Part1 2DCG とプログラミング入門

ここでは、2D（平面）上の図形の描き方と、プログラミングの基礎を学びます。

1. はじめの一歩
2. Hello world 解説
3. 画面上の位置、図形の描き方
4. 色、塗りつぶし
5. 変数
6. 計算
7. for 文
8. 規則的な模様
9. ランダムな模様

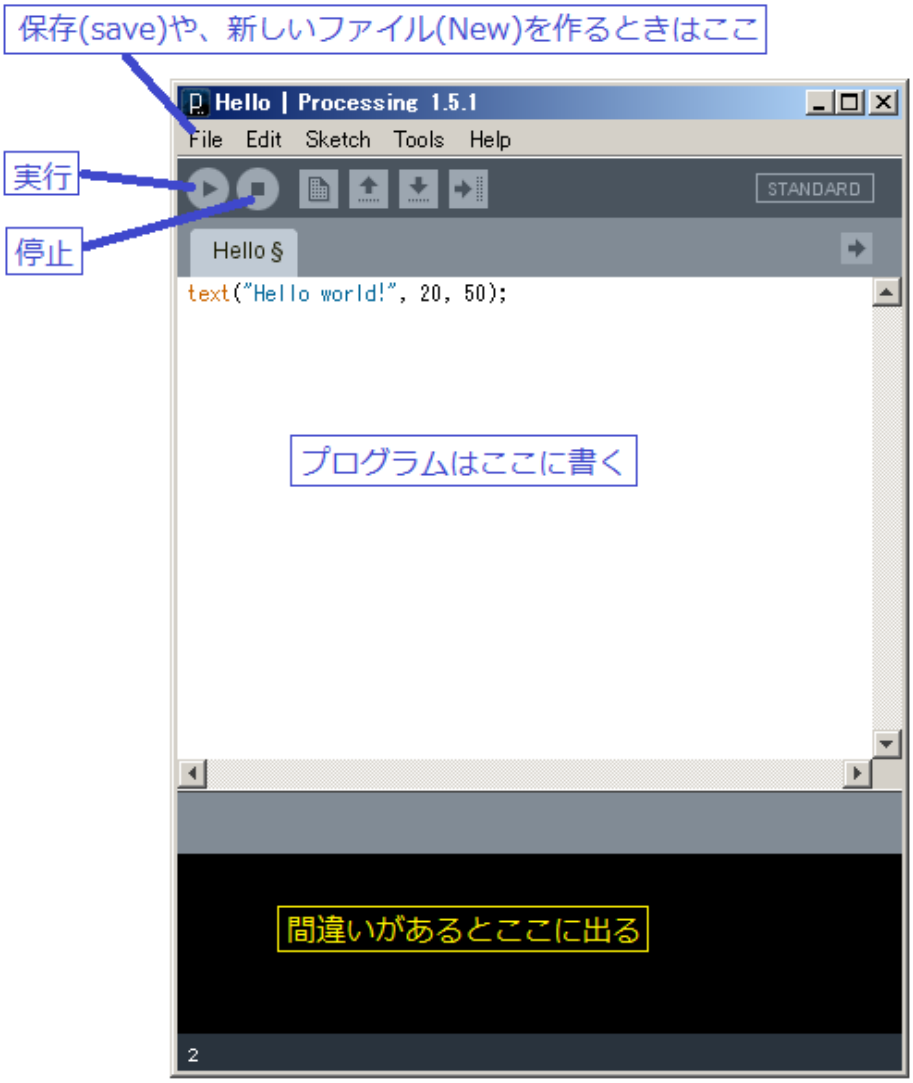
Part2 アニメーション

Part3 3DCG に挑戦

１．はじめの一歩

今日から４日間、グラフィックプログラミングを勉強していきましょう。
この講習では「Processing（プロセッシング）」というプログラミング言語を使います。

さっそく最初のプログラムを作って、実行してみましょう。



・上のメニューから「File」を選んでクリック→「New」を選んで**新しいファイルを作成**します
・真ん中の白いところに、上のとおりにプログラムを**入力**しましょう
・**保存**します。「File」から「Save」を選んで保存しましょう。
　ファイル名はわかりやすいように「Part1_01_Hello」などにします。
・**実行**します。もしバグ（間違い）があると下に赤い文字で色々書かれます。
　一文字でも違っていると動かない時もあるので注意。
・うまくいくと、画面が開いて真ん中に「Hello world!」と表示されます。

ね、簡単でしょ！

２．Hello world 解説

今書いた Hello プログラムは、「文」と「命令」と「データ」から成っています。

セミコロン「；」で終わっているのが「**文**」で、プログラムは基本的に文のあつまりでできています。何もしなければ上に書いた文から順番にコンピュータが実行していきます。

