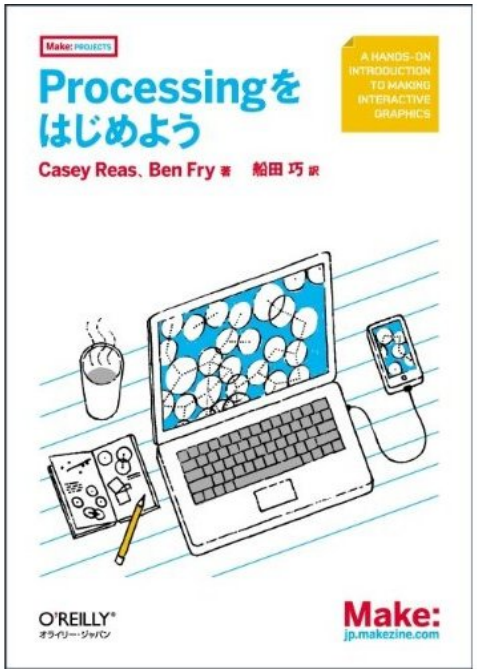
Practice # 2

文 法 の 基 礎

演習2A　繰り返し文（P45 - 51）



Processingをはじめよう
第二版
(Make: PROJECTS)

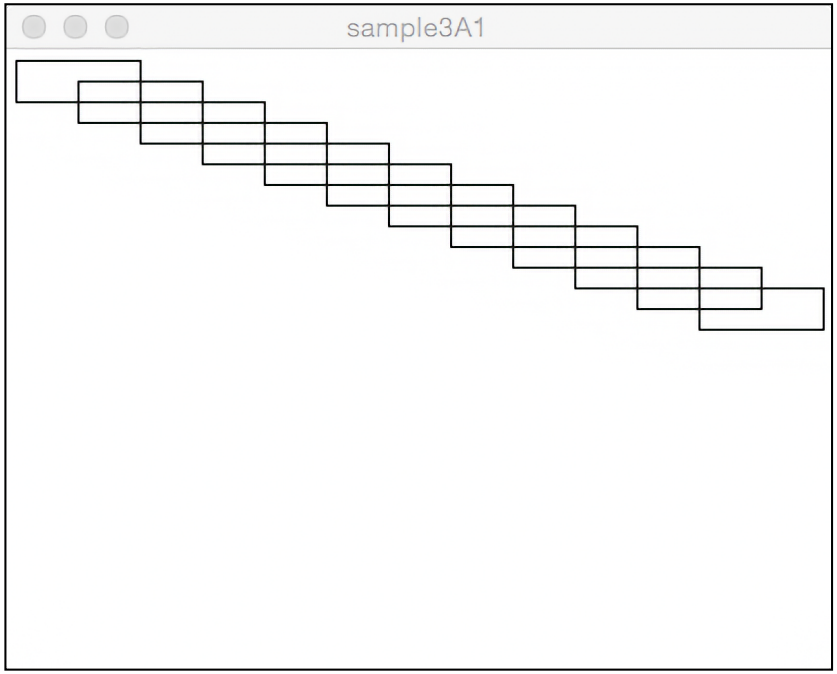
オンデマンド授業　5.08
課題学習　5.15

この本を参考書として使用します. 以下の資料のなかで、ページ数が書かれてあるものは, 教科書のページに対応しています. 自習復習, 辞書代わりに使用してください.

繰り返しの処理（繰り返し文を使わない）

```
1
2  size(400,300);  //サイズは任意
3
4  background(255); //背景白
5  stroke(0);       //線は黒
6  noFill();        //塗りつぶしなし
7
8  //四角形の左上頂点
9  int x = 5;  int y = 5;
10 //四角形の幅と高さ
11 int w = 60; int h = 20;
12
13 rect(x,y,w,h); //最初の長方形
14
15 //右下にずらしながら描画を繰り返す
16 x += w/2; y += h/2; rect(x,y,w,h);
17 x += w/2; y += h/2; rect(x,y,w,h);
18 x += w/2; y += h/2; rect(x,y,w,h);
19 x += w/2; y += h/2; rect(x,y,w,h);
20 x += w/2; y += h/2; rect(x,y,w,h);
21 x += w/2; y += h/2; rect(x,y,w,h);
22 x += w/2; y += h/2; rect(x,y,w,h);
23 x += w/2; y += h/2; rect(x,y,w,h);
24 x += w/2; y += h/2; rect(x,y,w,h);
25 x += w/2; y += h/2; rect(x,y,w,h);
26 x += w/2; y += h/2; rect(x,y,w,h);
27
```

sample2A_1.pde

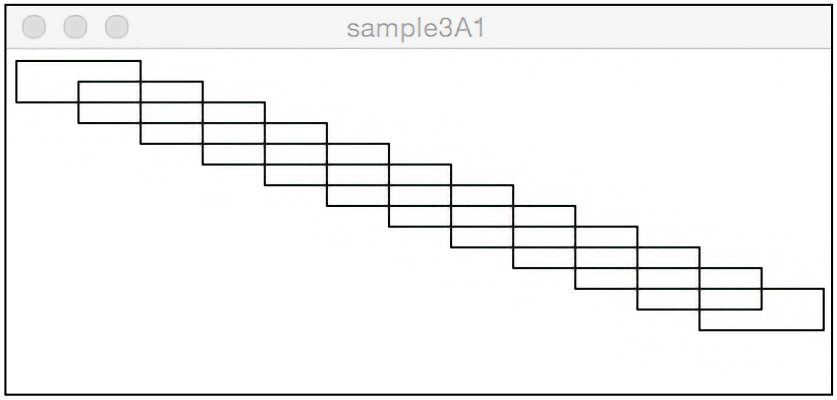


左のコードを作成してください. 長いですが, コピペを多用すれば, 比較的すぐ終わります.

繰り返しの処理（繰り返し文を使わない）

```
size(400,300);  //サイズは任意

background(255); //背景白
stroke(0);       //線は黒
noFill();        //塗りつぶしなし

//四角形の左上頂点
int x = 5;  int y = 5;
//四角形の幅と高さ
int w = 60; int h = 20;

rect(x,y,w,h); //最初の長方形

//右下にずらしながら描画を繰り返す
x += w/2; y += h/2; rect(x,y,w,h);
x += w/2; y += h/2; rect(x,y,w,h);
x += w/2; y += h/2; rect(x,y,w,h);
x += w/2; y += h/2; rect(x,y,w,h);
x += w/2; y += h/2; rect(x,y,w,h);
x += w/2; y += h/2; rect(x,y,w,h);
x += w/2; y += h/2; rect(x,y,w,h);
x += w/2; y += h/2; rect(x,y,w,h);
x += w/2; y += h/2; rect(x,y,w,h);
x += w/2; y += h/2; rect(x,y,w,h);
x += w/2; y += h/2; rect(x,y,w,h);
```

sample2A_1.pde



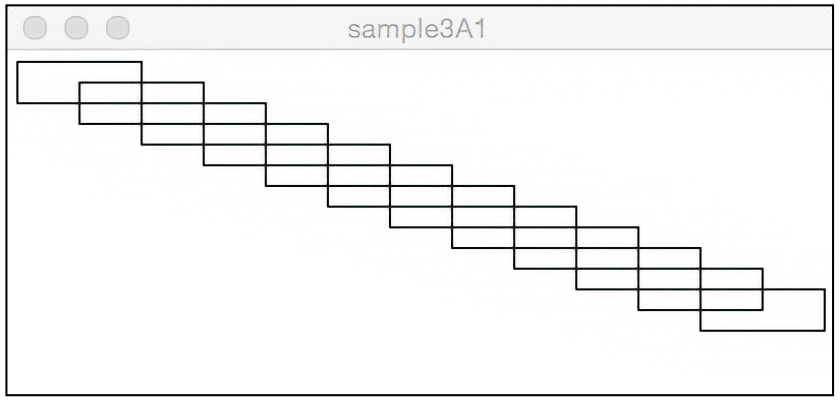
繰り返し文を使うと，11行がたったの3行に圧縮できました．

```
for(int i=0;i<11;i++){
  x += w/2; y += h/2; rect(x,y,w,h);
}
```

