K 会 2012 年度冬季講習　情報講座

リピーターコース　Part3　画像処理

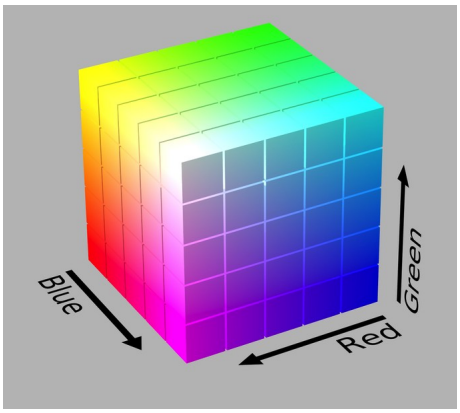
目次

3-1. 色の指定

（１）RGB 色空間と HSB(HSV)色空間

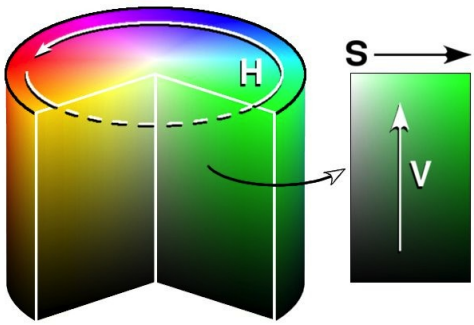
・RGB 色空間とは、色を光の三原色（R：赤、G：緑、B：青）の強さで表したものである。市販のカメラや人間の目は R, G, B のセンサーを持っているから、合理的な手法である。ディスプレイも R, G, B 発光素子の組み合わせで色を表現。

　混色（複数の光を混ぜた時にどんな色になるか）をするときに、普通にそれぞれの原色の値を足し算するだけでそれらしい値を得ることができる。



Created by Wapcaplet in the en:GIMP.

・HSB(HSV)色空間とは、H：色調、S：鮮やかさ、B(V)：明るさの 3 つの数値で色を表現したものである。画像の色調を補正したりするときに役に立つ。一対一ではないが、RGB と相互に変換できる（B が 0 のときは H や S がどんな値でも黒である）



POV-Ray Object Collection

Processing には色見本の機能があるので、使ってみよう（メニューの Tools→Color selector）

（２）colorMode 命令

colorMode 命令は、fill や stroke, 後述の color 命令や red, saturation 命令などでどの色空間と、値の範囲を使うのかを指定する命令である。この命令が実行されてから、次にまた colorMode 命令が実行されるまで有効となる。

colorMode命令

```
colorMode(RGB, M)
   各色が(0～M)のRGB色空間で色を表現する
colorMode(HSB, M)
   各色が(0～M)のHSB色空間で色を表現する

デフォルトは(RGB, 256)になっている
```

（３）color 変数と、色を得る命令
一色の色を表す color 型がある。

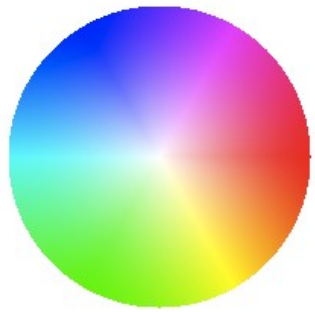
color型

color c0;	宣言
c0 = color(R, G, B);	代入（colorModeで指定した色空間で）
red(c0), green(c0), blue(c0)	R値、G値、B値をfloat型で得る
hue(c0), saturation(c0), brightness(c0)	H値、S値、B値をfloat型で得る

値はcolorModeで指定した値の範囲を取る。
colorModeがRGBでもH, S, B値を得ることはできる。

今まで fill や stroke では fill(R, G, B)のようにしていたが、colo 型の変数 c0 を使って fill(c0)などと書いても良い。

（４）課題
(i) colorMode(HSB, 100)と、color 変数を使って右の色円盤を描け。
B 値を変化させた時に色円盤がどう変化するか調べてみよ。



(ii) hue, saturation, brightness 命令を使って、黄色（RGB で(255, 255, 0)）の HSB 値を調べよ。

3-2. 画像の読み込みとピクセルの編集

（１）画像の読み込みと表示

写真や絵といった画像ファイルを読み込む方法を学ぶ。

画像の扱い

PImage img;
　画像を表すデータ型PImage
img = loadImage("ファイル名");
　imgに画像を読み込む
img.resize(W, H);
　読み込んだ画像の横、縦幅をW×Hに変える
image(img, X, Y);
　画像imgを場所(X, Y)が左上になるように描画する