文はいくつかの「**命令**」から成り立っています。このプログラムでは text 命令だけです。

命令は、命令の名前の後にカッコがあって、カッコのなかにカンマ「, 」で区切られた幾つかの**引数**が入ります。text 命令は、1 番目の引数の内容を、ウィンドウの左端から「2 番目の数字」、上端から「3 番目の数字」の位置に表示するという命令です。

命令の引数は、何番目が文字列で、何番目には数字が入るといった決まりがあります。“Hello world”は文字列、20 や 50 はもちろん数です。

・練習

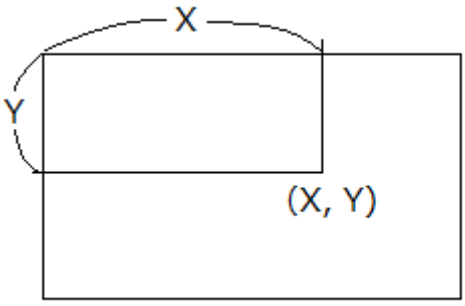
(1) Hello world とは違う文章を表示してみましょう

(2) 表示する位置を変えてみましょう。はみ出すと見えなくなるので注意。ウィンドウの横幅、縦幅は今は 100 です。

(3) 3 か所の違う位置に、別々の文書を表示してみましょう。

作ったら保存することを忘れずに！

３．画面上の位置、図形の描き方



コンピュータの画面は、小さな光の点が集まってできています。画面に近づいてじっと見ると、点々がわかる（かもしれない）はずです。

画面上の位置は、左端からＸ番目、上端からＹ番目の点（X, Y)のように表します。数学の授業でグラフを習った人は、数学とはＹ座標がひっくり返っていることに注意してください。

図形を書く命令を使ってみよう。

size(W, H)命令
　　画面の横幅をW,縦幅をＨにする。

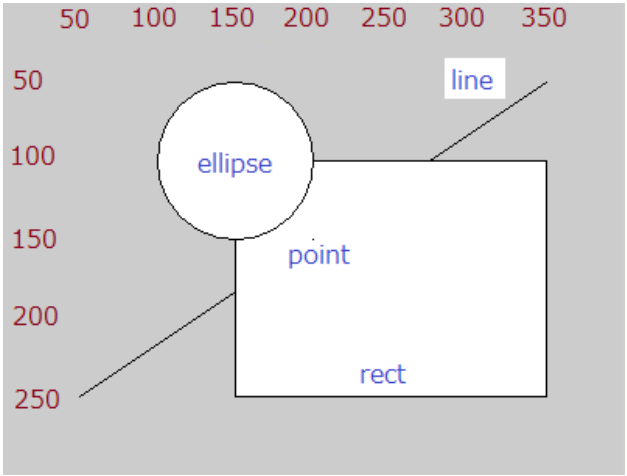
point(X, Y)命令
　　(X, Y)に点を書く

line(X0, Y0, X1, Y1)
　　(X0, Y0)から(X1, Y1)に線を引く。

rect(X, Y, W, H)
　　(X, Y)を左上の頂点とした、横W、高さＨの長方形を描く。

ellipse (X, Y, W, H)
　　(X, Y)を中心とした、横W、高さＨのだ円を描く。

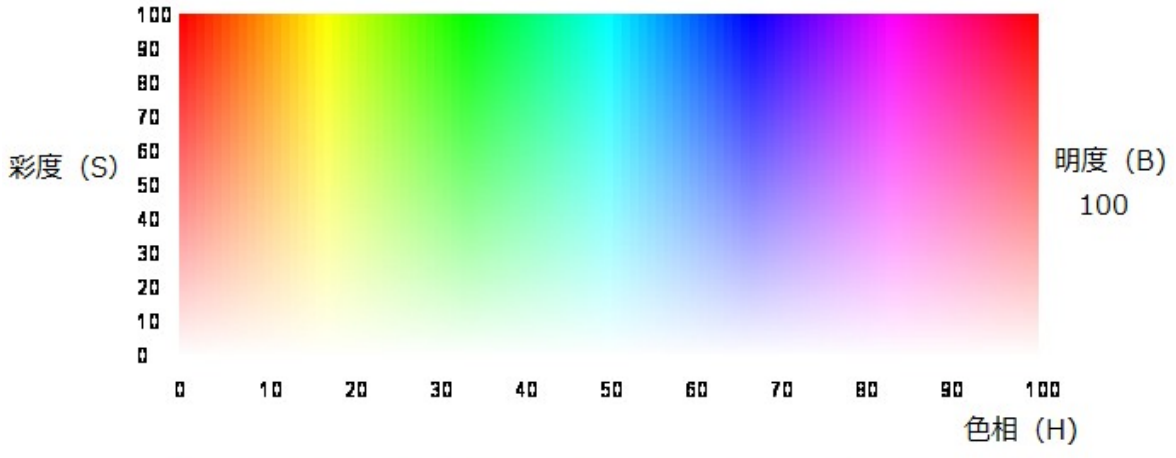
```
size(400, 300);
line(350, 50, 50, 250);
rect(150, 100, 200, 150);
ellipse(150, 100, 100, 100);
point(200, 150);
```

