

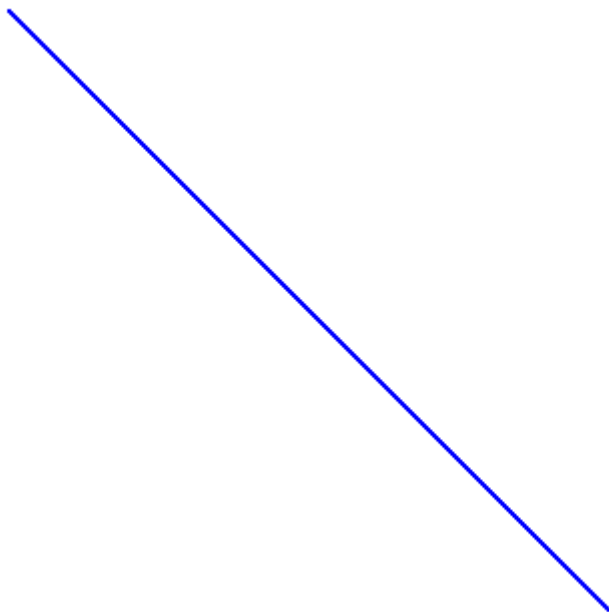
メディア技術基礎(web) 脇田玲先生(2010)、田中浩也(2011)

Processing [第一回]

基本的な描画

- ・線の描画
- ・長方形、円の描画
- ・for 文を用いた繰り返し
- ・ランダムな数値を用いた描画

■ サンプルプログラム ■



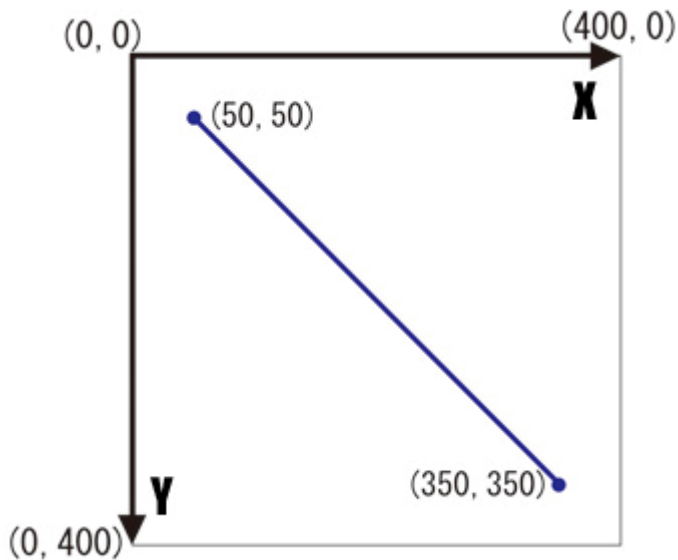
```
size(400, 400);           //ウィンドウのサイズを決定 (横幅、縦幅)  
colorMode(RGB, 255);     //カラーモードを設定 (モード、最大値)  
background(255, 255, 255); //背景の色を指定 (R, G, B) この場合は白
```

```
stroke(0, 0, 255);       //線の色を指定 (R, G, B)  
strokeWeight(2);        //線の太さを指定 (ピクセル)
```

```
line(50, 50, 350, 350); //線を引く (始点 x 座標, 始点 y 座標, 終点 x 座標, 終点 y 座標)
```

[Processing の座標系]

Processing のウィンドウでは、左上が原点となり、x は右、y は下が + 方向となる。



[線を描画する関数]

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

座標 (x1,y1) から座標 (x2,y2) までの線分を描画

[カラーモードを設定する関数する関数]

```
colorMode(mode,value);
```

カラーモードを mode (RGB or HSB) に設定、値を 0 ~ value の間で設定。

RGB モードの場合色を (Red:赤の強さ(0 ~ value)、Green:緑の強さ(0 ~ value)、Blue:青の強さ(0 ~ value)、Alpha:透明度(0 ~ value)) で設定

HSB モードの場合色を (Hue:色相(0 ~ value)、Satulation:彩度(0 ~ value)、Brightness:明度(0 ~ value)、Alpha:透明度(0 ~ value)) で設定

練習 1

1-1

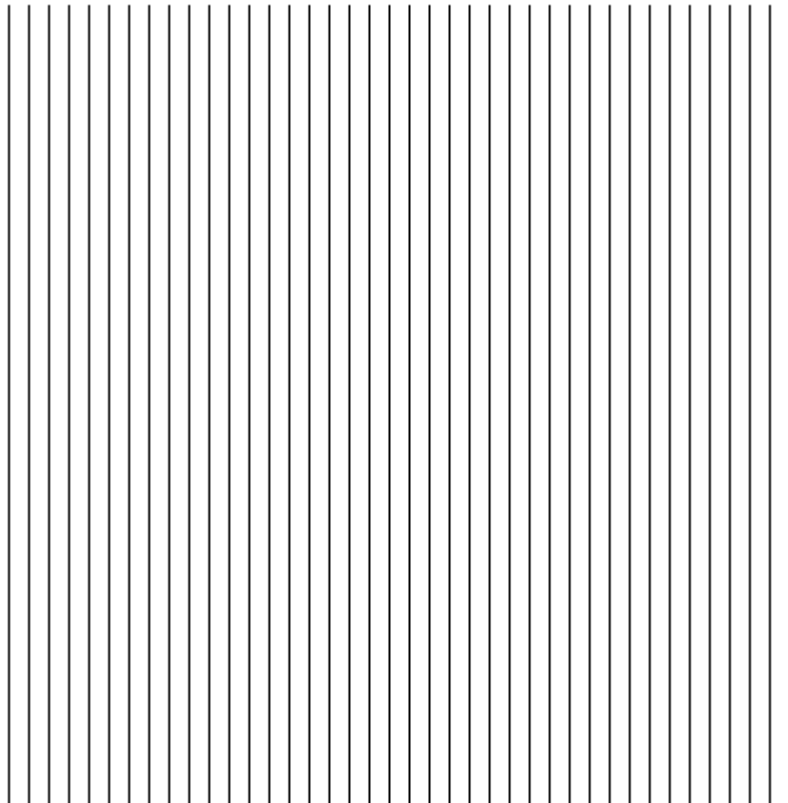
4点(100,100),(100,300),(300,100),(300,300)を結んで正方形を描画しましょう。

[\[課題 1-1 Processing 実行画面サンプル\]](#)

1-2(a)

繰り返し(for 文)を用いて、下のサンプルのような等間隔の縦ラインを描画しましょう。
サンプルでは、ライン同士の間隔は 10 ピクセルで 40 本のラインを描画しています。

[課題 1-2(a) 実行画面]



[for 文を用いた繰り返し]

```
int i;  
  
for (i=0; i<40; i++) {  
    (繰り返したい命令)  
}
```

まず繰り返しが何回目かをカウントする変数を定義する。

1. (繰り返しカウント 0) 1 回実行 {}内の命令
2. 命令を実行し終わったら変数を 1 カウントし繰り返す命令の最初に戻る
3. 繰り返し 1 回目(実行は 2 回目)実行 {}内の命令
4. 命令を実行し終わったら変数を 1 カウントし繰り返す命令の最初に戻る
5. 上の手順で実行を続ける
6. 繰り返しカウントが設定した上限まで行ったら繰り返し終了となる。

```
int i;
```

カウントするための整数を定義する

```
for(i=0; i<40; i++){
```

i(繰り返しカウント)が 0 のとき(初回実行時)から、40 未満(39 まで)の間、{}内の命令を繰り返す。(この場合 40 回繰り返す。)

i++は、i を 1 ずつカウントする。という定義。

1-2(b)

1-2(a)で描画したラインの色を、for 文(変数 i)を上手く利用して、グラデーションにしてみましよう。

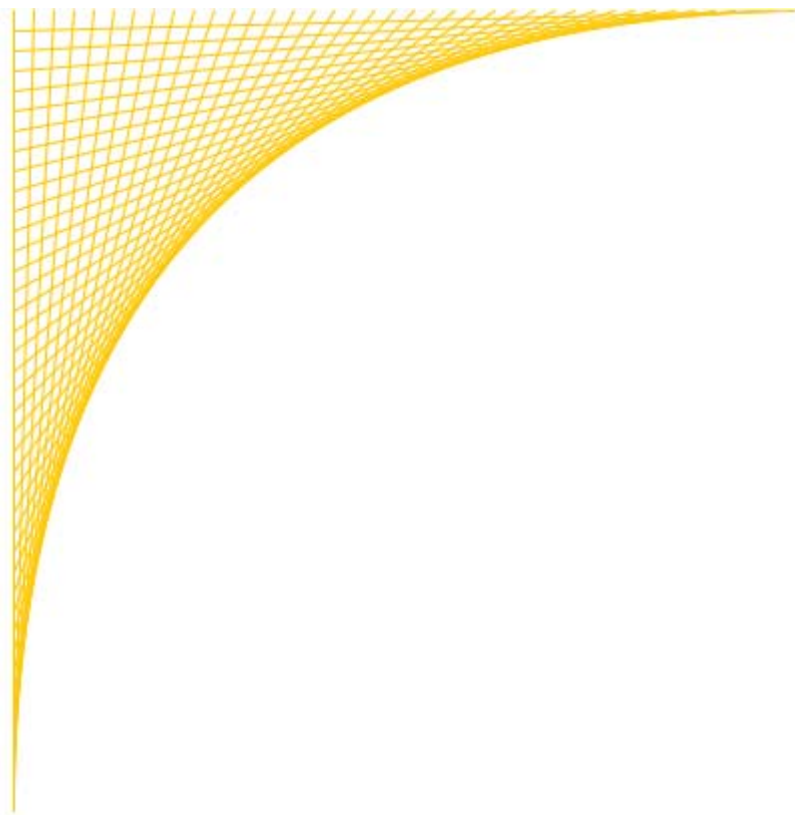
(線の x 座標値を変化させていく方法を stroke 関数にも応用)

[\[課題 1-2\(b\) Processing 実行画面サンプル\]](#)

1-3

下のような図形を描画してみましよう。

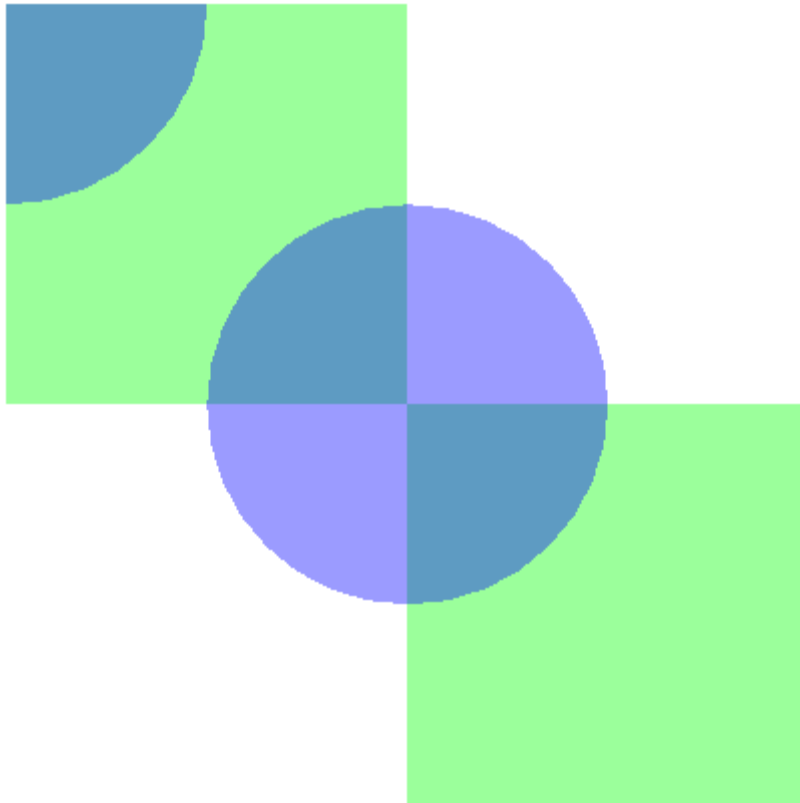
[課題 1-3 実行画面サンプル]



一見複雑に見えますが `line(0,0,0,400),line(10,0,0,390),...` と、始点の x 座標が 0 から 10 ずつ増えていくと同時に、終点の y 座標が 400 から 10 ずつ減っていく。という繰り返しで書くことができます。

