



プログラミング演習 (2)

相対位置を使って動かそう

中村, 高橋
小林, 橋本



- Processing に慣れ親しむ
- Processing で絵を描く
 - 色々な関数(命令)を試してみる
 - 順番の重要性を理解する
- 変数を使って絵を描く
 - 相対位置で描画してみる



命令名 (命令の詳細, 命令の詳細, ...);

- 例: size, background, line, ellipse, ...
- **すべて半角英数字**
 - 日本語はダメ！ 大文字小文字に注意！
- 命令の詳細は括弧の中に！
 - 複数あるときはカンマで区切る
- **最後はセミコロン！**
- プログラムは上から順に実行される



- Processing には、大きく分けて **setup (準備)** と **draw (描画)** がある
 - 「**準備**」では、プログラムが実行されるときに、**最初に1回だけ何を行うか**ということを記述する
 - ウィンドウサイズの指定
 - 利用する画像や音声の読み込み
 - 「**描画**」では、プログラムが実行されている際に、**毎回繰り返し何を描画するか**ということを記述する
 - 画面上での何らかのアニメーション
 - 画像の表示や音声の再生



```
void setup(){  
    size( 400, 300 );  
    background( 255, 255, 255 );  
}
```

- 「**void setup(){**」から「**}**」までの間に準備の内容を記述する
- 上記の例では、400x300のウィンドウを作り、背景の色を白色(255,255,255)に指定している



```
void draw(){  
    fill( 255, 0, 0 );  
    ellipse( 200, 150, 150, 150 );  
}
```

- 「**void draw(){**」から「**}**」までの間に毎回描画する内容を記述する
- 上記の例では、塗り色を赤色(255,0,0)に指定し、(200,150)の位置に直径150の円を描画
 - 円の位置が動かないのであまり意味が無いが...



void とか () とか {} とか

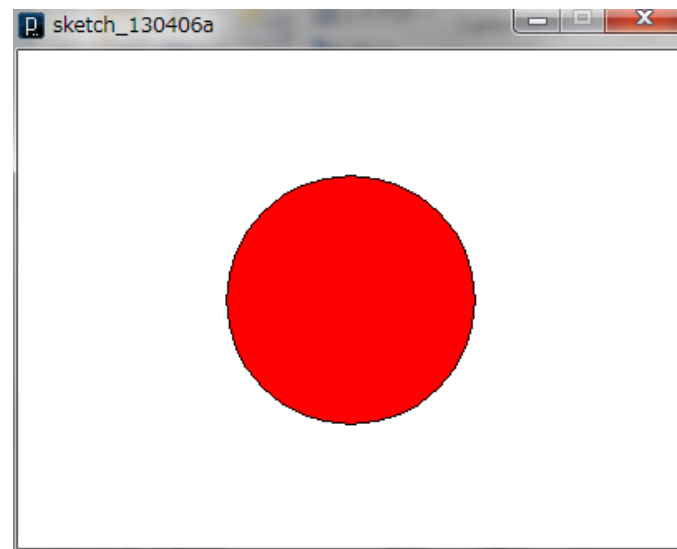
- setup や draw の前後の **void** と **()** は、まずはおまじないだと思って下さい(後に説明します)
- 重要なのは、**{**から**}**までが1まとまりで、括弧内が上から下にまとめて実行されるという事
- 開いたら閉じる！！
 - **(**には必ず対応する**)**が必要！
 - **{**には必ず対応する**}**が必要！
 - 後に登場しますが**[**にも**]**が必要！

日本の国旗の例を



- 試しに入力してみましよう

```
void setup(){  
  size( 400, 300 );  
  background( 255, 255, 255 );  
}  
  
void draw(){  
  fill( 255, 0, 0 );  
  ellipse( 200, 150, 150, 150 );  
}
```



日本の国旗の例を



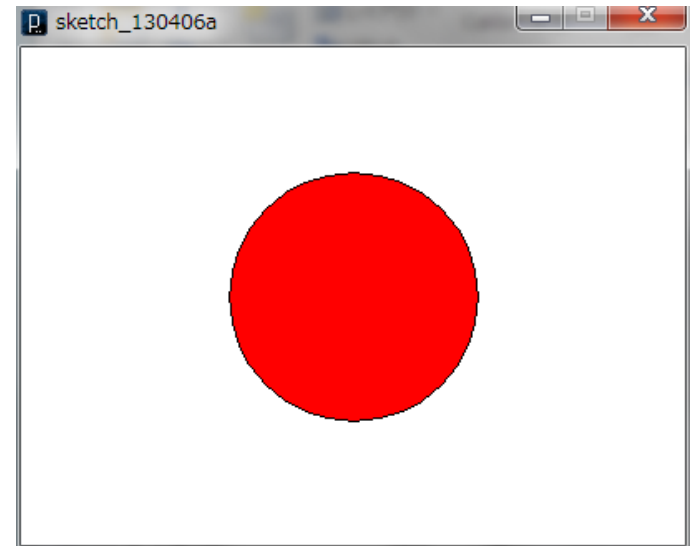
- 試しに入力してみましょう

```
void setup(){  
  size( 400, 300 );  
  background( 255, 255, 255 );  
}
```