繰り返しの処理（繰り返し文を使わない）

```
//右下にずらしながら描画を繰り返す
rect(x+0*w/2,y+0*h/2,w,h);
rect(x+1*w/2,y+1*h/2,w,h);
rect(x+2*w/2,y+2*h/2,w,h);
rect(x+3*w/2,y+3*h/2,w,h);
rect(x+4*w/2,y+4*h/2,w,h);
rect(x+5*w/2,y+5*h/2,w,h);
rect(x+6*w/2,y+6*h/2,w,h);
rect(x+7*w/2,y+7*h/2,w,h);
rect(x+8*w/2,y+8*h/2,w,h);
rect(x+9*w/2,y+9*h/2,w,h);
rect(x+10*w/2,y+10*h/2,w,h);
rect(x+11*w/2,y+11*h/2,w,h);
```

sample2A_1.pde



数式を使うと，左のように書き換えることもできます（3A1の13行目以降を置き換え）．
これを繰り返し文に書き換えると，以下のようになります．

```
for(int i=0;i<=11;i++){
  rect(x+i*w/2,y+i*h/2,w,h);
}
```

for文の使い方

```
for(int i=0;i<=11;i++){
  rect(x+i*w/2,y+i*h/2,w,h);
}
```

sample2A_2.pde

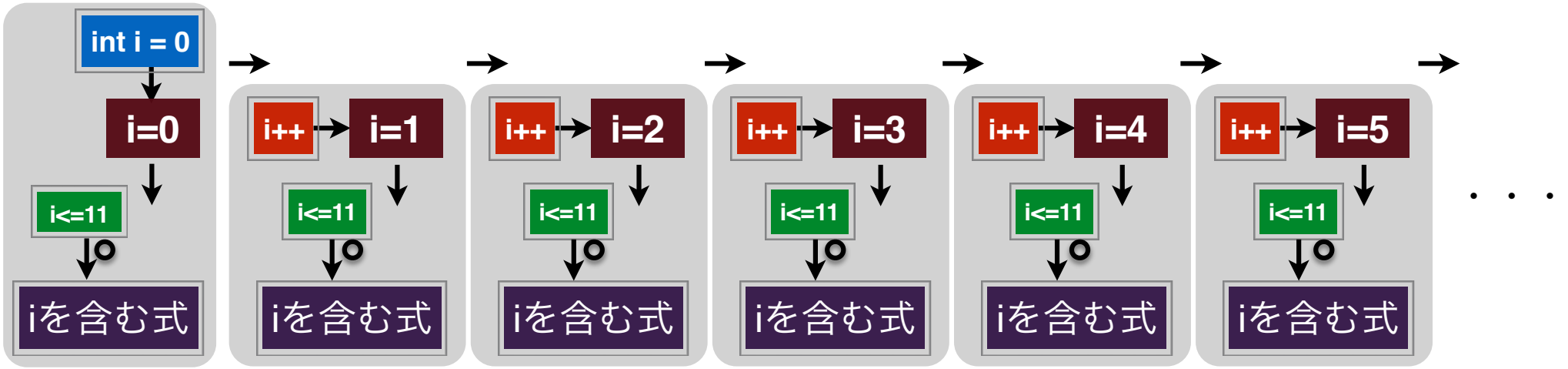
```
for( int i = 0 ; i<=11 ; i++ ){
    rect(x+i*w/2,y+i*h/2,w,h);
}
```

for文の使い方

`int i = 0`	1. 変数の初期化（初期値の設定）
`i<=11`	2. 繰り返しを持続する条件
`i++`	3. 繰り返し毎の変数の増減

```
for( int i = 0 ; i<=11 ; i++ ){
    rect(x+i*w/2,y+i*h/2,w,h);
}
```

繰り返しの対象となる式 (複数可)



for文の使い方

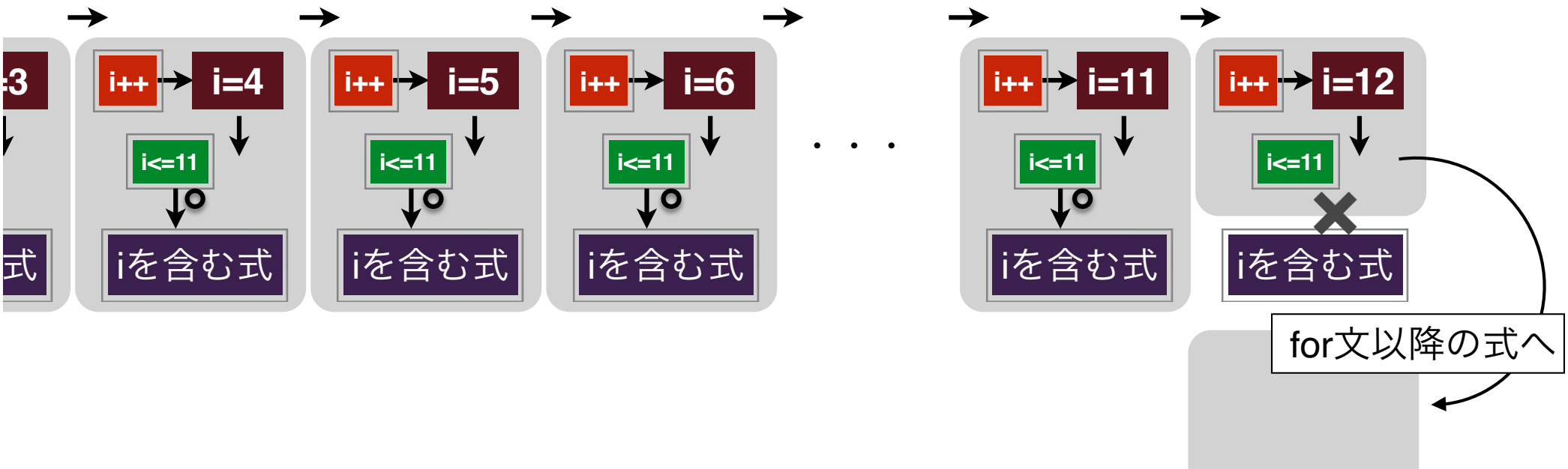
`int i = 0` 1. 変数の初期化（初期値の設定）

`i<=11` 2. 繰り返しを持続する条件

`i++` 3. 繰り返し毎の変数の増減

```
for( int i = 0 ; i<=11 ; i++ ){
    rect(x+i*w/2,y+i*h/2,w,h);
}
```

繰り返しの対象となる式 (複数可)



for文の使い方

`int i = 0` 1. 変数の初期化（初期値の設定）

- 役割的には, 繰り返し文の中で使用する（多くの場合, 何回目の繰り返しなのかを記憶するための）変数の初期値を決定するセクション.
- 既に宣言済み（箱が用意されている）の変数を使う場合は, 再度初期化することはできない（エラーになります）.

○

```
for(int i=0;i<=11;i++){
    ......
}
```

○

```
int i;
for(i=0;i<=11;i++){
    ......
}
```

×

```
int i;
for(int i=1;i<=20;i++){
    ......
}
```

- 変数としては, i, a, p, xがよく用いられる.

```
for( int i = 0 ; i<=11 ; i++ ){
```

for文の使い方

`i<=11` 2. 繰り返しを持続する条件

- 不等式は, true（真）かfalse（偽）を返す式.

関係演算子	trueを返す条件
a==b	aとbが等しい
a!=b	aとbが等しくない
a>b	aよりbが大きい
a>=b	aがbと等しいか, より大きい
a<b	aがbより小さい.
a<=b	aがbと等しいか, より小さい

p.46–47

```
for( int i = 0 ; i<=11 ; i++ ){
```

i<=11

2. 繰り返しを持続する条件

- 不等式は, true（真）かfalse（偽）を返す式.

関係演算子	trueを返す条件
a==b	aとbが等しい
a!=b	aとbが等しくない
a>b	aよりbが大きい
a>=b	aがbと等しいか, より大きい
a<b	aがbより小さい.
a<=b	aがbと等しいか, より小さい

p.46–47

```
int x, y;
boolean b;
```

```
x = 5;
b = (x>5);
```

```
x = 3;
b = (x!=3);
```

```
x = 10; y=10;
b = (x==y);
```

```
x = 0.1; y = 0.2;
b = (x-y>0.0);
```

```
println("b = " + b);
```

b = false

b = false

b = true

b = false

for文の使い方

i++

3. 繰り返し毎の変数の増減

- インクリメント式・デクリメント式がよく使われる.

```
for(int i=1;i<=20;i++)
{
   ......
}
```

```
for(int i=1;i<=20;i+=1)
{
   ......
}
```

```
for(int i=1;i<=20;i=i+1)
{
   ......
}
```

i=1, 2, 3, 4, 5, ...19, 20

```
for(int i=20;i>=1;i--)
{
   ......
}
```

```
for(int i=20;i>=1;i-=1)
{
   ......
}
```

```
for(int i=20;i>=1;i=i-1)
{
   ......
}
```

