

Graphics with Processing



2007-05 色彩とピクセル処理

<http://vilab.org>

塩澤秀和

5.1 色彩

色

- color型
 - 色を表す変数型
- color(成分1, 成分2, 成分3)
 - 色の生成
 - `color c = color(r, g, b);`
 - 初期モードはRGB & 0~255
- colorMode(色空間, 値範囲)
 - 色指定モードの設定
 - 色空間: RGB, HSB
 - 値範囲: 成分の上限値
 - `colorMode(色空間, 範囲1, 範囲2, 範囲3)` の形式もある
 - `fill()`, `stroke()` などにも影響
 - 例) `colorMode(HSB, 1.0);`
 - サンプル Basics → Color

色の混合

- 透明度(alpha)
 - 色の第4成分
 - 重ね塗りでの濃さを表す
 - `c = color(r, g, b, a);`
 - `fill()` や `stroke()` でも指定可
 - 例) `fill(255, 0, 0, 128);`
- blendColor(色1, 色2, 混色演算)
 - 色を混ぜて新しい色を作る
 - 混色法: BLEND, ADD など
(詳しくはマニュアル参照)

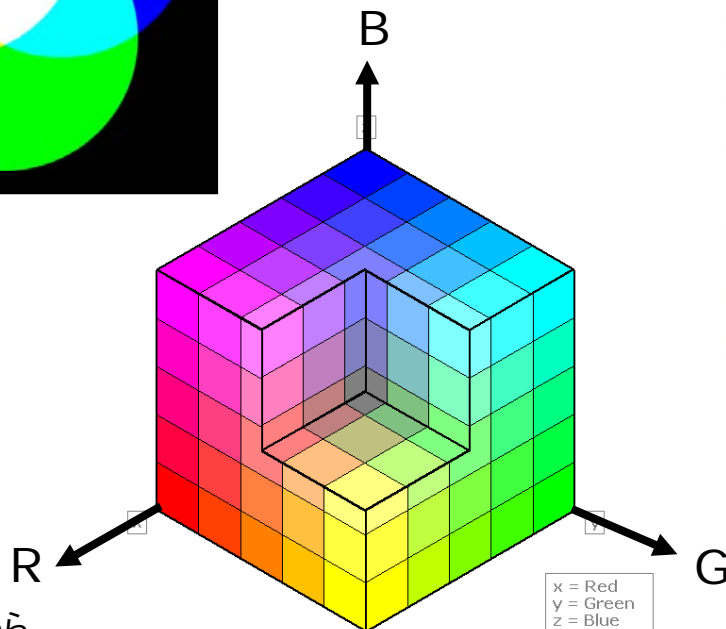
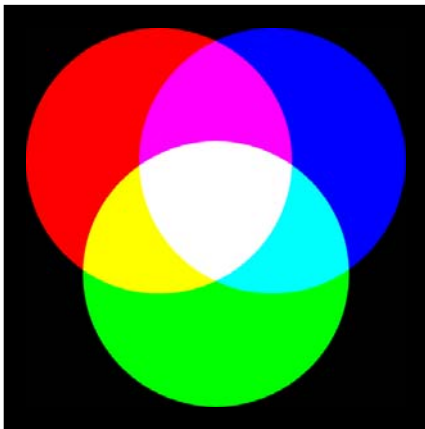
色の成分の取得

- `red(c)`, `green(c)`, `blue(c)`,
`hue(c)`, `saturation(c)`,
`brightness(c)`, `alpha(c)`