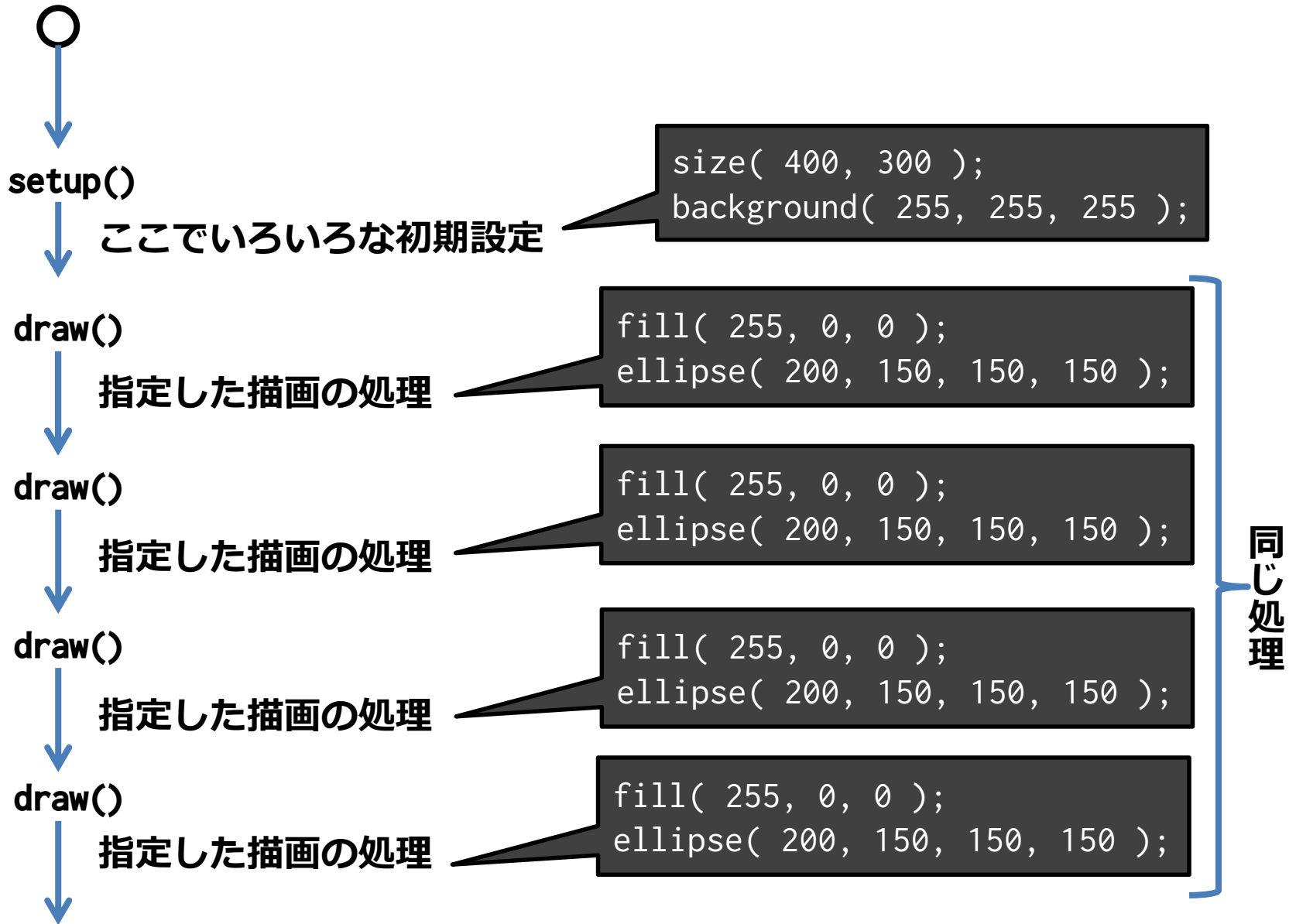
最初に一度だけ

```
void draw(){  
  fill( 255, 0, 0 );  
  ellipse( 200, 150, 150, 150 );  
}
```

