

Graphics with Processing



2010-04 色彩とピクセル処理

<http://vilab.org>

塩澤秀和

4.1 色彩

色の表現

- 色の指定方法
 - 1つの数値(グレースケール)
 - 3つの数値の組(カラー)
初期モードは RGB & 0~255
 - color 型の変数
- color(成分1, 成分2, 成分3)
 - 色データの生成
 - color 型の変数に代入できる
 - 例) `color c = color(r, g, b);`
- アルファ値(p.225)
 - 色の第4成分(透過処理用)
 - 重ね塗りでの濃さを表す
 - 例) `c = color(r, g, b, a);`
 - 例) `fill(255, 0, 0, 128);`

色のモード

- colorMode(モード, 値範囲)
 - 色指定モードの変更
 - モード: カラーモデル
RGB または HSB
 - 値範囲: 成分の上限値
 - colorMode(モード, 範囲1, 範囲2, 範囲3) の形式もある
 - 例) `colorMode(HSB, 1.0);`
 - サンプル Basics → Color

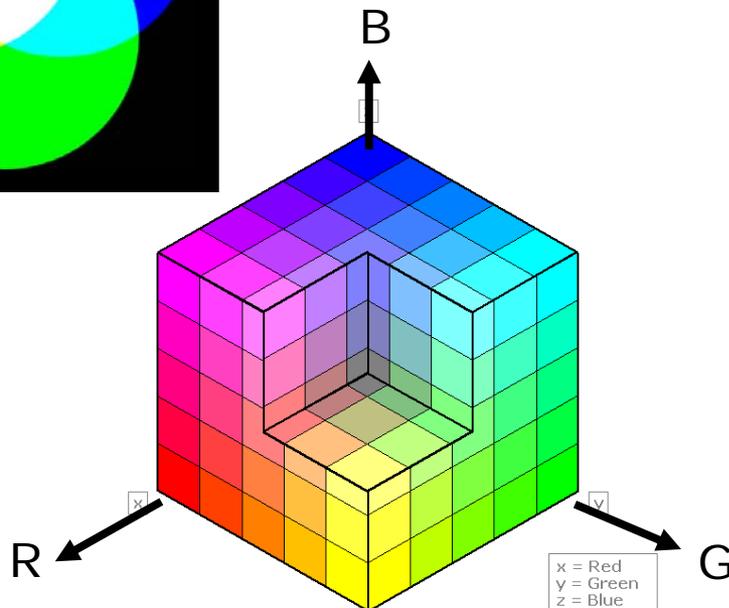
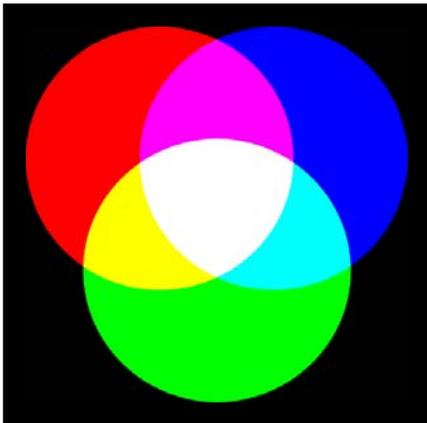
色の成分の取得

- `red(c)`, `green(c)`, `blue(c)`, `hue(c)`, `saturation(c)`, `brightness(c)`, `alpha(c)`
 - 色から各成分を取り出す

4.2 表色系/カラーモデル (p.201)