5.2 色空間

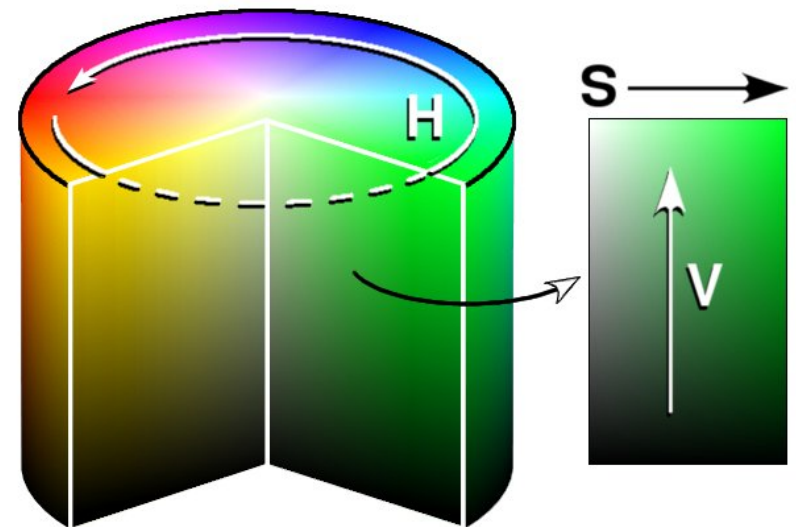
RGB色空間

- 光の三原色(赤, 緑, 青)



HSB (HSV) 色空間

- 色相 (H): 色あい
- 彩度 (S): あざやかさ
- 明度 (B or V): 明るさ



5.3 ピクセル処理

ピクセル処理

- pixels[]
 - 画面を構成する画素1点1点の色を格納している配列
 - color型の1次元配列
 - グローバル変数
 - 画面座標(x, y)の画素の色 pixels[y * width + x]
- loadPixels()
 - ピクセル処理の準備
 - 画面の画素ごとの色データを pixels[]に読み込む
- updatePixels()
 - ピクセル処理の終了
 - pixels[]を画面に反映する

