Processing

第 8 回

松田小学校／寄小学校

８ーステップ　０：前回の復習だよ

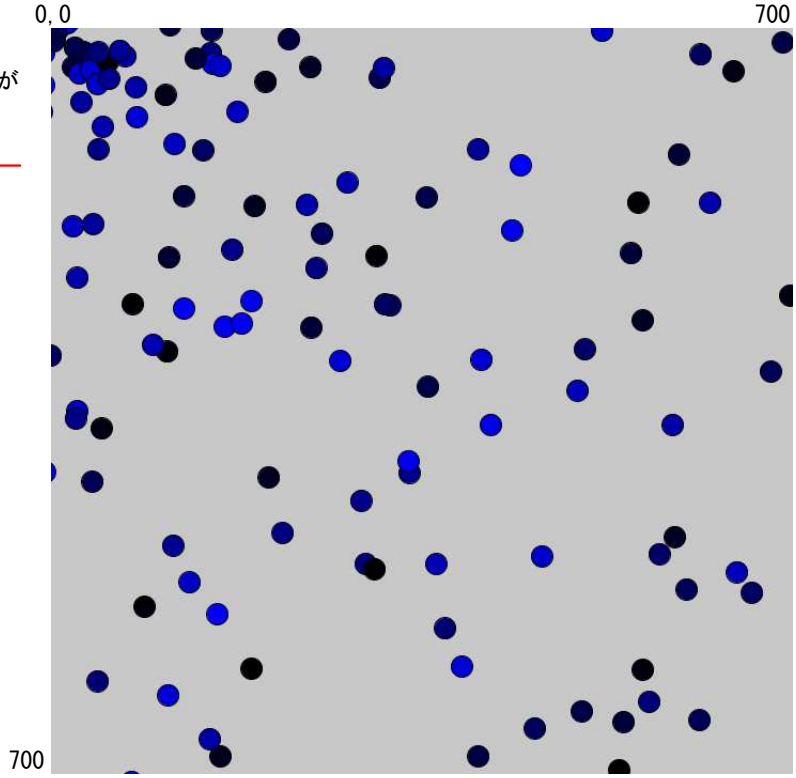
新規のファイルを開いて、下のコードを打ってみよう。

```
void setup() {
  size(700, 700);
  frameRate(10);
}

void draw() {
  background(200);
for(float a=0; a<700; a=a+0.5) {
  fill(0, 0, random(255));
  ellipse(random(-5, 5)*a, random(-5, 5)*a, 20, 20);
}}
```

random(-5,5) は、-5～4までの数字を不規則に発生させる。

Ｘ座標とＹ座標は、randomをつかって -5～4 間の数字を発生させて、それに 0 から700 までの数字 a をかけて決めている。

for(　　) のなかに、a=a+0.5 という小数があるから、int a=0; ではなく、float a=0; なんだ。



３行目に colorMode を加えるとどうなるかな？

```
void setup() {
  size(700, 700);
  colorMode(HSB, 360, 100, 100, 100);
  frameRate(10);
}

void draw() {
  background(200);
for(float a=0; a<700; a=a+0.5) {
  fill(0, 0, random(255));
  ellipse(random(-5, 5)*a, random(-5, 5)*a, 20, 20);
}}
```

hsb-2 で保存しよう。

８ーステップ　１：円弧（えんこ）を扱ってみよう

ファイルから新規を開いて 、下のコードを打ってみよう。

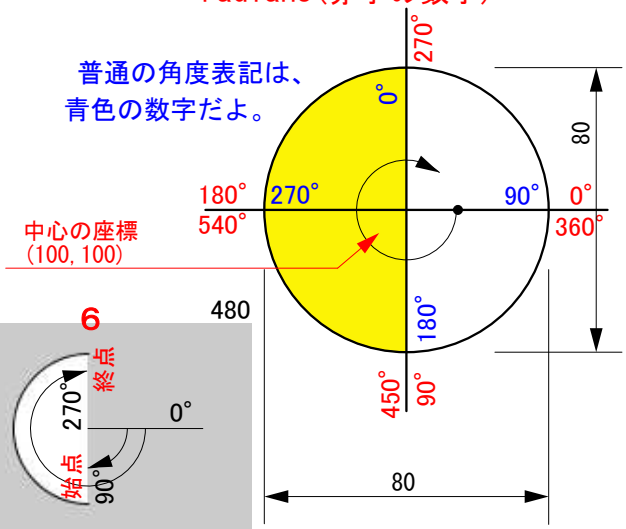
```
size(200,200);

arc(100,100,80,80,radians(90),radians(270));
```