i=20, 19, 18, 17, 16, ...2, 1

```
for(int i=1;i<=20;i+=2)
{
   ......
}
```

i=1, 3, 5, 7, 9, ...17, 19

```
for(int i=20;i>=1;i-=2)
{
   ......
}
```

i=20, 18, 16, ...4, 2

```
for(int i=1;i<=20;i*=2)
{
   ......
}
```

i=1, 2, 4, 8, 16

p.44

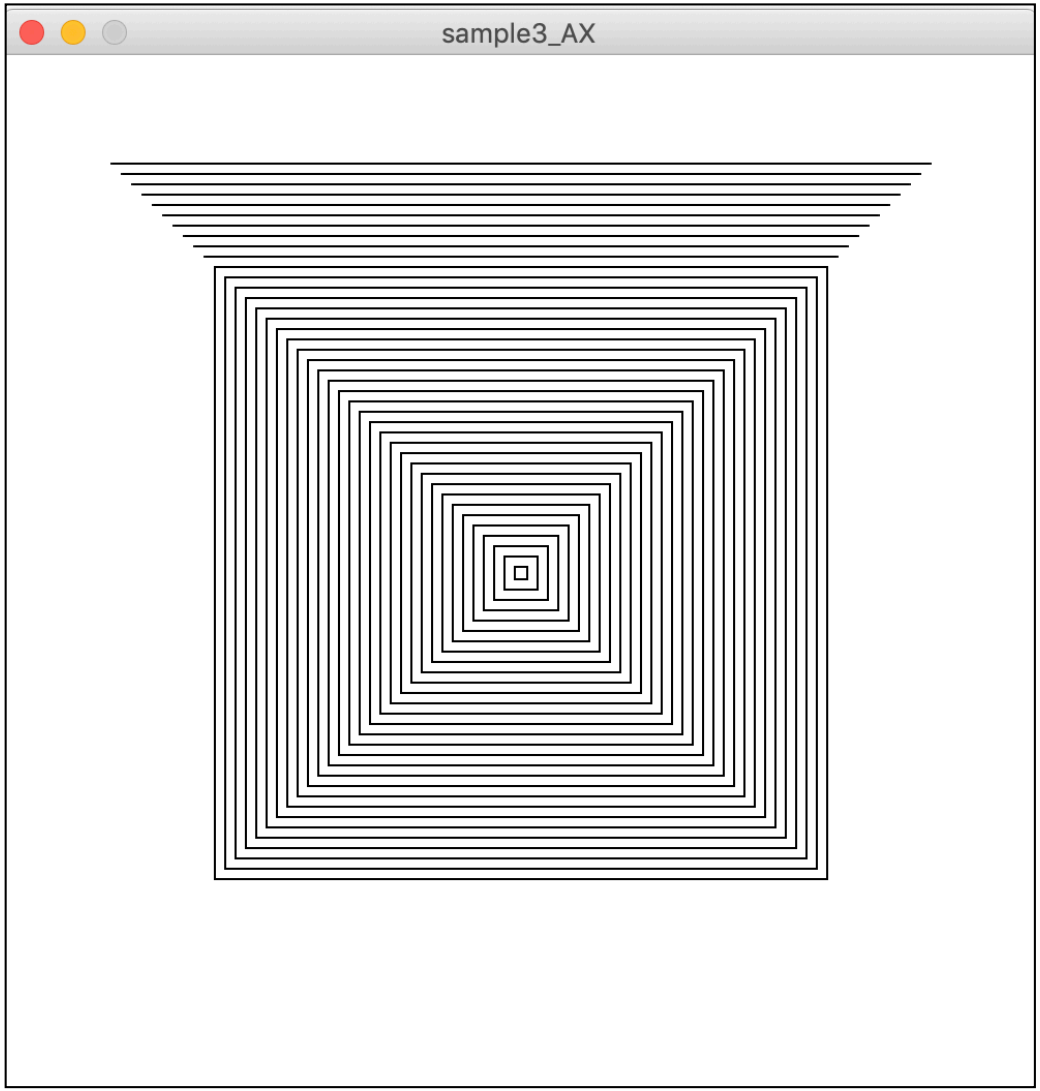
```
for( int i = 0 ; i<=11 ; i++ ){
```

小課題

```
size(500,500);

background(255); //背景白
stroke(0);       //線は黒
noFill();        //塗りつぶしなし

//四角形の一辺の長さの初期値
int r = 6;

//四角形の左上の座標（x,y）の初期値
float x = 0.5 * width - 0.5 * r;
float y = 0.5 * height - 0.5 * r;

for(int i=0;i<30;i++){
  rect(x,y,r,r);
  r += 10;
  x -= 5;
  y -= 5;
}

for(int i=0;i<10;i++){
  [          ]
  r += 10;
  x -= 5;
  y -= 5;

}
```

sample2A_X.pde

for文を使って, 以下のような図形を描画してみてください.



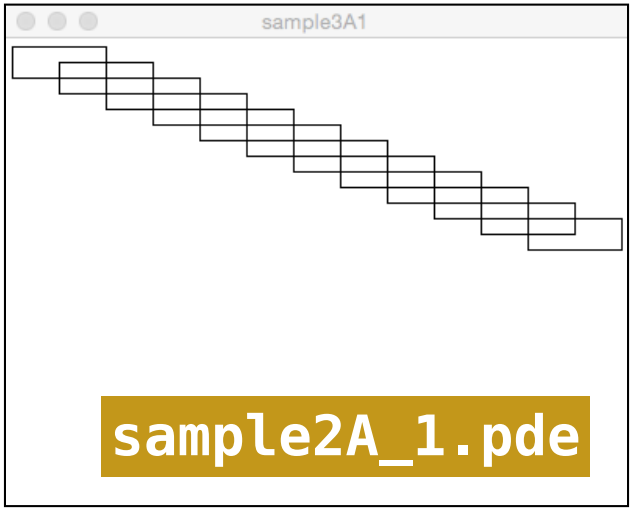
while文

```
size(700,300);  //サイズは任意

background(255); //背景白
stroke(0);       //線は黒
noFill();        //塗りつぶしなし

//四角形の左上頂点
int x = 5;  int y = 5;
//四角形の幅と高さ
int w = 60; int h = 20;

//四角形をずらしながら繰り返し描画する
for(int i=0;i<12;i++){
  rect(x,y,w,h);
  x += w/2;  y += h/2;
}
```

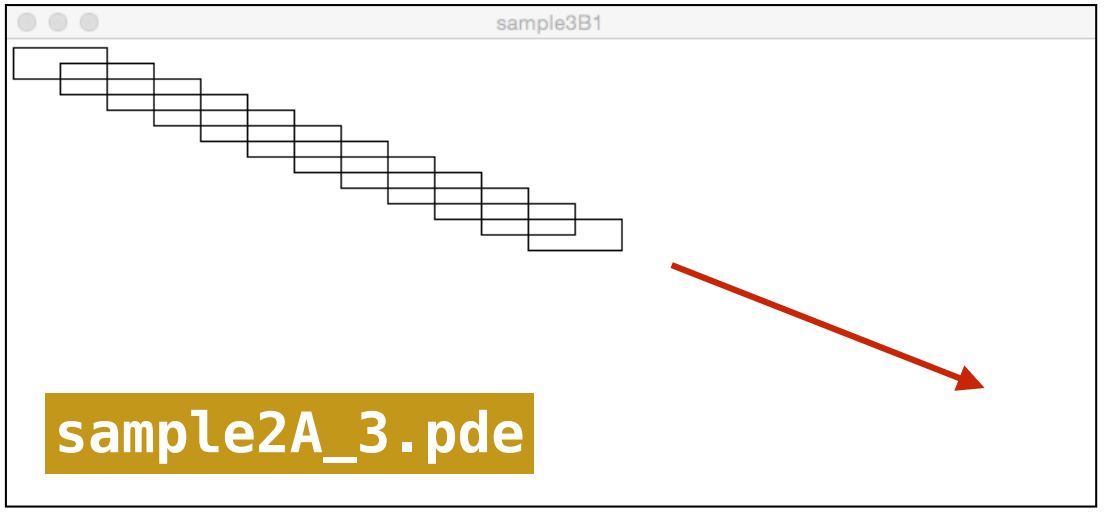
A1を, A3にコピーし, ウィンドウサイズの横幅を広げて実行してみよう.

繰り返し文の中身の順序が少し変わっています.

ウィンドウサイズに関わらず, ウィンドウサイズの右端ギリギリまで四角形を並べたい. どうすればよいか？



sample2A_1.pde

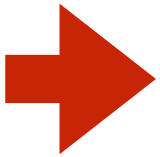


sample2A_3.pde

while文

以下のようにwhile文を使って書き換えてみましょう.

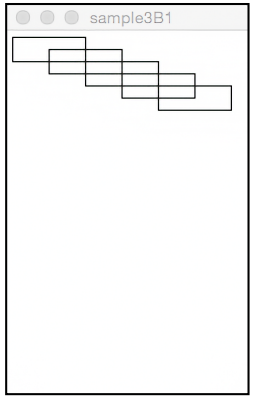
```
for(int i=0;i<12;i++){
  rect(x,y,w,h);
  x += w/2;  y += h/2;
}
```



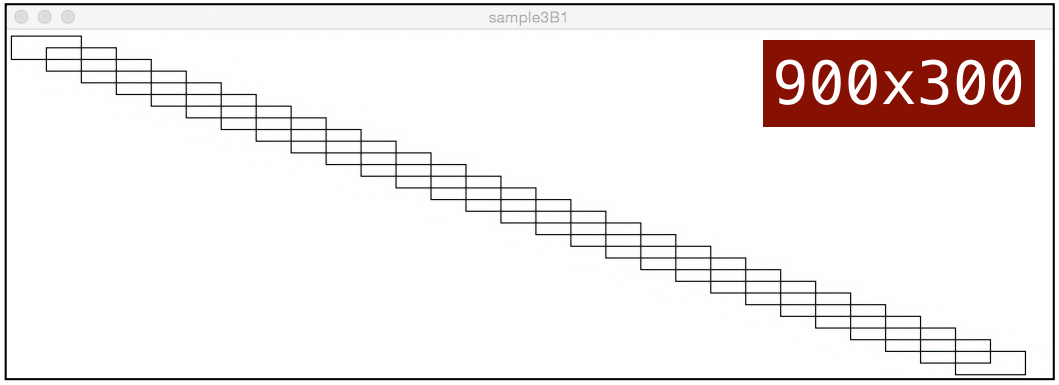
```
while(x+w<width){
  rect(x,y,w,h);
  x += w/2;  y += h/2;
}
```

どのようにサイズを変えても, ウィンドウの右端ギリギリのところで繰り返しが止まるようになりました.