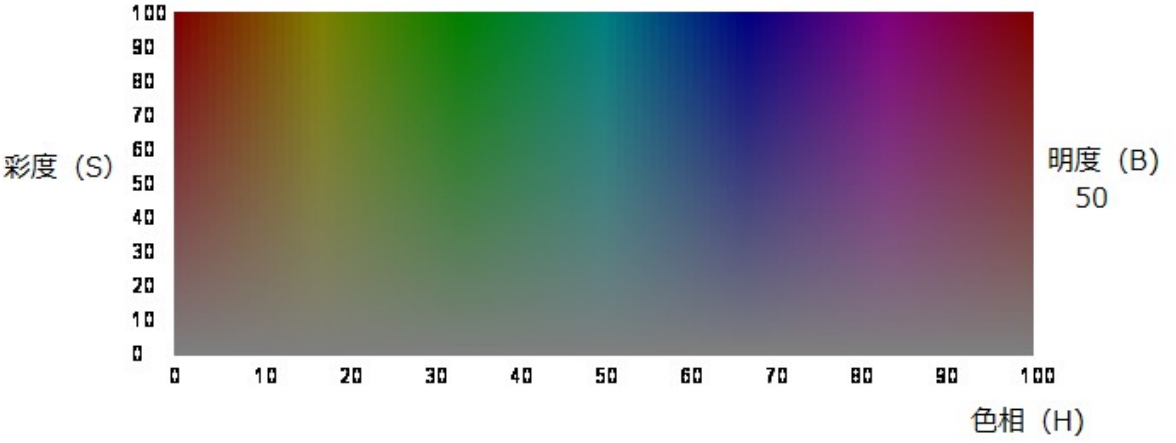


新しいファイルを作って上のプログラムを書いてみましょう。上の図のような絵が表示されれば正解です。ファイル名を「Part1_02_zukei」などにして保存しましょう。

・見にくいですが、(200,150)の位置にある小さい黒い点が、一つの光の点の大きさです。
・後に書いた文ほど上に描かれることに注意
・line 命令を３つ使って三角形を書いてみましょう

４．色、塗りつぶし





代表的な色

赤(0, 100, 100)
黄(18, 100, 100)
緑(25, 100, 60)
青(66, 100, 100)
黒(0, 0, 0)
灰(0, 0, 50)
白(0, 0, 100)

次は色をつけましょう。

色の表し方は色々ありますが、今回はHSBという方式を使います。HSBでは、色を
　　H：色相・・・赤、黄といった色合いを決める
　　S：彩度・・・鮮やかさを決める。これが低いほどグレーに近づく
　　B：明度・・・明るさを決める。これが高いほど明るくなる
の３つの数字で表します。どういう数字にするとどの色になるかは上の図を見てください。

色の付け方は、
　　stroke(H, S, B) ・・・りんかく線の色をH,S,Bで指定
　　fill(H, S, B)・・・塗りつぶしの色をH,S,Bで指定
　　noStroke()・・・りんかくなし
　　noFill()・・・塗りつぶしなし
の４つの命令があります。

どの命令も、そこから後に描く図形全てに影響するので、３つの赤い塗りつぶしの図形を書きたければ、最初に一回だけfill(0, 100, 100)と書くだけでOKです。

ついでに、widthとheightを覚えましょう。これらはそれぞれ「画面の横幅」と「画面の縦幅」を表します。前のページでやったように、size命令でこれらは指定されているから、
　　size(400, 300)
と書いたら、widthが400、heightが300です。
では、次のプログラムを書いて、練習しよう。

```
size(500, 500); //画面の大きさを500*500に指定
colorMode(HSB, 100, 100, 100);  //色をHSBで指定することにする

stroke(66, 100, 100);          //りんかく（線）の色を青に
line(0, 0, width, height);     //(0, 0)から(width, height)に線を引く。
                               //width、heightは画面の横、縦幅のことなので、対角線になる

noStroke();          //りんかく無し
fill(25, 100, 60);  //塗りつぶしを緑に
rect(0, 0, width/2, height);  //(0,0)を左上とする、幅がwidthの半分、高さheightの長方形を描く。
                              //「width/2」は横幅を２で割るという意味。
                              //これにより画面の左半分が緑色になる

stroke(0, 100, 100);  //輪郭を赤
noFill();       //塗りつぶし無し
ellipse(width/2, height/2, 50, 50);  //画面の中心を中心とする、直径50の円を描く
```

ファイル名は、「Part1_03_color」などにして保存しよう。実行したら、左半分が緑、中心に円、右下に青い直線が見えるはずだ。
これができたら、このプログラムを参考にして次の絵を描いてみよう。