中心の座標値　円弧円の幅　円弧円の高さ　始点の角度　終点の角度

円弧（えんこ）とは、円の一部を切り取ったものだよ。
Processingでは角度の表し方が、radians(　)を使うんだ。
(　)の中に入る数字は下の図や説明を見てね。

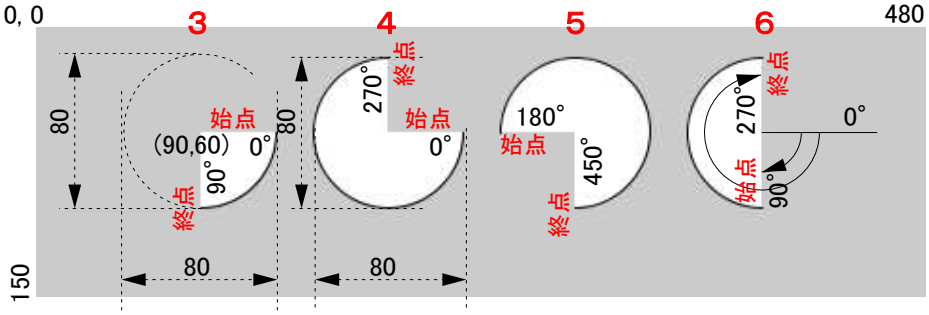
radians（ラジアン）の角度の表し方
radians（赤字の数字）

普通の角度表記は、
青色の数字だよ。



上のコードは、右の図の黄色い部分、
つまり左半分の円弧だよ。

中心の座標は前がＸ座標で、後ろがＹ座標だよね。
円弧の幅は円弧をつくる円の直径のこと、また、高さは円の直径のことなんだ。
始点と終点の数字は、右や下の図を見て考えてね。



下のコードを打って確かめたら、右のミッションをやってみよう。

```
size(480,150);
                    上の図の左から順に、3～6行目だよ！
arc(90,60,80,80,0,radians(90));
arc(190,60,80,80,0,radians(270));
arc(290,60,80,80,radians(180),radians(450));
arc(390,60,80,80,radians(90),radians(270));
```

コピペだよ！ 座標値や直しは後からね

ミッション（保存不要）

① 一番上のコードの円弧の幅を200に変えてみよう。
② 真下から始まり左真横で終わる円弧を描いてみよう。
③ 上半分の半円の円弧を描いてみよう。
④ 右真横から始まり右真横で終わる円を書いてみよう。

新規のファイルを開いて 、下のコードを打ってからミッションをやってみよう。

```
//float x;

void setup(){
  size(700,700);
}

void draw(){
  background(100);
  arc(350,350,                                    );
}
```

ここに直径が500で、真上から始まり真上で終わる円を書いてみよう。

ミッション

① 左のコード１行目の float x; を有効にし、始点から１回転するようにコードを変えてみよう。

ヒント

真上から始まるとは、始点が radians(270) ってことだね。１回転すると 360 増えるってことだから終点は 270+360 じゃないかな？　radians(270+360) と書けば、(　)内は Processing が計算してくれるよ。

動かすことは変数を、動く部分に組み込むんだよね。１回転で 360 だから、360 の部分を x で置き換えればＯＫじゃないかな。あとは、x が増えていく式を書けば良いんだね。