※`smooth()`; という関数を描画命令の前に書くと曲線などを滑らかに描画できます。

■ サンプルプログラム 2 ■



```
size(400, 400); //ウィンドウのサイズを決定（横幅、縦幅）
colorMode(RGB, 255); //カラーモードを設定（モード、最大値）
background(255, 255, 255); //背景の色を指定（R, G, B）この場合は白

noStroke(); //線をなしに設定
fill(0, 255, 0, 100); //塗りの色を設定（赤、緑、青、透明度）
rect(0, 0, 200, 200); //長方形を描画（左上の点の x 座標, y 座標,
//横幅, 縦幅）

rect(200, 200, 200, 200);
fill(0, 0, 255, 100); //楕円を描画（中心の x 座標, y 座標, 横幅,
//縦幅）
ellipse(200, 200, 200, 200);
```

[長方形を描画する関数]

```
rect(x,y,width,height);
```

座標(x,y)を基準(左上の角)とした、幅 width、高さ height の長方形を描画

※rectMode(CENTER) をrect関数の前に記述すると、基準点(x,y)を長方形の中心にすることができる。

[楕円を描画する関数]

```
ellipse(x,y,width,height);
```

座標(x,y)を中心とした、幅 width、高さ height の楕円を描画

[塗りつぶしの色を設定(RGB の場合)]

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

塗りつぶしの色を、赤の強さ red、緑の強さ green、青の強さ blue、透明度 alpha に指定。

alpha は 0 であれば透明、255 では不透明(通常の色)となる。

練習 2

2-1

(0,0),(200,200),(0,400),(400,0),(400,400)の点を中心とする半径 200 の円を描画してみましょう。

円の色を半透明にして描画してみましょう。

[\[課題 2-1 Processing 実行画面サンプル\]](#)

2-2(a)

for 文を用いて下の図形を描画してみましょう。

正方形の一辺の長さは 80 ピクセル、位置は(0,0),(80,80),(160,160),...となっています。透明度や色の設定はサンプル通りでなくても構いません。自由に設定してください。

[課題 2-2(a)実行画面サンプル]



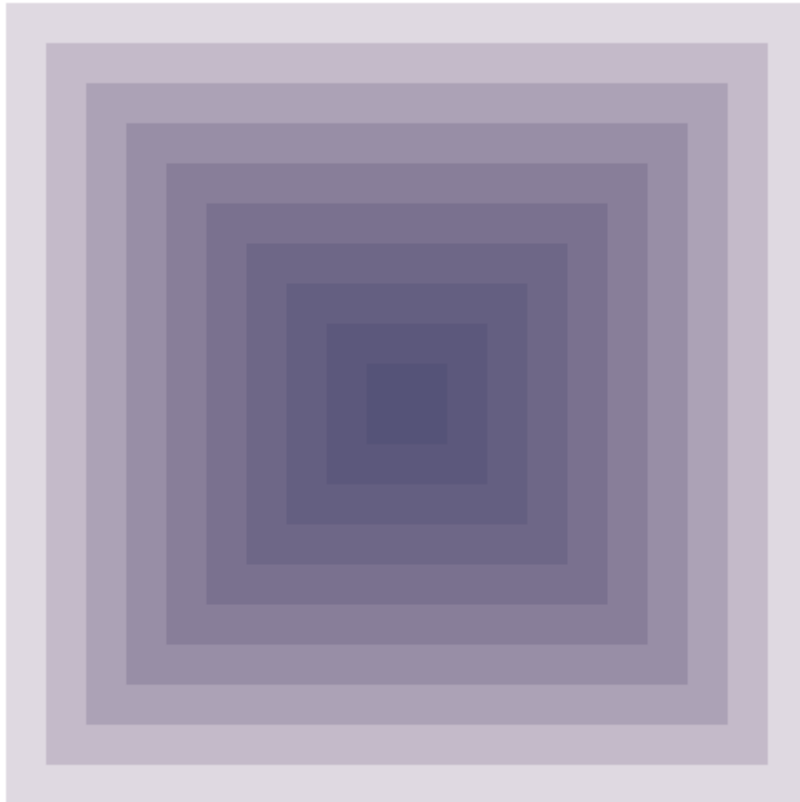
2-2(b)

for 文を用いて、半透明の正方形を大きさを変えながら重ねて描画し、以下のような描画を行ってみましょう。

サンプルでは書く正方形の透明度は 50、正方形のサイズが 40 ピクセルから 400 ピクセルまで 40 ずつ増えています。

`rectMode(CENTER)` を `rect` 関数の前に記述すると、基準点(x,y)を長方形の中心にすることができる。

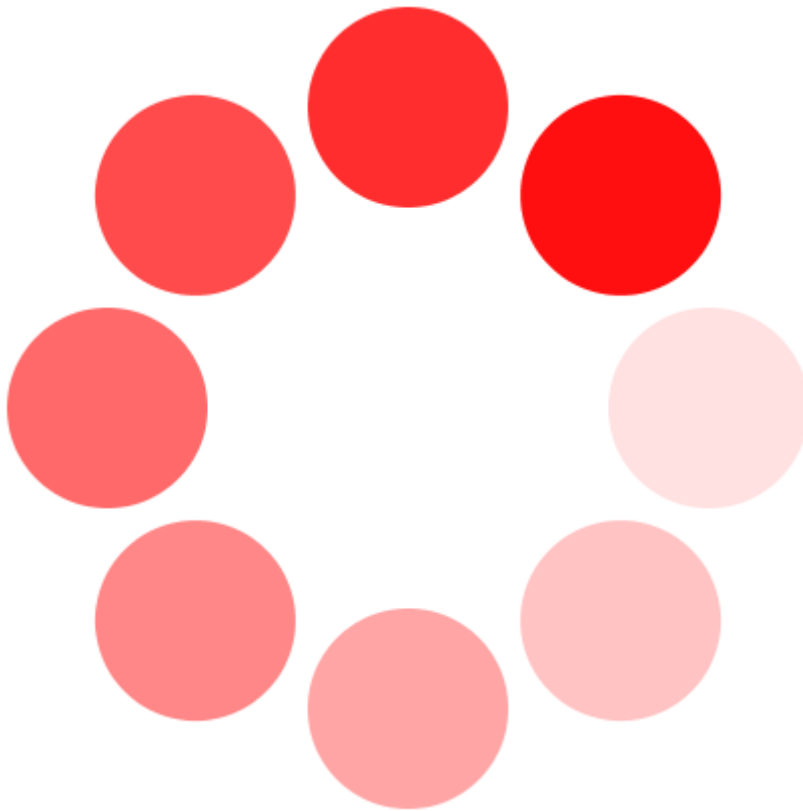
[課題 2-2(b)実行画面サンプル]



2-3

for 文を用いて、以下の例のような、円の中心点が円周上を移動して描かれる図形を描画してみましょう。

[課題 2-3 実行画面サンプル]



[Hint]

三角関数を使用して書いてみましょう。

`sin(radian),cos(radian),tan(radian)`

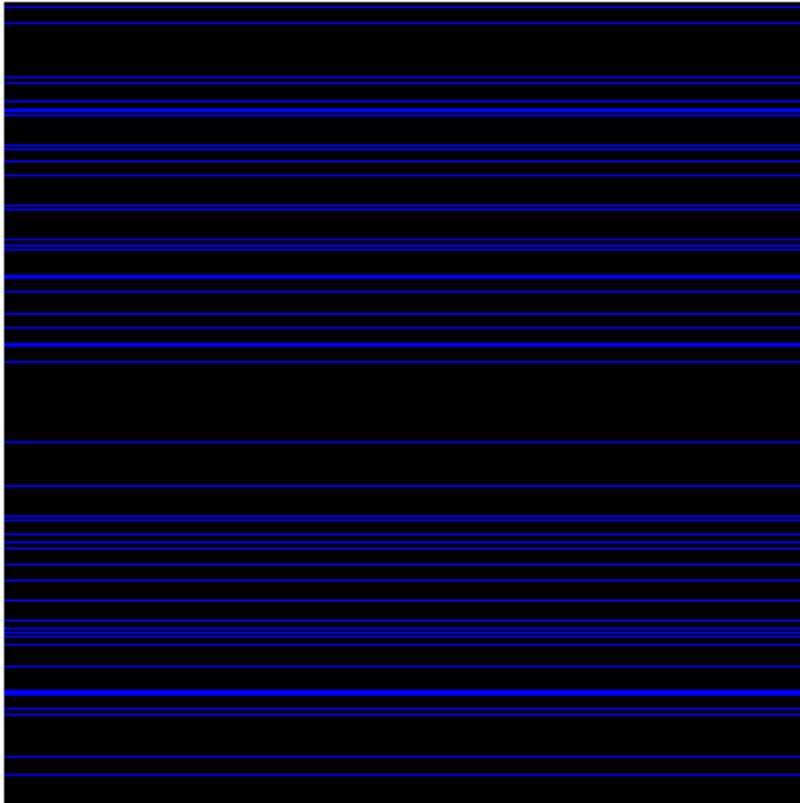
角度 radian(ラジアン)の時の `sin,cos,tan` それぞれの値を出力。

例えば、`ellipse(300*cos(PI/3),300*sin(PI/3),100,100);`等のように使用できる。

※ラジアン(rad)と度数の関係は、 0° のとき 0rad 、 180° の時 πrad 、 360° の時 $2\pi\text{rad}$ 。

Processing では、 π は `PI` と書くことで使用可能

■ サンプルプログラム 3 ■



```
size(400, 400); //ウィンドウのサイズを決定（横幅、縦幅）

colorMode(RGB, 255); //カラーモードを設定（モード、最大値）
background(0, 0, 0); //背景の色を指定（R, G, B）この場合は黒

for (int i=0; i<50; i++) {
  stroke(0, 0, 255, 255);
  float ry=random(height); //0〜height（ウィンドウの縦幅）までのラン
  ダムな数値を出力し、ryに代入
  line(0, ry, width, ry);
}
```

[ランダムに値を設定する]

- ・上限を設定する場合

```
random(value);
```

0～value までのランダムな値を出力する。

- ・範囲を指定する場合

```
random(min,max);
```

min～max までのランダムな値を出力する。

[ウィンドウの幅、高さの値を使用する]

変数としてウィンドウの幅、高さを使用したい場合、width,height という変数があらかじめ用意されているので、それを用いることで、サンプルプログラムのように、ウィンドウの高さ、幅を使用することができます。

練習 3

3-1(a)

水平な直線ではなく、ランダムな y 座標から、別のランダムな y 座標までの直線を描画するようにサンプルを変更してみましょう。

[\[課題 3-1\(a\) Processing 実行画面サンプル\]](#)

3-1(b)

座標値だけでなく、色、線の太さにもランダムな数値を用いて描画を行ってみましょう。サンプルでは、線の太さを 1 から 3 の間、青、緑の強さを 100 から 255 の間でランダムに設定しています。

範囲を決める時は、random(下限,上限);
を用います。

[\[課題 3-1\(b\) Processing 実行画面サンプル\]](#)

3-1(c)

```
line(0,random(400),random(400),0);
```

と指定すればウィンドウの左辺から上辺への斜めのランダムな線を描画することができます。

これを利用して、左辺から上辺、右辺から上辺、左辺から下辺、右辺から下辺すべて

についてランダムな線の集合を描画してみましょう。

[\[課題 3-1\(c\) Processing 実行画面サンプル\]](#)

3-2

画面に 5×5 の正方形を描画し、ランダムを利用して塗ってみましょう。

[Hint]

for 文のネスト(入れ子)を使う。for 文の中にもうひとつ for 文を作る

```
for (int i=0; i<5; i++) {  
    for (int j=0; j<5; i++) {  
    }  
}
```

[\[課題 3-2 Processing 実行画面サンプル\]](#)

課題

Processing で自由な描画を行い、Java アプレットとして自分のウェブに載せましょう。授業で習っていない描画方法などを使用していただいても結構です。(参考 URL[\[Processing.org Reference\]](#))

提出締切 : 5/27 (木) 23:59

[Processing で作ったファイルをウェブにアップする方法]

File メニューから、Export を選択 (Ctrl+E)

Processing のソースが保存されているディレクトリ(デフォルトでは、My Documents 以下の Processing ディレクトリ内、プログラム名のディレクトリ)に Applet というディレクトリが作られ、自動的に開かれ、中には

- ・index.html
- ・(プログラム名).pde Processing のソースファイル
- ・(プログラム名).jar プログラムのアプレット
- ・(プログラム名).java プログラムの Java のソースコード