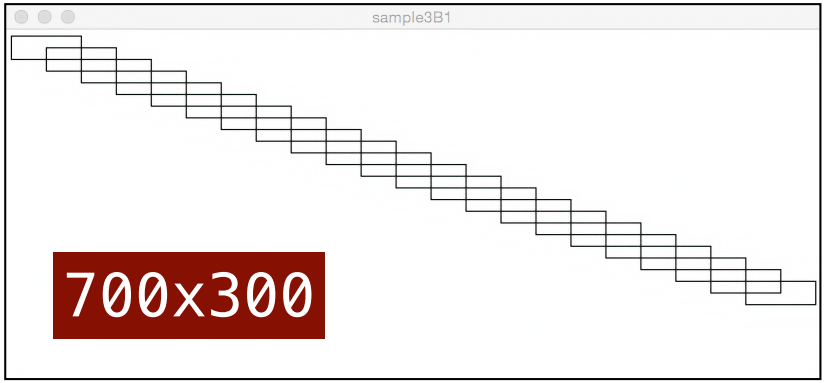
sample3A_3.pde



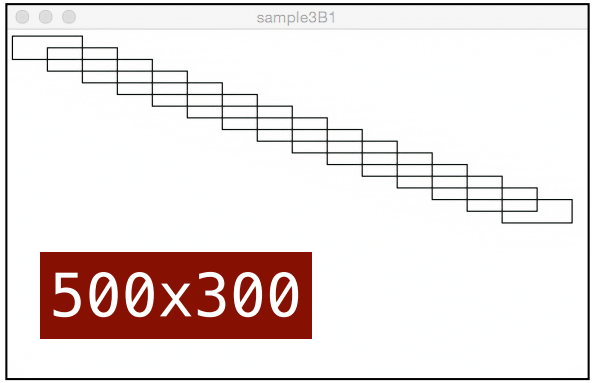
200x300



900x300



700x300



500x300

while文

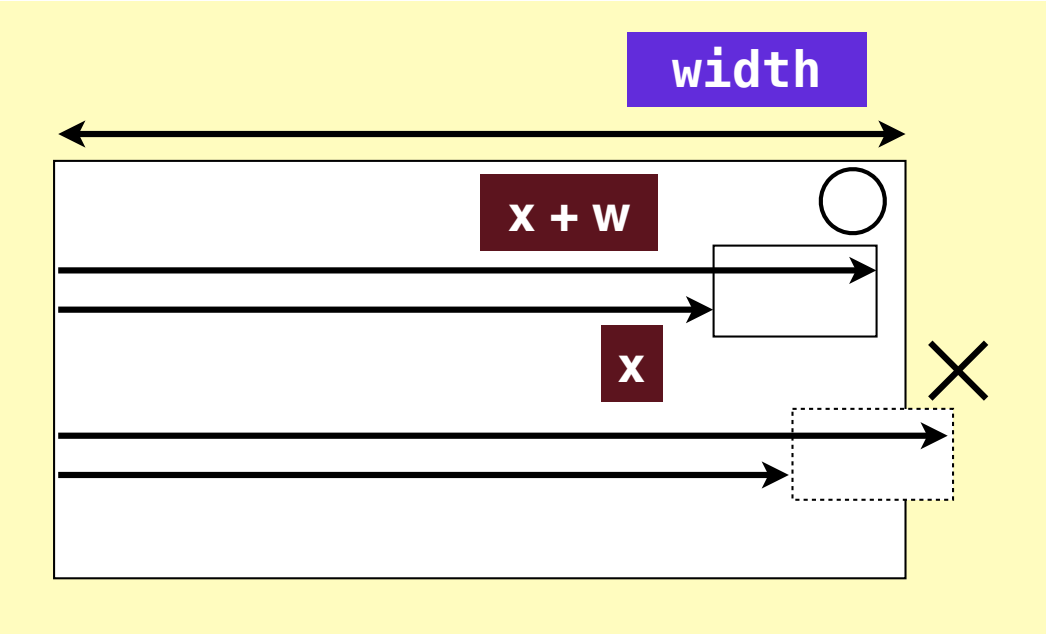
```
while( 繰り返しのための条件式 ){
    文A;
    文B;
    ...
}
```

p.237

繰り返しのための条件式
を満足する限り，
文A; 文B; ...
を繰り返し実行する.

「x+w」は，rect(x,y,w,h)実行時における，四角形の右辺のX座標に対応します．これが，ウィンドウの幅（width）を超える場合に，繰り返し文から脱出します．

```
while(x+w<width){
   rect(x,y,w,h);
   x += w/2;   y += h/2;
}
```



小課題

下に示したような図形を描くコードを, while文を使って書いてください. ただし, 円の半径（r）は下にずれるにつれて, 3ピクセルずつ大きくなります.

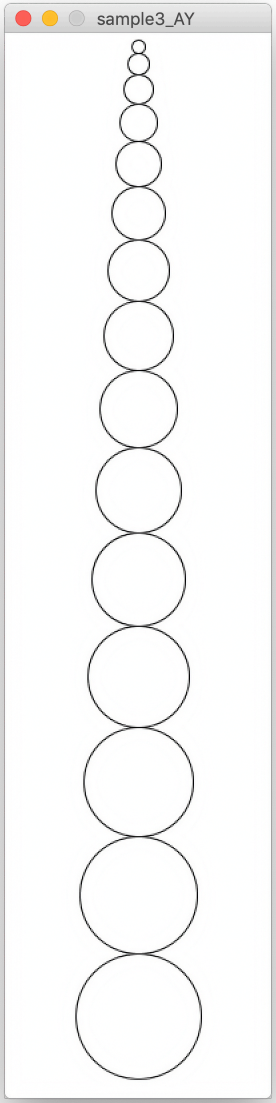
ここは自由に変えてください.

sample2A_Y.pde

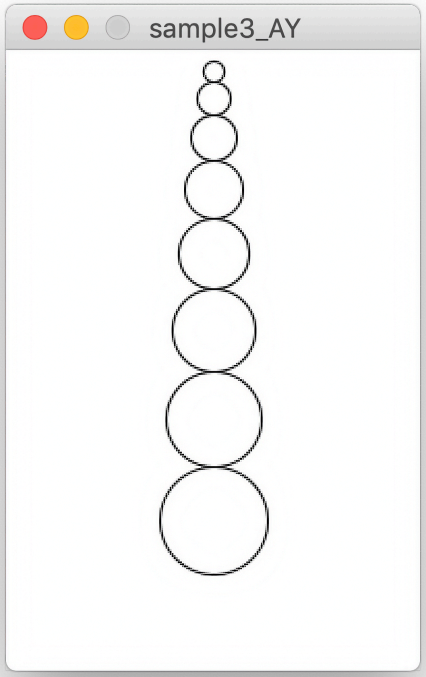
```
size(200,300);

background(255); //背景白
stroke(0);       //線は黒
noFill();        //塗りつぶしなし

int x = 100; //円中心のX座標
int y = 10;  //円中心のY座標
int r = 5;   //円の半径

while(                    ){

  ellipse(x,y,2*r,2*r);

                          }
```

この二箇所に, 適切な条件式・文を追加してください.

