メディアプログラミング演習―第１回（第１テーマ１日目）―

第１テーマ：画像の扱い，画像処理１

メディアとして画像を扱う．ビットマップ画像については，メディア演習において，ビットマップ画像の生成，グラデーションパターンの生成，ブレゼンハムアルゴリズム（ビットマップ画像としての線分描画）を扱った．また，静止画処理では，画像処理プログラムを用いて，画像変換を行った．

本テーマは，画像処理の方式（アルゴリズム）を実際の画像データに対するプログラムを作成することにより理解することを目的とする．合わせて，幾つかのプログラミングテクニックについても演習する．

テーマ１：カラー画像の表示をグレースケール処理

演習１－０：Processing での画像の構造：初めに processing での画像の扱いの基本を理解する。

＊プログラム（sample1）および class PImage（補助資料１－１）。

＊配列の添え数変換（補助資料１－２）

演習１－１：　カラーとグレースケール

カラー画像の構成とグレースケール画像への変換処理を学ぶ。

＊カラー：光の三原色（R：赤、G：緑、B：青）の加重平均で表現.

Windows などでは，それぞれに 8 ビットずつ割り当て。

＊グレースケール：白－灰色－黒．三原色を同一比で混合.

カラー画像をグレースケールに変換する式

GLAY = 0.3 * R + 0.59 * G + 0.11 * B

（人間の目は、緑に敏感で、青に余り敏感でない．この特性に従い，自然に見えるように、0.30, 0.59, 0.11 と重み付け）

実際のプログラム：

```
color c = img_in.pixels[pos];  // クラス color
  float r = red( c );        // color から、特定色の抽出
  float g = green( c );
  float b = blue( c );
  float gray = 0.3 * r + 0.59 * g + 0.11 * b;
img_out.pixels[pos] = color(gray,gray,gray); //カラー変数への代入
```

```
// sample1

PImage img_in, img_out;

{
  img_in = loadImage("sample1-1.jpg"); // 画像読み込み
  img_out = createImage( img_in.width, img_in.height, RGB );  //結果領域
  size(img_in.width*2, img_in.height);  // 表示領域
  noLoop();
}

void draw()
{
  image(img_in,0,0);  //元画像の表示
  img_in.loadPixels();  // Pixcel へのコピー
  for ( int y = 0; y < img_in.height; y++)   // すべての縦方向
  {
    for ( int x = 0; x < img_in.width; x++)  //すべての横方向
    {
      int pos = x + y*img_in.width;
      img_out.pixels[pos] = img_in.pixels[pos];  Pixcel のコピー
    }
  }

  img_out.updatePixels();  // Pixcel からのコピー
  image(img_out,img_in.width,0); //結果の表示
  img_out.save("result.jpg");    //処理画像の保存
}
```

プログラム(sample1)

テーマ２：色抽出

演習１－２：(a)赤色のみ表示

```
img_out.pixels[pos] = color(r, 0, 0)
```

同様に、(b)緑色、(c)青色のみの表示

**補助資料１－２：２次元配列の扱い**

＊２次元ビットプレーンと１次元配列

PImage クラスの配列 pixels は，左上から右方向、右端まで行ったらその下の行と一元配列で表現されるが，イメージは２次元的に扱えた方がよい.

２次元(x,y)ー＞１次元(k)への変換は、以下の通りである。

配列の位置(k) = 横方向の位置(x) + 縦方向の位置(y) ＊１行分の画素

( int pos = x + y * img.width;)

これにより,

```
for ( int y = 0; y < img_org.height; y++)       // 縦方向 y 番目の
   {
     for ( int x = 0; x < img_org.width; x++)  // 横方向 x 番目
       {
           :
          int pos = x + y*img_org.width;       //ピクセル (x,y) の位置(pos)
           :
       }
   }
```