・loading.gif アプレットロード時用の画像
という 5 つのファイルがつくられます。

index.html には、すでにアプレットが埋め込まれた状態で html になっているので、このまま Applet ディレクトリごとアップして、ディレクトリ内の index.html にリンクを張ることで、プロセッシングで作ったものを Web にアップすることが可能です。

課題の提出方法

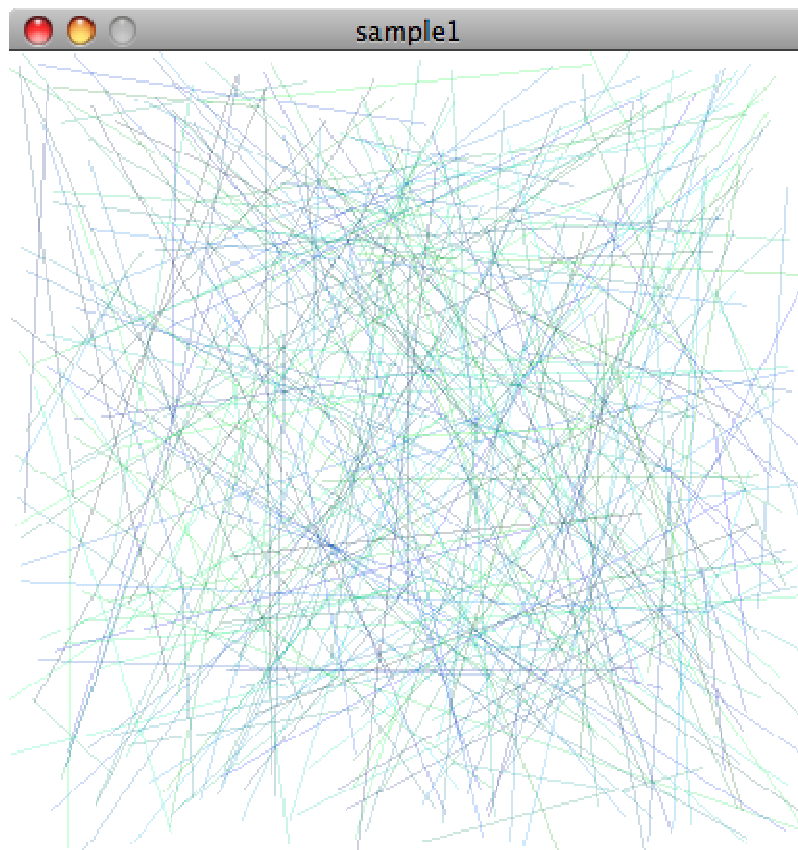
課題を個人のディレクトリに以下の URL になるように配置します。

Processing [第二回]

アニメーションの基本

- ・ランダムな描画
- ・速度を用いた運動の表現
- ・条件分岐 (if,else)

■ サンプルプログラム ■



[サンプルを](#)

[表示](#)

```
void setup() {  
  size(400, 400);           //ウィンドウのサイズを決定(横幅、縦幅)  
  colorMode(RGB, 255);     //カラーモードを設定(モード、最大値)  
  background(255, 255, 255); //背景の色を指定(R, G, B) この場合は白
```

```
    frameRate(30); //フレームレートを設定(fps:frame per
seconds)
}

void draw() {
    stroke(0, random(255), random(255), 50);
    line(random(width), random(height), random(width), random(height));
}
```

[アニメーションの基本]

Processing では、メインループ関数(draw 関数)内に書かれた命令を繰り返し実行することによって動的な描画を行います。

サンプルでは、ランダムな点からランダムな点まで線を引くという命令を一秒間に 30 回のペースで繰り返し実行するというプログラムで描画を行っています。

[初期化関数]

```
void setup(){
(初期化する内容)
}
```

プログラムの最初(実行時)に一度だけ実行される命令を書きます。

- ・ウィンドウサイズの指定
- ・色モードの指定
- ・背景色の指定
- ・フレームレートの設定

等、初期化で一度だけ設定すればいい命令などを書きます。

[メインループ関数]

```
void draw(){
(ループさせたい命令)
}
```

ここに書いた命令は無限ループのように繰り返し実行されます。

繰り返しのインターバルは、フレームレートの設定に依存します。

draw 関数内のプログラムを上から順に実行→一定時間(サンプルでは 1/30 秒)→繰り返し…というようになります。

[フレームレートを設定する関数]


```
frameRate(fps);
```

一秒間に fps 回繰り返し描画する。(1/30 毎にメインループ関数内の命令を繰り返す)
フレームレートの単位は、fps:frame per seconds(一秒間に描画するフレーム数)で表
します。

このとき、フレームとは映画のフィルムや、アニメのセル画のようなものと考えることが
できます。

練習 1

1-1

終点(もしくは始点)をウィンドウの中心に設定して、放射線状に線を描画してみましょ
う。

[\[練習 1-1 Processing 実行画面サンプル\]](#)

1-2

ランダムな1点で交差する x 軸と平行な直線と y 軸と平行な直線を描画してみましょ
う。

[Hint]

ランダムな y 座標をとる、x 軸と平行な線(水平線)を引く場合

```
float ry=random(height);
```

```
line(0,ry,width,ry);
```

というように指定すればよい。

[\[練習 1-2 Processing 実行画面サンプル\]](#)

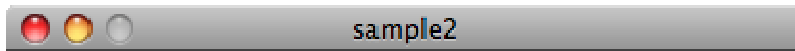
1-3

ランダムな位置に正方形を描画してみましょ
う。

(位置が重なることがあるので、半透明な塗りで描画してみましょ
う)

[\[練習 1-3 Processing 実行画面サンプル\]](#)

■ サンプルプログラム 2 ■



[サンプルを](#)

[表示](#)

```
int x=0; //x 座標の初期値を設定(グローバル変数)
int speed=10; //速度を設定(グローバル変数)

void setup() {
  size(400, 400); //ウィンドウのサイズを決定(横幅、縦幅)
  colorMode(RGB, 255); //カラーモードを設定(モード、最大値)
  background(255, 255, 255); //背景の色を指定(R, G, B) この場合は白
  frameRate(30); //フレームレートを設定(fps:frame per
seconds)
}

void draw() {
```

```

noStroke(); //線をなしに設定

fill(255, 255, 255, 255); //塗りの色を白(背景色)に設定
rect(0, 0, width, height); //背景色でウィンドウを塗りつぶす(ウィンドウをリセット)

fill(50, 50, 50, 255);
ellipse(x, height/2, 50, 50); //座標(x, height/2)に円を描画

x=x+speed; //描画後 x を speed(10) だけ増やす(次に描画する位置を 10 ピクセル動かす)
}

```

[変数を用いたアニメーション]

今回のサンプルでは、円をこちらが意図したとおりに動かしています。このとき、1フレームごとに画面をリセット(サンプルの場合はウィンドウを白く塗る)し、x座標を10ピクセルずつ動かしながら繰り返し描画することにより、円が右に移動しているように見えるアニメーションを作っています。

[画面のリセット]

```
fill(255,255,255,255);
rect(0,0,width,height);
```

塗りの色を白に設定し、ウィンドウの幅、高さの四角を描画しています。つまり、ウィンドウを白く塗って、画面をリセット(初期化)しています。この部分を書かずに描画してみるとわかりますが、ウィンドウをリセットしないと、それまで描画した円が消えないため、右に移動するアニメーションではなく、徐々に右に伸びていくアニメーションになってしまいます。また、メインループプログラムは、上から順に実行した結果を出力するので、画面のリセットを描画命令より下を書いてしまうと、描画命令の内容を白で塗りつぶした結果が出力されてしまうため、注意してください。

```
fill(255,255,255,100);
rect(0,0,width,height);
```

のように、半透明の白で塗りつぶすことで、球の動きの軌跡を残すこともできます。

[サンプルの円の動きを軌跡を残しながら描画した例]

[グローバル変数]

サンプルの1行目に書いてある

```
int x=0;
```

は、円を描画する位置(変化する値)を保存する変数と初期値を定義する命令です。

```
int speed=10;
```

は、円の移動する速度を保存する変数です。初期値は 1/30 秒に 10ピクセル移動する速度に設定しています。

関数内で変数を定義した場合、その関数内でしか定義した変数は使用できません(これをローカル変数といいます)。

たとえば、初期化関数(setup)内で `int x=0;` と書いても、プログラムの最初に実行され、`x` に値が入れられますが、メインループ関数(draw)内でこの `x` を使うことができません。

このため、どの関数からでもアクセスできる変数(こちらをグローバル変数と呼びます)を定義し、初期値を設定したい場合は、初期化関数やメインループ関数の外で定義を行います。

このサンプルでは `int speed=10;` を描画関数(draw)内に記述しても描画には問題ありませんが、後の練習で変化させる必要が出てくるためグローバル変数として定義しています。

[変数の操作]

移動するアニメーションを作るには、描画するたびに座標が変わるようにする必要があります。

サンプルでは、メインループ関数の最後に

```
x=x+speed;
```

という命令を書いています。これは、`x` に `speed(10)` 足したものを `x` に代入するという命令で、`x` を 10 増やすという意味です。

このサンプルでは、メインループの描画の最後にグローバル変数 `x` に `speed(10)` をプラスしているため、次に描画する際 `x`(円を描画する `x` 座標)が 10 大きく(10ピクセル右)になっています。

この場合は、10ピクセルというのは `x` が 1/30 秒間に進む距離になるため、`speed` の値を変えることにより円の運動の速さを変えることができます。

練習 2

2-1

サンプルでは、 x が 400 を超えてしまうと円がウィンドウの外まで動いていってしまい、見えなくなってしまう。

x がウィンドウの右端まで行ったら、また左端に戻るようプログラムを変更してみましょう。

[\[練習 2-1 Processing 実行画面サンプル\]](#)

[if 文を用いた条件分岐]

この練習では、「もし x が 400 を超えたら x が 0 に戻る」というような命令を書く必要があります。

このような、ある条件にあてはまったときに命令を実行する、というようにしたい場合、if 分を使います。

```
if(条件式){
  (条件式が true なときに実行する命令)
}
```

このように記述し、条件式(不等式や等式など)が true、つまり条件に当てはまる場合括弧{}内の命令を実行する。というものです。もし条件に当てはまらない場合は括弧{}内の命令は実行されません。

条件式の書き方は、たとえば x が 0 の時実行する命令を書きたい場合 `if(x=0){}`、 y が 200 以下の時に実行する命令を書きたい場合は `if(y<=200){}` というように書きます。

2-2

サンプルのように、円が右端まで動いたら、壁にぶつかって跳ね返るように運動の方向を左に変え、左端に行ったらまた右に動くような、左右に往復するアニメーションをさせてみましょう。

[\[練習 2-2 Processing 実行画面サンプル\]](#)

[Hint]

else if()を用いて条件分岐をします。

[2 つ以上の条件を用いた場合分け]

```
if(条件式 1){
    Case 1:(条件式 1 が true なときに実行する命令)
}
else if(条件式 2){
    Case 2:(条件式 1 が false で、条件式 2 が true なときに実行する命令)
}
```