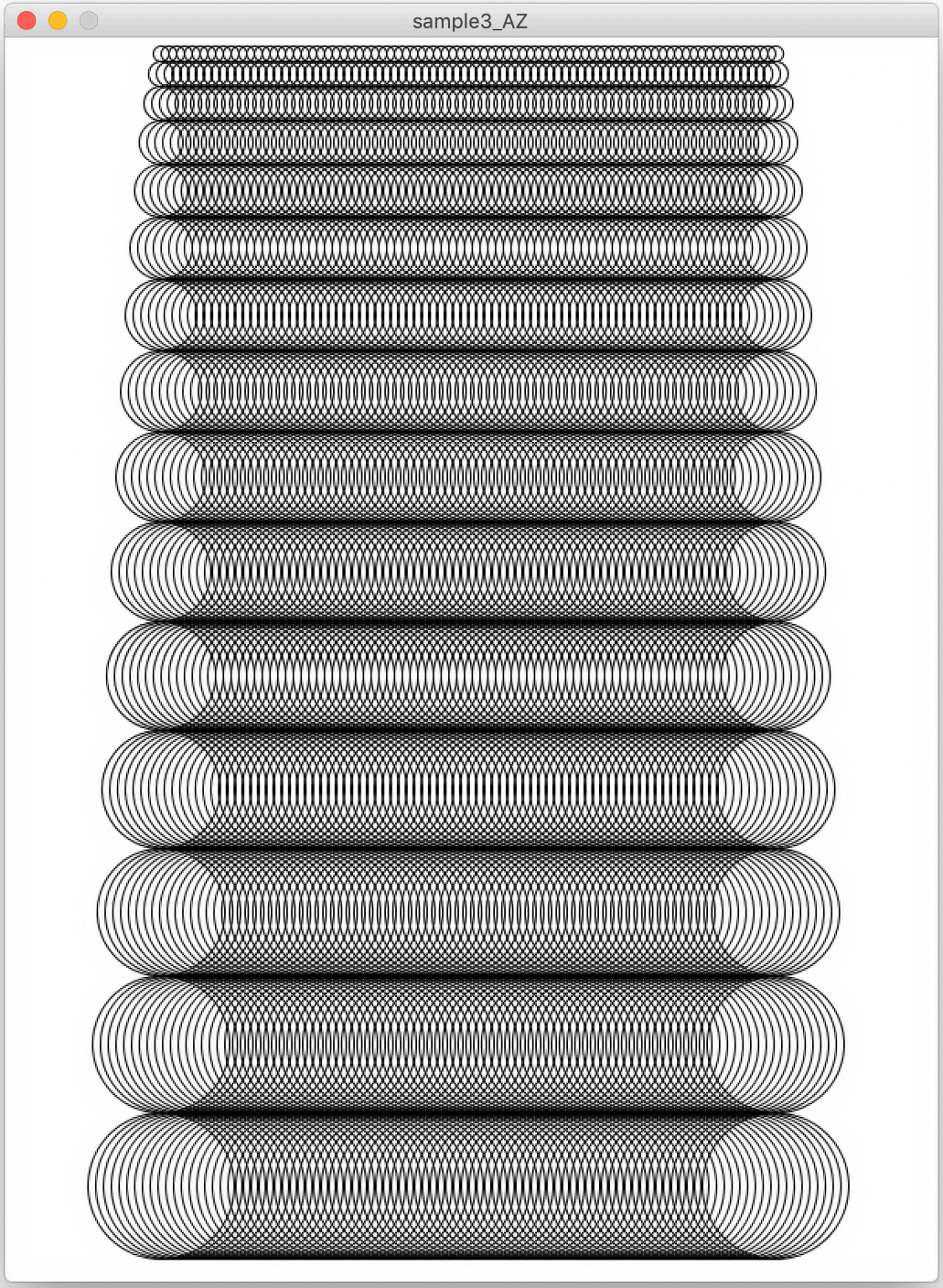
200x800



200x300



小課題



AYでつくった「塔」を，さらに横に100個並べるプログラムを作成してください．

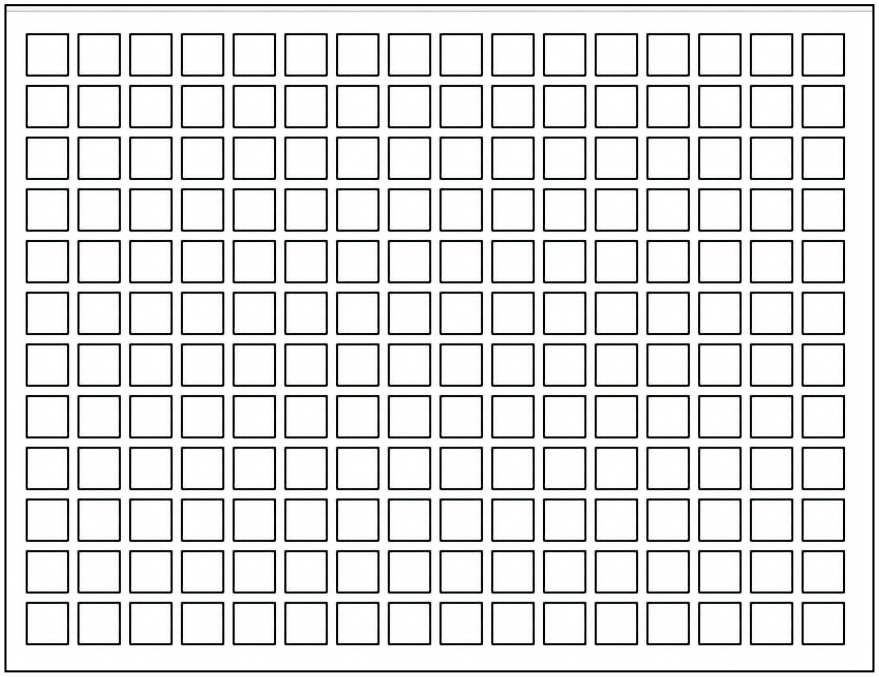
sample2A_Z.pde

FOR文のネスト（入れ子）

```
1  size(420,700);  //サイズ
2
3  background(255); //背景白
4  stroke(0);       //線は黒
5  noFill();        //塗りつぶしなし
6
7  //X座標の左端, 上端
8  int ix = 10; int iy = 10;
9  //正方形の一辺の長さ, 正方形の間隔
10 int w = 20; int space = w+5;
11 //正方形の左上頂点
12 int x,y;
13
14 for(int i=0;i<16;i++){
15   x = ix + i*space;
16   for(int j=0;j<12;j++){
17     y = iy + j*space;
18     rect(x,y,w,w);
19   }
20 }
```