次のプログラムを書こう。

空欄のところは、好きな画像を使おう（デスクトップの image フォルダにいくつか写真を用意してある。自分で画像を描いたり、ネットから落としてきたりしてもいい）。

画像によっては縦横比が 600*450 になっていないが、resize 命令の 600, 450 のどちらかを 0 にすると、空気を読んで縦横比を維持して拡大縮小してくれる。

```
PImage img;

void setup(){
  size(600, 450);
  img = loadImage(                );
  img.resize(600, 450);
  image(img, 0, 0);
}

void draw(){
}
```

（２）ピクセルの直接編集

color 型の 1 次元配列として画像を扱うことができる。すると、画像の各座標(x, y)の色（ピクセルという）を直接操作することができる。

ピクセルの直接編集

img: PImage型のデータとする

img.pixels[y*img.width + x]
　この配列に画像の(x, y)の位置の色がcolor型で格納されている

img.loadPixels();
　...
img.updatePixels();
　pixels配列を使うには、loadPixels()を実行する必要がある。
　pixels配列を書き換えた後にupdatePixels()命令を実行すると、
　書き換えが画像に反映される。

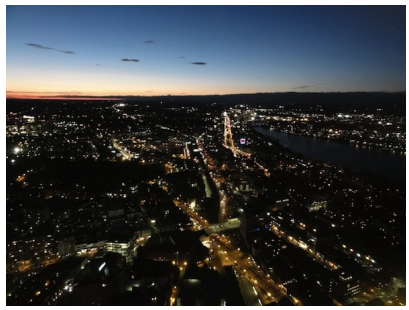
（３）例

(i) マウスがある位置の色を RGB, HSB で表示するプログラム

```
PImage img;

void setup(){
  size(600, 450);
  img = loadImage("photo1_2.jpg");
  img.resize(600, 450);
  image(img, 0, 0);
}

void draw(){
  img.loadPixels();
  color c = img.pixels[mouseX+img.width*mouseY];
  println(red(c) + "  " + green(c) + "  " + blue(c) + "  :  "
    + hue(c) + "  " + saturation(c) + "  " + brightness(c));
}
```

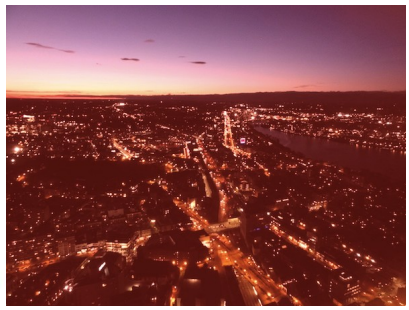


(ii) 全てのピクセルの R 値を 100 増やすプログラム

```
PImage img;

void setup(){
  size(600, 450);
  img = loadImage("photo1_2.jpg");
  img.resize(600, 450);
  noLoop();
}

void draw(){
  img.loadPixels();
  for(int ix=0; ix<img.width; ix++){
    for(int iy=0; iy<img.height; iy++){
      color c = img.pixels[ix+img.width*iy];
      img.pixels[ix+img.width*iy] = color(red(c)+100, green(c), blue(c));
    }
  }
  img.updatePixels();
  image(img, 0, 0);
}
```



（４）課題

(i) RGB フィルタ

ピクセルの RGB 値を変更するフィルタを作ってみよう。例えば右図は青色だけを透過する透明な板を通してみた先ほどの写真である。

(ii) HSB フィルタ

マウスの位置によって HSB の H 値と S 値が変わるフィルタを作ろう。H 値は上限を超えると強制的に上限にされてしまうので、たとえば上限 256 で H に 100 を足したいなら、(H+100)%256 と書くと良い。

左の写真は色調を変えた写真、右は HSB の変更で白黒にしてみた写真である。

(iii) 複合カラーフィルタ

HSB や RGB をいろいろいじると、次のような古くなった写真が作れる。

3-3. 画像合成

右のプログラムは、背景画像 bg の上に前景画像 fg を重ねて描画するプログラムである。夜景の画像と花火の画像をこのプログラムで合成すると、次のような画像になる。

```
PImage bg, fg;
int X = 100;
int Y = 50;

void setup(){
  size(400, 400);
  bg = loadImage("../photo7.jpg");
  bg.resize(400, 400);
  fg = loadImage("../photo11.jpg");
  fg.resize(270, 180);
}

void draw(){
  bg.loadPixels();
  fg.loadPixels();
  for(int ix=0; ix<270; ix++){
    for(int iy=0; iy<180; iy++){
      color cf = fg.pixels[(iy*fg.width + ix)];
      bg.pixels[(Y+iy)*bg.width + (X+ix)] = cf;
    }
  }
  bg.updatePixels();
  image(bg, 0, 0);
}
```