RGBカラーモデル

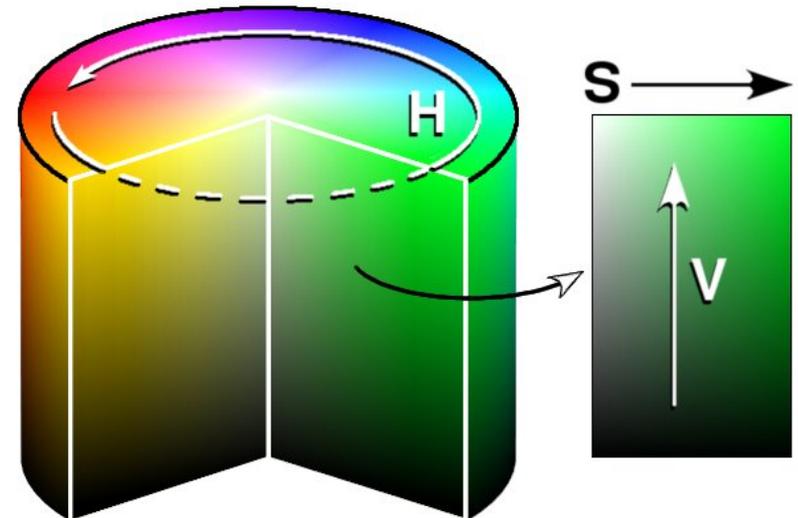
□ 光の三原色(赤, 緑, 青)



HSB (HSV/HSI) カラーモデル

□ 光の三属性

- 色相(H): 色あい
- 彩度(S): あざやかさ
- 明度(B/V/I): 明るさ
- メニュー Tools → Color Selector



4.3 ピクセル処理

ピクセル配列

- ピクセルとは(p.11)
 - 画面を構成する画素1点1点
(pixel ← picture cell)
- pixels[]
 - 各画素の色(color型のデータ)を格納する1次元配列
 - 画面座標(x, y)の要素は
pixels[y * width + x]
- loadPixels()
 - ピクセル処理の開始処理
 - 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処理
 - pixels[]を画面に反映する