条件式 1 が true(条件 1 に当てはまる場合)、Case 1 の命令を実行。

条件式 1 が false(当てはまらない)で、条件式 2 が true(当てはまる)なとき、Case 2 の命令を実行

条件式 1 も 2 も共に false(当てはまらない)な場合はどちらも実行されません。

というような仕組みになっています。else if(条件式 2){} は何個でも記述できるため、2 つ以上の場合分けもすることができます。

また

```
else{
    (どの条件にも当てはまらない時に実行する命令)
}
```

を用いることで、どの条件にも当てはまらない場合の命令も設定することができます。

if 文 より細かい条件分岐

|| と書くと or

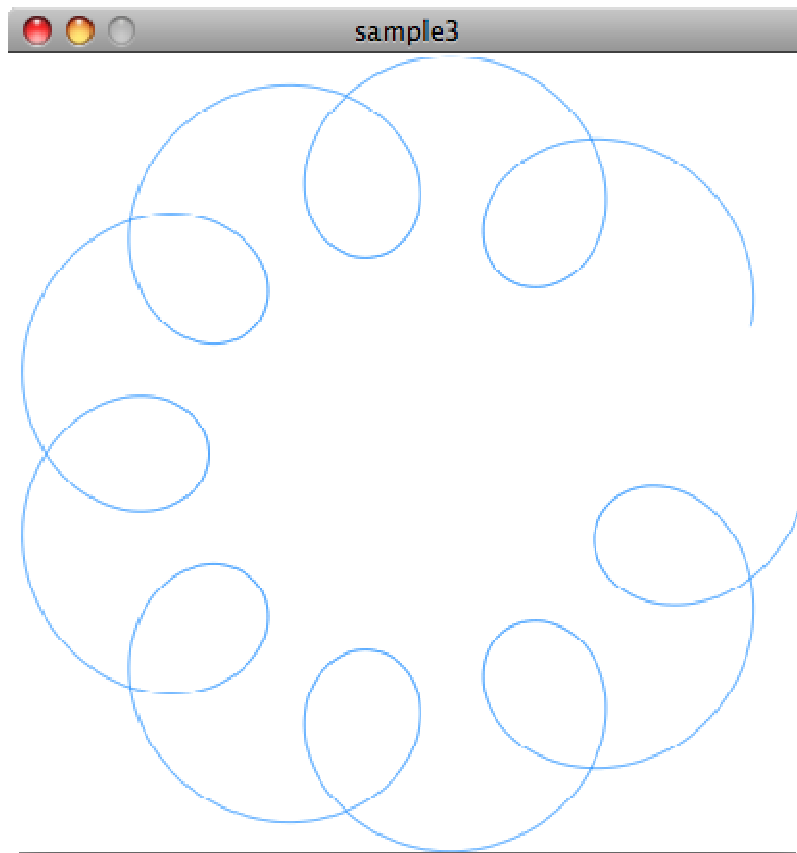
&& と書くと and

つまり

```
if( ( x < 10 || x > 20 ) && x < 30 )
```

と書くと x が 10 未満、もしくは x が 21 以上で 30 以下の場合を指す。

■ サンプルプログラム 3 ■



[サンプルを](#)

[表示](#)

```
float d1=0; //一つ目の円運動の角度を定義
(d:degree )
float d2=0; //二つ目の円運動の角度を定義(d:degree)
float av1=0.1; //一つ目の円運動の角速度を定義
(av:angular velocity)
float av2=1; //二つ目の円運動の角速度を定義
(av:angular velocity)
int r1=150; //一つ目の円運動の半径を定義(r:radius)
int r2=50; //二つ目の円運動の半径を定義(r:radius)

void setup() {
  size(400, 400); //ウィンドウのサイズを決定(横幅、縦幅)
  colorMode(RGB, 255); //カラーモードを設定(モード、最大値)
```

```

    background(255, 255, 255); //背景の色を指定 (R, G, B) この場合は白
    frameRate(30); //小刻みな描画のためフレームレートを
    高めに設定
}

void draw() {
    smooth();

    stroke(0, 125, 255, 100);
    line(width/2+r1*cos(radians(d1))+r2*cos(radians(d2)),
        height/2+r1*sin(radians(d1))+r2*sin(radians(d2)),
        width/2+r1*cos(radians(d1+av1))+r2*cos(radians(d2+av2)),
        height/2+r1*sin(radians(d1+av1))+r2*sin(radians(d2+av2)));

    d1=d1+av1; //角速度分だけ角度を増加
    d2=d2+av2; //角速度分だけ角度を増加
}

```

[角度をラジアンに変換する関数]

radians(degrees);

は degrees° のときのラジアン(rad: 弧度)を出力する。

radians();関数を使用することにより、0° ~360° の角度をラジアンに変えることができます。

たとえば、radians(180)は PI、radians(120)は 2*PI/3 というよう出力されます。

[線を引くアニメーション]

サンプルの場合、ウィンドウの中央を中心とした半径 r1(150)の円周上を回転する半径 r2(50)の円の円周上の一点が描く軌跡を描画しています。

複雑な式になっていますが、一つ一つソースを読めば理解できると思います。

このサンプルでは、もとの点から半径 r1 の円が 0.1°、半径 r2 の円が 1° 動いたところまで直線を引くことを繰り返し、細かく直線を引き続けて曲線に見えるように描画しています。

円を 1° ずつ描画するアニメーションを書く場合は

```

void draw(){
    line(width/2+radius*cos(radians(degrees)),

```



```
    height/2+radius*sin(radians(degrees)),  
    width/2+radius*cos(radians(degrees+1)),  
    height/2+radius*sin(radians(degrees+1)));  
degrees=degrees+1;  
}
```

となります。(radius は半径、degrees は角度です。)

練習 3

3-1

サンプルは、ウィンドウの中央を中心とする大きい円の円周上を小さい円が回転する時の小さい円の円周上の一転の軌跡を描画しています。

サンプルは大きい円が一周する間に小さい円は 10 周するようなプログラムになっています。

これを、小さい円が一周する間に大きい円が 10 周するように変えてみましょう。

ソースが複雑でわかりづらいと思いますが、一つ一つ読んでみてどの変数を変えれば良いか考えてみましょう。

[\[練習 3-1 Processing 実行画面サンプル\]](#)

3-2

サンプルのように、それぞれの角速度の整数比に最小公倍数があるとき、その公倍数になるたびに図形が閉じられます。(サンプルは大円が 1 回、小円が 10 回回転した時点で閉じます。サンプルでは描画するたびに少しずれが生じますが、理論上は常に同じ軌道を通り続けます。)

無理数を用いて、無限に重ならず描画し続けるプログラムを書いてみましょう。

[\[練習 3-2 Processing 実行画面サンプル\]](#)

[Hint]

ルートを使って無理数を表現しましょう。

無理数を用いる際、扱いやすいものに平方根があります。

```
sqrt(number);
```

で、number の平方根を出力します。

例えば、 $\sqrt{4}$ は2、 $\sqrt{2}$ は、1.41421356...(無理数)になります。

サンプルでは、大円の角速度 1 に対し、小円の角速度を $(\sqrt{5})/10$ と設定しています。

課題

Processing で自由なアニメーションを作りを行い、前回同様 Java アプレットとして自分のウェブに載せましょう。

授業で習っていない描画方法などを使用していただいても結構です。(参考 URL[[Processing.org Reference](#)])

提出締切 : 6/3 (木) 23:59

課題の提出方法

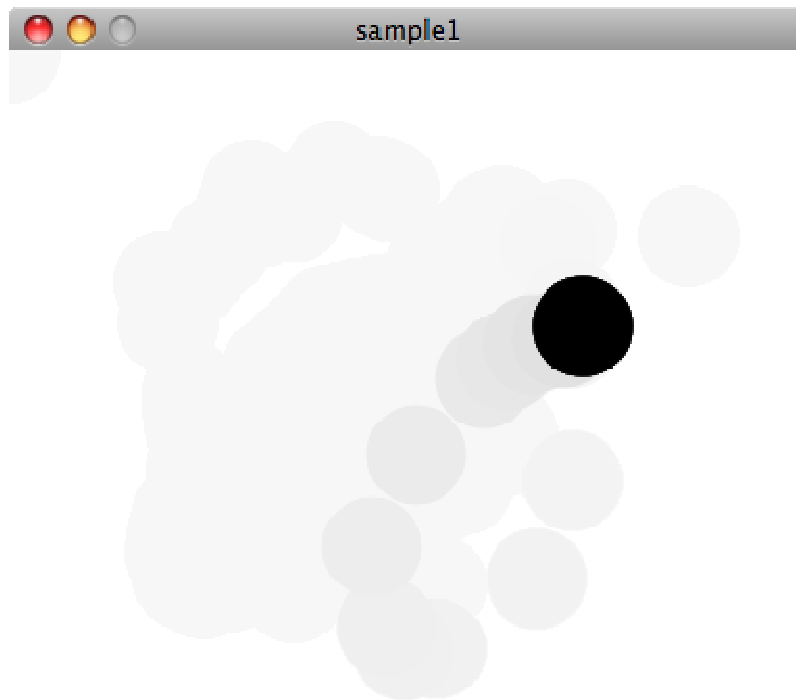
課題を個人のディレクトリに以下の URL になるように配置します。

Processing [第三回]

インタラクティブ

- ・マウスに反応させる
 - ・描写アプリの作成
 - ・ボタンの作成
 - ・キーボード入力
-

■ サンプルプログラム ■



[サンプルを](#)

[表示](#)

```
void setup() {  
  size(400, 400);           //ウィンドウのサイズを決定(横幅、縦幅)  
  colorMode(RGB, 255);     //カラーモードを設定(モード、最大値)
```

```

background(255, 255, 255); //背景の色を指定 (R, G, B) この場合は白
frameRate(30); //フレームレートを設定 (fps:frame per
seconds)
}

void draw() {
  noStroke(); //線をなしに設定

  fill(255, 255, 255, 30); //塗りの色を白(背景色)の半透明に設定
  rect(0, 0, width, height); //背景色でウィンドウを塗りつぶす(ウィ
ンドウをリセット)

  fill(0, 0, 0, 255); //円の色を黒に設定
  ellipse(mouseX, mouseY, 50, 50); //マウスの位置に円を描画
}

```

[マウス操作の基本]

Processing では、現在のマウスの座標を取得するには以下のように記述します。

```

mouseX 現在のマウスの X 座標
mouseY 現在のマウスの Y 座標

```

練習 1

1-1

マウスの位置によって色を変えてみましょう。

[\[練習 1-1 Processing 実行画面サンプル\]](#)

1-2

マウスの左ボタンが押されている時だけ、円が描写されるようにプログラミングしてください。

[Hint]

マウスが押されているかどうかは、mousePressed で取得することができます。

<http://processing.org/reference/mousePressed.html>

[\[練習 1-2 Processing 実行画面サンプル\]](#)

1-3

マウスのスピードによって円の大きさを変えてみましょう。

[Hint]

マウスの1フレーム前の座標を取得するには以下のように記述します

pmouseX 1フレーム前のマウスの X 座標

pmouseY 1フレーム前のマウスの Y 座標

<http://processing.org/reference/pmouseX.html>

<http://processing.org/reference/pmouseY.html>

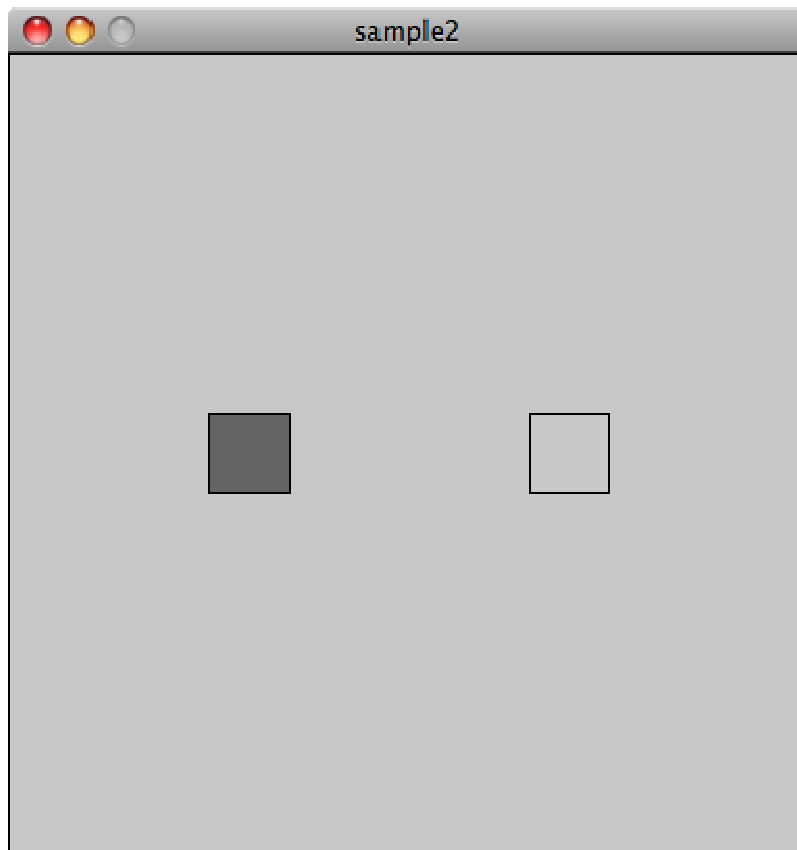
2点間(ポイント A,ポイント B)の距離を求めるには以下のように記述します。

dist(ポイント A の x 座標, ポイント A の y 座標, ポイント B の x 座標, ポイント B の y 座標);

http://processing.org/reference/dist_.html

[\[練習 1-3 Processing 実行画面サンプル\]](#)

■ サンプルプログラム 2 ■



[サンプルを](#)