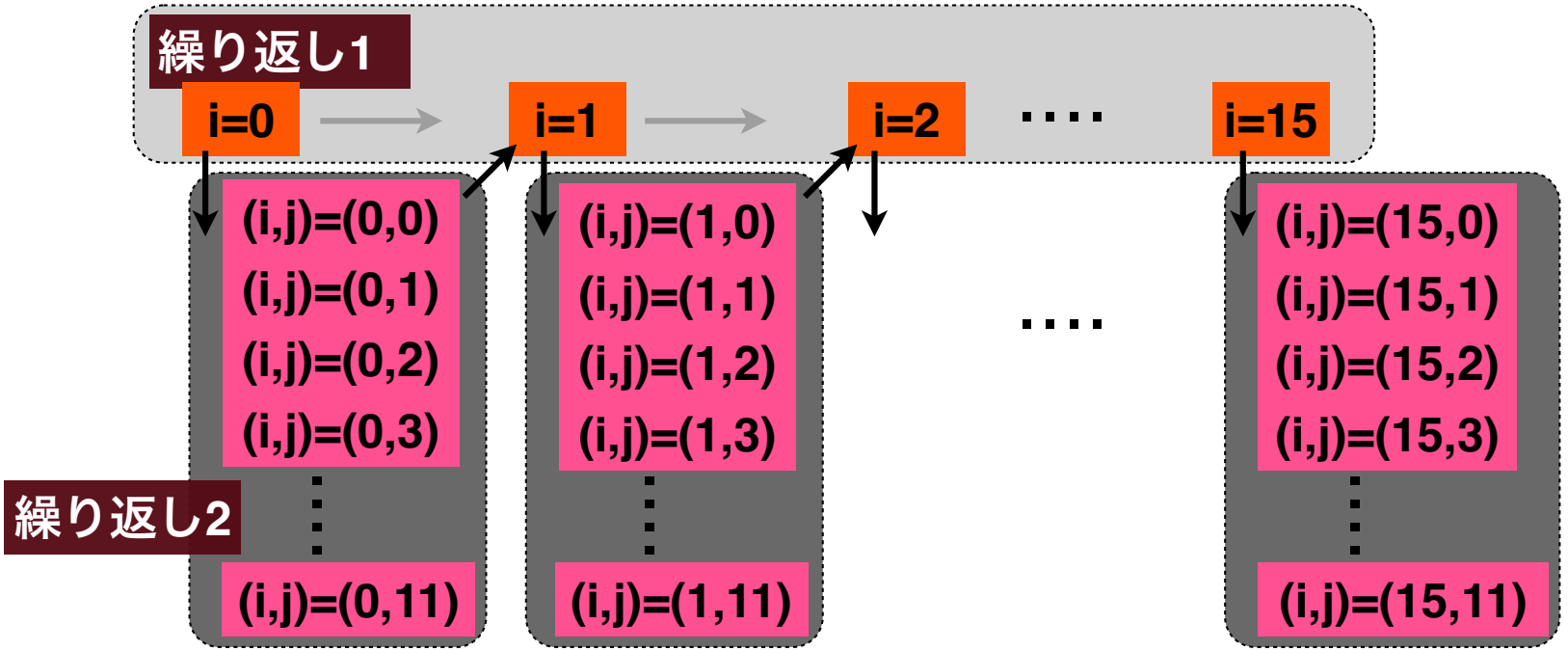
sample2A_4.pde

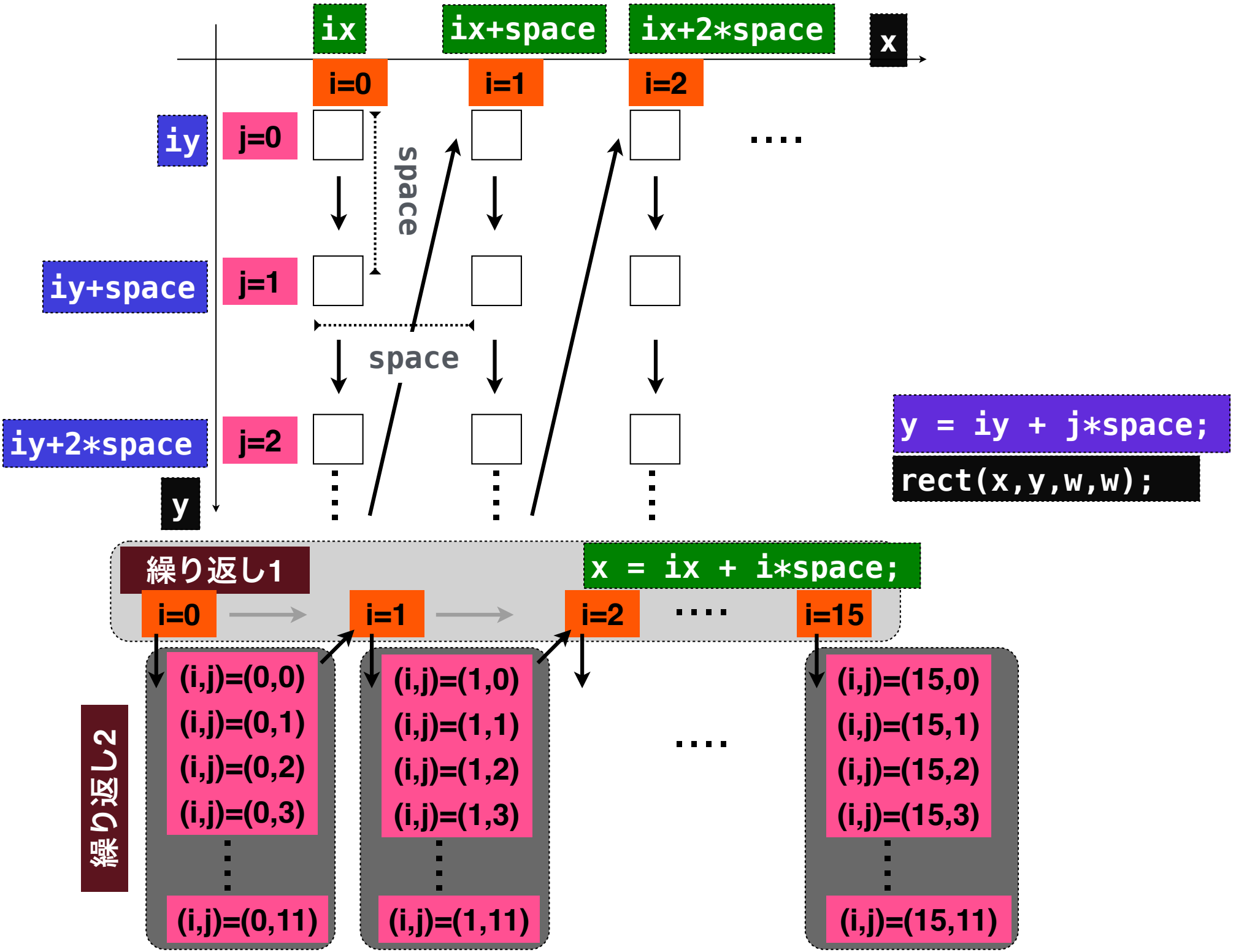
実行結果



FOR文のネスト（入れ子）

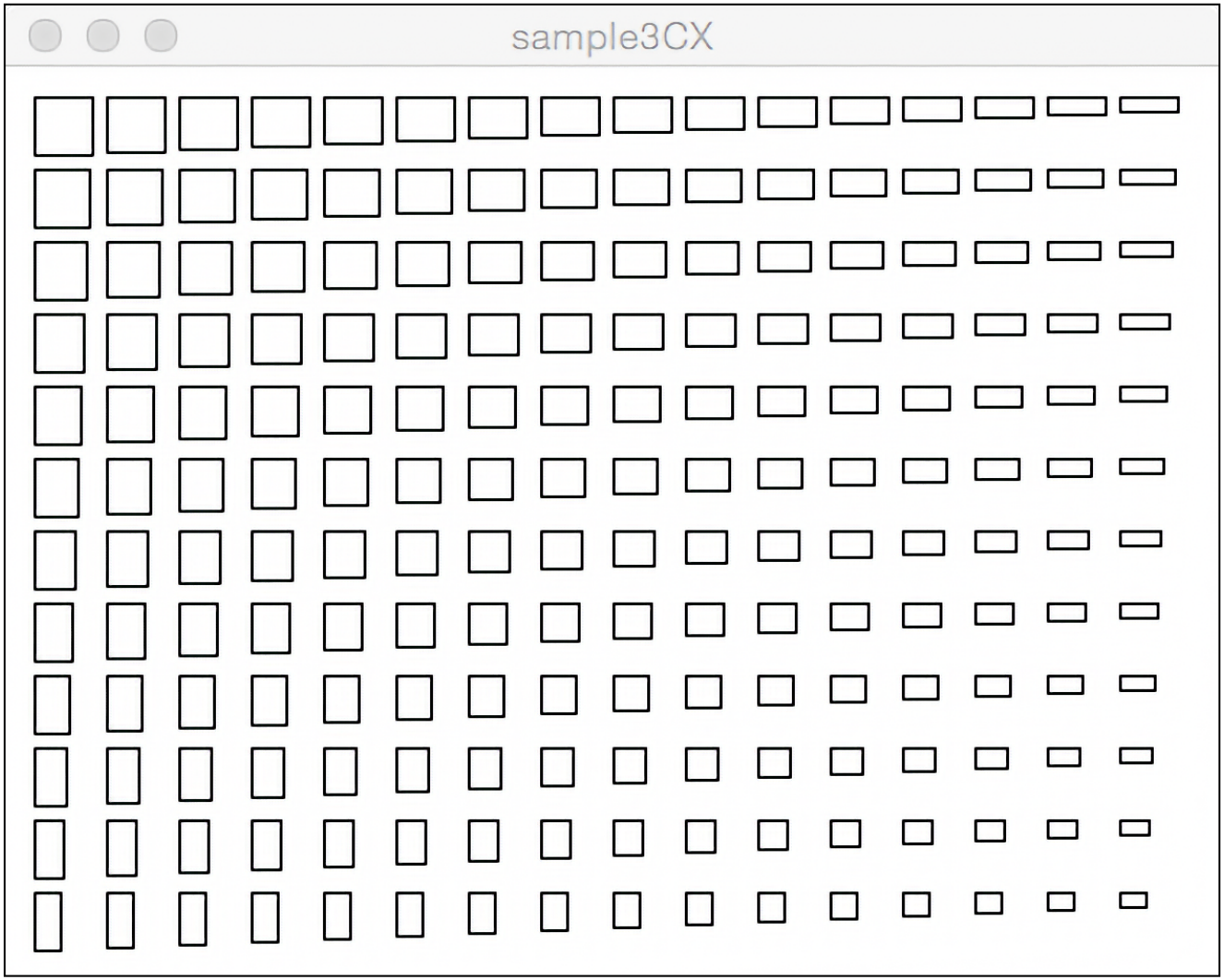
```
//くりかえし文の実行
for(int i=0;i<16;i++){                    繰り返し1
    int x = ix + i*space; //図形を右方向にspaceずらす
    for(int j=0;j<12;j++){                繰り返し2
        int y= iy + j*space; //図形を下方向にspaceずらす
        rect(x,y,w,w);
    }
}
```




ix
ix+space
ix+2*space
x
i=0
i=1
i=2
iy
j=0
space
iy+space
j=1
space
iy+2*space
j=2
y
y = iy + j*space;
rect(x,y,w,w);
繰り返し1
x = ix + i*space;
i=0
i=1
i=2
i=15
繰り返し2
(i,j)=(0,0)
(i,j)=(0,1)
(i,j)=(0,2)
(i,j)=(0,3)
(i,j)=(0,11)
(i,j)=(1,0)
(i,j)=(1,1)
(i,j)=(1,2)
(i,j)=(1,3)
(i,j)=(1,11)
(i,j)=(15,0)
(i,j)=(15,1)
(i,j)=(15,2)
(i,j)=(15,3)
(i,j)=(15,11)

小課題

sample2A_4.pdeを修正し，下図のように，右に行くほど高さが減り，下に行くほど幅が狭くなるようにしてください．



sample2A_4X.pde

FOR文のネストのネスト（入れ子）

記法

階層はどこまでも深くできる.

```
for(int i=0;i<16;i++){
     x = ix + i*sp;
     for(int j=0;j<12;j++){
          y= iy + j*sp;
          for(int k=0;k<8;k++){
          }
      }
 }
```

階層が深くなるにつれて, アルファベットを一つずつ後ろにずらしていくとわかりやすい.

