Processing による画像の取り扱いと単純な縮小

Processing による画像の取り扱い

本演習では 1 年の基盤ユニットで扱った Processing を用いて、簡単な画像処理の基礎を学びます。混乱をさけるために外部ライブラリは用いずに、Processing 本体の機能のみを使うことにします。

Processing で画像を扱うためには、PImage というデータ型を使います。Processing サイトの PImage についてのリファレンス(https://processing.org/reference/PImage.html)には次のような説明があります。「Processing では、.gif、.jpg、.tga、.png 画像を表示できます。 画像は 2D および 3D スペースで表示できます。 画像を使用する前に、loadImage () 関数でロードする必要があります。 PImage クラスには、画像の幅と高さのフィールドのほか、画像内のすべてのピクセルの値を含む、pixels []と呼ばれる配列が含まれています。以下で説明する方法を使用すると、画像のピクセルとアルファチャネルに簡単にアクセスでき、合成プロセスが簡略化されます。pixels[]配列を使用する前に、必ず loadPixels（）メソッドを使用して、ピクセルデータが正しく読み込まれていることを確認してください。新しい画像を作成するには、createImage（）関数を使用します。 構文 new PImage（）は使用しないでください。」

ピンとこない方もおられると思うので、一つずつ確認していきましょう。まずは、以下のようなサンプルプログラムで既存の画像を読み込んで、表示してみましょう。

```
PImage photo;

void setup() {
  size(100, 100);
  photo = loadImage("test1.jpg");
}

void draw() {
  image(photo, 0, 0);
}
```

[自己確認問題 1]上記プログラム中の画像ファイル名を自分の PC にある適当な画像ファイル名に置き換えて、実行してみてください。プログラムから画像ファイル名だけで画像を読み込むには何かしなくちゃいけなかったことを思い出してください。思い出せない方は、ネットで検索するか、基盤ユニットのテキストを引っ張り出して調べてください。

ここで、画像サイズと表示される Window のサイズが合っていないために、手作業で Window のサイズを変更しなければならないことに不満が残ります。これを解決するために、先ほどのプログラムを以下のように変更してみます。

```
PImage photo;

void setup() {
  photo = loadImage("test1.jpg");
  size(photo.width, photo.height);
}

void draw() {
  image(photo, 0, 0);
}
```

[自己確認問題 2]上記プログラム中の画像ファイル名を自分の PC にある適当な画像ファイル名に置き換えて、実行してみてください。

すると、「Please fix the size() line to continue.」というエラーが出てしまいます。setup()関数の中では、最初に size()関数を呼び出して Window のサイズを決めてやる必要があるのです。実は、この解決策が「Changes in 3.0 - processing/processing Wiki(https://github.com/processing/processing/wiki/Changes-in-3.0)」というサイトに載っています。まずは、ここに掲載されているプログラムを実行してみましょう。

```
void setup() {
  size(400, 400);
  surface.setResizable(true);
}

void draw() {
  background(255);
  line(100, 100, width-100, height-100);
}

void keyPressed() {
  surface.setSize(round(random(200,500)),round(random(200,500)));
}
```

[自己確認問題 3]上記プログラムを実行して何が起こるか確認してみてください。

[演習問題 1]上記プログラムを参考に、読み込んだ画像サイズに合わせて Window のサイズを自動調整するプログラムを作成してください。ソースプログラムと複数の実行結果をつけて報告してください。

次は、画像の保存です。保存は、画像情報が含まれる PImage に対して save()メソッドを呼び出すだけです。例えば、「test1.jpg」を読み込んで「test2.png」という名前で保存するプログラムは以下のようになります。

```
PImage photo;

void setup() {
  size(100, 100);
  photo = loadImage("test1.jpg");
  photo.save(“test2.png”);
}

void draw() {
  image(photo, 0, 0);
}
```