描画の度に



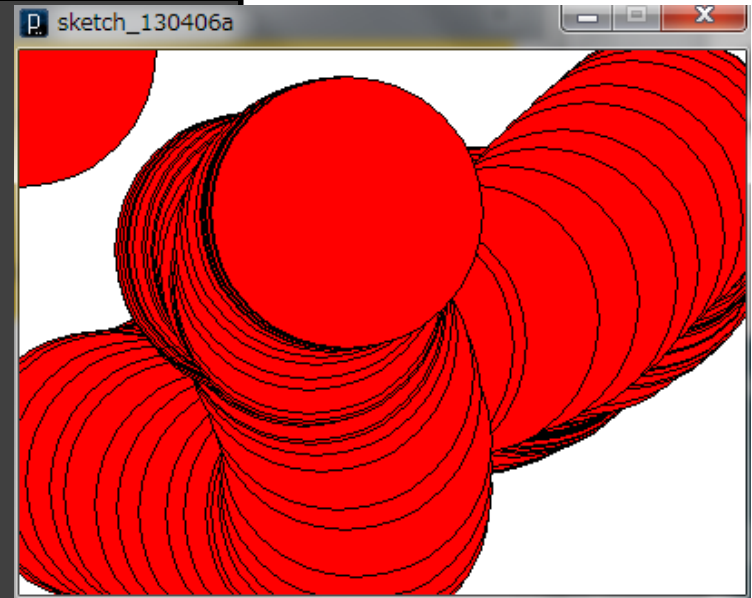
流れ



赤丸をマウスの場所に



```
void setup(){  
  size( 400, 300 );  
  background( 255, 255, 255 );  
}  
  
void draw(){  
  fill( 255, 0, 0 );  
  ellipse( mouseX, mouseY, 150, 150 );  
}
```



マウスのXY座標

ぐちゃぐちゃになってしまう



流れ(解決)



背景を白色で塗り直す！



setup()

ここでいろいろな初期設定

```
size( 400, 300 );
```

draw()

指定した描画の処理

```
background( 255, 255, 255 );  
fill( 255, 0, 0 );  
ellipse( mouseX, mouseY, 150, 150 );
```

draw()

指定した描画の処理

```
background( 255, 255, 255 );  
fill( 255, 0, 0 );  
ellipse( mouseX, mouseY, 150, 150 );
```

draw()

指定した描画の処理

```
background( 255, 255, 255 );  
fill( 255, 0, 0 );  
ellipse( mouseX, mouseY, 150, 150 );
```

draw()

指定した描画の処理

```
background( 255, 255, 255 );  
fill( 255, 0, 0 );  
ellipse( mouseX, mouseY, 150, 150 );
```

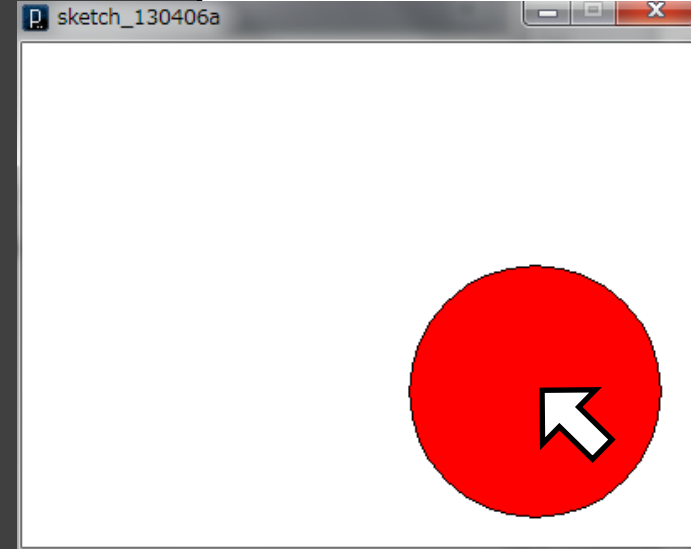
赤丸をマウスの場所に



```
void setup(){  
  size( 400, 300 );  
}  
  
void draw(){  
  background( 255, 255, 255 );  
  fill( 255, 0, 0 );  
  ellipse( mouseX, mouseY, 150, 150 );  
}
```

マウスのXY座標

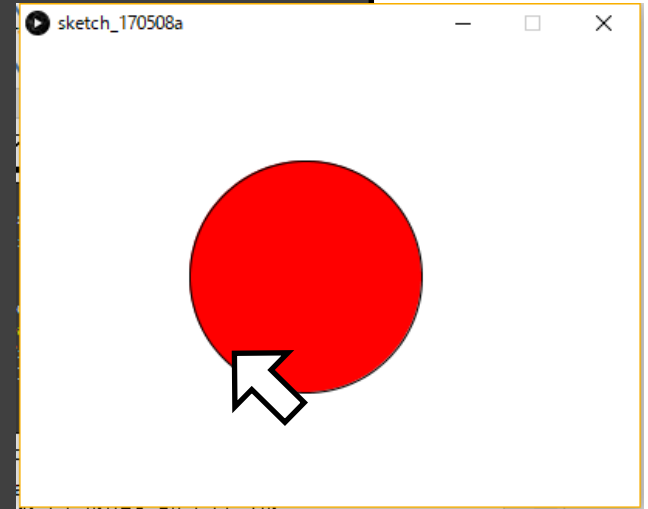
- mouseX と mouseY はカーソルの位置
- draw の中で mouseX や mouseY を利用するとその点に絡めた描画が可能



カーソルからずらすには？



```
void setup(){  
  size( 400, 300 );  
}  
  
void draw(){  
  background( 255, 255, 255 );  
  fill( 255, 0, 0 );  
  ellipse( mouseX+50, mouseY-50, 150, 150 );  
}
```