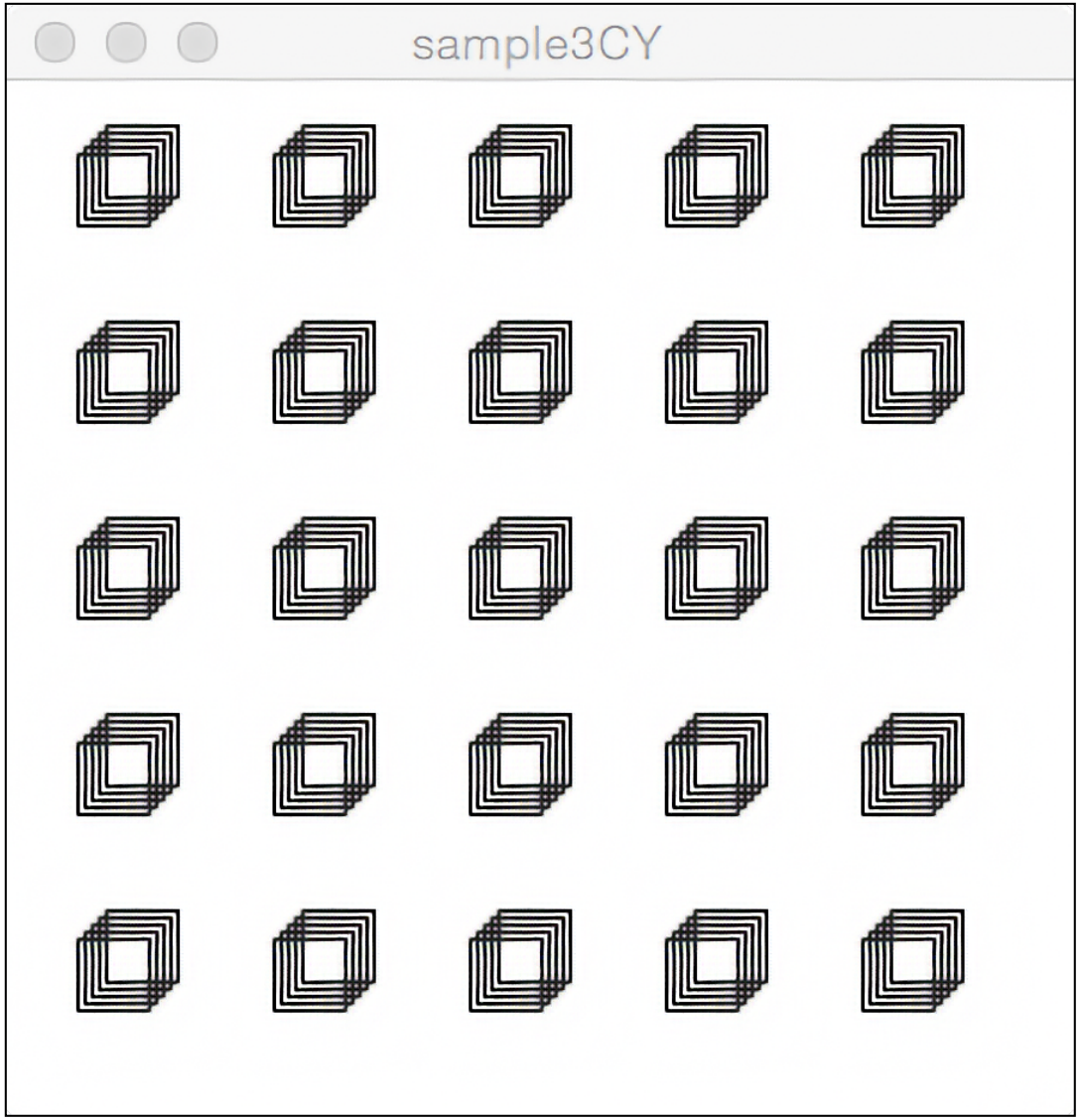
a, b, c,...	i, j, k,...
p, q, r,...	x, y, z,...

小課題（ネストのネスト）

sample2A_5.pde

```
1  size(300,290);  //サイズ
2
3  background(255); //背景白
4  stroke(0);       //線は黒
5  noFill();        //塗りつぶしなし
6
7  //X座標の左端, 上端
8  int ix = 20; int iy = 20;
9  //正方形の一辺の長さ, 正方形の間隔
10 int w = 20; int space = w+35;
11
12 for(int i=0;i<5;i++){
13   int x = ix + i*space;
14   for(int j=0;j<5;j++){
15     int y = iy + j*space;
16
17     for(int k=0;[redacted]k++){
18       int x2 = x[redacted]
19       int y2 = y[redacted]
20       rect(x2,y2,w,w);
21     }
22   }
23 }
```

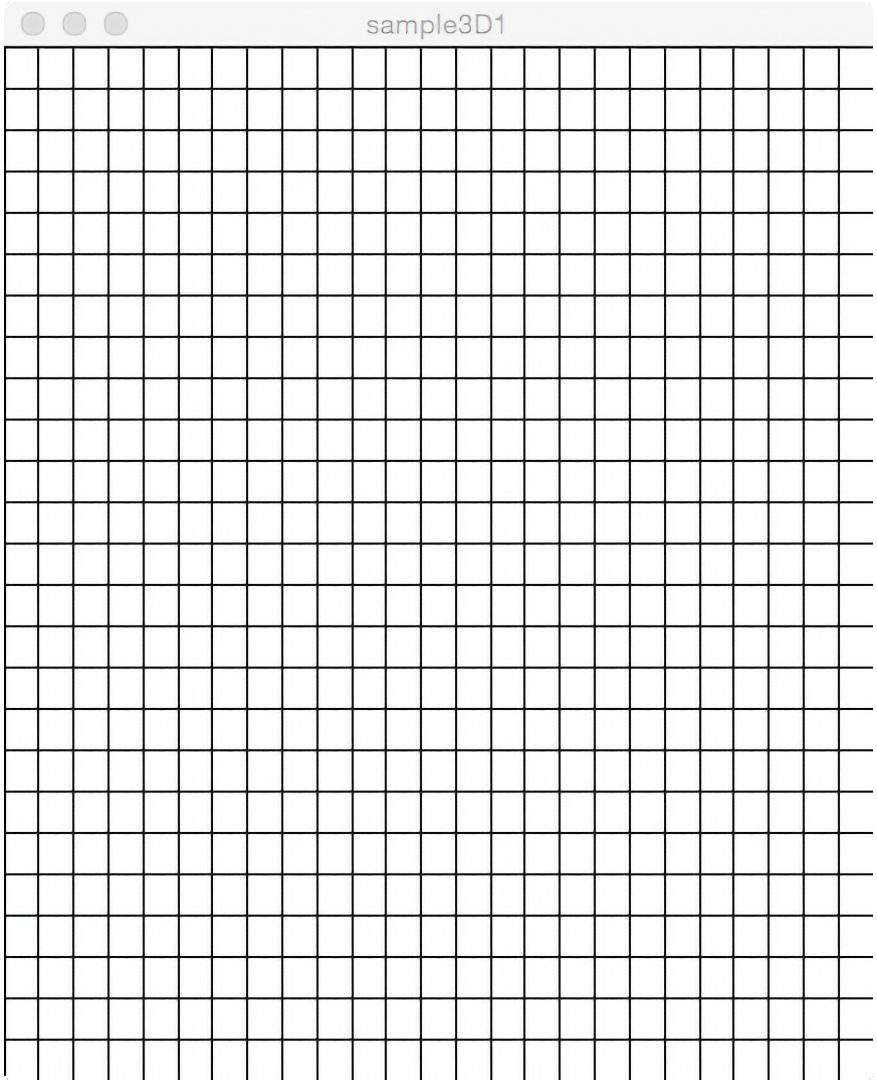
左のコードを参考として, 下のような図形のパターンを描画してみてください.