信号機（黄、青は明度を下げる）

スイスの国旗

中の十字はりんかくが無いことに注意

虹（少しずつ色を変えた直線を10本くらい並べる。コピー＆貼付けを使わないと大変なことになるので注意）

某機械猫の（著作権的にやばそうなので）一部。
進みの速い人は挑戦してみよう。
楽勝って人はヒゲと口も描こう。

５．変数

前回 width、height という、「文字なんだけど数字を表してるもの」が出てきました。
これを「変数」といいます。変数には数などを保存しておくことができるのです。

変数を作るには、次のように書きます。

```
float a;
```

float は小数が入る変数という意味で、a という名前の変数がこれで作られます。
変数に数を記録するには、「＝」記号を使います。この操作を「代入」といいます。

```
a = 100;
```

同じ変数に二回代入すると、一回目に代入した数は消えてしまいます。

さて、平行な直線３本を描くことを考えてみましょう。

line 命令には始点、終点の X,Y 座標を表す４つの引数が必要だから、今までの描き方だと、合わせて 12 個の数を書かないといけない。これは面倒だし、間違える可能性も上がる。
ここで、変数が使えます。
一番目の直線の始点を(x0, y0)， 終点を(x1, y1)としよう。また、直線の x 方向（左右方向）の間隔を d としよう。
すると、二本目の直線は(x0+d, y0)から(x1+d, y1)に引かれたものであることがわかる。
三本目は(x0+2*d, y0)から(x1+2*d, y0)である（*記号はかけ算の意味）。
これで 12 個の数ではなく、x0, y0, x1, y1, d の 5 つの変数で同じものが書けることになります。

```
float x0;  //小数が入る変数x0を作る
float x1, y0, y1;  //何度もfloatと書くのは面倒なので、「,」で区切っていくつも同時に変数を作れる。
float d;  //これには直線の間隔を記録することにしよう。

x0 = 10;
y0 = 10;
x1 = 20;
y1 = 30;
d = 10;
line(x0, y0, x1, y1);  //一本目の直線。(x0, y0)から(x1, y1)に直線を引く

x0 = x0 + d;  //x0に、今のx0にdを足したものを記録しなおす。つまり、x0は10増える。つまり、始点が右に10動く。
x1 += d;      //上の計算は、このように略して書いても良い。終点も右に10動く。
line(x0, y0, x1, y1);  //二本目の直線。この部分はさっきと同じなのでコピー＆ペースト

x0 += d;
x1 += d;
line(x0, y0, x1, y1);  //三本目
```

これをプログラムにすると、次のようになります。ファイル名「Part1_04_hennsuu」として作成しよう。これから、特に指示しませんが、プログラムが出てきたら、**適当なファイル名をつけて作ってください。**
変数に入れる値を変えて、描かれる図形の変化を観察しよう。変数 d の値を増やすと直線の間隔はどうなるだろうか？

６．計算

　いままで足し算とわり算が出てきましたが覚えてますか？ここでは計算をする方法をまとめます。なんといってもコンピュータの強みは計算力なのです。

・基本となる計算は次の記号を使います。

足し算：　+
引き算：　-
かけ算：　*
わり算：　/　　（分数のイメージ。1/2 は 2 分の 1 っぽく見える）
かっこ：（　）

・変数に対しては、次の特別な記号があります。

```
X = 5;   :代入
X += 5;  :X に 5 を足す(-,*,/も同様)
X++;     :X に 1 を足す
X--;     :X から 1 を引く
```

・二つの数、変数を比較することもできます。「X が Y より大きかったら～する」みたいな処理で使います。

```
X == Y   :等しい
X != Y   :等しくない
X <  Y   :X が Y より小さい
X <= Y   :X が Y 以下
X >  Y   :X が Y より大きい
X >= Y   :X が Y 以上

(X==Y) && (Y==Z)   :X が Y と等しく、かつ、Y が Z と等しい
(X==Y) || (Y==Z)   :X が Y と等しい、または、Y が Z と等しい
```

・整数と小数

　float で小数型の変数を作るというのをやりました。
int と書くと、整数型の変数が作れます。整数型の変数には小数を入れることができません。
また、X、Y を整数、F を小数としたとき、

```
X+Y :整数     X+F :小数
```

となります。

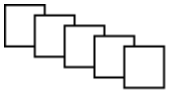
・次のプログラムを書いてみましょう。println 命令は、() の中に書かれたものを、下の黒いところに出力するという命令です。

```
println(1+2+3);
println(1+2*3+4);
println((1+2)*(3+4));
println(10/3);
println(10.0/3);
println(10.0/3.0);
println(PI/2);
```

1+2*3+4 は、2*3 が先に実行されます。1+2 を先に計算したかったら、カッコでくくりましょう。
10/3 は 3 になって、10.0/3 は 3.333333 になります。整数同士の割り算の場合、小数部分は切り捨てられます。10.0 のように、**.0 をつける**だけで小数扱いになるので覚えておきましょう。

