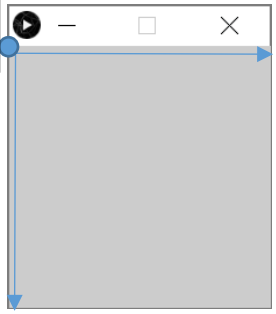
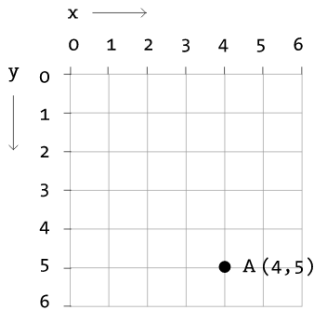
ボールをバウンドさせよう by Processing プログラミング

プログラミングは特に学習の最初の時点でのハードルが高く、慣れるまでは大変に感じるかもしれませんが、とても便利かつ数学や英語の必要性を実感できます。Processing は初心者でも扱いやすいプログラミングソフトです。ウェブ[1]で無料でダウンロードできて、Windows でも Mac でも動きます。元々はヴィジュアルアートを目的として開発されました。他の本格的な言語（C 言語など）とも似ていますし、マウスやカメラによる外部入力や、様々なライブラリと呼ばれる機能を比較的簡単に使うことができます。

1．グラフィックス　　：プログラミングでボールを描画

1.1, ウィンドウを作る





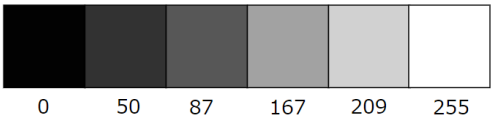

```
size(300,300);
//size(横幅、縦幅)
```

;これはセミコロン。プログラムの命令ごと（大体は行の最後）に必ず付ける決まり。

//これはコメントアウト。
この行はプログラムから無視されます。

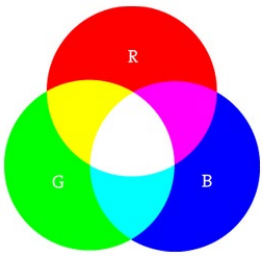
左図は Processing で作った 300x300 ピクセルのウィンドウ。真ん中の図は座標を表す。普通の座標と違って、y 方向（上下）が逆なので注意。右図はプログラミングの内容。

1.2, 背景色を変える



1.1 のプログラムの最後に background(255);を追加してください。画面の色が白で塗りつぶされます。色は 0~255 で強さを表します。

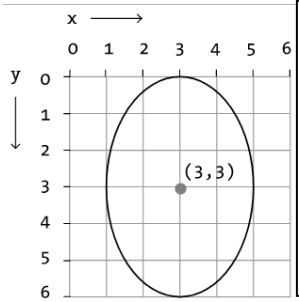
次に、background(255,0,0);に変えてみましょう。パラメータが 3 つあるときは、それぞれ R,G,B(red, green, blue)を表し、3 色を混ぜて様々な色を表現します。絵具のような減法混色ではなく、加法混色は沢山混ぜるほど明るい色＝白に近づきます。



課題 1.1a. ウェブで「RGB　カラーチャート」で検索。例えばこのようなサイト[2]を見る。
課題 1.1b. Photoshop または win のペイントなどのカラーパレットを見てみる
課題 1.1c プログラムで background に RGB の値を指定して、色々な背景色を表示する。

[1] Processing ウェブページ, https://processing.org/
[2] Web Safe Color, https://www.scollabo.com/banban/lectur/websafe.html

1.2, ボールを描く




```
size(300,300);
background(255);
ellipse(3,3,4,6);
//ellipse(x 位置,y 位置,円の横幅、円の縦幅)
```

楕円という意味の ellipse という命令は 4 つパラメータがあります。左図のように円を描くとき、右図のようにプログラムします。

◇課題 1.2 直径 100 ピクセルの円を 2 つ、違う場所に描画しよう。

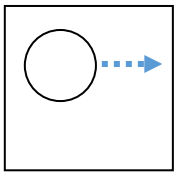
2．アニメーション　　：ボールを動かす

2.1, setup と draw

```
void setup(){
        size(300,300);
        println("SET ");
}
void draw()
{
        println("DRAW ");
}
```