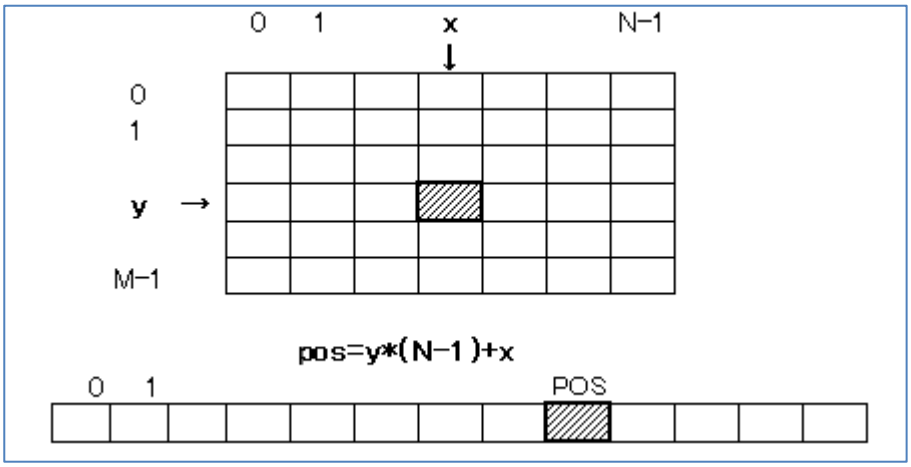


**図１－１：Pixels[]の２次元的解釈**

補助資料１－２　：　PImage について

**Description：**

must be loaded with the **loadImage()** function. The **PImage** class contains fields for the width and height of the image, as well as an array called **pixels[]** that contains the values for every pixel in the image. The methods described below allow easy access to the image's pixels and simplify the process of compositing.

Before using the pixels[] array, be sure to use the **loadPixels()** method on the image to make sure that the pixel data is properly loaded. To create a new image, use the **createImage()** function.

**Fields：**

| | |
|---|---|
| pixels[] | Array(liner) containing the color of every pixel in the image |
| width | Image width |
| height | Image height |

**Methods：**

| | |
|---|---|
| loadPixels() | Loads the pixel data for the image into its pixels[] array |
| updatePixels() | Updates the image with the data in its pixels[] array |
| save() | Saves the image to a TIFF, TARGA, PNG, or JPEG file |

**Constructor:**

PImage(width, height)

**Related:**

loadImage(), createImage()

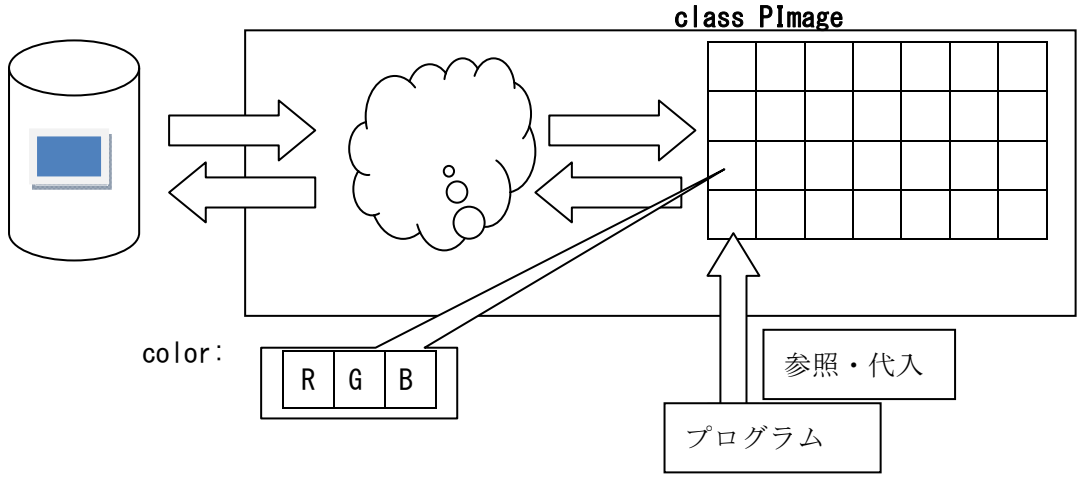


**図１－２　クラス PImage の構造**