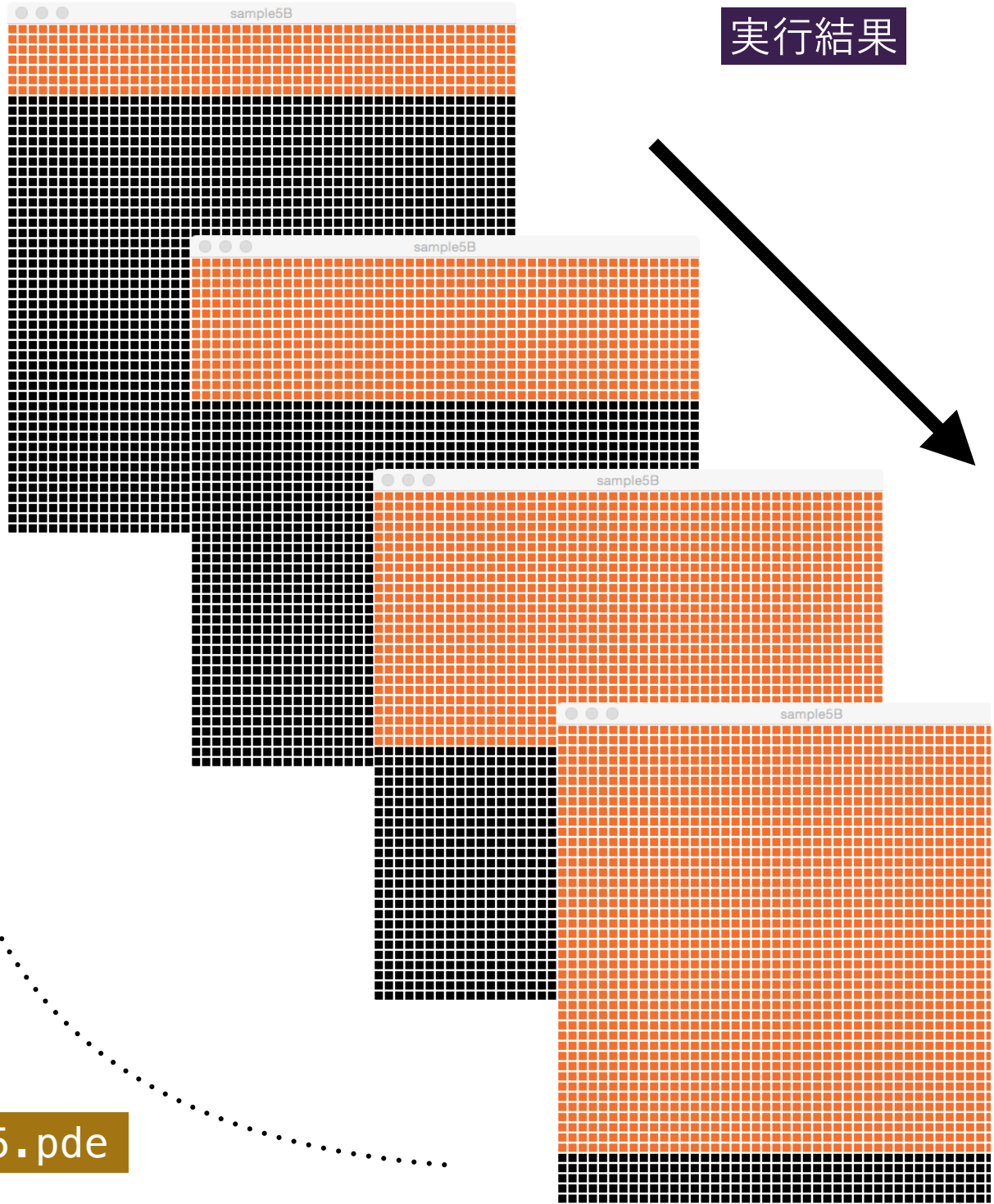
アニメーション

```
int count; //カウンター

void setup(){
  size(500,500); //サイズ
  //背景白、線なし
  background(255);  noStroke();
  //50fps
  frameRate(50); //50fps

  //カウンターを0に初期化
  count = 0;
}
```

```
void draw(){

  for(int i=0;i<50;i++){
    for(int j=0;j<50;j++){

      float x = 1 + 10*i;
      float y = 1 + 10*j;

      if(j<count%50){
        fill(255,100,0);
      }else{
        fill(0);
      }
      rect(x,y,8,8);
    }
  }
  count++;
}
```