"は半角英数なので注意。

print はコンソールと呼ばれる画面下の方の黒い画面に文字を出力する。
setup はプログラムを実行すると、一回だけその内容が処理される。
その後、draw の内容は何度も繰り返し呼び出されて処理される。
なので、左のプログラムを実行すると、コンソールに「SET DRAW DRAW …」と表示される。
※左のソースをコピペすると""が全角になってしまいます。word の仕様みたいで、今のところ皆さんに手でなおしてもらうしかないみたいです🙇

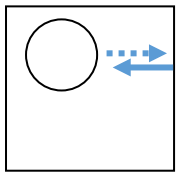


2.2, draw でボールを動かす

```
int c=0;
//setup は略。さっきと同じ
void draw(){
        c = c+1;
        println(c);
        //ellipse(c, 50, 100, 100);
}
```

int は整数(…,-1,0,1,2,…)を表す。
整数のデータ c を作成し、draw で+1 し続けると、「SET 1234…」と表示される。
c をボール(ellipse)の位置のパラメータにしたら、ボールが移動する。
ボールの描画が重なるので、background を draw 内の最初にすると、前の描画が消える。

◇課題 2.2 ボールの速度を変えてみよう。加速させる必要はなく、速度を今と別の定数に変更するだけでよい。



2.3, ボールの位置を変数にする+壁でバウンド

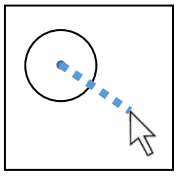
```
int bx=200;//ball の x 位置
int by=100;//ball の y 位置
int vx = 10;//ball の x 方向の速度 velocity
int vy = 10;//ball の y 方向の速度 velocity
int bs   =100;
//setup は略
void draw(){
        background(255);
        ellipse(bx, by, bs, bs);
        bx = bx + 1;//+vx に変更してみる


        if(bx > width){vx = -1 * vx;}
}
```

ボールの位置を表す変数 bx, by を用意。
x 方向のボールの速度を表す vx も用意。
※vx と vy は速度ベクトル。
ボールの大きさ bs も用意。

右の壁に当たったら跳ね返る場合、速度を逆にしたらよい。
if を使って、「もしボールの位置が画面の一番右より大きくなったら」「x 方向の速度を逆にする」という処理をする。
width にはあらかじめ画面の横幅が入っている。画面の縦幅は height。

◇課題 2.3a ボールが左側の壁に当たったら跳ね返るようにしよう。
◇課題 2.3b ボールが上下の壁に当たったら跳ね返るようにしよう（y 方向の処理)。



3．インタラクション　：ボールに現実世界から影響を与える

ユーザの動かすマウスに当たったら跳ね返るようにしましょう。

```
void draw(){
  //if より上は略(2.3 の内容)
  //マウスとインタラクション
  int dist = (bx-mouseX)*(bx-mouseX) +
             (by-mouseY)*(by-mouseY);
  if(dist < bs/2*bs/2){
    vx = vx * -1;
    vy = vy * -1;
    //bs = bs - 1;
  }
```

マウスの位置は mouseX, mouseY です。

マウスとボールの距離がボールの半径より小さい時、マウスはボールに触っているか、ボールの内側に入っています。

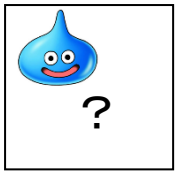
このとき、ボールを反対に跳ね返らせます。

他にも、ボールとマウスが触れたとき、ボールのサイズを小さくしてみるなど。

◇課題 3 ユーザの動かすマウスにボールが当たったら跳ね返るようにしましょう。

４．ボールを好きな画像に変えよう＆ゲームを考えよう

これまで学んだこと＋自分のアイデアでゲームや面白い表現を考えてみましょう。例えば、ボールを自分で選んだ絵に置き換えて、マウスや壁に当たった時にサイズを変えるなど（例を見せます）。




```
//以上は略してます
PImage img;//①

void setup()
{
  img = loadImage("sura.png");//②
  size(500, 500);
}

void draw()
{
  background(255);
  ellipse(bx, by, bs, bs);
  image(img, bx-bs/2, by-bs/2, bs, bs);//③
  //以下は略してます
}
```