[自己確認問題 4]上記プログラム中の画像ファイル名を自分の PC にある適当な画像ファイル名に置き換えて、実行してみてください。

しかし、入力画像ファイルの名前を指定するのはともかく、処理した結果を保存する際のファイル名も指定しなければならないのは釈然としません。例えば、「test1.jpg」に対して何らかの処理をした場合には、「test1_xxxx.jpg」などと元の画像ファイルとの関連をファイル名に残すということです。しかも、規則的な変化なのでプログラムでファイル名を自動生成できるというメリットもあります。

[演習問題 2]〇X〇X.jpg という文字列を与えると、〇X〇X_raw.jpg という文字列を返すユーザ定義関数を作成し、上記プログラムと融合することによって入力画像ファイル名と関連のある名前でそのまま保存するプログラムを作成してください。ソースプログラムと複数の実行結果をつけて報告してください。ちなみに、raw というのは「生(なま)」という意味で、何も処理していないということです。

最後は、画像ファイルからデータを読み出して PImage 型の変数を作成するのではなく、

画像サイズと色解像度を指定してメモリ中に PImage 型の変数を作り出す方法についてです。これは、入力画像と処理結果のサイズや色解像度が異なる場合に必要になります。creaeImage()という関数に幅、高さ、色解像度をパラメータとして与えると、注文通りの PImage 型変数を作成して返してくれます。色解像度の指定には、RGB(フルカラー), ARGB(透明度つきフルカラー), ALPHA(グレースケール)の３種類が使えます。以下は、日本の国旗である日の丸を生成し、「japan_flag.png」という名前で保存するプログラムです。ちなみに規定によれば、「旗の形は縦が横の 3 分の 2 の長方形。日章の直径は縦の 5 分の 3 で中心は旗の中心。地色は白色、日章は紅色」とされているそうです。

```
int w, h, radius;
PImage flag;

float calcDistance(int x1, int y1, int x2, int y2) {
  int d1 = x1 - x2, d2 = y1 – y2;
  return sqrt(d1*d1 + d2*d2);
}

void setup() {
  size(600, 400);
  w = width; h = height; radius = int(h * 3.0 / 5.0 / 2.0);
  flag = createImage(w, h, RGB);
  int cx = w / 2, cy = h / 2;
  color red = color(255, 0, 0), white = color(255, 255, 255);
  for (int y = 0; y < h; y++) {
    for (int x = 0; x < w; x++) {
      if (calcDistance(x, y, cx, cy) <= radius) {
         flag.pixels[y * w + x] =  red;
      }
      else {
         flag.pixels[y * w + x] =  white;
      }
    }
  }
  flag.save("japan_flag.png");
}
```

```
void draw() {
  image(flag, 0, 0);
}
```

[演習問題 3]上記プログラムを参考に、読み込んだカラー画像ファイルをグレースケールに変換して保存するプログラムを作成してください。保存する際のファイル名は、入力画像ファイル名が〇X〇X.jpg であれば、〇X〇X_gray.jpg にしてください。ソースプログラムと複数の実行結果をつけて報告してください。

単純な縮小

ここでは、画像の縦横を整数分の 1 に縮小するために、ピクセルを間引くだけの単純な縮小をやってみます。例えば、幅 77 ピクセル、高さ 60 ピクセルの画像を 4 分の 1 に縮小するとしましょう。その場合の手順は以下のようになります。

1.画像ファイルを読み込みます。

2.入力画像のサイズ(ここでは、幅 77 ピクセル、高さ 60 ピクセル)と縮小率(ここでは、1/4)から処理結果の画像サイズ(ここでは、幅 int(77/4)=19、高さ int(60/4)=15)を決定し、対応する PImage 型の変数を作成します。

3.x については 0,4,8,…,76、y については 0,4,8,…,56 の値をもつ座標のピクセル値のみを、処理結果の変数に格納する。

[演習問題 4]上記手順を参考に、読み込んだカラー画像ファイルを間引くことによって縦横を整数分の 1 に縮小して保存するプログラムを作成してください。保存する際のファイル名は、入力画像ファイル名が〇X〇X.jpg であり、4 分の 1 に縮小した場合には、〇X〇X_d4.jpg にしてください。ソースプログラムと複数の実行結果をつけて報告してください。ちなみに、d は decimate(間引く)の頭文字を表しています。