７．for 文

「（少しずつ違った）同じような命令をなんどもくり返す」とき、同じような文をその回数書くのはなんか無駄な気がします。こういったときに使うのが「for」構文です。



たとえばこんな図を描いてみましょう。幅 20 の正方形が横方向に間隔 15、縦方向に間隔 5 で重なっています。

```
rect(0*15, 0*5, 20, 20);
rect(1*15, 1*5, 20, 20);
rect(2*15, 2*5, 20, 20);
rect(3*15, 3*5, 20, 20);
rect(4*15, 4*5, 20, 20);
```

と書けば確かにこの図が描けます。しかし、四角形が 100 個とかだったらやってられません。そこで、**for 文**を使います。

```
for(int i=0; i<5; i++){
  rect(i*15, i*5, 20, 20);
}
```

このように書くと、上の rect ５つと同じ意味になります。0,1,2,3,4 となってた部分が、変数「i」に変わっていることに注目してください。for の中の 5 を変えれば、100 個でも 1000 個でも描くことができます。便利でしょう？

for 文のカッコの中には３つの部分があり、その後の中カッコ {} の中に書かれた命令を繰り返します。

```
for(　　;　;　)
```

この３つの部分は、次のような役割がある。

１つ目：**くり返しの前に行う命令**。「int i」は整数しか入らない変数「i」を作るという意味。Float は小数の変数だったっことを思い出して。
「int i = 0」と書くと、整数の変数 i を作って、0 を記録する、という意味になる。
２つ目：**くり返しを行う条件**。「i<5」は、i が 5 より小さいという意味。他には「i<=5」：i が 5 以下、「i>=5」：i が 5 以上などがある。
３つ目：**一回くり返すごとに行う命令**。「i++」は i に 1 を足すという意味。
「i=i+1」や、「i+=1」と書いても同じ意味になる。

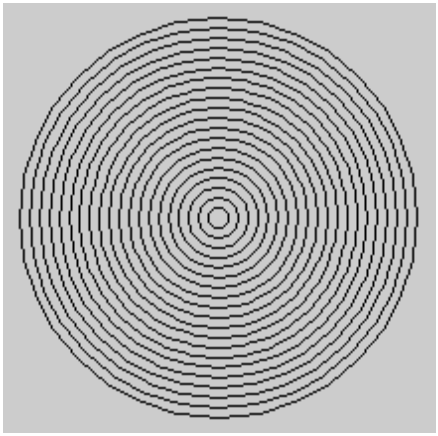
つまり、この for 文は、

１．整数型の変数 i を作って 0 を入れる
２．i が５より小さければ３．に進む。そうでなければ終わる。
３．中カッコ｛　｝の中身を実行する
４．i に 1 を足して２．に戻る

という動作をしているのだ。

さて、次で for 文を使ったプログラムの練習をしましょう。

８．規則的な模様



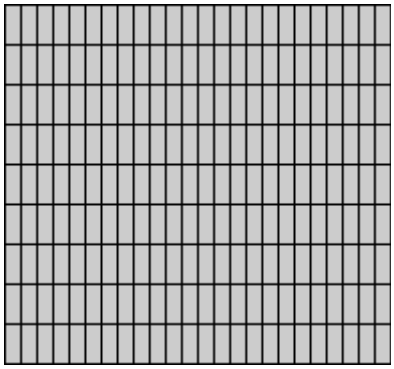
・例題：同心円

図のような、20 重の同心円（中心が同じな円たち）を描きましょう。
空欄を埋めて、プログラムを完成させてください。

```
size(500, 500);
noFill();
for(        ;      ;    ){
  ellipse(width/2, height/2, i*10, i*10);
}
```

・問題１：格子

図のような格子を描いてください。画面の大きさや格子の間隔は何でもいいです。

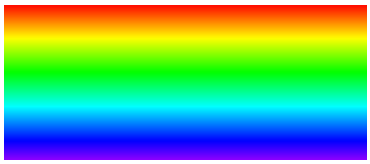


```
size(          );

for(                    ){
  line(                  );
}
for(                    ){
  line(                  );
}
```

