Processing の遊び方

Yuri Kazuna

2015 年 8 月 23 日

1　一歩目

1.1　Processing とは

Processing はプログラミング言語の一つです。もうちょっと詳しく言うと、Java の機能を制限してお絵かきに特化させたものです。書き方は C 言語によく似てるので、馴染みやすいかと思います。C 言語を覚えていなくてもそれはそれで大丈夫です。

1.2　Processing を拾う

https://processing.org/download/?processing

図 1　Processing 置き場

ここに、Processing があります。使っている OS にあわせてダウンロードしましょう。

ちなみにこの Processing のバージョンは 3.0 です。この 8 月にリリースされたので、Web 上の情報と実際の動きが合わない (情報が古い) ことがあります。変だなあと思ったらゆーりに相談してください。

1.3　Processing を動かす

ダウンロードした書庫 (zip とかtar.gz とか) を解凍すると、processing(.exe) というファイルがあると思います。それを実行しましょう。(インストール不要なんです)

そうするとポップアップウィンドウに「Get Started」みたいなのがあるので、それをクリックしてポップアップウィンドウを消します。

図 2　ポップアップウィンドウは消して

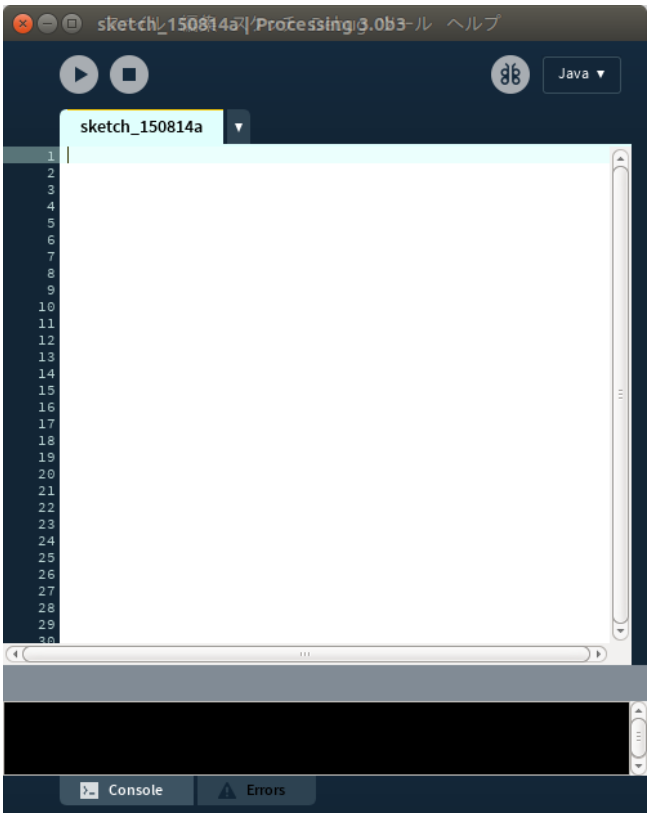

図 3　エディタ画面と向き合う

エディタ画面がありますね？ それの左上にある三角形を押しましょう。
すると小さなウィンドウが出てきましたね？

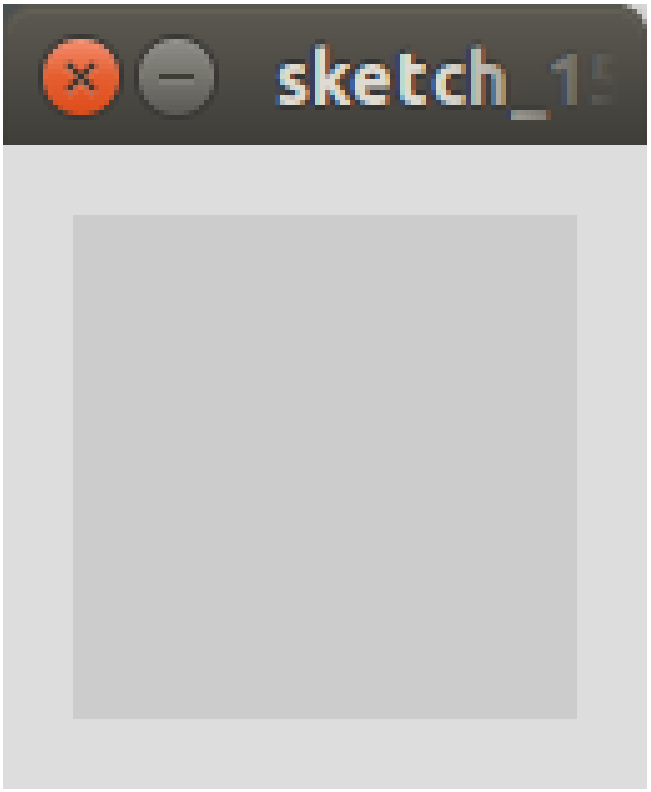

図 4　実行画面

おめでとうございます！ あなたの書いたコードが動きました！ (えっ、まだ何も書いてないって？)

2　二歩目

C 言語では黒い画面を相手にプログラムを書いていましたが、Processing では今出てきた実行画面のウィンドウを相手にします。
二歩目では、ウィンドウにお絵かきをする方法を紹介していきます。

2.1　文法

関数の呼び出し (実行) は、
関数名 (引数をいくつか);
というふうにします。引数の数が 3 だと、カッコの中は (引数 1, 引数 2, 引数 3) となります。
実行の仕方は次章で説明しますが、この章で扱う内容を試したいなら、Processing のエディタ画面に直接、

```
ellipse(30,30,40,40);
```