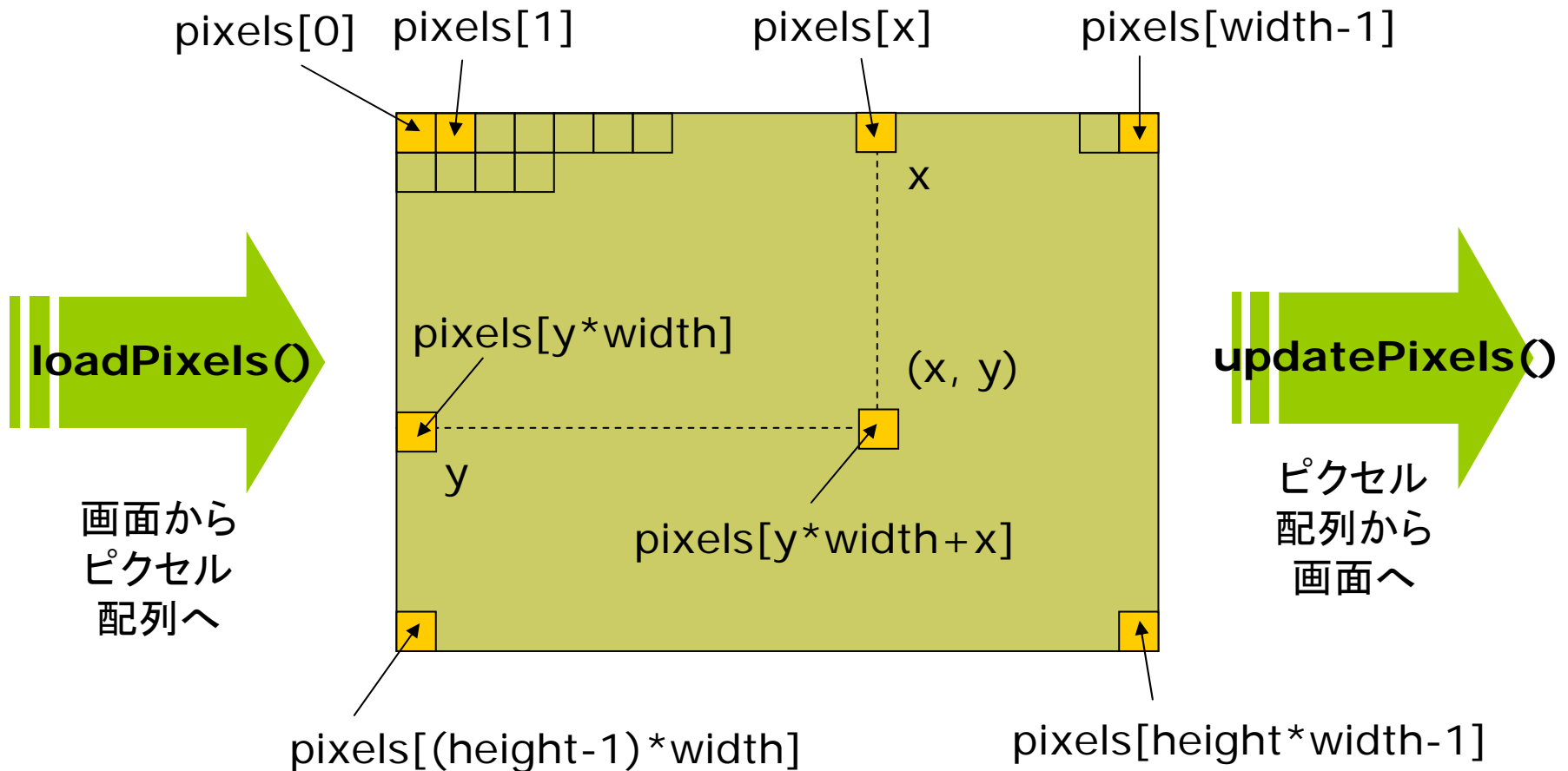
ピクセル配列の操作

- ピクセルの読み出し
 - color c;
 - c = pixels[y * width + x];
- ピクセルの書き込み
 - pixels[y * width + x] = c;

画面領域の一括操作

- copy(x1, y1, w1, h1, x2, y2, w2, h2)
- blend(x1, y1, w1, h1, x2, y2, w2, h2, 混色演算)
 - 画面領域を別の場所にコピー
- get(), get(x, y, 幅, 高さ)
 - 画面領域を画像として取得

5.4 ピクセル配列



5.5 画像

画像

- PImage型
 - 画像を表す変数型
 - 通常, グローバル変数で用意
PImage img;
 - サンプル Examples →
Basics → Image → Sprite
- loadImage("ファイル名")
 - 画像の読み込み
 - 通常, setup()の中で使う
img = loadImage("a.jpg")
 - Sketch → Add File...で, あらかじめ, 画像ファイルをデータフォルダにコピーしておくこと
 - 対応形式 .gif .jpg .png .tga

画像表示

- image(画像, x, y)
 - 画像の描画
- image(画像, x, y, 幅, 高さ)
 - サイズを変更して画像を描画
- imageMode(モード)
 - rectMode/ellipseModeと同

画像の部分表示

- copy(画像, x_{画像}, y_{画像}, w_{画像}, h_{画像}, x, y, w, h)
- blend(画像, x_{画像}, y_{画像}, w_{画像}, h_{画像}, x, y, w, h, 混色演算)
 - 画像の指定領域を描画

5.6 オブジェクト指向基礎

オブジェクト指向

□ オブジェクト

- 関連するデータをまとめて、その操作方法とパックにしたもの
- Processing ← 実体はJava
- 例) PImage img

□ オブジェクト指向用語

- 「クラス」: オブジェクトの型
 - 例) PImage
- 「インスタンス」: オブジェクト変数
 - 例) img
- 「フィールド」: オブジェクトの属性
 - 例) img.height
- 「メソッド」: オブジェクトの操作
 - 例) img.save()

PImage型

□ フィールド

- 自分の属性を表す変数
- `img.width`, `img.height`
- `img.pixels[]`

例) `img.pixels[y*img.width+x]`

□ メソッド

- 自分を操作するための関数
- `img.save("ファイル名")`
 - 画像にファイル名をつけて保存
- `img.get(x, y, 幅, 高さ)`
 - 画像の一部を画像として取り出す
- `img.copy(マニュアル参照)`
 - 画像内での領域コピー
または他の画像の埋め込み