draw 命令の中の for 二重ループで、fg の(ix, iy)のピクセルを bg の(X+ix, Y+iy)の位置に重ね書きしている。

・課題

（１）さっきの画像では、花火画像の黒い部分がそのまま重ね書きされてしまっている。このプログラムを改良して、右のような花火の部分だけを重ね書きした合成写真を作れ。

（２）同様にして空の画像と飛行機の画像を合成せよ。



3-4. 光の演出

（１）光球の描き方

ここでは右図のような光の描き方を学ぶ。
このプログラムは、まず写真の代わりに空白の画像を用意する。

createImage
createImage(W, H, RGB)
W*Hの画像を作る。
RGBをARGBにすると透明色が使えるようになる。

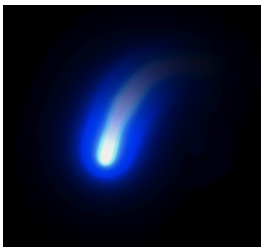
draw が実行された時の処理は次のとおりである。

・全部のピクセルについて、
　マウスとの距離 d を計算（dist 命令）
　マウスに近いピクセルほど RGB 値を加算する。ここでは 1/(d*d)に比例して加算している。加算した結果が colorMode の上限 100 を超えると 100 に切り捨てられるので、マウス付近は白(100, 100, 100)になる。
　RGB 値を数%減らす。こうするとマウスから遠い場所の色は急速に黒に減衰していく。

```
PImage img;

void setup(){
  size(400, 400);
  img = createImage(400, 400, RGB);
  colorMode(RGB, 100);
}

void draw(){
  img.loadPixels();
  for(int ix=0; ix<400; ix++){
    for(int iy=0; iy<400; iy++){
      float d = dist(mouseX, mouseY, ix, iy);
      color c = img.pixels[iy*img.width+ix];
      if(d<150){
        c = color(red(c)+(1000)/d/d, green(c)+(2000)/d/d, blue(c)+(10000)/d/d);
      }
      img.pixels[iy*img.width+ix] = color(red(c)*0.92, green(c)*0.92, blue(c)*0.92);
    }
  }
  img.updatePixels();
  image(img, 0, 0);
}
```

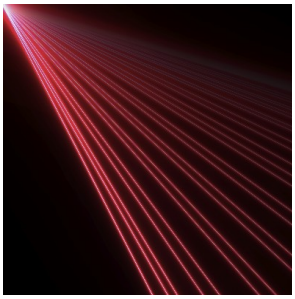


色や光球の大きさを変えてみよう。また、マウスではなく変数 x, y, vx, vy を作って壁で反射する等速直線運動を行う光球にしてみよう。

（２）レーザー

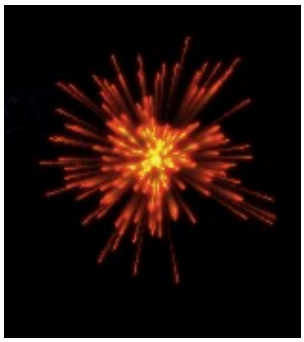
次の点と直線の距離を求める命令を使ってレーザーを作ろう。

```
// (ax, ay)と(bx, by)を通る直線と、点(px, py)との距離
float line_dist(float ax, float ay, float bx, float by, float px, float py){
  float dx = bx-ax;
  float dy = by-ay;
  float t = (dx*(px-ax) + dy*(py-ay))/(dx*dx + dy*dy);  // 内分(外分)する比率
  float cx = (1-t)*ax + t*bx;
  float cy = (1-t)*ay + t*by;  // 内分(外分)点の位置
  return sqrt((px-cx)*(px-cx) + (py-cy)*(py-cy));
}
```



（３）爆発

(1)で作った光球を花火のように放射状に複数飛ばし、爆発の演出を作ってみよう。プログラムが重くなりやすいので注意。さっきのプログラムでは一つの光球について、画面のすべてのピクセルとの距離を計算していたが、光球の大きさが小さくていいなら、光球の周囲 10*10 マスなどだけ計算することでプログラムを軽くできる。