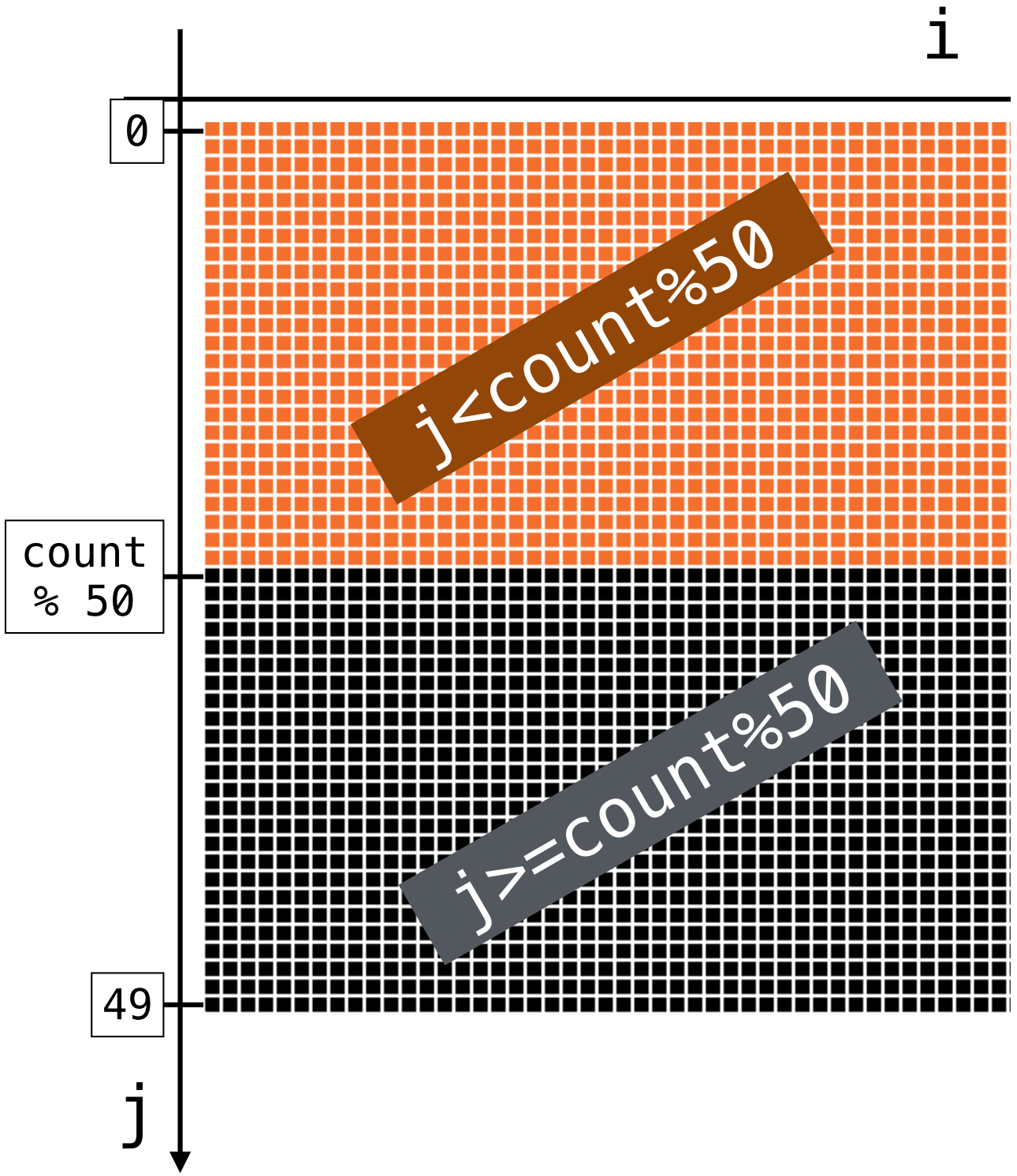
sample4_A5.pde

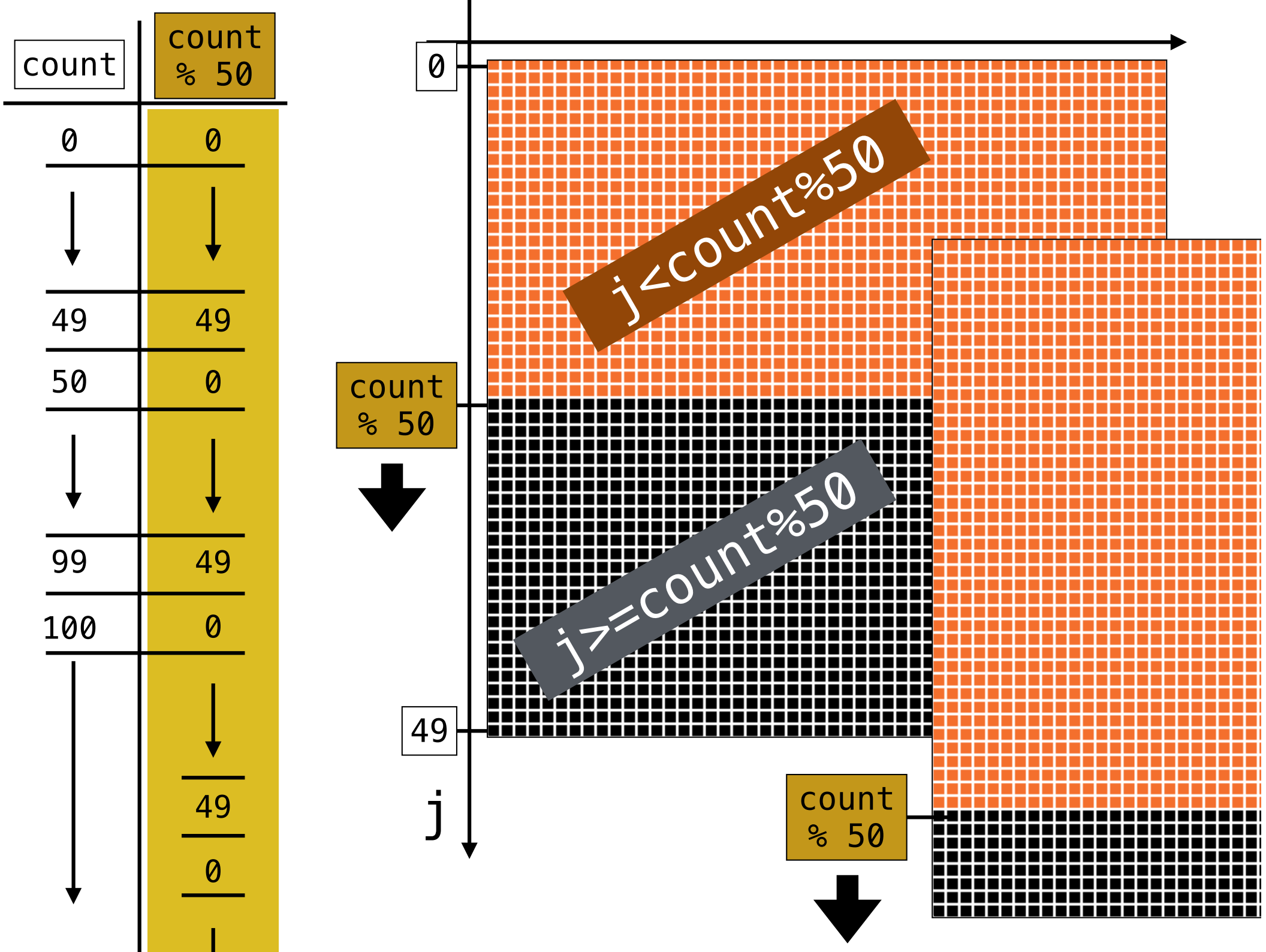
実行結果



```
14 void draw(){
15
16   for(int i=0;i<50;i++){
17     for(int j=0;j<50;j++){
18
19       float x = 1 + 10*i;
20       float y = 1 + 10*j;
21
22       if(j<count%50){
23         fill(255,100,0);
24       }else{
25         fill(0);
26       }
27       rect(x,y,8,8);
28     }
29   }
30   count++;
31 }
```