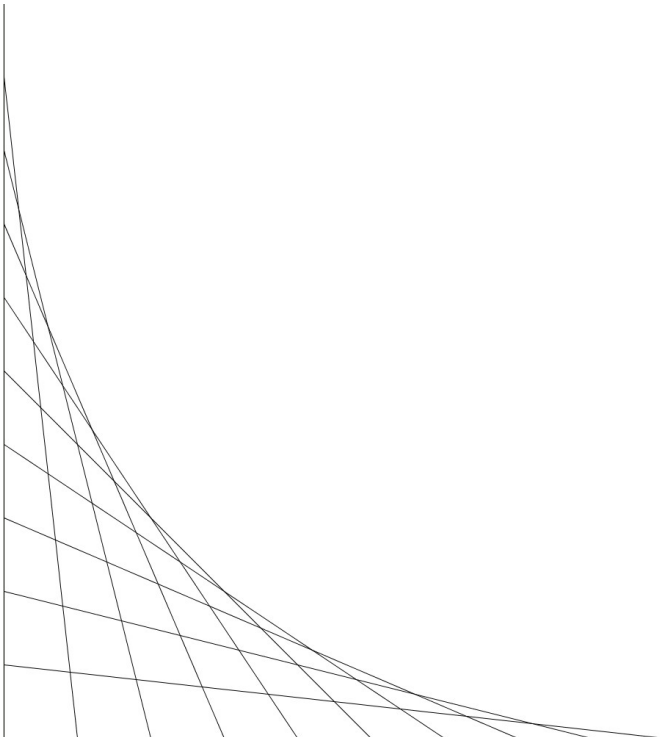
・・問題２：虹

for 文で虹を描いてみましょう



・問題３：曲線っぽいの

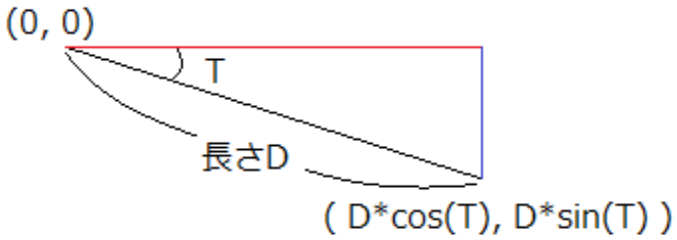
直線だけで次のような曲線のような図形が描けます。
For 文を使って描いてみましょう。



・補足：角度について

「角度〇〇°の直線が引きたい」といったときどうすればいいかを説明します。
まず、コンピュータの世界では、計算の都合で角度を 0 ～ 360° では扱いません。
その代わりに、
0 ～ 2*PI　　（PI は円周率 3.141592・・・）
で角度を表します（「度」ではなく「ラジアン」という）。
だから、180° は PI(=3.141592...)、30° は PI/6、1° は PI/180 です。
X° は、X*PI/180
となります。少々ややこしいので注意してください。

で、下の図のように角度 T、長さ D の直線を引くことを考えます。



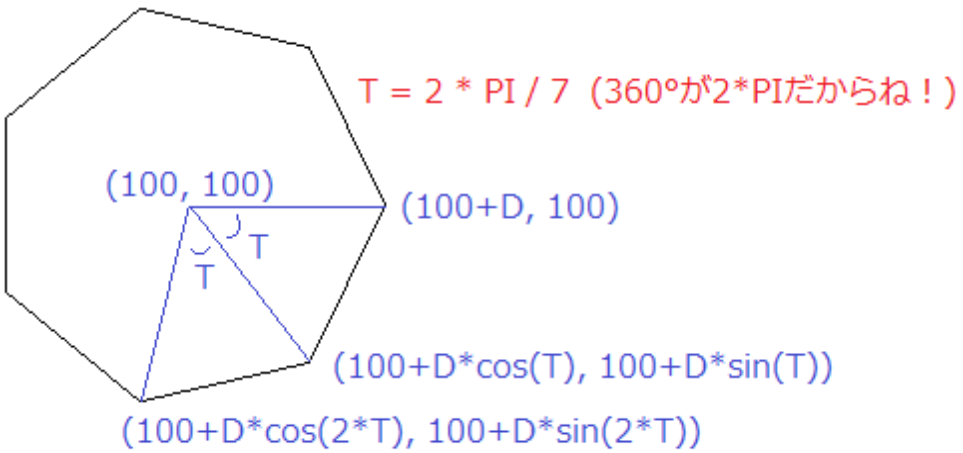
line 命令を使うためには始点と終点の位置が必要でした。
実は Processing には、こういうときに終点の位置を計算する便利な関数 **cos**（コサイン）と **sin**（サイン）があるのです。
cos(T)は上の黒線と赤線の長さの比を、sin(T)は黒線と青線の長さの比を教えてくれます。
だから、図のように始点を(0, 0)とした時の終点の位置は、(D*cos(T), D*sin(T))となります。
これをプログラムにすると、次のようになります。

ここでは長さ 100、角度 20° ＝20*PI/180 の　　直線を引きます。

```
float D = 100;
float T = 20 * PI / 180;
line(0, 0, D*cos(T), D*sin(T));
```