できたら、arc-1 で保存しよう。

８ーステップ　２：円弧を動かしてみよう

arc-1 のファイルを開いて、２周目は色が変わるコードを考えてみよう。

ヒント

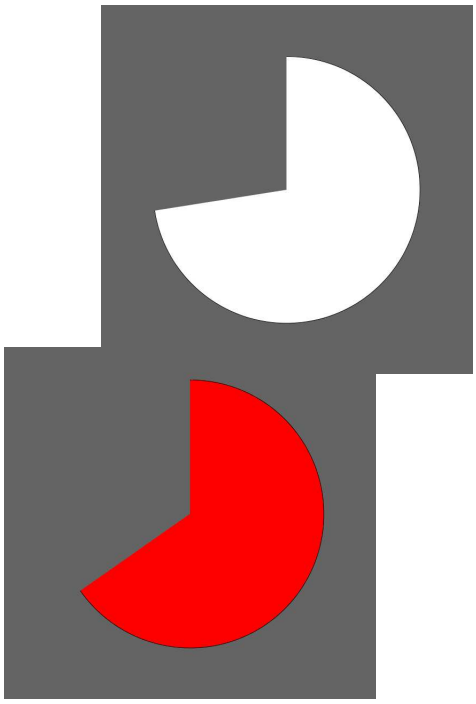
１周目は、270° → 270+360° の間を移動していった。２周目に入るってことは、もし x が 360 よりも大きいと考えればどうだろう？　x > 360; のときに、fill(255,0,0); が有効になるような式を足せば良いと言うことになるね。

```
1  float x;
2
3  void setup(){
4    size(700,700);
5  }
6
7  void draw(){
8    background(100);
9
10   arc(350,350,500,500,radians(270),radians(270+x));
11   x=x+1;
12
13 }
```

これは arc-1 のコードだ。
これに何を足せば良いのだろう？

if (x>360) { fill (255,0,0);}

12行目を足したら、実行してみよう 。



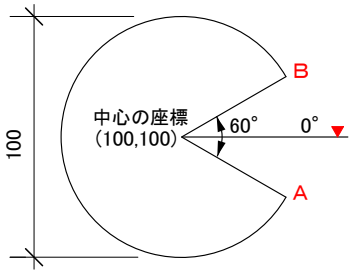
12行目にif 文を加えたけど、右のようにならないで、ずっと赤色の円のままだね。それは次の理由によるんだ。

最初、始点270、終点270ではじまり、終点が 270+360 で一周するあいだは白色だよね。次に x>360 によって fill(255,0,0); で赤色になるけど、２週目の始点は 270 のままで、終点が 270+361→270+362→270+363……へと連続して増加していくからずっと赤色で進んでしまうんだ。そこで、

fill(255,0,0); の後ろに x=0; を入れると、始点270、終点270になるから、最初の状態に戻るんだ。そこで fill(255,0,0); が働いて、赤色の円弧が 0 から始まるというわけだね。

できたら実行してから、arc-2 で保存しよう。

ファイルから新規を開いて 、
右の図になるようなコードを打ってみよう。




```
1  size(800,300);
2
3  arc(100,100,100,100,                    );
4
```

始点と終点を入れてね

0° は 60° の中心だよ。始点Ａと終点Ｂを考えよう。
radians(　) だよ。size は 800,300 でやってみよう。

第５回ステップ４でやった ball-3 のファイルを開いて、上のコードと合体して円弧を動かそう。

```
1  int a=25; //X座標の初めの値
2  int b=1;  //進む方向
3
4  void setup(){
5    size(600,200);
6  }
7
8  void draw(){
9    background(100*a);
10   a=a+b;
11   if(a>=width-25){ b=-1; fill(255,0,0);}
12   if(a<=25){ b=1; fill(0,0,255);}
13   ellipse(a,100,50,50);
14 }
```

◯は円の半径だよ。
円弧の半径に変えてね。

できたら、arc-3 で保存しよう。

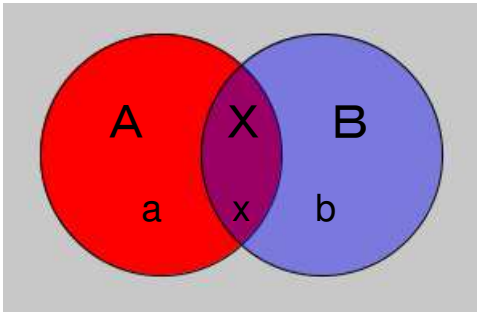
動かすためには、void setup() には最初の一度だけ設定する命令を、void draw() には繰り返し使う命令を書くんだったよね。

background(100*a); で背景を変化させているから背景の色も変わっていくね。このコードに円弧を組み込めば良いんだ。まずsize(　); を直そう。

円が移動するのを、円弧が移動するに変えるんだから、ellipse(　); を上で書いた arc(　); と入れ替えれば良いはずだ。できたら実行してみよう。あれっ！
両端で少し壁にもぐっちゃうね。これはball-3 の円が直径50だったから、半径25の分だけ前で跳ね返るようにしたからなんだね。円弧の直径は100だから、半径50だけ前で跳ね返るように変えてやらなくちゃならないね。