[表示](#)

```
float button1_x;    //左のボタンの x 座標
float button1_y;    //左のボタンの y 座標

float button2_x;    //右のボタンの x 座標
float button2_y;    //右のボタンの y 座標

float button_scale; //ボタンの大きさ

float bgColor;      //背景の色

void setup() {
  size(400, 400);    //ウィンドウのサイズを決定(横幅、縦幅)
  colorMode(RGB, 255); //カラーモードを設定(モード、最大値)
```

```

background(255, 255, 255); //背景の色を指定 (R, G, B) この場合は白
frameRate(30); //フレームレートを設定 (fps:frame per
seconds)

button1_x = 100; //ボタンの場所を設定
button1_y = height/2;

button2_x = 260;
button2_y = height/2;

button_scale = 40; //ボタンの大きさを設定

bgColor = 255; //背景色の初期化
}

void draw() {

stroke(0); //線を黒色に設定

fill(bgColor); //塗りの色をを bgColor に設定
rect(0,0, width , height ); //背景色でウィンドウを塗りつぶす

fill(100); //左のボタンの色を設定
rect(button1_x, button1_y, button_scale, button_scale); //左のボタ
ンを作成

fill(200); //右のボタンの色を設定
rect(button2_x, button2_y, button_scale, button_scale); //右のボタ
ンを作成

//マウスの位置が左のボタンの上にあるかどうかを判断

if ((mouseX >= button1_x) && (mouseX <= button1_x + button_scale)
    && (mouseY >= button1_y) && (mouseY <= button1_y + button_scale))
{

```

```

    bgColor = 100;                //背景の色を設定し直す。
}

//マウスの位置が右のボタンの上にあるかどうかを判断

else if((mouseX >= button2_x) && (mouseX <= button2_x + button_scale)
        && (mouseY >= button2_y) && (mouseY <= button2_y + button_scale))
{
    bgColor = 200;                //背景の色を設定し直す。
}
}

```

[マウスの位置による判定]

今回のサンプルでは、マウスがどこにあるのかを判断をしています。

マウスがボタンの上にあるときを検出したい場合は、

マウスの座標が、ボタンの枠内(四角形の内側)にあるかどうかを判断すれば可能となります。

このとき、重要なのは、x 座標と、y 座標をそれぞれ別々に判断し、if 文(条件分岐)の中で && を用いて範囲を特定することです。

サンプルでの

(mouseX >= button1_x) && (mouseX <= button1_x + button_scale)とは、
(マウスの x 座標 >= ボタンの右端) かつ (マウスの x 座標 <= ボタンの左端) という
ことです。

同じように、&& で上端、下端を設定することで、マウスがボタンの上にあるかどうか
が分かります。

練習 2

2-1

ボタンを3つにし、背景色を、赤、緑、青に切り替えられるようにしてください。

[Hint]

float bgColor; の代わりに

```
float redColor;  
float greenColor;  
float blueColor;
```

など、3つの変数を用意してください。

[\[練習 2-1 Processing 実行画面サンプル\]](#)

2-2

ボタンをクリックしたときに背景色が切り替わるようにしてください。

[\[練習 2-2 Processing 実行画面サンプル\]](#)

■ サンプルプログラム 3 ■



[サンプルを](#)

[表示](#)

PFont myFont; //フォントを格納するための箱を用意します。

```
void setup() {
  size(400, 400); //ウィンドウのサイズを決定(横幅、縦幅)
  colorMode(RGB, 255); //カラーモードを設定(モード、最大値)
  background(255, 255, 255); //背景の色を指定(R, G, B) この場合は白
  frameRate(30); //フレームレートを設定(fps:frame per
seconds)
  noStroke(); //線をなしに設定

  myFont = loadFont("HelveticaNeue-48.vlw");
}
```

```

void draw() {

    fill(255, 255, 255, 50);           //背景の塗りを白の半透明に設定
    rect(0, 0, width, height);       //背景色でウィンドウを塗りつぶす

    fill(100);                         //テキストの色をグレーに設定

    textFont(myFont, 24);             //フォント、フォントの大きさを指定
    text("press any key ", 220, height-10); //ナビゲーション用のテキストを表示

    textFont(myFont, 48);             //フォント、フォントの大きさを指定

    //キーボードが押されている場合
    if (keyPressed) {
        text(key, random(width), random(height)); //押されたキーの文字を画面上の任意の点に描写
    }
}

```

[テキスト描写の基本]

Processing で文字を描写したいとき、いくつか注意点があります。

[前準備]

processing では好きなフォントを使用できます。

どのフォントを使用するか指定するために、フォントを格納する箱を用意する必要があります。

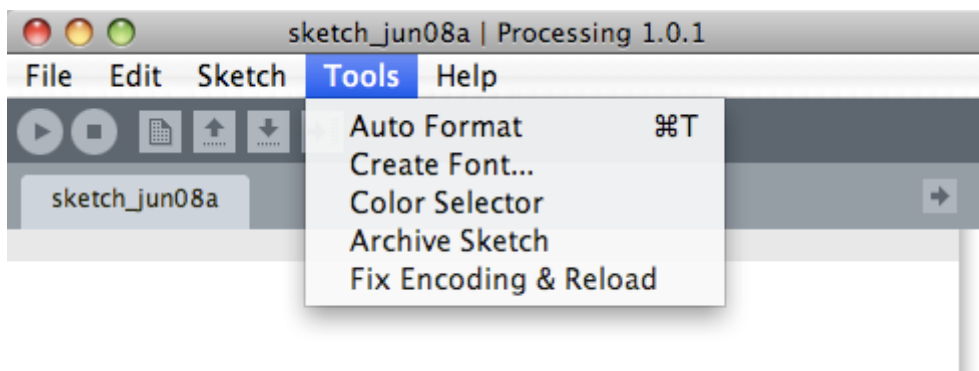
PFont 任意の名前;(サンプル1行目)

PFont の後に、任意の名前で箱を作成してください。変数を作るときと同じです。

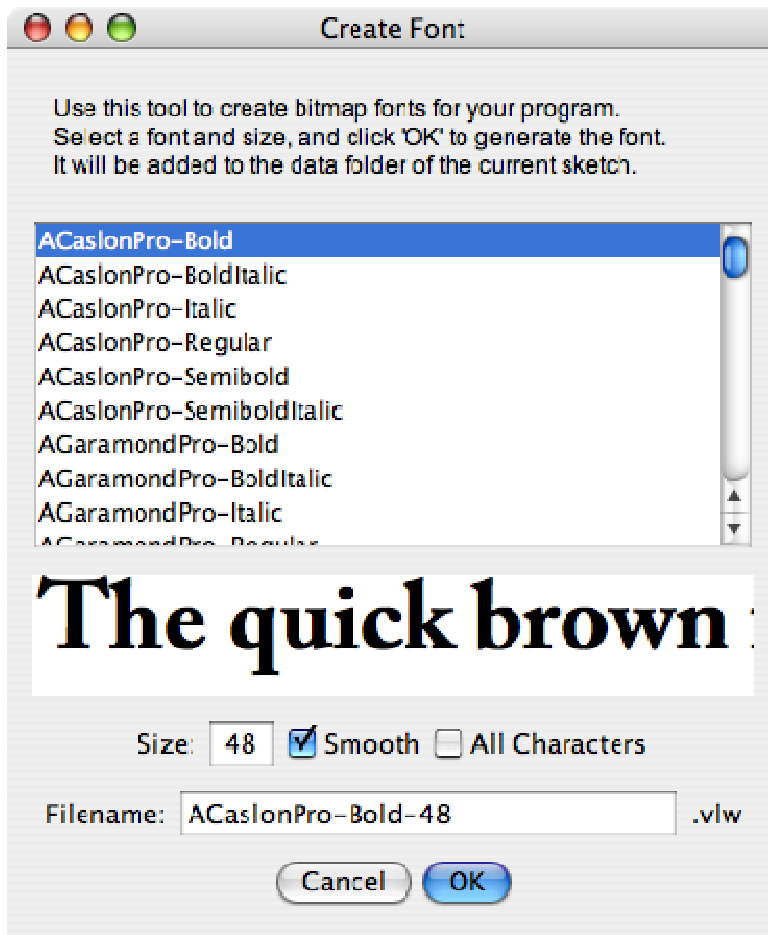
<http://processing.org/reference/PFont.html>

次に、用意した箱に、フォントのデータを埋め込んであげます。
プログラミングの前に、以下の作業を行いフォントデータを読み込みましょう。

まず、processing のメニューの「Tools」→「Create Font...」を選択します。



すると、以下のような画面が出てきますので、使用したいフォントを選んでください。



このとき、選択した Filename をコピーしておいてください。
その後、先ほど用意した箱に使用するフォントを格納します。

上で作成したフォントの箱 = loadFont(“フォントのファイル名”);(サンプル10行目)

読み込み作業で選択した Filename をそのまま書いてください。(.vfw を忘れずに)
http://processing.org/reference/loadFont_.html

[テキストの描写]

テキストの描写では基本的に以下の2つの分が必要となります。

textFont(フォントの箱,フォントのサイズ);(サンプル20行目)

text(表示したいテキスト,x 座標,y 座標);(サンプル21行目)

まず、textFont で使用したいフォントを指定し、フォントの大きさを設定します。

http://processing.org/reference/textFont_.html

次に、text で表示する内容と、表示する場所を設定すれば OK。

http://processing.org/reference/text_.html

[まとめ]

テキスト描写には以下の5つのプロセスが必要になります。

「Tools」→「Create Font...」から使用するフォントを選択

PFont 任意の名前;(サンプル1行目)

フォントの箱 = loadFont(“フォントのファイル名”);(サンプル10行目)

textFont(フォントの箱,フォントのサイズ);(サンプル20行目)

text(表示したいテキスト,x 座標,y 座標);(サンプル21行目)

練習 3

3-1

上下ボタンでフォントの大きさを変えられるようにしてください。

[Hint]

矢印キーのような特殊なキーの取得は keyCode を用いると簡単にできます。

<http://processing.org/reference/keyCode.html>

[\[練習 3-1 Processing 実行画面サンプル\]](#)

課題

Processing で自由なアニメーションを作りを行い、前回同様 Java アプレットとして自分のウェブに載せましょう。

授業で習っていない描画方法などを使用していただいても結構です。(参考 URL[\[Processing.org Reference\]](#))

提出締切 : 6/10 (木) 23:59

課題の提出方法

課題を個人のディレクトリに以下の URL になるように配置します。

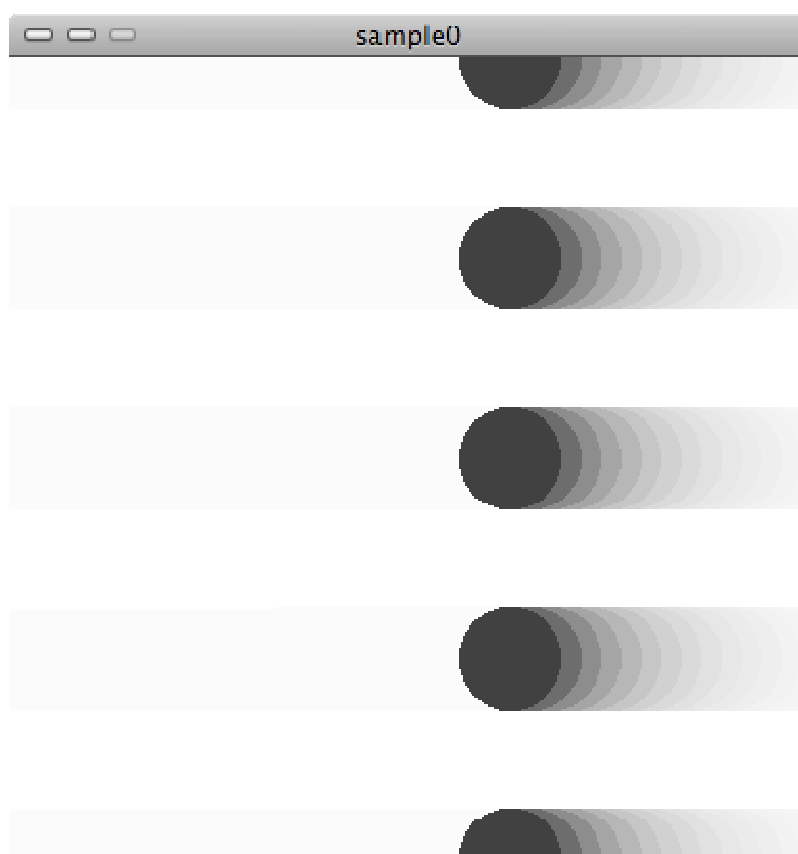
Processing [第4回]

配列

・配列の理解

[配列の有効性]