D や T の大きさを変えて、どうなるか調べてみてください。

・例題：正７角形を描くプログラム
これでどんな角度の直線でも引けるようになりましたね。
これを応用して、正７角形を描いてみましょう。



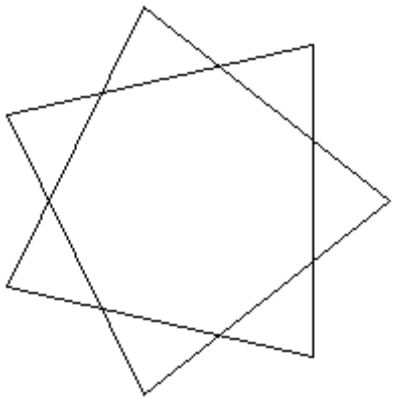
中心を(100, 100)として、一番右の頂点を(100+D, 100)としましょう。すると、
　　この頂点、中心、このひとつ下の頂点
のなす角度は 360/7° = 2*PI/7 です。
ということは、さっきの話から、二番目の頂点の位置は
　　(100+D*cos(T), 100+D*sin(T))
です。同じように、三番目の頂点について、
　　右端の頂点、中心、３番目の頂点
のなす角度は 2 * 2 * PI / 7 です。ということは三番目の頂点の位置は分かりますね！
　　(100+D*cos(2*T), 100+D*sin(2*T))
このようにして、頂点たちの位置がわかります。これらを一個ずつつないでいけば正７角形が描けます。

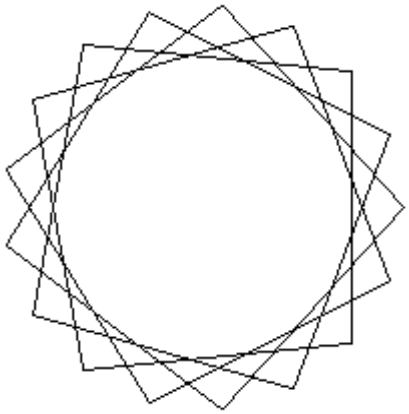
```
float D = 100;
int N = 7;
for(int i=0; i<N; i++){
  float T = 2 * PI / N;
  line(100+D*cos((i-1)*T), 100+D*sin((i-1)*T), 100+D*cos(i*T), 100+D*sin(i*T));
}
```

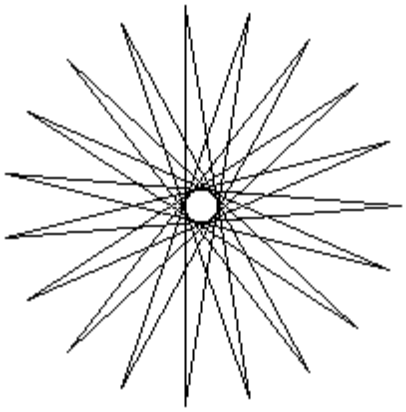
７角形がはみ出さないように size 命令を付け足してください。
このプログラムの変数 N に代入する数をかえると、違う頂点数の正 N 角形が描けます。
N を 100 くらいまで増やすとどうなりますか・・・？

・問題：星

今のプログラムを改良して、右のような星型の図形を描いてみましょう。ヒントは、「何頂点飛ばしで頂点を結んでいくか」を変数 M として記録しておくことです。
あとはさっきと同じように for 文で辺を一本ずつ描いていけば良いのです。







９．ランダムな模様

ここでは、「乱数」、「透明度」、「弧」の３つを学びます。

・乱数

乱数とは、何になるかわからないような（ランダムな）数のことで、コンピュータの中でサイコロをふるようなものです。

```
x = random(a, b);
```

と書くと、x には a 以上 b 未満の小数のうちのどれかが入ります。何になるかは実行するまで分からないし、実行するたびに結果は変わります。
何の役に立つのかといえば、例えばゲームの敵キャラの動きなどに使えます。
他にも、乱数をうまく使うと、きれいな絵が描けることがあります。

```
size(500, 500);
colorMode(HSB, 100, 100, 100);
background(0, 0, 0);  //背景：黒

for(int i=0; i<100; i++){
  stroke(0, 0, 100);  //白
  point(random(0, width), random(0, height));
}
```

このプログラムが何を描くか考えてみてください。なんとなく分かったら、実際に書いて実行してみましょう。

・透明度

色は H、S、B の３つの数で指定できるといいましたが、実は H、S、B、A の４つの数で指定することもできます。
この A は透明度といい、0～100 の値をとります。A が少ないほど、次から描かれる色は透明になり、下の色が透けて見えるようになります。

```
size(200, 200);
colorMode(HSB, 100, 100, 100);

fill(0, 100, 100, 50);  //赤半透明
rect(0, 0, 150, 150);
fill(66, 100, 100, 50); //青半透明
rect(50, 50, 150, 150);
fill(30, 100, 100, 50); //緑半透明
rect(100, 0, 100, 100);
fill(18, 100, 100, 50); //黄半透明
rect(0, 100, 100, 100);
```

このプログラムは、半透明な四角形を４枚重なるように描きます。Fill 命令の中の、４つ目の数字が透明度 A です。透明度を変えてみるなどして、透明度の働きを実感してみてください。

・弧、扇形

ellipse の拡張として、弧、扇形を描く命令があります。

arc(X, Y, W, H, A, B)