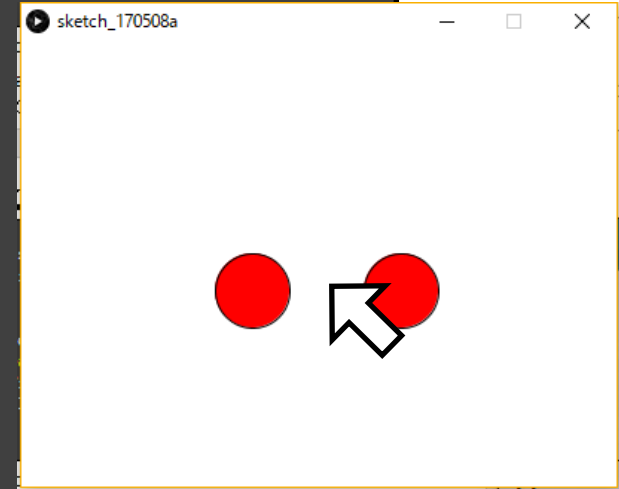
マウスのXY座標からの差分で指定

- mouseX と mouseY はカーソルの位置
- 「+」や「-」で場所を動かす

2つ描くには...



```
void setup(){  
  size( 400, 300 );  
}  
  
void draw(){  
  background( 255, 255, 255 );  
  fill( 255, 0, 0 );  
  ellipse( mouseX-50, mouseY, 50, 50 );  
  ellipse( mouseX+50, mouseY, 50, 50 );  
}
```



マウスのXY座標からの差分で指定

- mouseX と mouseY はカーソルの位置
- 「+」や「-」で場所を動かす

違いを見るために



`println("表示したい文字列");`
(ぷりんと**える**えぬです. ぷりんとあいえぬではないです)
を入れてどのように動作しているか調べてみよう

```
void setup(){
  size( 400, 300 );
  println( "background" );
  background( 255, 255, 255 );
}

void draw(){
  println( "fill" );
  fill( 255, 0, 0 );
  println( "ellipse" );
  ellipse( mouseX, mouseY, 150, 150 );
}
```

```
void setup(){
  size( 400, 300 );
}

void draw(){
  println( "background" );
  background( 255, 255, 255 );
  println( "fill" );
  fill( 255, 0, 0 );
  println( "ellipse" );
  ellipse( mouseX, mouseY, 150, 150 );
}
```

エディタ下の表示に注目



The screenshot shows the Processing IDE window titled "sketch_140415b | Processing 2.1.1". The code editor contains the following code:

```
void setup() {  
  size( 400, 300 );  
  println( "background" );  
  background( 255, 255, 255 );  
}  
  
void draw() {  
  println( "fill" );  
  fill( 255, 0, 0 );  
  println( "ellipse" );  
  ellipse( mouseX, mouseY, 150, 150 );  
}
```

The console output at the bottom shows the following text:

```
background  
fill  
ellipse  
fill  
ellipse  
fill  
ellipse  
fill  
ellipse  
fill  
ellipse  
fill  
ellinse  
13
```

An orange callout box with white text is overlaid on the console output, stating: "ここには println で指定された文字が表示される" (Here, the text specified by println is displayed).

The screenshot shows the Processing IDE window titled "sketch_140415b | Processing 2.1.1". The code editor contains the following code:

```
void setup() {  
  size( 400, 300 );  
}  
  
void draw() {  
  println( "background" );  
  background( 255, 255, 255 );  
  println( "fill" );  
  fill( 255, 0, 0 );  
  println( "ellipse" );  
  ellipse( mouseX, mouseY, 150, 150 );  
}
```

The console output at the bottom shows the following text:

```
background  
fill  
ellipse  
background  
fill  
ellipse  
background  
fill  
ellipse  
background  
fill  
ellipse  
background  
13
```

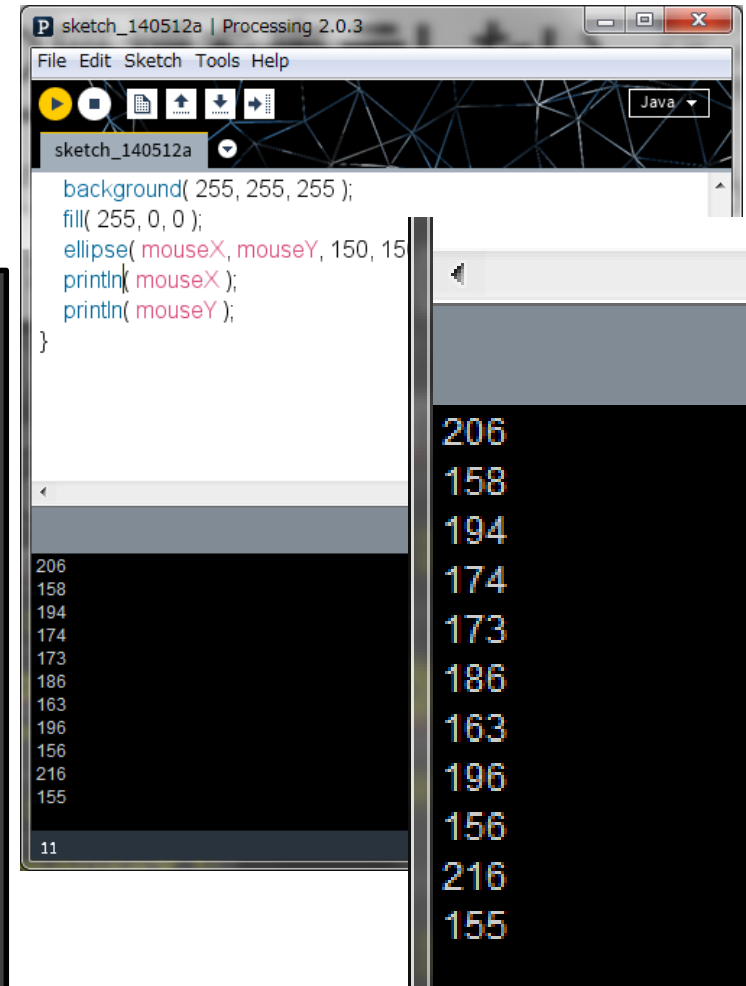
An orange callout box with white text is overlaid on the console output, stating: "ここには println で指定された文字が表示される" (Here, the text specified by println is displayed).