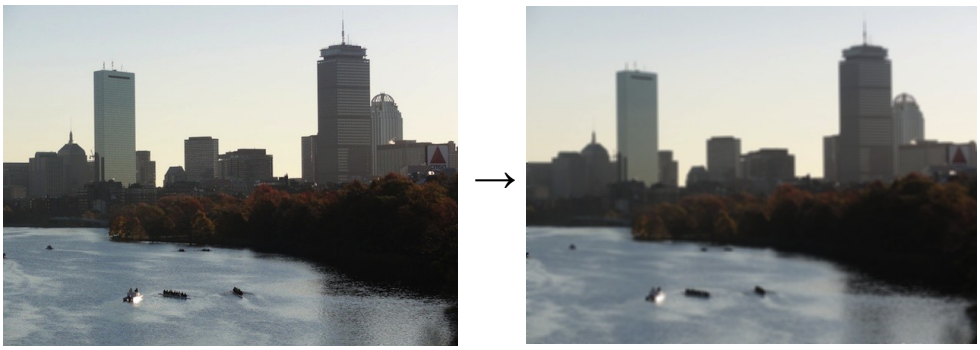


3-5. 周囲を見るフィルタ

（１）ぼかしフィルタ

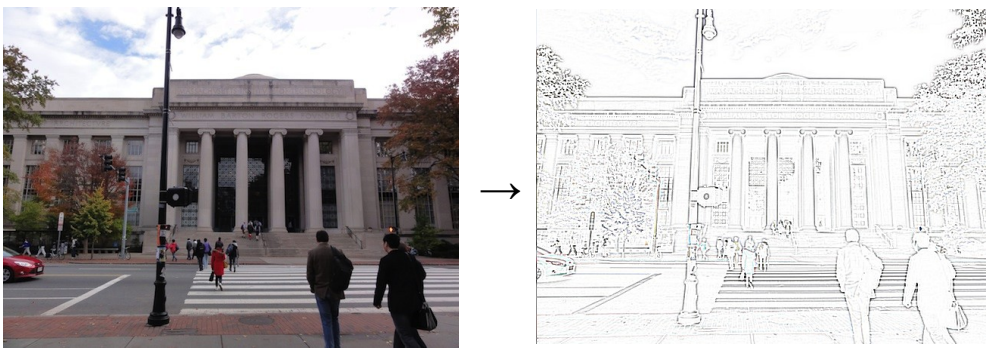
各ピクセルの周囲 5*5 ピクセルの RGB 値の平均をとった画像を作ると、下の右図のようにぼかしがかかる。このプログラムを実装せよ。

・Pimage img0 に元の写真を読み込む
・Pimage img を img0 と同じサイズで createImage で作る。
　これが処理後の写真となる。
・img の各ピクセルについて、img0 の周囲数ピクセルの平均値を代入する
　（端に飛び出してしまうことに注意）



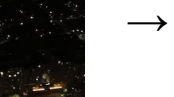
（２）輪郭検出フィルタ

次のような輪郭を抽出するフィルタを、どう作ればいいか考えて実装せよ。
ヒント：輪郭は、色が大きく変化している点である



3-6. 画像合成フィルタ

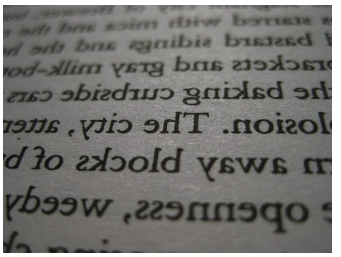
（１）きらきらフィルタ
　写真の明るい部分にだけ別の画像（十字の形の光）を合成すると、右のようなキラキラ感が増した写真が作れる。このプログラムを実装せよ。

（２）鉛筆画フィルタ
　写真を鉛筆画に変換するフィルタを作ろう。
　合成に使うのは次のような斜めの線の画像である。

　さらに 3-5 の輪郭フィルタを重ね書きすると、次のような画像ができる。


3-7. 変形フィルタ



（１）鏡

最後に、ピクセルの位置を移動させて画像を変形するタイプのフィルタをやろう。まずは最も簡単な、左右反転を作る。

img が表示する画像、img0 が元の画像だ。

```
for(int ix=0; ix<img.width; ix++){
  for(int iy=0; iy<img.height; iy++){
    img.pixels[iy*img.width+ix] = img0.pixels[iy*img.width+(img.width-1-ix)];
  }
}
```

（２）レンズ

右のような、マウスの周辺が拡大される拡大鏡を作れ。マウスとの距離の半分の位置にあるピクセルを持ってくれば、長さが 2 倍に拡大される。拡大範囲を表す長方形は後から書いている。

（３）ブラックホール

右のような、マウスに画像が吸い込まれるようなプログラムを作れ。画面端の扱いに注意。