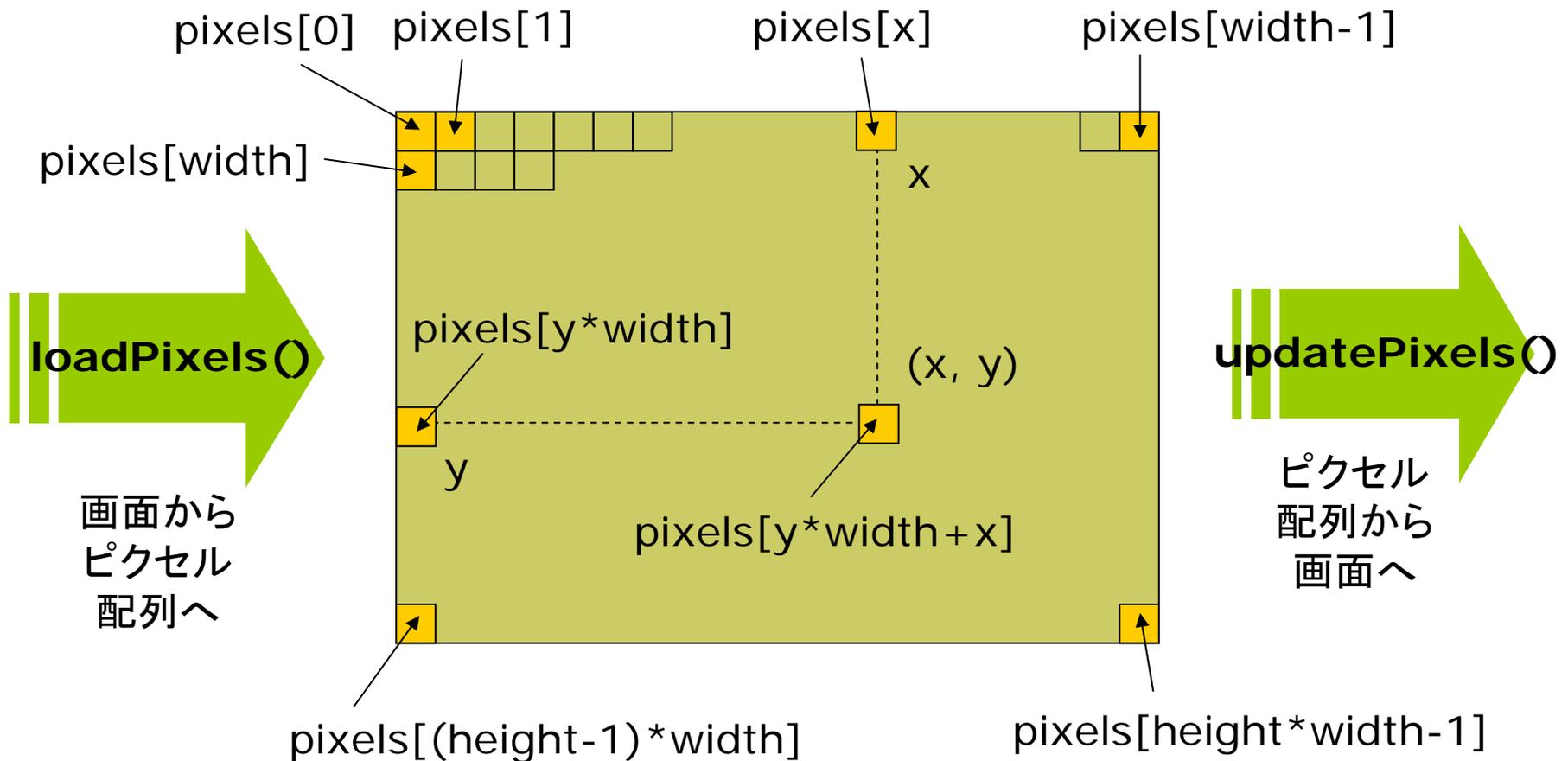
ピクセル配列の操作

- 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

4.4 ピクセル配列



4.5 画像データ

画像データ

- 画像の利用
 - サンプル Basics → Image → LoadDisplayImage など
 - 対応形式: jpg gif png tga
- PImage型
 - 画像を扱うには, PImage型のグローバル変数を用意しておく
PImage img;
- loadImage("ファイル名")
 - 画像データの読み込み
 - 通常, setup()で1回だけ行う
img = loadImage("a.jpg")
 - ファイルは, 事前にメニューの Sketch → Add File...でデータフォルダにコピーしておく

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と同様

画像の部分表示

- copy(画像, $x_{\text{画像}}$, $y_{\text{画像}}$, $w_{\text{画像}}$, $h_{\text{画像}}$, x, y, w, h)
 - 画像の指定領域だけを描画
- blend(画像, $x_{\text{画像}}$, $y_{\text{画像}}$, $w_{\text{画像}}$, $h_{\text{画像}}$, x, y, w, h, 混色演算)
 - 指定した方法で画像を重ね塗り

4.6 オブジェクト指向基礎

オブジェクト指向

- オブジェクトとは
 - データとその操作をセットにして、使いやすくしたもの
 - 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型の例

- フィールド
 - img.width, img.height
 - 画像のサイズ(横・縦の幅)
 - img.pixels[]
 - 画像データのピクセル配列
- メソッド(一部)
 - 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.resize(幅, 高さ)
 - 画像のサイズを変更する
 - img.loadPixels(), img.updatePixels()
 - ピクセル処理のためのメソッド

4.7 演習課題

準備課題

- 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きなさい
 - 実行したら、ウィンドウ上でマウスをドラッグしてみなさい
 - さらに、★の範囲を
if (random(1.0) < 0.2)
というif文で囲んでみなさい