マウスの座標を知りたい



- println で mouseX の値を表示
- println で mouseY の値を表示

```
void setup(){  
  size( 400, 300 );  
}  
  
void draw(){  
  background( 255, 255, 255 );  
  fill( 255, 0, 0 );  
  ellipse( mouseX, mouseY, 150, 150 );  
  println( mouseX );  
  println( mouseY );  
}
```



値を表示できた！けど
ちょっとわかりにくい...

マウスの座標を表示したい



- 例えば (200, 300) にマウスの座標がある場合には、 $x = 200$, $y = 300$ と表示したい！
 - print で「x =」を書いて
 - print で mouseX を表示して
 - print で「,」を書いて
 - print で「y =」を書いて
 - println で mouseY を表示

```
sketch_140512a | Processing 2.0.3
File Edit Sketch Tools Help
sketch_140512a
background( 255, 255, 255 );
fill( 255, 0, 0 );
ellipse( mouseX, mouseY, 150, 150 );
print( "x = " );
print( mouseX );
print( ", " );
print( "y = " );
println( mouseY );
}

x = 228, y = 108
x = 230, y = 108
x = 231, y = 108
x = 231, y = 108
x = 232, y = 108
x = 232, y = 108
x = 233, y = 108
x = 234, y = 108
x = 234, y = 109
x = 234, y = 110
x = 234, y = 114
x = 231, y = 118
17
```

マウスの座標を表示したい



- 例えば (200, 300) にマウスの座標がある場合には, $x = 200$, $y = 300$ と表示したい!

```
void draw(){
  background( 255, 255, 255 );
  fill( 255, 0, 0 );
  ellipse( mouseX, mouseY, 150, 150 );
  print( "x = " );
  print( mouseX );
  print( ", " );
  print( "y = " );
  println( mouseY );
}
```

マウスの座標を表示したい



- ちなみに、下記のように書くことも可能
–「+」は文字と文字をくっつけるという意味

```
void draw(){  
  background( 255, 255, 255 );  
  fill( 255, 0, 0 );  
  ellipse( mouseX, mouseY, 150, 150 );  
  println( "x=" +mouseX+ ", y=" +mouseY );  
}
```

相対座標で描こう！



```
void setup(){  
  size( 800, 600 );  
}
```

```
void draw(){  
  background( 255, 255, 255 );  
  drawCharacter( mouseX, mouseY );  
}
```

