画素へのアクセス

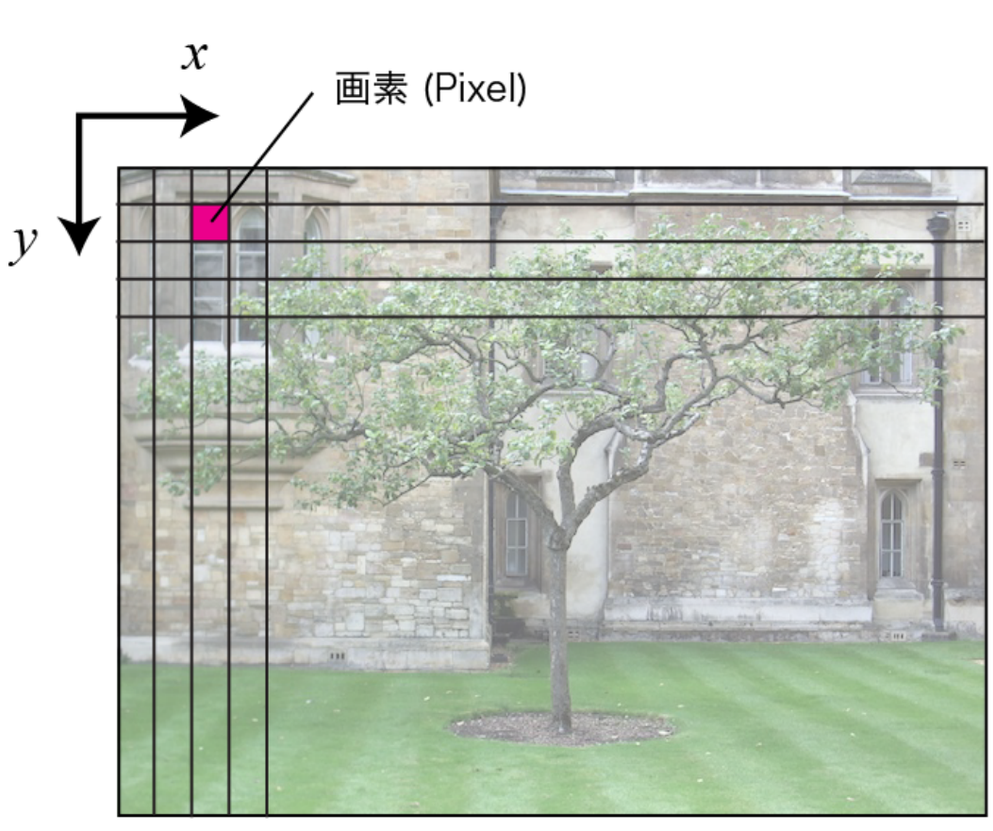
予定

- 画素値
  - 色値・輝度値
- 画像の走査 (スキャン)