グリッド線を引く

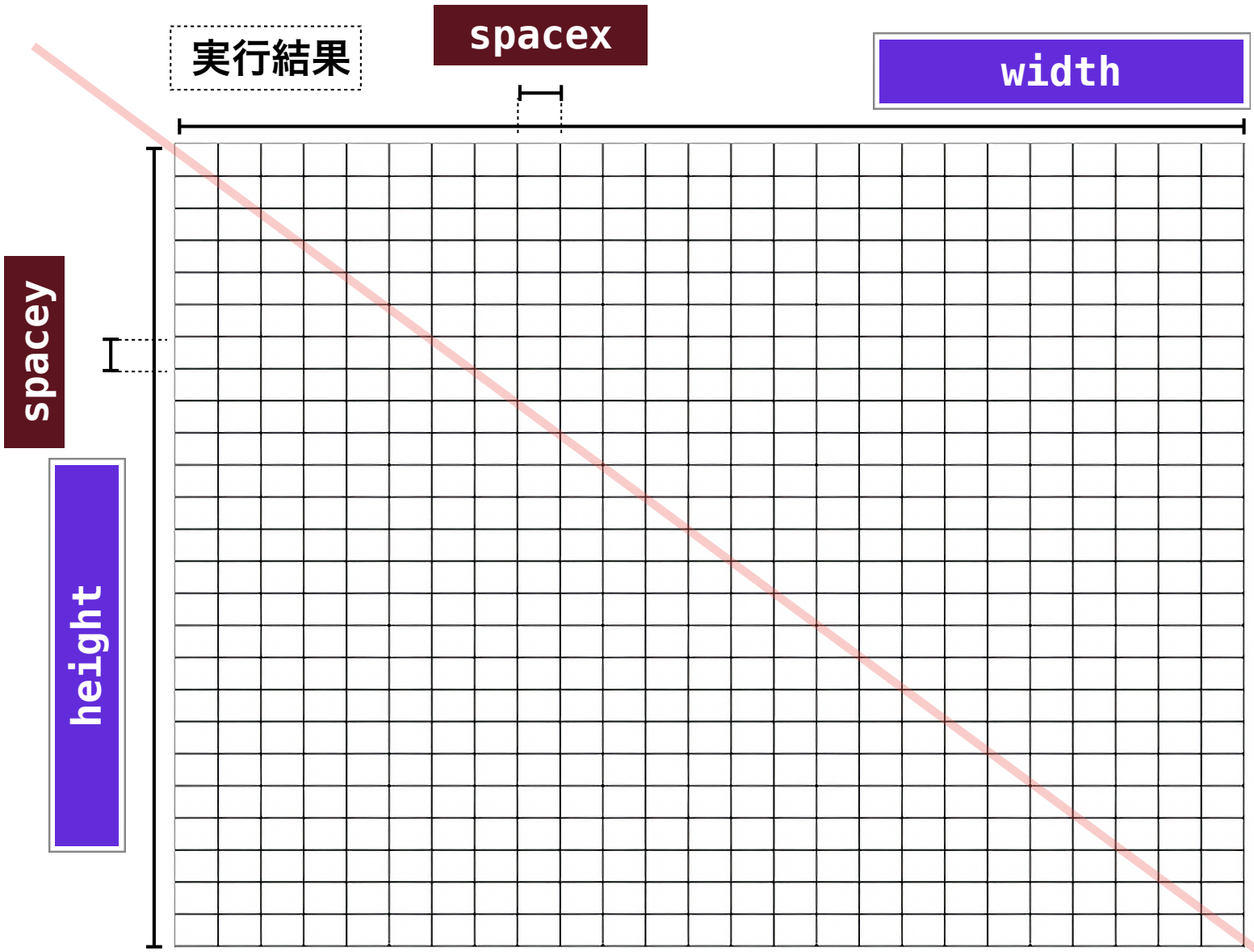
```
1  size(420,500);  //サイズ
2
3  background(255); //背景白
4  stroke(0);       //線は黒
5  noFill();        //塗りつぶしなし
6
7  float linesum = 25; //線の数
8  float spacex = width/linesum;  //横方向の間隔
9  float spacey = height/linesum; //縦方向の間隔
10 float x, y;  //各線のX座標・Y座標
11
12 for(int i=0;i<linesum;i++){
13   x = i * spacex;
14   line(x,0,x,height);
15 }
16
17 for(int i=0;i<linesum;i++){
18   y = i * spacey;
19   line(0,y,width,y);
20 }
```

sample2A_6.pde



実行結果

グリッド線を引く



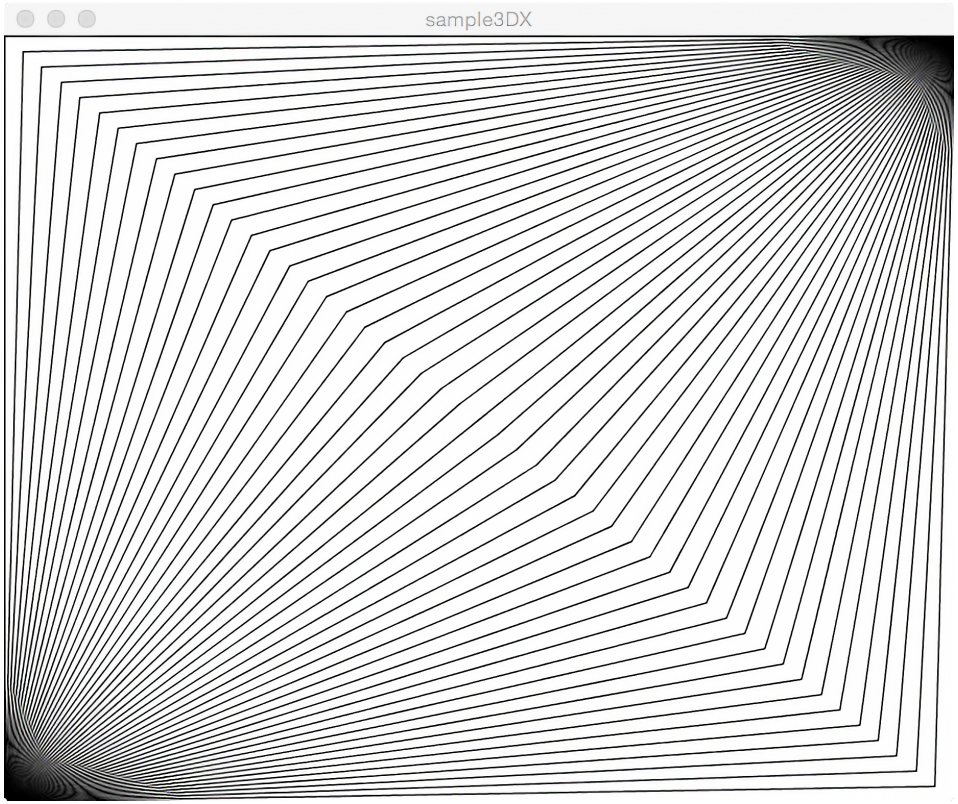
spacex = width / linesum=25（線の数）;

spacey = height / linesum=25（線の数）;

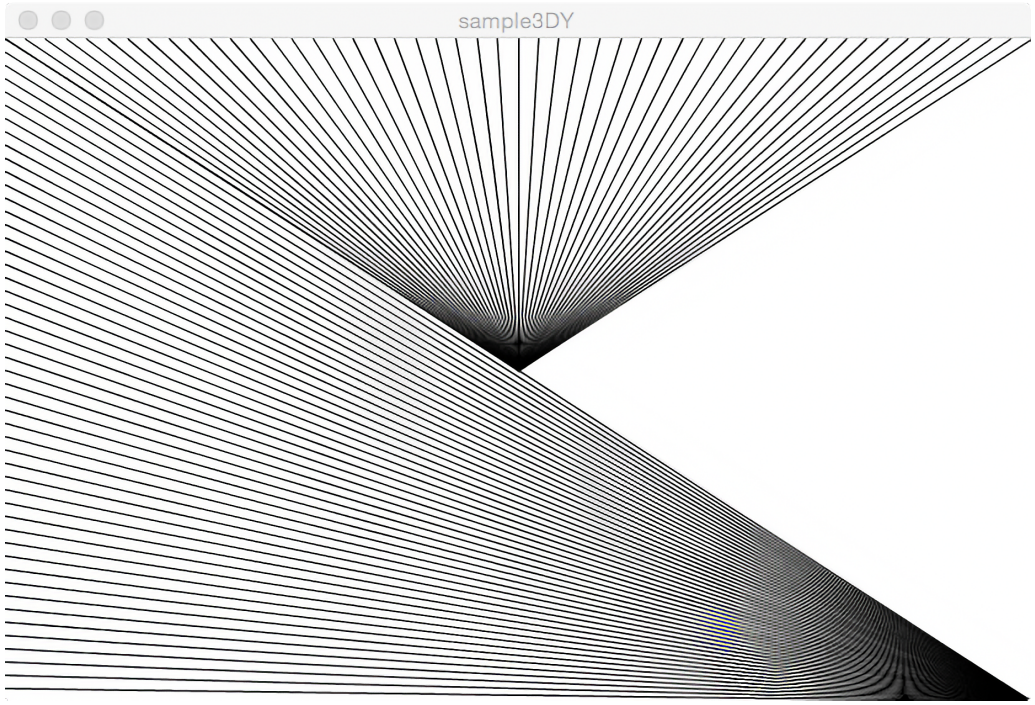
練習

以下のような図形を描画してください.
線の間隔は自由に変えてください.

sample2A_6X.pde



sample2A_6Y.pde



上部の放射線はウィンドウの中点を起点としています.