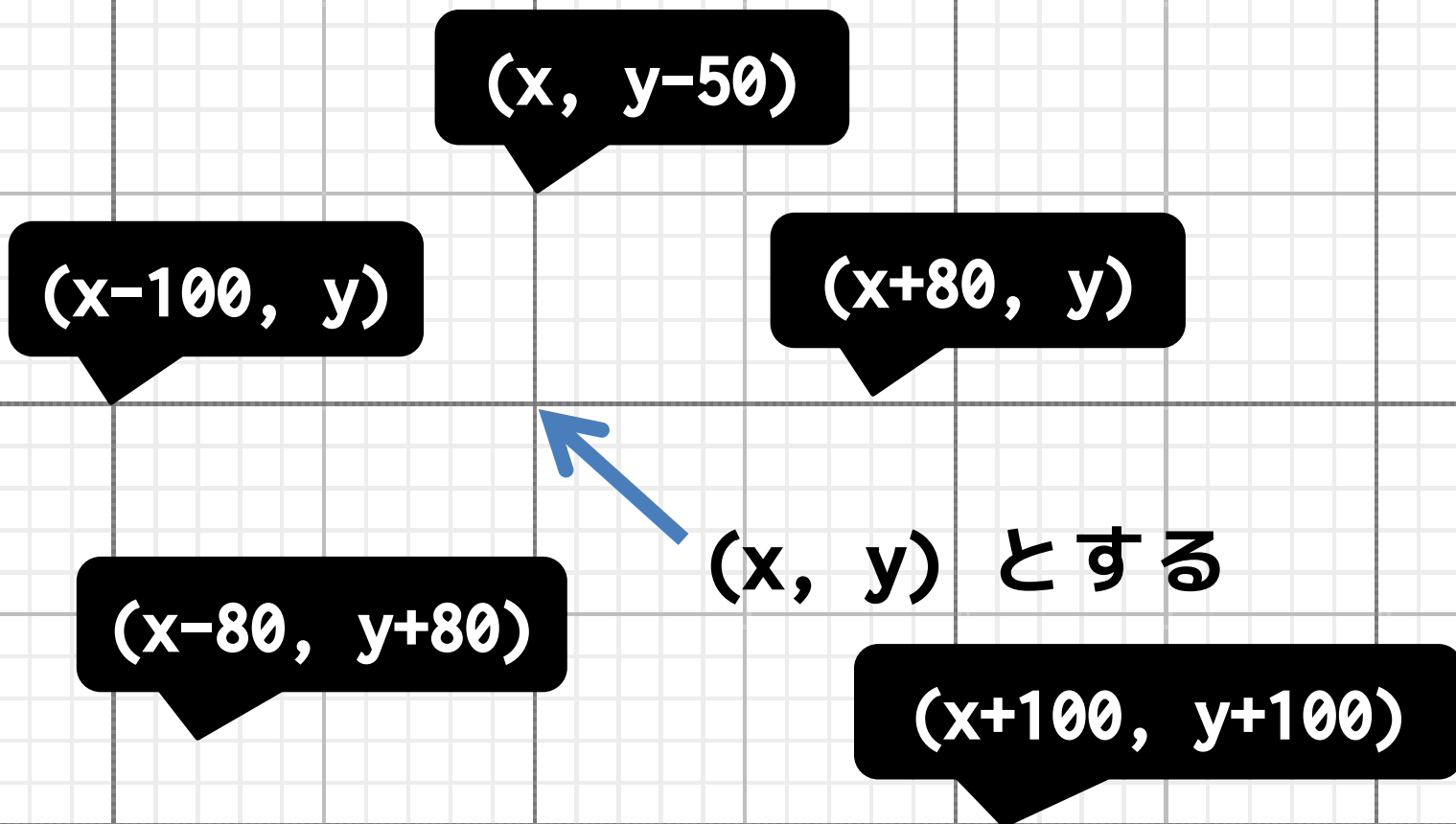
```
// xとyを使ってキャラクタを描こう！  
// 足し算と引き算の組み合わせで描画してみる！！
```

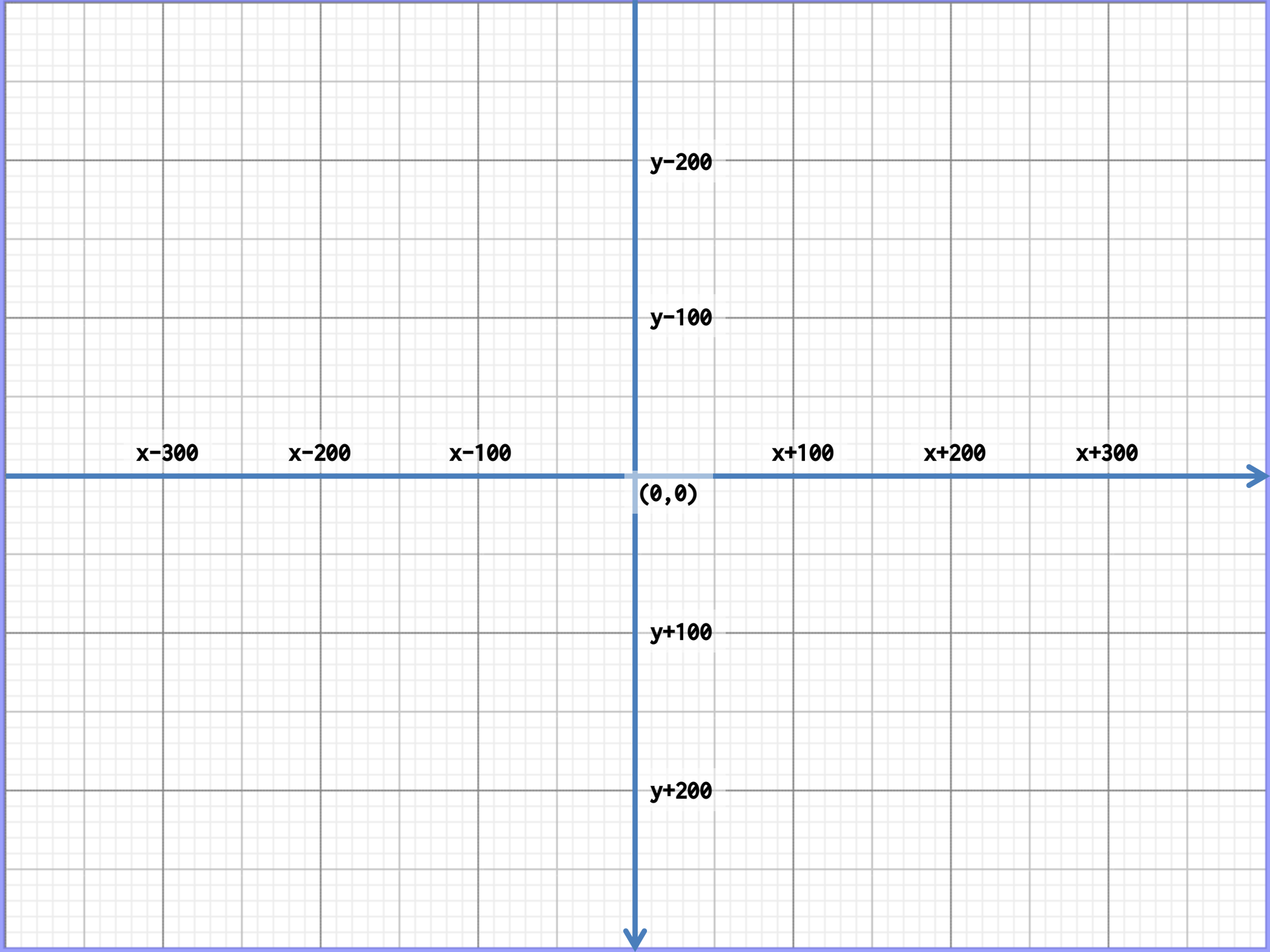
```
void drawCharacter( int x, int y ){  
  
}
```