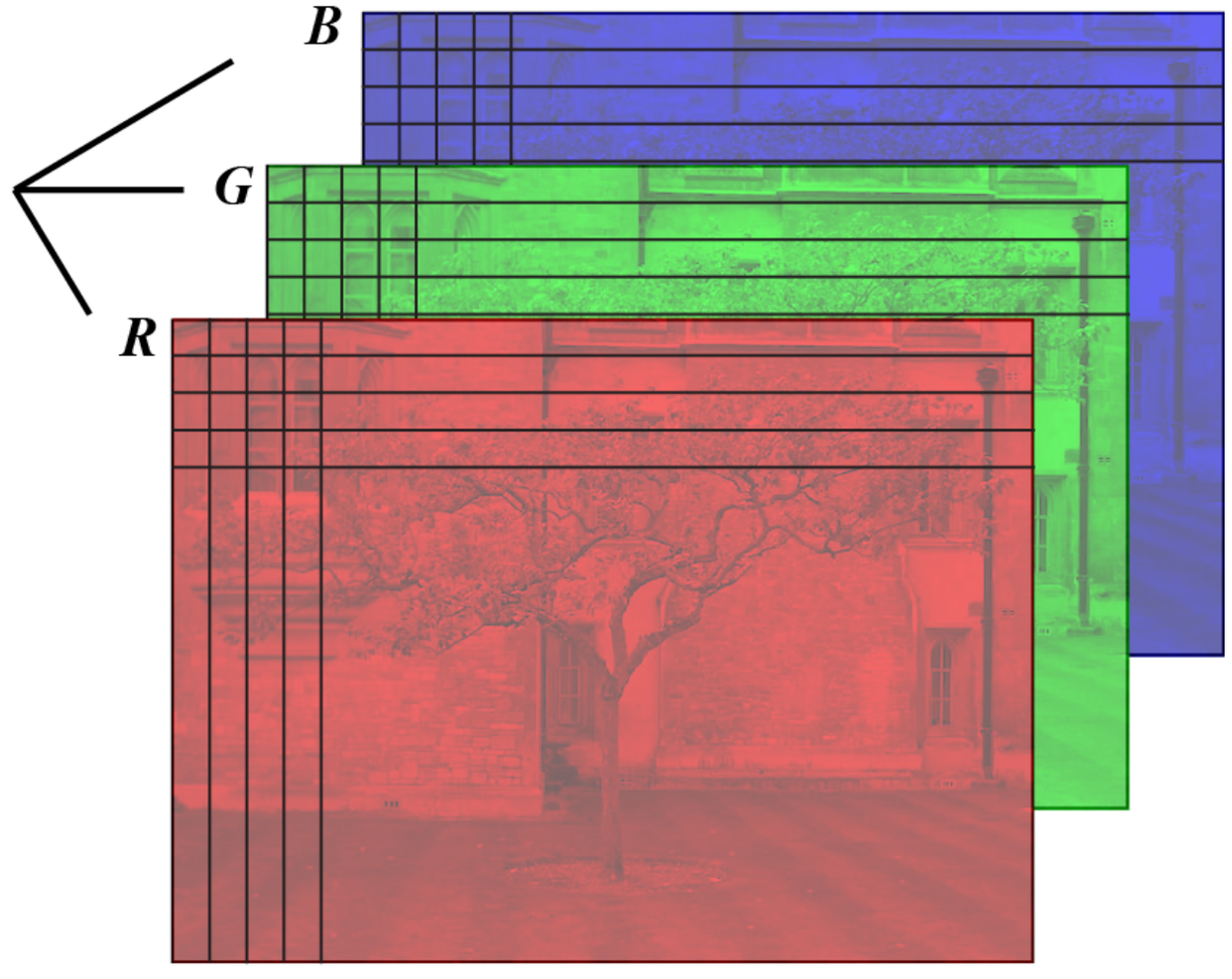
カラー画像とチャンネル

復習

- 1画素は3要素 (チャンネル) で構成
- OpenCVでは**B**, **G**, **R**, **B**, **G**, **R**, …の順でデータ格納



カラー画像



RGB 3チャンネル

画素値 (色値・輝度値)について

- **カラー画像 (RGB) の色値**
  - 例: (R, G, B) = (255, 0, 0)
  - 例: (R, G, B) = (255, 0, 255)
- **モノクロ画像 (単色画像)の輝度値**
  - 0: 黒色
  - 255: 白色