「配列」ってなに？その前に、配列をどういう時に使いたくなるか、実際に例を出して説明します！



[サンプルを表示](#)

上のサンプルは、[第7回講義の課題 2-2 の運動](#)を5つ並べたものです。

まだ良く理解出来ていない人は、SA を捕まえて復習してしまいましょう。

さて、最初のサンプルのように、5つ並べたいときはどうしましょう。

今まで学んできたことでやると、以下ようになります。

```
int x0=0; //x 座標の初期値を設定(グローバル変数)
int x1=0; //x 座標の初期値を設定(グローバル変数)
int x2=0; //x 座標の初期値を設定(グローバル変数)
int x3=0; //x 座標の初期値を設定(グローバル変数)
int x4=0; //x 座標の初期値を設定(グローバル変数)

int speed0=10; //速度を設定(グローバル変数)
int speed1=10; //速度を設定(グローバル変数)
int speed2=10; //速度を設定(グローバル変数)

int speed3=10; //速度を設定(グローバル変数)
int speed4=10; //速度を設定(グローバル変数)

void setup() {
  size(400, 400); //ウィンドウのサイズを決定(横幅、縦幅)
  colorMode( RGB, 255); //カラーモードを設定(モード、最大値)
  background(255, 255, 255); //背景の色を指定(R, G, B) この場合は白
  frameRate(30); //フレームレートを設定(fps:frame per
seconds)
}

void draw() {
  noStroke(); //線をなしに設定

  fill(255, 255, 255, 50); //塗りの色を白(背景色)に設定
  rect(0, 0, width, height); //背景色でウィンドウを塗りつぶす(ウィ
ンドウをリセット)

  fill(50, 50, 50, 255);
```



```

ellipse(x0, 0, 50, 50); //座標(x0, height/2)に円を描画
ellipse(x1, 100, 50, 50); //座標(x1, height/2)に円を描画
ellipse(x2, 200, 50, 50); //座標(x2, height/2)に円を描画
ellipse(x3, 300, 50, 50); //座標(x3, height/2)に円を描画
ellipse(x4, 400, 50, 50); //座標(x4, height/2)に円を描画

if (x0 < 0 || x0 > width) { //x0 が画面からはみ出したら
    speed0 = -1*speed0; //円の進む方向を逆方向にする。
}
if (x1 < 0 || x1 > width) { //x1 が画面からはみ出したら
    speed1 = -1*speed1; //円の進む方向を逆方向にする。
}
if (x2 < 0 || x2 > width) { //x2 が画面からはみ出したら
    speed2 = -1*speed2; //円の進む方向を逆方向にする。
}
if (x3 < 0 || x3 > width) { //x3 が画面からはみ出したら
    speed3 = -1*speed3; //円の進む方向を逆方向にする。
}
if (x4 < 0 || x4 > width) { //x4 が画面からはみ出したら
    speed4 = -1*speed4; //円の進む方向を逆方向にする。
}

x0=x0+speed0;
x1=x1+speed1;
x2=x2+speed2;
x3=x3+speed3;
x4=x4+speed4;
}

```

赤い部分のように、描写する円の数だけ対応する変数を増やしてあげれば出来ますよね。

それでは、円の数 を 10 個にしてみましょう！

それができたら、円の数 を 100 個にしてみましょう！

それができたら、円の数 を 1000 個にしてみましょう！

...

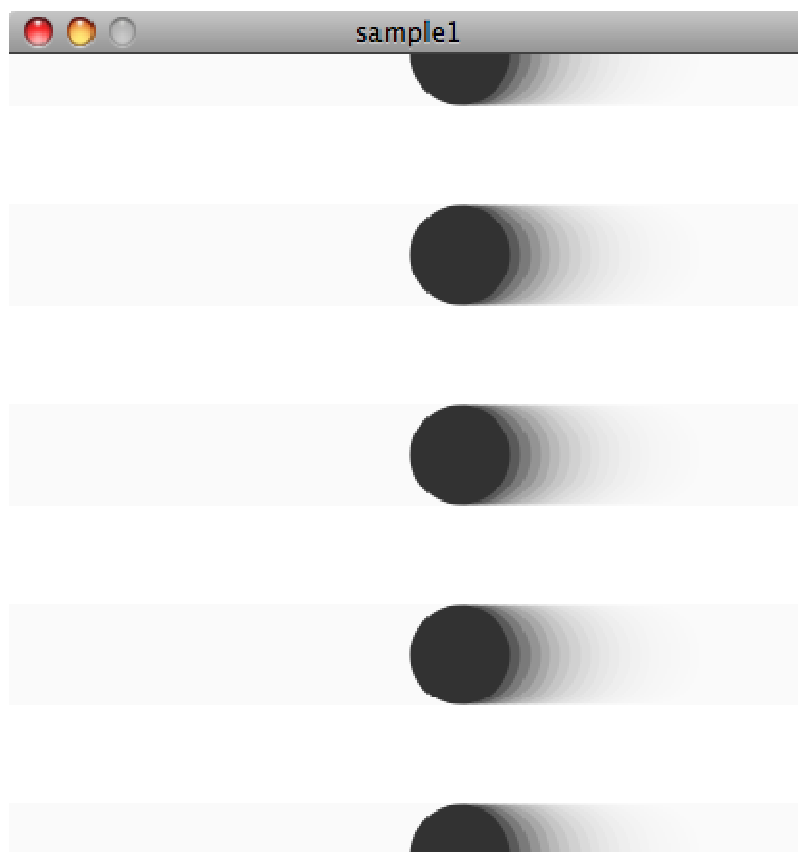
嗚呼、それはさすがに大変です。

上のソースでは、赤い部分はほとんど同じ事を書いています。

こういった、x0,x1,x2,x3...といったまとまったデータを扱うときに便利なものが配列です！

配列を使用すると、上のプログラムがどうなるか示してみましよう。

■ サンプルプログラム ■



[サンプルを](#)

[表示](#)

```
int cNum = 5;           //円の数を保存する変数
```

```

int[] x = new int[cNum];           //x 座標を保存する配列
準備
int[] speed = new int[cNum];      //速度を保存する配列を
準備

void setup() {
    size(400, 400);               //ウィンドウのサイズを決定(横幅、縦幅)
    colorMode( RGB, 255);         //カラーモードを設定(モード、最大値)
    background(255, 255, 255);    //背景の色を指定(R, G, B) この場合は白
    frameRate(30);               //フレームレートを設定(fps:frame per
seconds)

    for(int k = 0;k < cNum; k++){
        x[k] = 0;                //x 座標を初期化
        speed[k] = 5;           //スピードを設定(初期化)
    }
}

void draw() {
    noStroke();                  //線をなしに設定
    smooth();
    fill(255, 255, 255, 50);     //塗りの色を白(背景色)に設定
    rect(0, 0, width, height);  //背景色でウィンドウを塗りつぶす(ウィ
ンドウをリセット)

    fill(50, 50, 50, 255);
    for(int i = 0;i < cNum; i++){
        ellipse(x[i], i*height/(cNum-1), 50, 50); //座標(x, height/2)に
円を描画

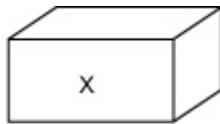
        if(x[i] > width || x[i] < 0){
            speed[i] = -1 * speed[i];
        }
        x[i]=x[i]+speed[i];
    }
}

```

[配列のとは？]

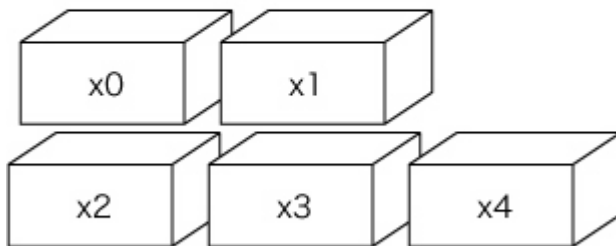
少し、シンプルになりました。コードの説明をする前に配列の説明をします。

今まで、「数や文字」を操作するために「箱」(変数)を用意する方法を学んできました。

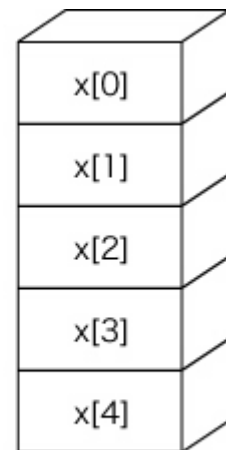


`int x;`

最初の例では、`x0,x1,x2...`と5つ別々の「箱」を用意してましたが、配列ではこの5つを一つのまとまりとして扱うことができます。



`int x0, x1, x2, x3, x4;`



`int[] x = new int[5];`

配列で扱うメリットは「**x の配列の何個目**」という使い方ができるようになることです。サンプルに沿って具体例をあげると、5つの円を描く場合、以下のように記述することができます。

```
ellipse(x0, 0, 50, 50);  
ellipse(x1, 100, 50, 50);  
ellipse(x2, 200, 50, 50);  
ellipse(x3, 300, 50, 50);  
  
ellipse(x4, 400, 50, 50);
```

→

```
for(int i = 0; i < 5; i++) {  
    ellipse(x[i], i*100, 10, 10);  
}
```

配列 x の1つ目、2つ目という概念があるため、for 文でくり順番に処理することが可能となります。

5つだと、大して労力は変わらない気もしますが、円を 100 個書きたい場合も、配列を使えば上のように3行で記述することができるのです。

便利な気がしてきたでしょう？

[processing で配列を使用する方法]

<http://processing.org/reference/Array.html>

変数を使用前に宣言したように、配列も使用する前には必ず「配列使うよ〜」と宣言しましょう。

型名[] 並列名 = new 型名[配列の数];

(例)int[] x = new int[5];

[Hint]

配列の要素(箱)を取り出した場合、

hoge = x[0]; ←要注意

のように取り出すことができます。

このとき、配列要素は 0 番から始まるため、配列要素を5つ用意した場合は、x[0],x[1],...,x[4]の5つとなり、x[5]はありません。

練習 1

1-1

円の数を10個にしてみましょう。

[\[練習 1-1 Processing 実行画面サンプル\]](#)

1-2

円の移動スピードをそれぞれ変えてみましょう。

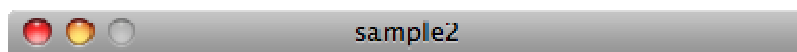
[\[練習 1-2 Processing 実行画面サンプル\]](#)

1-3

円の色をそれぞれ変えてみましょう。

[\[練習 1-3 Processing 実行画面サンプル\]](#)

■ サンプルプログラム 2 ■



[サンプルを](#)

[表示](#)

```
PVector pos, next;           //座標用の変数

void setup() {
  size(400, 400);           //ウィンドウのサイズを決定(横幅、縦幅)
```

```

    colorMode( RGB, 255 );           //カラーモードを設定 (モード、最大値)
    background( 255, 255, 255 );    //背景の色を指定 (R, G, B) この場合は白
    frameRate( 30 );               //フレームレートを設定 (fps:frame per
seconds)
    pos = new PVector( 0, 0 );
    next = new PVector( 0, 0 );
}

void draw() {
    noStroke();                    //線をなしに設定
    smooth();
    fill( 255, 255, 255, 255 );    //塗りの色を白(背景色)に設定
    rect( 0, 0, width, height );  //背景色でウィンドウを塗りつぶす(ウィ
ンドウをリセット)

    next.x = pos.x + (mouseX - pos.x)*0.1;
    next.y = pos.y + (mouseY - pos.y)*0.1;

    fill( 50, 50, 50, 255 );
    ellipse( next.x, next.y, 10, 10 );

    pos = next;
}

```

練習 2

2-1

円の数を10個にしてみましょう。

[\[練習 2-1 Processing 実行画面サンプル\]](#)

課題

Processing で配列を用いた自由なアニメーションを作りを行い、前回同様 Java アプレットとして自分のウェブに載せましょう。

授業で習っていない描画方法などを使用していただいても結構です。(参考 URL[[Processing.org Reference](http://processing.org/Reference)])

Processing [第三回]