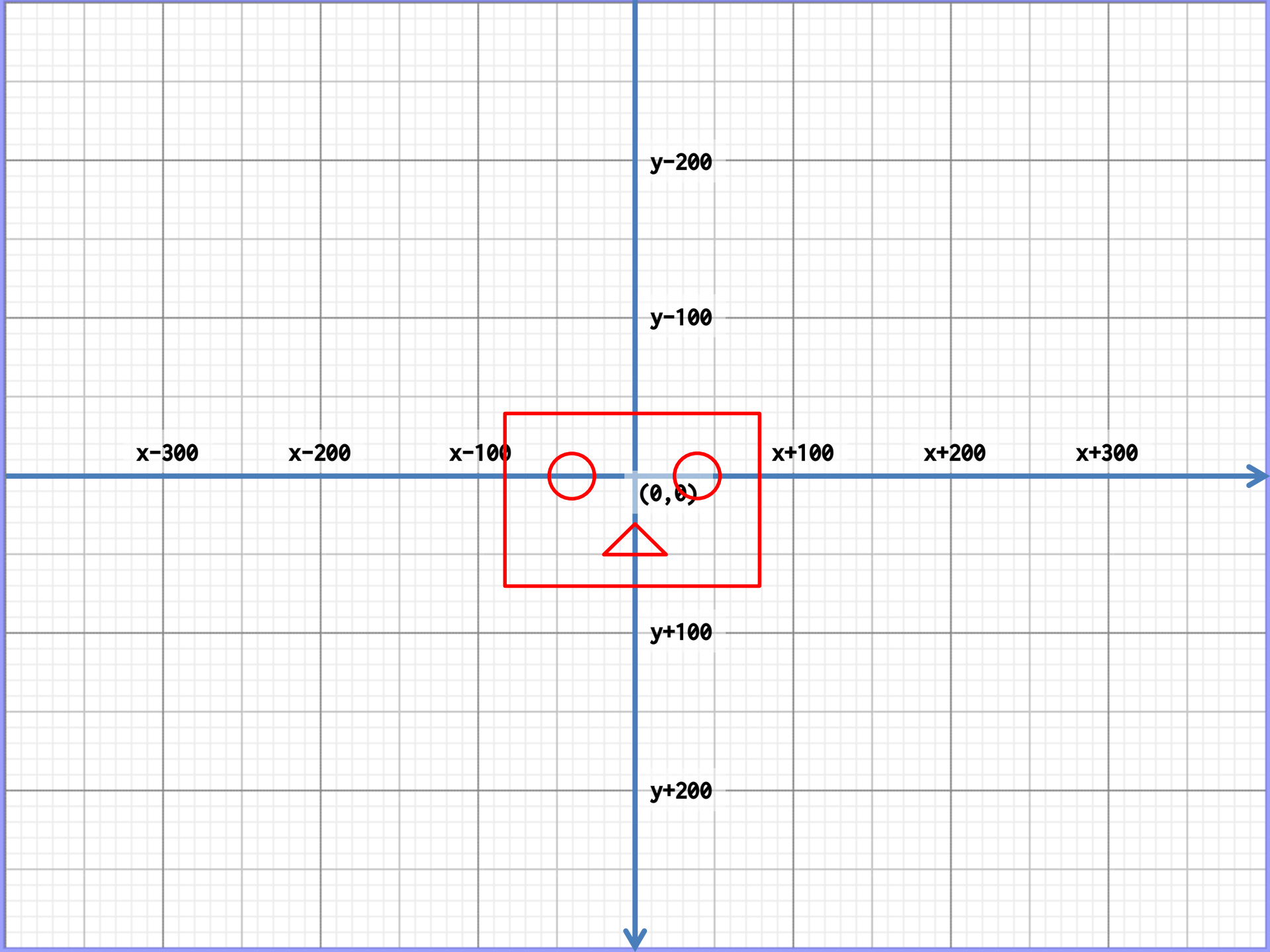
今は drawCharacter って何？
と考えずに、とりあえず
書いてxとyを使って
描画してみてください