5.7 演習課題

準備

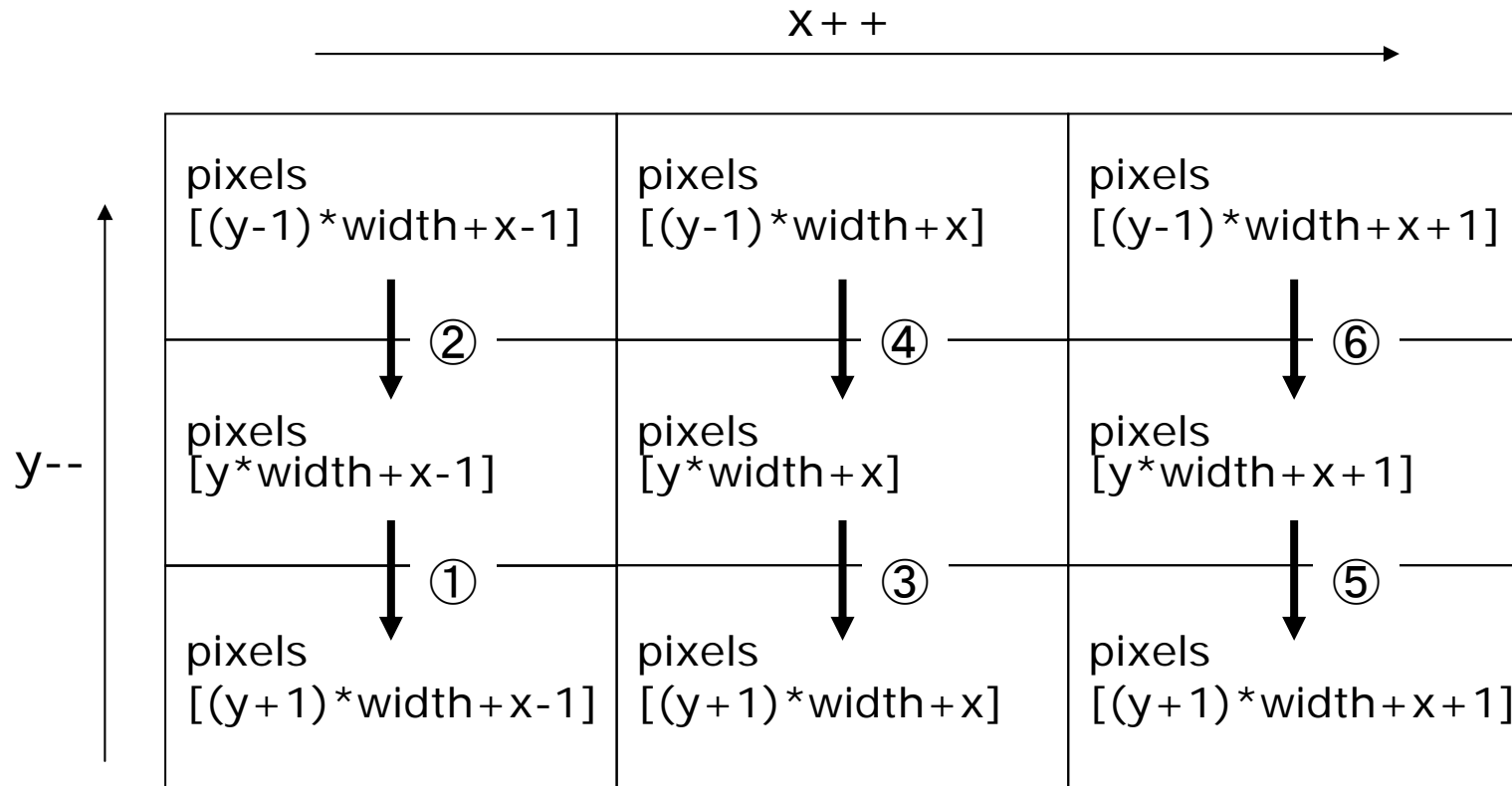
- 右のプログラムに適切な setup 関数を補って正しく動作するプログラムにしてください
- プログラムを実行したら、ウィンドウ上で、マウスのボタンを押してドラッグしてみてください
- さらに、3行目～10行目のforループの内側に
if (random(1.0) < 0.2) { ...
という条件判定を入れてください

課題

- 上から下に流れる模様を、右から左に流れるように変更してください
- 右→左以外のものは認めない

```
void draw() {  
  loadPixels();  
  for (int x = 0; x < width; x++) {  
    for (int y = height-1; y > 0; y--) {  
      pixels[y*width + x] =  
        pixels[(y-1)*width + x];  
    }  
    pixels[x] = color(0, 0,  
      frameCount % 256);  
  }  
  updatePixels();  
  if (mousePressed) {  
    noStroke();  
    fill(255, 220, 220, 200);  
    ellipse(mouseX, mouseY, 20, 20);  
  }  
}
```


【ヒント】座標 (x,y) 周辺のピクセル配列



pixels[y*width + x] = pixels[(y-1)*width + x];
 の意味と処理の順序をよく考えてください