などと書けばよいです。(この時、関数名の部分が ellipse に変わって、いくつかの引数がが 4 つの数字に変わったと考える) main 関数などは必要ありません。

書いたらエディタ画面左上の三角形を押して実行です。

2.2　図形

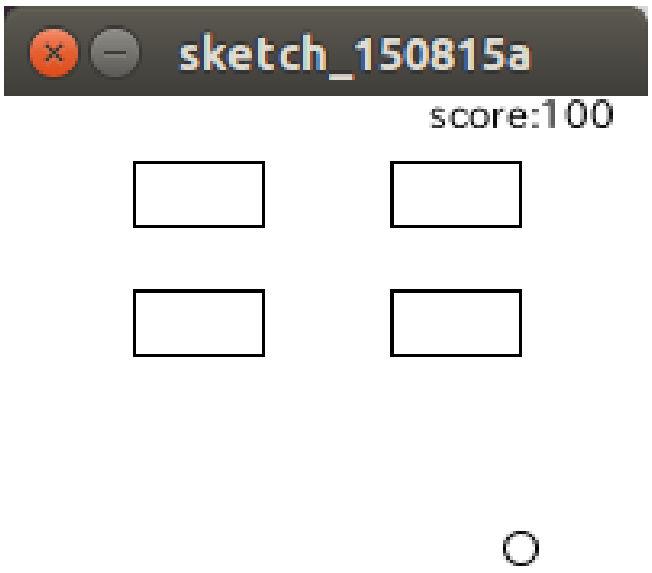

図 5　よくあるブロック崩し

上のよくある感じのブロック崩しを考えてみましょう。見えるものはなんですか？

「ボールの座標」？ それは見えてないですよね。

「ボール」？ ボールなんて描いた記憶が無いですね。

円ですね。円です。

あとは、四角、線、文字。

これだけあれば結構なものが表現できます。画像については後述します。

2.2.1　円を描く:ellipse(float,float,float,float)

色指定:stroke, fill

円を描くには ellipse 関数を使います。

```
ellipse(中心の X 座標，中心の Y 座標，幅，高さ);
```

2.2.2　長方形を描く:rect(float,float,float,float)

色指定:stroke, fill

長方形を描くには rect 関数を使います。

```
rect(左上の X 座標，左上の Y 座標，幅，高さ);
```

2.2.3　直線を描く:ellipse(float,float,float,float)

色指定:stroke

直線を描くにはline 関数を使います。

```
line(始点の X 座標，始点の Y 座標，終点の X 座標，終点の Y 座標);
```

2.2.4　文字を描く:text(string,float,float)

色指定:fill

文字を画面に描くにはtext 関数を使います。

```
text("書きたい文字列"，左上の X 座標，左上の Y 座標);
```

書きたい文字列は半角ダブルクオーテーション (", Shift を押しながら 2 を押すと出ることが多い) でくくらないと表示できないと思います。

2.3　色

ブロック崩しの例では黒と白しかありませんが、なにかを作るときには黒と白以外の色も必要になりますね。

以下に、色を扱う関数を紹介します。

2.3.1　画面を塗りつぶす:background(float)/background(float,float,float)

画面を 1 色で塗りつぶしたいときには background 関数を使います。

ところで、上には関数の引数のとり方が 2 種類あるように書いてますが、実際そうです。以下に、使い方を示します。

```
background(明度);
```

または

```
background(赤の強さ (R)，緑の強さ (G)，青の強さ (B));
```

下の例のように RGB 値で背景色を指定する場合、それぞれの値を同じ値にすると、その色を明度とした上の例での塗りつぶしと同じ色で塗りつぶされます。

？？？ と思ったら、やってみるのが早いです。

また、各値の最小は 0、最大は 255 です。それ以下・以上は無視されます。

2.3.2　線の色を変える:stroke(float)/stroke(float,float,float)

前項で示した図形のうち、色指定に stroke を含むものの色を変えます。先ほどと同様に色指定の方法は 2 種類あります。

```
stroke(明度);
```

または

```
stroke(R,G,B);
```

この関数は **stroke 関数以降の図形関数の色だけ**を変えます。stroke 関数以前に実行された描画については色を変えません。

2.3.3　塗りつぶしの色を変える:fill(float)/fill(float,float,float)

前項で示した図形のうち、色指定に fill を含むものの色を変えます。例によって色指定方法は 2 種類あります。

```
fill(明度);
```

または

```
fill(R,G,B);
```

この関数は **fill 関数以降の図形関数の色だけ**を変えます。fill 関数以前に実行された描画については色を変えません。

2.3.4　線を表示しない:noStroke()

次に stroke 関数が呼ばれる (実行される) まで、色指定に stroke を含む関数で線を描きません。

```
noStroke();
```

例によって、noStroke が呼ばれた後の図形関数にのみ影響します。

ちなみに、noStroke が有効になっていると、line 関数で何も描画されません。

2.3.5　塗りつぶしをしない:noFill()

次に fill 関数が呼ばれる (実行される) まで、色指定に fill を含む関数で塗りつぶしをしません。

```
noFill();
```

例によって、noFill が呼ばれた後の図形関数にのみ影響します。

ちなみに、noFill が有効になっていると、text 関数で何も描画されません。

2.4　問題

二歩目を歩けているかどうか、確認の問題を出します。次の画面を出力してください。

関数は並べれば上から下に実行されます。どのように書けば図形がうまく隠れるか、どうすれば綺麗に塗り分けられるか、いろいろ考えてみてください。

図 6　二歩目の問題

位置は完全に同じでなくてもいいですが、テキストの表示、隠れ具合はだいたい同じにするのが良いです。

色の値は 0 と 255 しか使ってません。適当に探してください。