API

すごい人たちが作った有難いライブラリを使おう。

processing には、世界中のプログラマーによって便利なライブラリが作られています。今回は、世界中のすごい人たちが作った有難いライブラリの使い方を学びましょう。

processing の本サイトでいくつかライブラリが紹介されています。

<http://processing.org/reference/libraries/index.html>

今回はライブラリの使用方法を学びましょう。

Processing にはもともといくつかのライブラリがインストールされています。

その他にも世界中にたくさん存在するライブラリをインストールすることで使用する事ができます。

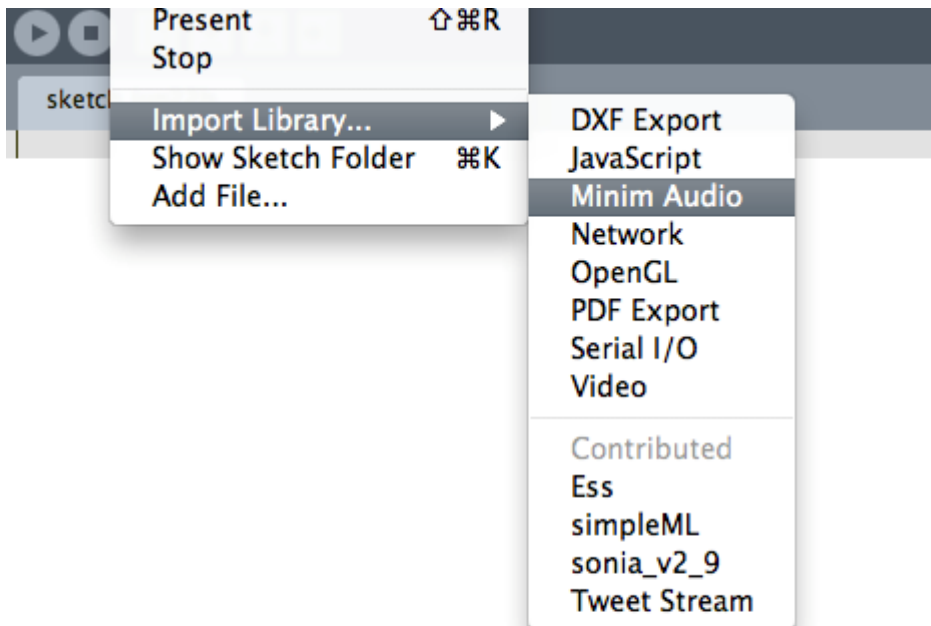
まずは、最初からインストールされているライブラリを使用してみましょう。

まずは音を扱うライブラリ「MinimAudio」です。

processing を起動してみましょう。

メニューから「Sketch」→「Improt Library」の中に、使用できるライブラリの一覧が表示されます。

その中にある「MinimAudio」を選択。



すると

```
import ddf.minim.*;
```

という1文が追加されます。

これで、ESS のライブラリを使用する準備ができました。

では、サンプルを見てみましょう。

■ サンプルプログラム ■

[サンプルを表示](#)

```
import ddf.minim.*; // minim ライブラリを import
Minim minim; // 変数定義
AudioPlayer player;
```

```
void setup() {  
  minim = new Minim(this); // インスタンス生成  
  player = minim.loadFile("test.mp3"); // ロード  
}  
  
void draw() {  
  player.play(); // 再生  
}
```

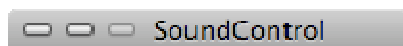
[Hint]

- mp3 ファイルが用意出来ない人は、[こちらのサンプル](#)を使って下さい。
 - 一度、名前を付けてファイルを保存して下さい。
 - そのディレクトリの中に「data」というディレクトリを作成し、その中に再生したい音楽ファイルを入れておきましょう。
-

練習 1

1-1

サンプルに再生ボタンと停止ボタンを作成せよ。



[\[練習 1-1 Processing 実行画面サンプル\]](#)

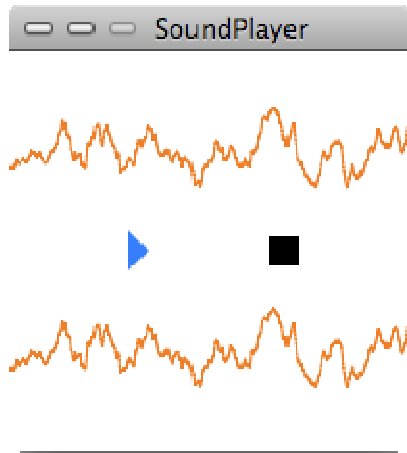
[Hint]

・[第8回のボタンの復習](#)

・停止 : 「AudioPlayer 変数名.close();」

1-2

ビジュアライザを追加せよ。



[\[練習 1-2 Processing 実行画面サンプル\]](#)

[Hint]

・ビジュアライザは以下のプログラムによって描画出来ます。

```
for(int i = 0; i < player.bufferSize()-1; i++) {  
    float x1 = map(i, 0, player.bufferSize(), 0, width);  
    float x2 = map(i+1, 0, player.bufferSize(), 0, width);  
    line(x1, 50 + player.left.get(i)*50, x2, 50 +  
player.left.get(i+1)*50);  
    line(x1, 150 + player.right.get(i)*50, x2, 150 +  
player.right.get(i+1)*50);  
}
```

ここまでくればライブラリの使い方がわかってきたかな。

世界中のすごい人達たちは、まだまだ沢山ライブラリ作ってくれました。次は、物理的な動きを表現するのに便利な「Physics」というライブラリを紹介します。

TRAER.PHYSICS : <http://www.cs.princeton.edu/%7Etraer/physics/>

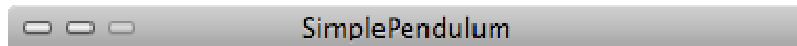
ダウンロードしたフォルダには、何やら解り難い名前がついています。この名前を「traer」(このライブラリの作者)に変えておきましょう。Processing が置いてあるフォルダの中の「libraries」というフォルダを作成してください。

作った「libraries」のフォルダの中にダウンロードしたライブラリを「traer」ごと移動するとインストール完了です。

確認

Processing / libraries / traer / physics / library / physics.jar

■ サンプルプログラム ■



[サンプルを表示](#)

```
import traer.physics.*;

ParticleSystem physics;
Particle p;
Particle anchor;
Spring s;

void setup() {
  size( 400, 400 );
  smooth();
  fill( 0 );
  ellipseMode( CENTER );

  physics = new ParticleSystem( 1, 0.05 );
```

```

    p = physics.makeParticle( 1.0, width/2, height/2, 0 );
    anchor = physics.makeParticle( 1.0, width/2, height/2, 0 );
    anchor.makeFixed();
    s = physics.makeSpring( p, anchor, 0.5, 0.1, 75 );
}

void mousePressed() {
    p.makeFixed();
    p.position().set( mouseX, mouseY, 0 );
}

void mouseDragged() {
    p.position().set( mouseX, mouseY, 0 );
}

void mouseReleased() {
    p.makeFree();
}

void draw() {
    physics.tick();
    background( 255 );

    line( p.position().x(), p.position().y(), anchor.position().x(),
anchor.position().y() );
    ellipse( anchor.position().x(), anchor.position().y(), 5, 5 );
    ellipse( p.position().x(), p.position().y(), 20, 20 );
}

```

次は今流行っている twitter を、ライブラリを使用して Processing 上で表示してみよう。

twitter の情報を扱うためには twitter の XML を読み込む必要があります。
XML とは情報交換するためのものと考えて下さい。

まず XML のライブラリをインストールします。

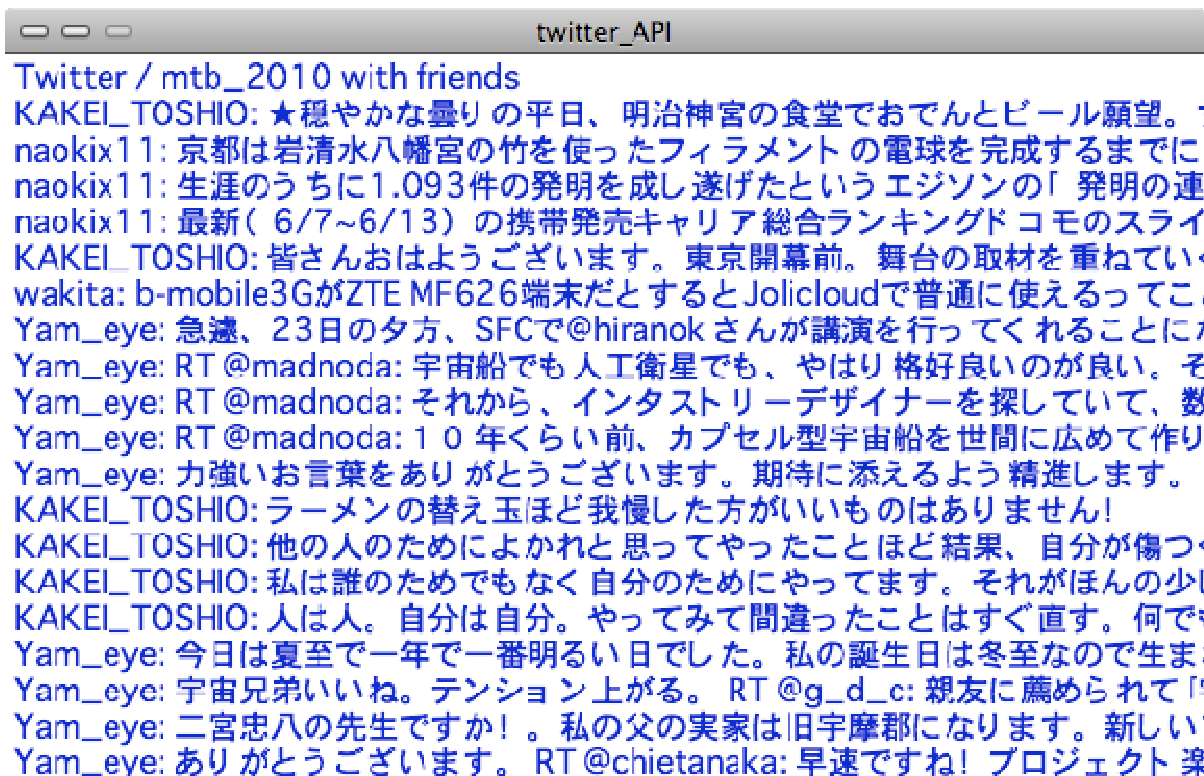
simpleML : <http://www.learningprocessing.com/downloads/>

先程作った「libraries」のフォルダの中にダウンロードした「SimpleML」を移動すればインストール完了です。

確認

Processing / libraries / SimpleML / library / SimpleML.jar

■ サンプルプログラム ■



```
twitter_API
Twitter / mtb_2010 with friends
KAKEI_TOSHIO: ★穏やかな曇りの平日、明治神宮の食堂でおでんとビール願望。
naokix11: 京都は岩清水八幡宮の竹を使ったフィラメントの電球を完成するまでに
naokix11: 生涯のうちに1.093件の発明を成し遂げたというエジソンの「発明の連
naokix11: 最新(6/7~6/13)の携帯発売キャリア総合ランキングドコモのスライ
KAKEI_TOSHIO: 皆さんおはようございます。東京開幕前。舞台の取材を重ねてい
wakita: b-mobile3GがZTE MF626端末だとするとJolicloudで普通に使えるってこ
Yam_eye: 急遽、23日の夕方、SFCで@hiranokさんが講演を行ってくれることに
Yam_eye: RT @madnoda: 宇宙船でも人工衛星でも、やはり格好良いのが良い。そ
Yam_eye: RT @madnoda: それから、インタストリーデザイナーを探していて、数
Yam_eye: RT @madnoda: 10年くらい前、カプセル型宇宙船を世間に広めて作り
Yam_eye: 力強いお言葉をありがとうございます。期待に添えるよう精進します。
KAKEI_TOSHIO: ラーメンの替え玉ほど我慢した方がいいものはありません!
KAKEI_TOSHIO: 他の人のためによかれと思ってやったことほど結果、自分が傷つ
KAKEI_TOSHIO: 私は誰のためでもなく自分のためにやっています。それがほんの少
KAKEI_TOSHIO: 人は人。自分は自分。やってみて間違ったことはすぐ直す。何で
Yam_eye: 今日は夏至で一年で一番明るい日でした。私の誕生日は冬至なので生ま
Yam_eye: 宇宙兄弟いいね。テンション上がる。RT @g_d_c: 親友に薦められて「
Yam_eye: 二宮忠八の先生ですか!。私の父の実家は旧宇摩郡になります。新しい
Yam_eye: ありがとうございます。RT @chietanaka: 早速ですね! プロジェクト楽
```

[サンプルを表示](#)

```

///// ライブラリの読み込み /////
import simpleML.*; // simpleML の読み込み
import java.net.Authenticator;
import java.net.PasswordAuthentication;

///// javaにおける http 認証 /////
String user = "username"; // ユーザーネームの入力
String pass = "password"; // パスワードの入力
String url = "http://twitter.com/statuses/friends_timeline.rss";
// RSS の URL を入力

///// クラスの定義 /////
class BasicAuth extends java.net.Authenticator {
    private String user;
    private String pass;
    public BasicAuth(String user, String pass) {
        this.user = user;
        this.pass = pass;
    }
    protected PasswordAuthentication getPasswordAuthentication() {
        return new PasswordAuthentication(user, pass.toCharArray());
    }
}

///// 初期設定 /////
XMLRequest xmlRequest;
void setup() {
    size(screen.width/2, screen.height/2);
    background(255, 255, 255);
    BasicAuth tw_authenticator = new BasicAuth(user, pass); // ユー
ユーザー情報を設定
    Authenticator.setDefault(tw_authenticator); // 認証設定
    xmlRequest = new XMLRequest(this, url);
    xmlRequest.makeRequest(); // XML のリクエストを送信
}

```



```

///// tweet を表示する関数を定義 /////
void netEvent(XMLRequest ml) {
    PFont fontA = loadFont("Osaka-16.vlw"); // フォントの読み込み
    textFont(fontA, 16); // フォントを設定
    String titles[] = ml.getElementArray("title");
    for (int i=0; i < titles.length; i++) {
        fill(0, 0, 200); // 文字色の設定
        text(titles[i].replaceAll("\n", ""), 5, 18*(i+1)); // tweet を表
示
    }
}

void draw() {
    background(255);
    netEvent(xmlRequest);
}

```

補足

フォントを設定する時に日本語対応フォントを選択。
 また All Characters にチェックを入れる。
 これを忘れると日本語が正しく表示されません。

今回、今まで使用してこなかった言葉が所々ありましたが全部理解出来なくて大丈夫です。

ライブラリを使用するとこんな事もできるんだと実感できれば OK.