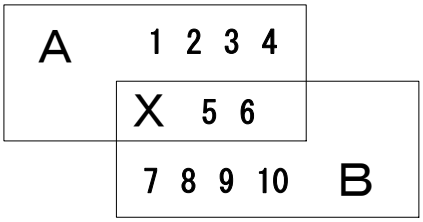
８ーステップ ３：円弧を動かしてみようの続き

arc-３ では、同じ方向を向いたまま往復していたね。今度は、右端に行ったら振り向いてから進ませるようにしよう。この先は、ちょっと難しいから事前の勉強だ！



左の図は赤色の部分＝Ａと、青色の部分＝Ｂとあるね。a はＡの中にあるし、b はＢの中にある。では、x はどちらにあるんだろうか？これは x はＡかつＢの部分にあると言うんだ。つまりＡＢが重なった Ｘ の部分にある場合だ。それを A&&B と表すんだよ。

それに対して、a, x , b はＡやＢだけの中じゃなくて、Ａ＋Ｂの中にあるとも言えるね。これは、a, x , b はＡまたはＢの部分にあると言うんだ。こちらは A||B と表すよ。



問題① 左の図を考えてみよう

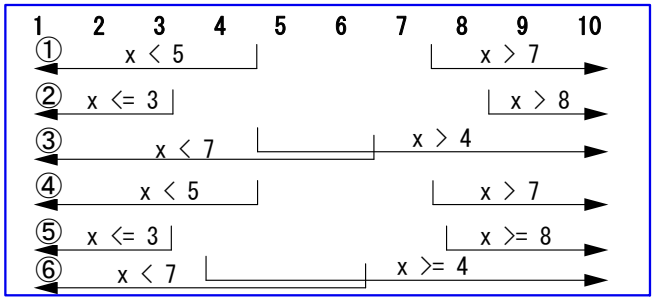
① Ａに含まれるのは、1, 2, 3, 4, 5, 6 だ。では、Ｂに含まれるのは？
② A&&B に含まれる数字は？
③ A||B に含まれる数字は？

となりのトモちゃんとお互いに確認してね。

問題② 右のヒントを見て よーく 考えてみよう

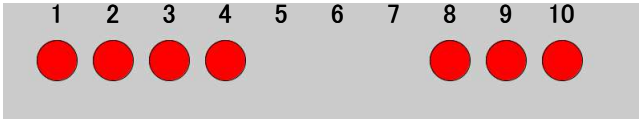
① if (x < 5 || x > 7) に含まれるのは、どんな数字だ？
② if (x <= 3 || x > 8) に含まれるのは、どんな数字だ？
③ if (x > 4 || x < 7) に含まれるのは、どんな数字だ？
④ if (x < 5 && x > 7) に含まれるのは、どんな数字だ？
⑤ if (x <= 3 && x >=8) に含まれるのは、どんな数字だ？
⑥ if (x >= 4 && x < 7) に含まれるのは、どんな数字だ？

ヒント



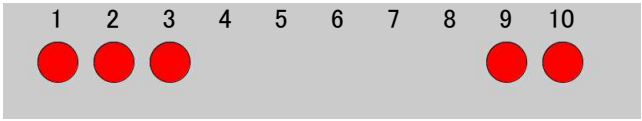
上の問題についてコードを書いて確認しよう。

①
```
size(800, 150);
for(int x=1; x<=10; x=x+1){
  if( x<5 || x>7 ){fill(255, 0, 0);
  ellipse(70*x, 75, 50, 50);
}}
```
赤線部分だけが違うんだよ！



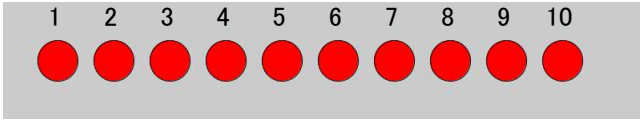
(x<5 || x>7) に含まれるのは、 1、2、3、4、8、9、10 だね。

②
```
size(800, 150);
for(int x=1; x<=10; x=x+1){
  if( x<=3 || x>8 ){fill(255, 0, 0);
  ellipse(70*x, 75, 50, 50);
}}
```
赤線部分だけが違うんだよ！



(x<=3 || x>8) に含まれるのは、 1、2、3、9、10 だね。

③
```
size(800, 150);
for(int x=1; x<=10; x=x+1){
  if( x<7 || x>4 ){fill(255, 0, 0);
  ellipse(70*x, 75, 50, 50);
}}
```
赤線部分だけが違うんだよ！