sample4A_5.pde



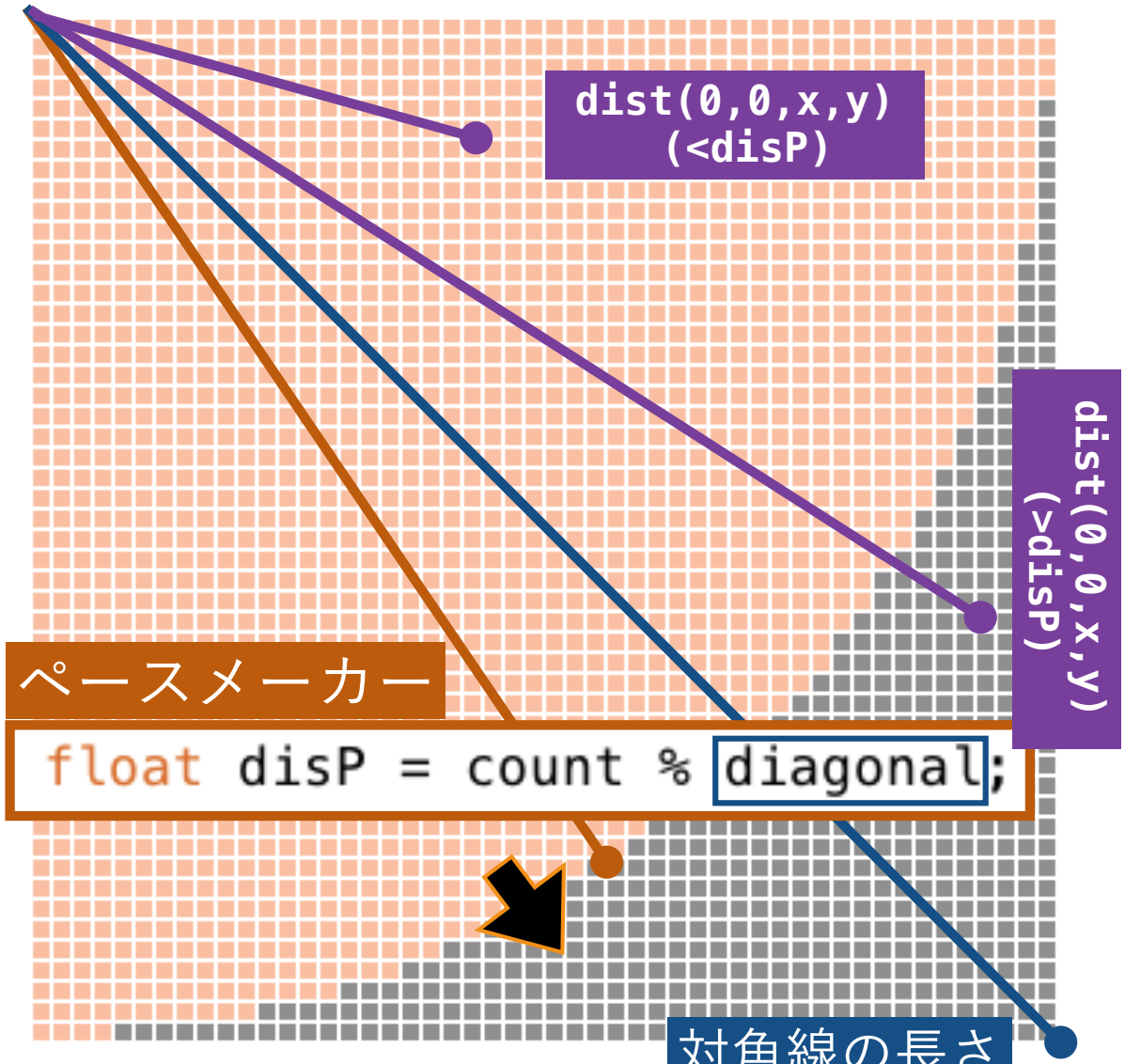

count
count % 50
0
0
49
49
50
0
99
49
100
0
49
0
0
49
count % 50
j<count%50
j>=count%50
49
j
count % 50

練習

```
int count; //カウンター
float diagonal; //対角線の長さ

void setup(){
  size(500,500); //サイズ
  //背景白、線なし
  background(255);  noStroke();
  //50fps
  frameRate(50);

  //対角線の距離を計算
  diagonal = dist(0,0,width,height);
  //カウンターを0に初期化
  count = 0;
}
```

```
void draw(){
  //ペースメーカー
  float disP = count % diagonal;

  for(int i=0;i<50;i++){
    for(int j=0;j<50;j++){

      float x = 1 + 10*i;
      float y = 1 + 10*j;

      if(dist(0,0,x,y)<disP){
        fill(255,100,0);
      }else{
        fill(0);
      }
      rect(x,y,8,8);
    }
  }
  count++;
}
```

sample4A_6.pde




```
diagonal = dist(0,0,width,height);
```

```
float dist(x1,y1,x2,y2);
```

(x1,y1)と**(x2,y2)** の距離を計算します.
sqrt((x2-x1)*(x2-x1)+(y2-y1)*(y2-y1))と同じです.

マウス座標の取得

mouseX　mouseY

マウスのX座標とY座標
（ウィンドウ領域内にある時のみ追尾）

```
int count; //カウンター
float cx = 0; float cy = 0; //円
boolean mode = false; //描画の有無

void setup(){
  size(500,500);
  background(255);  noStroke();
  frameRate(50);
  count = 0;
}
```

描画の有無の切り替え

```
void mousePressed(){
  cx = mouseX;  cy = mouseY;
  count = 0;
  mode = true;
}
```

マウスが押されたた場所を(cx,cy)とし，countを初期化し，描画モードとします．

```
void mouseReleased(){
  mode = false;
}
```