ボールの代わりに画像を表示する場合は、左の①～③を追加します。

ただし、下線部は自分の使いたい画像の名前にします。

更に、プログラムのあるフォルダに使いたい画像ファイルを置くと、使えます。

画像はウェブで検索し、
　画像を右クリック、
　名前を付けて画像を保存、
　デスクトップなどに保存、
等の後、プログラムのあるフォルダに移動します。

◇課題４　ボールを画像に変えた上で、何かのゲーム性を付与してください。

編集後記

※Pong というゲームの一人バージョンは今日の内容で作れそう

四角形を描画する
rect(x 位置,y 位置,横幅,縦幅)が要りますが。

※モンストもある程度作れそう？

※インベーダーゲームもある程度作れそう？

Processing でのプログラミング TIPS

1．Processing のプログラムファイルなどの保存方法

メニューの　ファイル＞保存
でプログラムをデフォルトの場所に保存できます。
※保存された場所は 2 の方法で確認できます。参照 URL にデフォルトの場所の変え方があります。

ファイル名は自動で「**Sketch_140915a**」などになります。数字はその日の日付で、a,b,c,…はその日何番目に作られたファイルかで決まります。

また、ファイルはメインの pde ファイルと同じ名前のフォルダ内に、一緒に作成されます。
Processing のプログラムファイルは同じ名前のフォルダ内に配置されていないと実行できません。

参照　https://r-dimension.xsrv.jp/classes_j/start/

2．プログラムファイルがあるフォルダの簡単なアクセス方法：

メニューの　スケッチ＞フォルダーを開く
を選ぶと該当フォルダが開きます。

画像を置くときなどに活用しましょう。

3．拡張子の表示方法

Mac の場合　https://support.apple.com/ja-jp/guide/mac-help/mchlp2304/mac
Windows の場合　https://www.pc-koubou.jp/magazine/36291

上記の URL が無効になっている場合「Mac　拡張子　表示」などで検索すると出てきます。

これも画像を読みこむ際に必要です。
理系なら、拡張子は表示しておきましょう！

４．コードの自動整形

Win なら Ctrl＋T、
Mac だとコマンドキー＋T
でコードをきれいに整形できます。

きれいなコードはバグよけに必須です！
他にも以下もよく使いますが、**コード整形はコードを書きながら逐次行うこと必須！**

プログラムの実行：Win　Ctrl＋R
※よく使う。▶ボタン押さなくていい

コメントアウト： Win　Ctrl＋/
※よく使う。//を自動でいれたり、削除。複数行選択していれば複数行に対して行える

５．拡張子 exe と pde

exe は実行ファイルという、アプリを実行するファイルの拡張子です。
processing.exe は Processing というアプリを実行するファイルなのですね。
sketch_22414a.pde などというファイルは書いたコードのテキストファイルです。

おまけ：正しい反射

初回でやるのはボリュームが多いかなと思うので省いている内容です。ボールの反射は壁に当たった時は単純でよいのですが、マウスに当たった時は違和感があるでしょう。本当はマウスが小さなボールだとみなして接面で反射させるのが正しいです。下記のサイトを見ると、ボールの正しい反射方向の計算の仕方がわかります。また、その実装例を以下にのせていますが、これから教えることがたくさん含まれています・・・。

参考：反射ベクトルを求める https://qiita.com/edo_m18/items/b145f2f5d2d05f0f29c9

```
//反射面のベクトル：マウスとの反射の場合、ボールとマウスの接面
float vecx = mouseX - bx;//ボールからマウスへのベクトル
float vecy = mouseY - by;//ボールからマウスへのベクトル
float dist = sqrt(vecx*vecx +vecy*vecy);

if (dist>=radi)return;//ボールが衝突してたら以下の計算をする。
float mx = mouseX;
float my = mouseY;

//反射面のベクトル：マウスとの反射の場合、ボールとマウスの接面
PVector F = new PVector(vx, vy);//ボールの進行ベクトル
PVector N = new PVector(bx-mx, by-my);//面の垂直ベクトル
N = N.normalize();
float a = -F.dot(N);//N に投影した F の大きさ
PVector R = F.add(N.mult(2.0*a));

//以下の t は参考サイトにはありません。ボールのめり込みを解消しつつ、不自然な挙動を減らす目的のものです。
float t = radi-dist;
t *= sq((1.0-t)/radi);

vx = R.x+N.x*t;
vy = R.y+N.y*t;

ellipse(mx, my, 10, 10);
line(mx, my, mx+N.x*10, my+N.y*10);
```

衝突するボール同士の場合も同じ原理ですが、同じ重さ・跳ね返り係数１の場合、単純に二つのボールの速度を交換すればよいです。（でも原理を知っておくのは大事ですよね！）