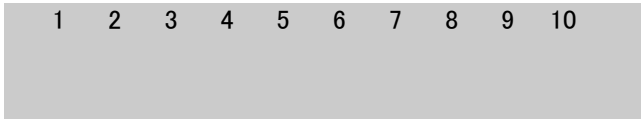


(x<7 || x>4) に含まれるのは、 1～10 の全部 だね。

問題②の①～③について、ＡまたはＢの部分を調べてみたんだ。
Ａ||ＢはＡ部分でもＢ部分でも、どちらにあっても良いんだから、①は、x<5 で 1～4 と x>7 で 8～10の合計７つの数字があてはまるね。真ん中の②は①と同じように両側だけだ。下の③は、指定している範囲が重なっているので、結局すべての数字があてはまってしまうんだね。
ヒントの ◀┘└▶ で表した図とも一致しているから、このコードは正しいよね。次にＡ&&ＢつまりＡかつＢの部分を調べてみよう。

８ーステップ ４：もう１つの円弧を動かそう

④
```
size(800, 150);
for(int x=1; x<=10; x=x+1){
  if( x<5 && x>7 ){fill(255, 0, 0);
  ellipse(70*x, 75, 50, 50);
}}
```
赤線部分だけが違うんだよ！

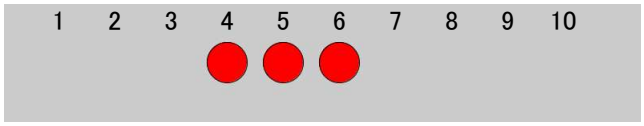


(x<5 && x>7) に含まれるのは、 1つもないね。

⑤
```
size(800, 150);
for(int x=1; x<=10; x=x+1){
  if( x<=3 && x>=8 ){fill(255, 0, 0);
  ellipse(70*x, 75, 50, 50);
}}
```
赤線部分だけが違うんだよ！

(x<=3 && x>=8) に含まれるのは、 1つもないね。

⑥
```
size(800, 150);
for(int x=1; x<=10; x=x+1){
  if( x<7 && x>=4 ){fill(255, 0, 0);
  ellipse(70*x, 75, 50, 50);
}}
```
赤線部分だけが違うんだよ！



(x<7 && x>=4) に含まれるのは、 4、5、6 だね。

問題②の④～⑥について、ＡかつＢの部分を調べてみたんだ。Ａ&&ＢはＡ部分でしかもＢ部分にもなければダメだね。言いかえると、重なっている部分にある数字だけがあてはまるんだから、④と⑤では該当する数字がなかったんだ。ちなみに、&& を使ったこのコードは左下のようにも書けるんだ。

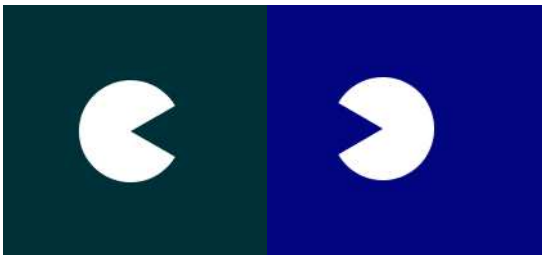
```
size(800, 150);
for(int x=1; x<=10; x=x+1){
  if( x<7 ){if( x>=4 ){ fill(255, 0, 0);
  ellipse(70*x, 75, 50, 50);}
}}
```
赤線部分だけが違うんだよ！

左のコードは、第４回ステップ５の難問ミッションで書いたものだ。Ａ&&ＢはＡかつＢという形で、同時に絞り込んでいるのに対して、第４回のコードは x<7で、しかも x>=4 という具合に、２段階で絞り込んでいる。でも結果は同じだ。

問題②の①～⑥のコードも上のコードも保存する必要はないよ。

arc-3 を開いて arc-4 に名前を変えて保存して、円弧が往復するコードに変えてみよう。

行き　　帰り



arc-2 とどこが変わったのだろうか？

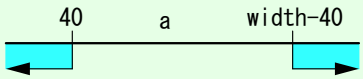
X座標が 0 → 39

```
int a = 39;  //最初のＸ座標
int b=1;     //進む方向

void setup(){
size(600,200);
}

void draw(){
    a=a+b;
    background(100*a);


  if((a<40)||(a>width-40))  {b =- b;}
  if(b == 1)  {arc(a,100,80,80,radians(30),radians(360-30));}
  if(b == -1)  {arc(a,100,80,80,radians(180+30),radians(450+60));}
}
```