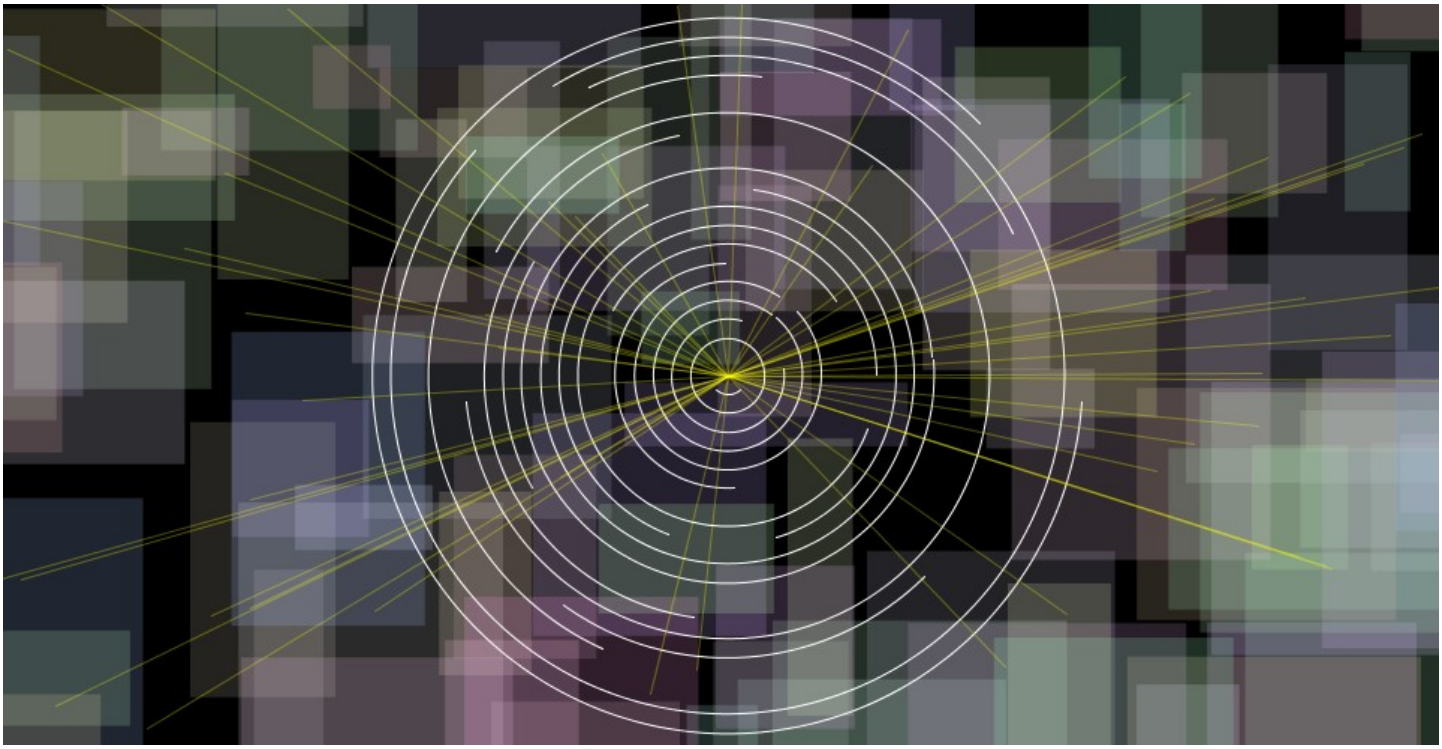
最初の４つは ellipse と同じで、 (X,Y)を中心とする幅 W、高さ H の円を表します。
次の二つが、「この円のどの部分を切り出すか」を表しており、
右方向を角度０として、**角度 A から角度 B まで**
の範囲が描画されます。

```
size(200, 200);
colorMode(HSB, 100, 100, 100);
background(0, 0, 0);  //背景：黒

noStroke();
fill(12, 100, 100);
arc(100, 100, 100, 100, 2*PI*0/6, 2*PI*1/6);
arc(100, 100, 100, 100, 2*PI*2/6, 2*PI*3/6);
arc(100, 100, 100, 100, 2*PI*4/6, 2*PI*5/6);
fill(0, 100, 100);
ellipse(100, 100, 25, 25);
```

扇形と円を組み合わせて最近話題の放射能マークを描くプログラムです。書けたら、どの arc 文がどの扇形に対応しているか考えてみて下さい。

・問題



今までの総まとめとして、こんな感じのイラストを描いてみましょう。

１．バックの四角形
真っ黒な背景に、乱数を使って、ランダムな位置、ランダムな大きさ、ランダムな色の半透明な長方形をたくさん描きましょう。どういった大きさ、色の長方形にするときれいに見えるか試してみましょう。

２．同心弧
arc を noFill()で描くと、上の図のような円弧が描けます。同心円のプログラムを参考に、ランダムな角度で描かれた同心弧を描いてみましょう。

３．光線
画面の中心から、ランダムな位置に向けてたくさんの半透明な直線を引きましょう。

第一のテーマはここまでです。もし時間の余った人は、好きなイラストを考えて自分で描いてみましょう。