マウスが離された時に，描画モードをオフとします．

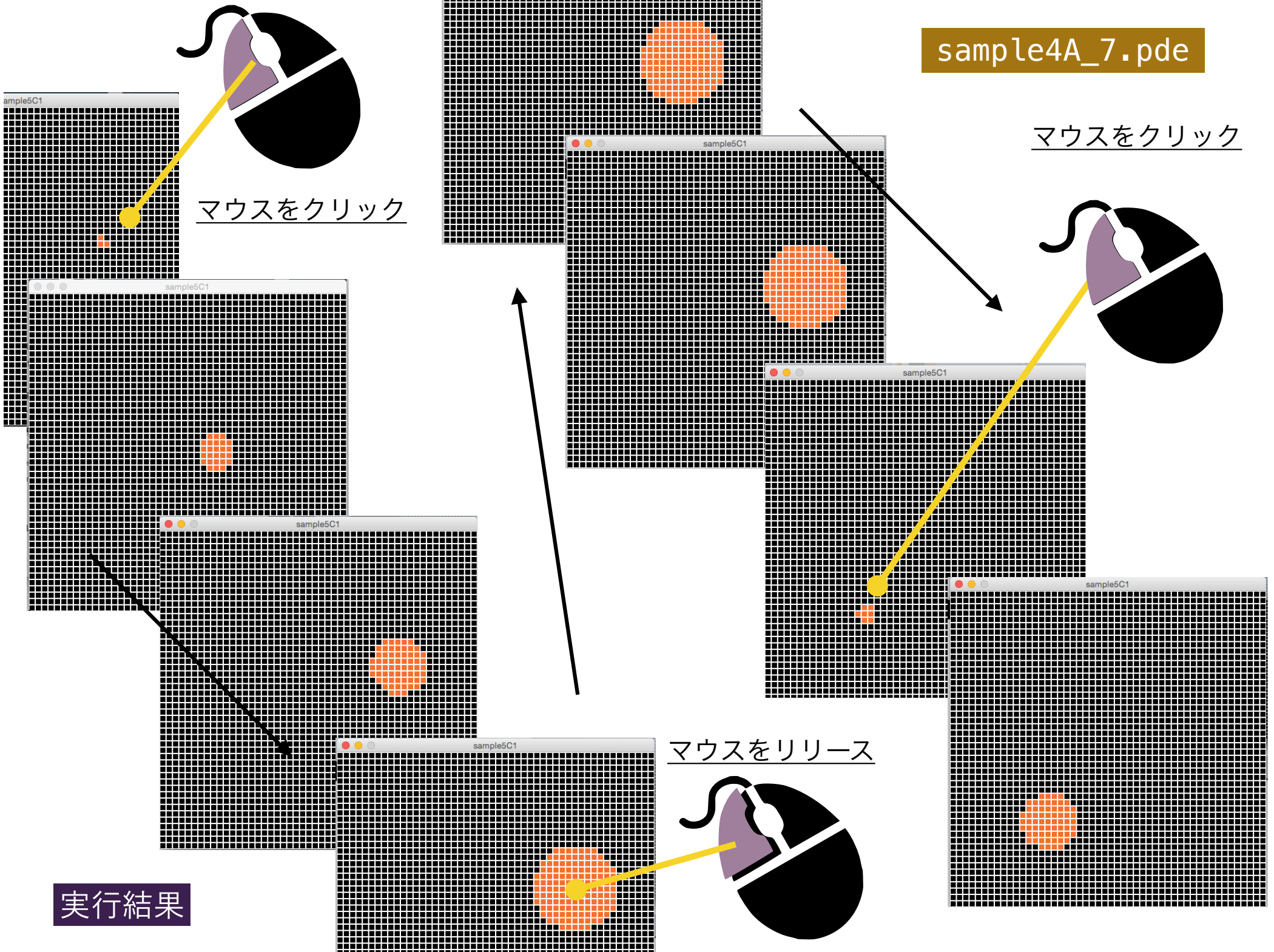
```
void draw(){

  for(int i=0;i<50;i++){
    for(int j=0;j<50;j++){

      float x = 1 + 10*i;
      float y = 1 + 10*j;

      if(dist(cx,cy,x,y)<count){
        fill(255,100,0);
      }else{
        fill(0);
      }
      rect(x,y,8,8);
    }
  }
  if(mode){
    count++;
  }
}
```

(cx,cy)からの距離がcountのときのみ色を橙色にそれ以外は黒とします．

modeがtrueのときのみ，countを増やします．

sample4_A7.pde

sample4A_7.pde



実行結果

練習

「sample4_A7.pde」の一部を修正して, マウスを離してもこれまでの描画が消えずに蓄積していくように変更してみてください.

sample4A_7X.pde