13 行目以降が大きく変わった。
詳しい説明は、右を見てね。

方向転換
右行き
左行き
コピペだよ、手直しはその後でね。
プロセッシングが(　　) の中で計算してくれるから、式のままでOKだよ。

int a=39;にしたのは、a の範囲を a<40 にしたので、39 より大きくないと動き始めないんだ。試しに int a=30;にして実行してみて。13行目以降は下のような具合だ。



(a<40)||(a>width-40) だから、a が 40 未満か width-40 より大きい、つまり青い部分で、b が -b に変わるんだ。青い部分にさしかかると、＋ → － か － → ＋ してUターンするんだ。だから、往復するんだね。

向きを変えるのは、b == 1 つまり行きでは radians(30),radians(360-30) で右向きだった。 帰路は b == -1 となると radians(180+30), radians(450+60) で左向きになるんだ。

arc-4 のまま保存しよう。Ctrl を押しながら S を押すと、保存されているコードに上書きされるんだ。

８ーステップ ５：もう１つ円弧を動かそうの続き

arc-４ のミッション

① if((a<40)&&(width-40)) に書き換えてみよう。どんな動きになるだろうか。なぜこう動くのか考えてみよう。
② if((a<40)||(width-40)) に戻し、青色で帰るようにしてみよう。２度目の行きが青色にならないようにね。
③ 赤色の四角形(100*100)で戻ってくるようにしてみよう。２度目の行きはどうなったかな？

arc-4 に手直しして、円弧を四周の壁で跳ね返らせてみよう

```
1  int m = 6;    //X方向の速さ
2  int n = 4;    //Y方向の速さ
3  int x = 40;   //最初のX座標
4  int y = 40;   //最初のY座標
5
6  void setup() {
7    size(500, 500);
8    noStroke();
9  }
10
11 void draw() {
12   background(255);
13   fill(255,0,0);
14   arc(x,y,80,80,radians(30),radians(360-30));
15
16   x = x-m;    //X座標を m ずつ飛ばし速度を変えたように見せている
17   y = y-n;    //Y座標を n ずつ飛ばし速度を変えたように見せている
18
19   if ((x > width-40) || (x-40 < 0)) { m = -m; }
20   if ((y > height-40) || (y-40 < 0)) { n = -n; }
21 }
```

２方向なので変数も２つだ。

縦横座標なので変数も２つだ。

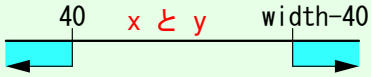
arc-4 で使えるコードは、消さずに使い廻ししよう。

X座標とY座標の両方に変数が必要だ。

19,20行目は、+- が反対だから方向転換だ。

arc-5 で保存しよう。

12 行目の background(255); までは arc-3とほとんど変わらないね。２方向へ動かしたいので、変数がそれぞれ２つずつだね。



arc-3 と違って２方向だけど、範囲指定は同じだね。Ｘ方向とＹ方向を別々に２本のif 文で書いているんだ。
いろいろ数字などを入れ替えて、動きを確認してみよう。

|| を論理和、&& を論理積と言うんだけれど、とても難しくて勘違いしやすいんだ。論理和とか論理積なんて言葉は覚える必要はないよ。分からなくなったら上の図を思いだしてね。

arc-5 を手直しして、もう１つ反対向きの円弧を入れてみよう。

```
1  int m = 6;    //X方向の速さ
2  int n = 4;    //Y方向の速さ
3  int x = 40;   //最初のX座標
4  int y = 40;   //最初のY座標
5
6  void setup() {
7    size(500, 500);
8    noStroke();
9  }
10
11 void draw() {
12   background(255);
13   fill(255,0,0);
14   arc(x,y,80,80,radians(30),radians(360-30));
15
16   x = x-m;    //X座標を m ずつ飛ばし速度を
17               //変えたように見せている
18   y = y-n;    //Y座標を n ずつ飛ばし速度を
19               //変えたように見せている
20
21   if ((x > width-40) || (x-40 < 0)) { m = -m; }
22   if ((y > height-40) || (y-40 < 0)) { n = -n; }
23 }
24
```