**必ず課題で出すので
やっておきましょう！**

相対座標で描こう！





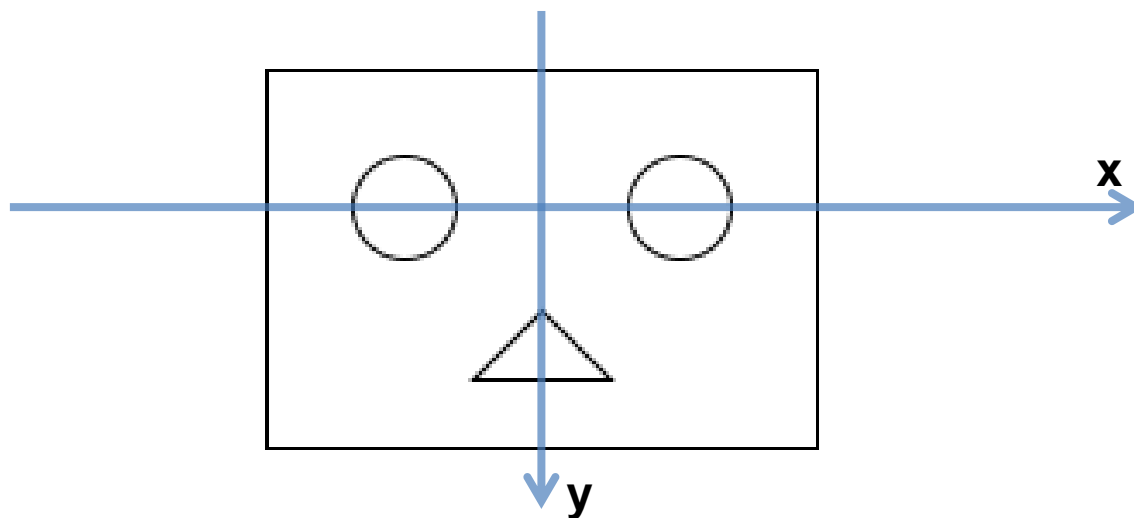


相対座標で描こう！



```
// xとyを使ってキャラクターを描こう！  
// 足し算と引き算の組み合わせで描画してみる！！  
void drawCharacter( int x, int y ){  
    rect( x-80, y-40, 160, 110 );  
    ellipse( x-40, y, 30, 30 );  
    ellipse( x+40, y, 30, 30 );  
    triangle( x, y+30, x+20, y+50, x-20, y+50 );  
}
```

例



[超重要] インデント



- プログラムのどこからどこまでが一緒に処理されるものかを把握するために利用

どちらが見やすいですか？

```
void setup(){
size(400,400);
}
void draw(){
background(255,255,255);
if( mouseX<200 ){
fill(255,0,0);}
else {
fill(0,0,0);}
ellipse(200,200,100,100);
}
```

```
void setup(){
  size(400,400);
}
void draw(){
  background(255,255,255);
  // マウスのx座標で色を変える
  if( mouseX<200 ){
    fill(255,0,0);
  } else {
    fill(0,0,0);
  }
  ellipse(200,200,100,100);
}
```

[超重要] インデント



• インデント

- 「{」と「}」との間を右に1段段下げして左を揃える
- ゲシュタルトの心理学の良い連続！

```
void setup(){
```

```
  Tab size( 400, 300 );
```

```
  Tab background( 255, 255, 255 );
```

```
}
```

```
void draw(){
```

```
  Tab fill( 255, 0, 0 );
```

```
  Tab ellipse( 200, 150, 150, 150 );
```

```
}
```

Tab はキーボードの「Tabキー」の事

インデントのポイント



- 色々な記法があるのですが…
 - {} の中は1段右へ
 - {} の中に {} があるとさらに1段右へ
 - 段を右へ動かす場合はタブを利用
 - 「{」や「}」は1つの行にする(特に, 「}」については, それだけの行を作ったほうがわかりやすい)

Edit → Auto Format ですべてインデント
されるので困ったら試そう
Ctrl+T を押すだけでもOK!

コメントとは



- プログラムとしては実行されない，人間用の説明文で，後で読むためにどんどん書く！
- コメントは「**//**」または「**/***」と「***/**」のペアを利用
 - 「**//**」は「**//**」以降行末までをコメントとして解釈
 - 「**/***」と「***/**」は，セットで間を全てコメントとして解釈

```
// 背景を白色に設定
```

```
background( 255, 255, 255 );
```

```
fill( 255, 0, 0 ); // 赤色で塗りつぶす
```

```
// fill( 0, 0, 255 ); // 青色で塗りつぶす
```

```
/* ここから  
   ここも
```

```
   ここまでもコメントですよー */
```

動かない時に消さずコメントアウトしよう！



- 短いプログラムは良いのだけれど、長くなるとすぐにワケが分からなくなります
- 下記の2つの点に注意しましょう！
 1. コメントをしっかり書く
 2. インデントでわかりやすく！



- Processing入門
 - <http://www.cp.cmc.osaka-u.ac.jp/~kikuchi/kougi/simulation2009/processing0.html>
- Processing リファレンス
 - <http://processing.org/reference/>