提出課題

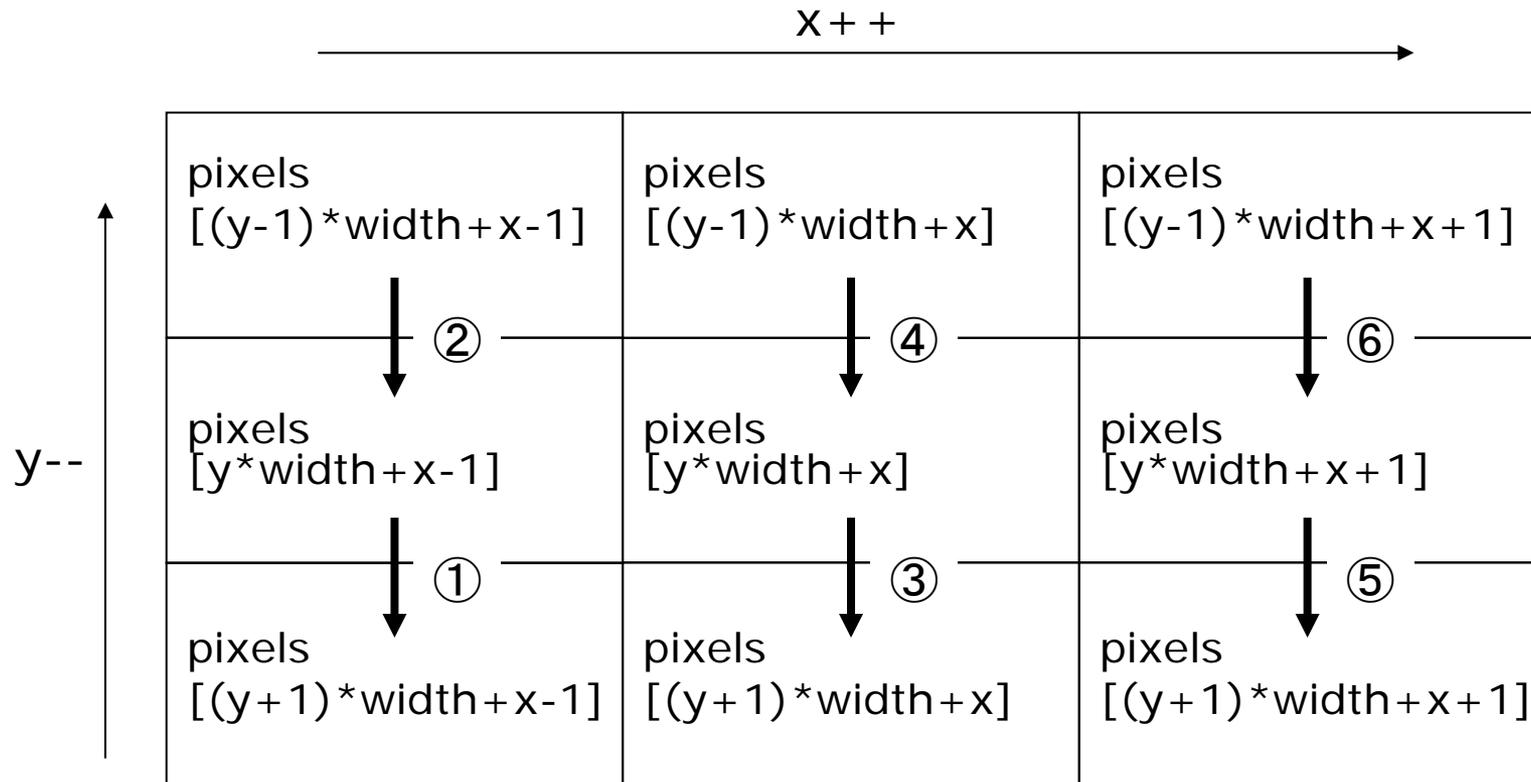
- 上から下に流れる模様を、右から左に流れるように変更きなさい
 - **右→左以外のものは認めない**
 - 簡単にできた人は、円以外の図形や画像を表示させてみなさい

```
void draw() {
  color c;

  loadPixels();
  for (int x = 0; x < width; x++) {
    // ★
    for (int y = height-1; y > 0; y--) {
      c = pixels[(y - 1) * width + x];
      pixels[y * width + x] = c;
    }
    pixels[x] =
      color(0, 0, frameCount % 256);
    // ★
  }
  updatePixels();

  if (mousePressed) {
    noStroke();
    fill(255, 220, 220, 200);
    ellipse(mouseX, mouseY, 20, 20);
  }
}
```

4.8 ヒント: 座標 (x,y) 周辺のピクセル配列



```
c = pixels[(y-1)*width + x];
pixels[y*width + x] = c;
```

の意味と処理の順序をよく考えてください