1～4行目を、5行目にコピペして変数を m→m1 のように変える

13～22行目を、23行目にコピペしてすべての変数を m→m1、x→x1 のように変える

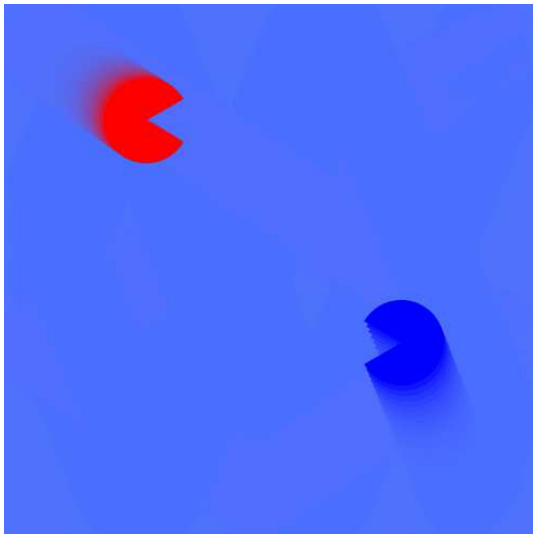
できたら、arc-6 で保存しよう。

ヒント

① もう１つの円弧だから、arc( ); を加えれば良いはずだね。
② arc( ); は、13～22行目がセットになって、円弧の動きを制御しているんだ。
③ 23行目の } を１行下げて空欄にしてから、13～22行目をコピー＆貼り付けしてみよう。
④ 動かすためには変数が必要だから、これもコードを書かなければいけないけど、1～4行目をコピーしてから５行目に貼り付ければ良いんだ。
⑤ このまま実行することはできないね。同じ変数を使うことはできない決まりなんだ。
⑥ １つ目と２つ目を区別するために、２つ目の名前を変えよう。int m = 6; を int m1 = 6; とするんだよ。n, x, y も同じようにやってみよう。
⑦ ここで一度、実行してみよう。
⑧ １つしか動いてないね。でも心配はないよ。２つが重なって１つに見えているだけだから、貼り付けた式の変数 m,n,x,y をm1,n1,x1,y1 に変えて、２つ目に違う動きを与えてやれば良いんだ。
⑨ ５、６行目の速さを変える。次に７、８行目の最初の座標値を変える。これで２つが動くはずだ。
⑩ もう１つは青色に変えよう。arc(～); の前の行に fill( ); で指定してやれば良い。
⑪ 最後に２つ目の円弧の向きを変えてやれば完了だ。向きの変え方をわすれた人は、隣のトモちゃんに助けてもらおう。

８ーステップ ６：もう１つ円弧を動かそうの続きの続き

arc-6 は右の図のようになったと思うけど、arc-6 のコードでは、円弧が影を引いてないよね。影を付けてみよう。



ヒント

arc-6 でいえば、12行目の background(77, 114, 255); が背景を反映しているんだね。これをウィンドウ全面の四角形に変えて、四角形の色をバックグラウンドと同じにする。
そして、第４の数字＝透明度を加えてやれば、左の図のようになるよ。いくつくらいの数字を使えば良いのかは、各自で考えて欲しい。

できたら arc-7 で保存しよう。

arc-7 に書き加えて
下の図のようにしてみよう。

ヒント

Class is over. は、上へと動いているんだ。だから、新しい変数が必要だね。int z = 1; として、変数の最後の行に書き加えてね。その下のコードは、Ｙ座標が 300 で止まる命令だよ。これでバッチリ決まりだ。
下のコードの空欄をおぎなって、どこかに挿入して欲しい。どこに挿入するか、よーく考えてみよう。

```
textSize(30);
fill(255, 0, 255);
text("          ",          , height+20+z);
z=z-0.5;
if(z<-300)  z=z+0.5;
```

Ｙ座標は上が 0 で、下が 500 だから、上に登るのは数字が小さくなっていくんだよ。

Ｙ座標が 300 で止まれという命令だ。(z==-300 でもOK)

Class is over. とは、授業は終わりと意味だよ。
arc-8 で保存して、今日の授業を終わろう。

次回の予告

プロセッシングが使えるようになったから、次回は復習をかねて木々の向こうをバスが走っている、そんな風景とバスのアニメーションを描いてみよう。

Bye-bye!