課題

Processing で Minim、SimpleML、又は、他のライブラリを用いた自由なアニメーション作りを行い、

前回同様 Java アプレットとして自分のウェブに載せましょう。

授業で習っていない描画方法を使用していただいても結構です。(参考

URL[Processing.org Reference])

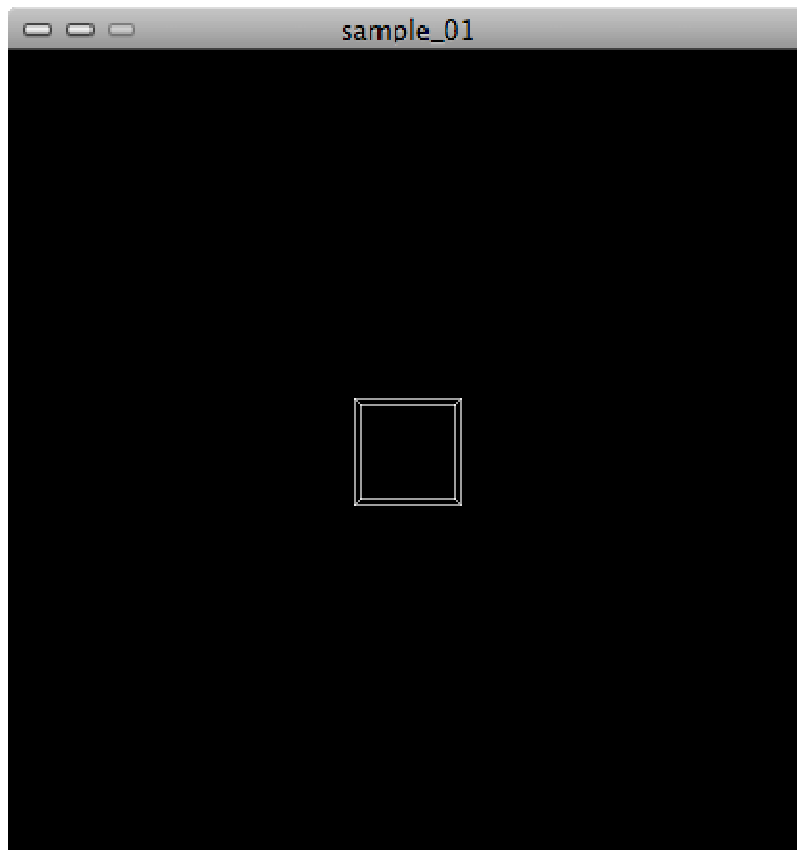
Processing [第六回]

座標, 3D

「僕、3D やりたいんです」「私、3D なんです」と言う皆様の期待に応えて、今回のテーマは、3D についてという事になりました。

最初のサンプルはとても簡単。
画面の真ん中に立方体を表示させるだけです。

■ サンプルプログラム - 1 ■



サンプルを表示

```
int sz = 50;

void setup() {
  size(400, 400, P3D);
  colorMode(RGB, 255);
  frameRate(30);
  fill(0, 0);
}

void draw() {
  background(0, 100);

  translate(width/2, height/2, 0);
```

```
stroke(255, 100);  
box(sz);  
}
```

[Hint]

- ・「[box\(size\);](#)」は、直方体の形しか指定する事は出来ないなので、必ず原点に描画されます。
- ・そこで、直方体を任意の場所に描画する為には「[translate\(x, y, z\);](#)」を使って軸を移動させる方法を探ります。

でもこれだけではあまり 3D っぽくないし、何だか物足りませんよね。
それでは、この立方体を並べてみましょう。
少しは 3D っぽく見えて来るかも知れません。

■ サンプルプログラム - 2 ■



サンプルを表示

```
int numX = 30, sz = 10;
```

```
void setup() {  
  size(400, 400, P3D);  
  colorMode(RGB, 255);  
  frameRate(30);  
  fill(0, 0);  
}
```

```
void draw() {  
  background(0, 100);  
  
  translate(width/2-sz*numX/2, height/2, 0);
```

```
for (int j=0; j<numX; j++) {
    stroke(255, 100);

    pushMatrix();
    translate(sz*j, 0, 0);
    box(sz);
    popMatrix();
}
}
```

[Hint]

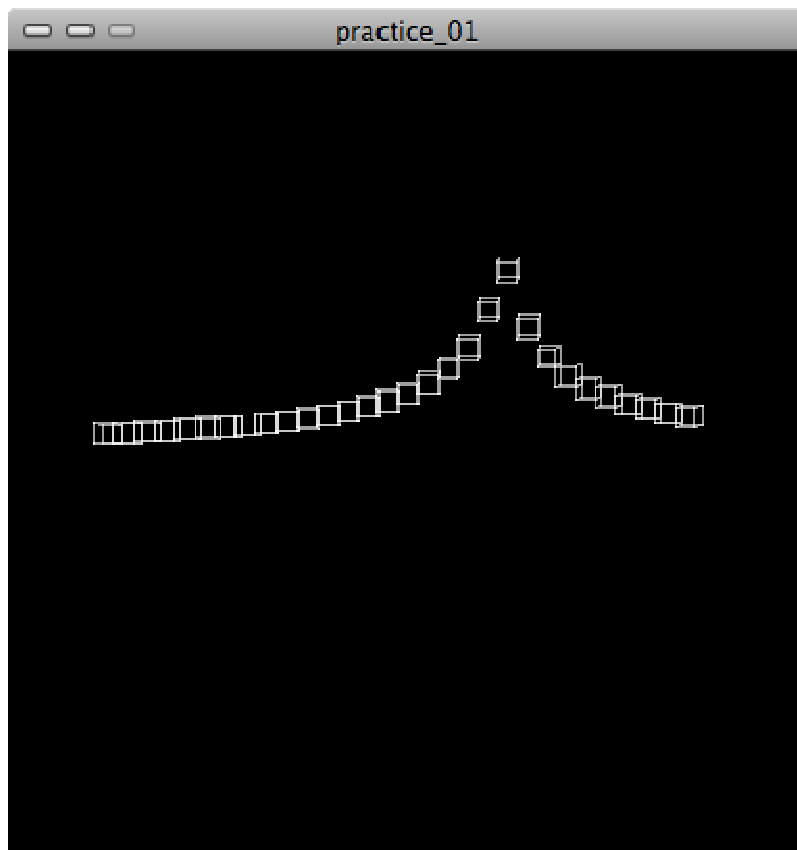
- ・並んだ立方体の内の、真ん中にある立方体が画面中央に来る様に、上手く「translate(x, y, z);」の値を調節してみましょう。
- ・1度 translate を行うと、その状態は自動では初期化されないなので、次のフレームにも適用されてしまいます。
- ・「[pushMatrix\(\);](#)」を使うと、その直後に記述された内容は「popMatrix();」後にリセットされます。
- ・translate で平行移動した座標のイメージが解らなくなってしまった人は、座標系を確認したい場所に以下のコードを入れてみましょう。

```
line(-400, 0, 0, 400, 0, 0); // DRAW X-AXIS
line(0, -400, 0, 0, 400, 0); // DRAW Y-AXIS
line(0, 0, -400, 0, 0, 400); // DRAW Z-AXIS
```

うーん、これでもやっぱりイマイチ 3D って感じがしませんね。
それでも、translate の使い方は何となく解ってきたのではないのでしょうか。
この考え方を応用して、早速今回の練習問題に挑戦してみましょう！

練習 1

マウスの X 座標に合わせて並んだ立方体の高さが変わる様にしましょう。



[\[練習 1 Processing 実行画面サンプル\]](#)

[Hint]

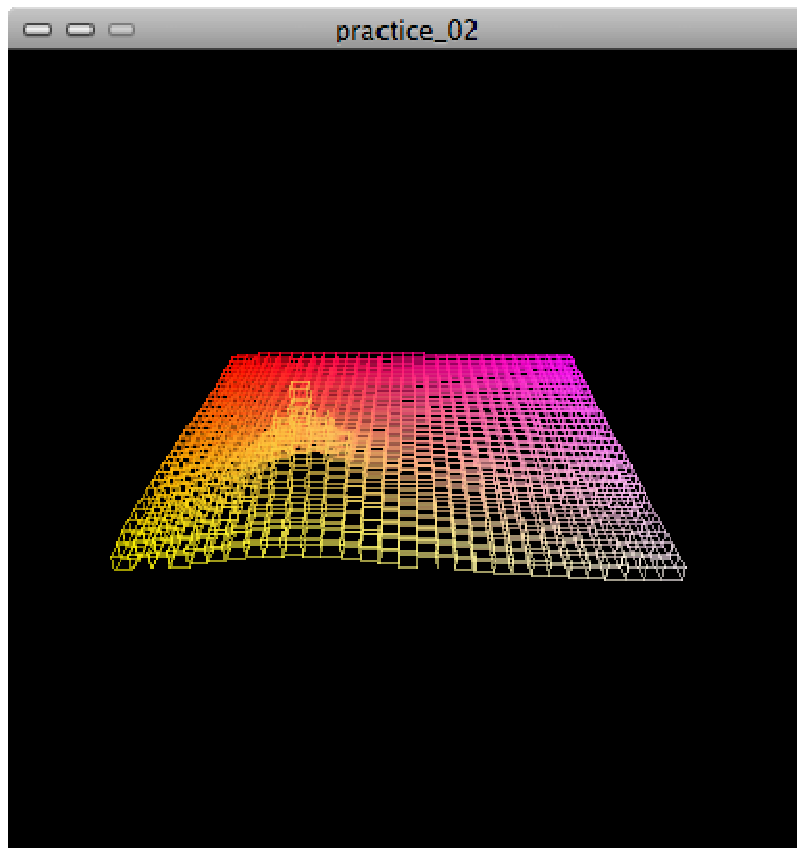
・「[dist\(x1, y1, z1, x2, y2, z2\)](#)」を使えばマウスの座標と立方体の距離を測る事が出来ます。

練習 2

立方体を平面上に配置しましょう。

マウスの Y 座標にも合わせて立方体の高さが変わる様にしてみましょう。

また、マウスを縦方向にドラッグすると、カメラの角度が変わる様にしましょう。



[練習 2 Processing 実行画面サンプル]

[Hint]

- ・「[rotate\(\);](#)」によって軸を回転させる事が出来ます。
- ・今回の場合は「[rotateX\(\);](#)」を使うと良いでしょう。

課題

Processing で 3D を用いた自由なアニメーション作りを行い、前回同様 Java アプレットとして自分のウェブに載せましょう。

授業で習っていない描画方法などを使用して頂いても結構です。(参考 :

[\[Processing.org Reference\]](#))

提出締切 : 7/1 (木) 23:59

課題の提出方法

課題を個人のディレクトリに以下の URL になるように配置します。