$$1\text{バイト}=2^8$$
$$=256$$

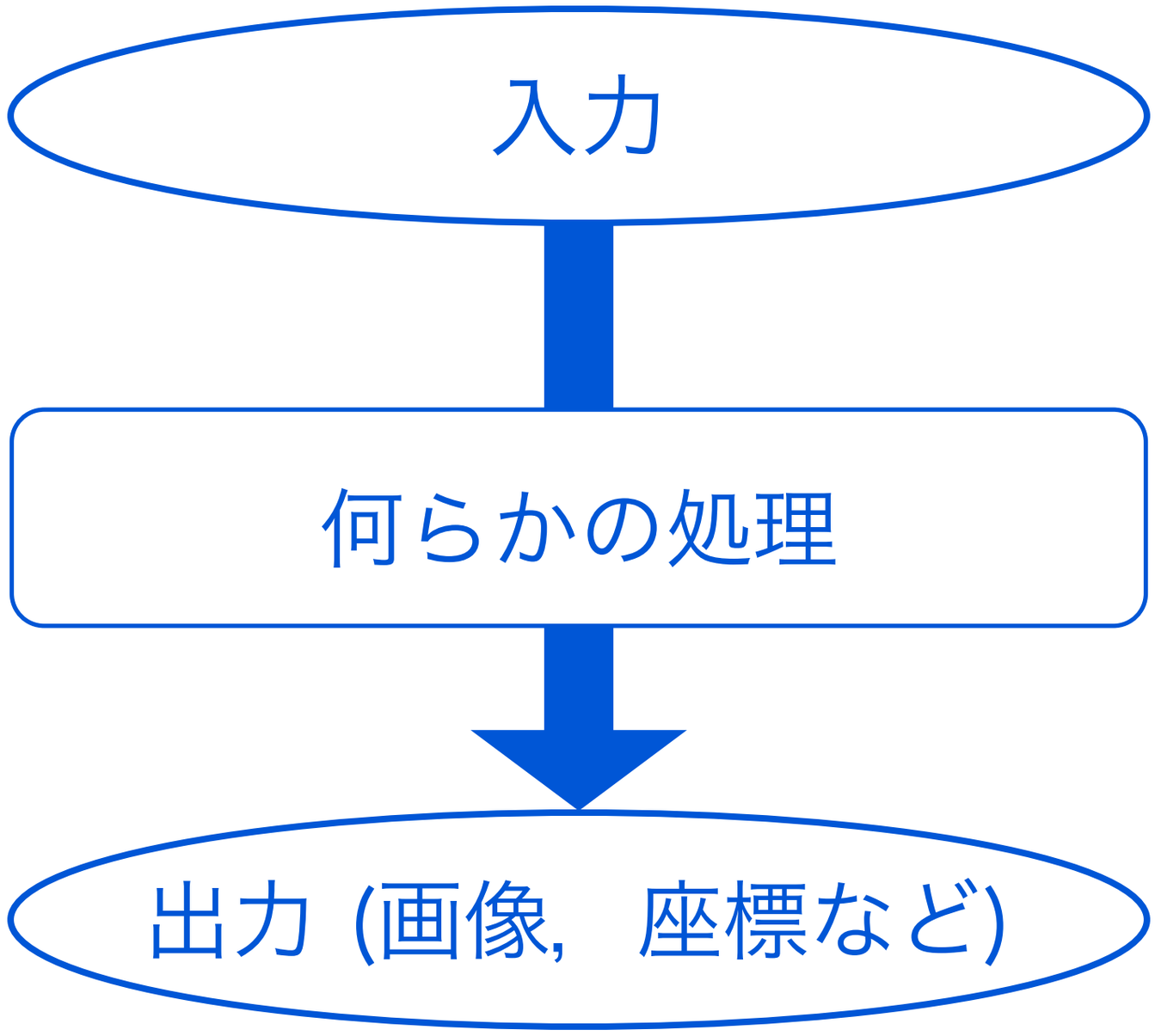
画素値の取得

- プロジェクト名: pixelAccess
- 実施内容
  - scanfで座標の決定
  - 指定された座標の画素値の取得
  - 指定された座標が負であれば終了

```
width=600, height=800
200 200
200, 200: (153, 141, 143)
100 100
100, 100: (77, 90, 145)
-1 -1
終了です
```

オレンジ文字: 入力

画像処理の基本



```
#include <stdio.h>
#include <opencv2/opencv.hpp> //OpenCV

//画像ファイル（サイズは小さめが良い）
#define FILE_NAME "../Debug/aquarium.jpg"
#define WINDOW_NAME "output"


int main (int argc, const char *argv[]) {
    //画像の入力
    cv::Mat src_img; //画像の型と変数
    src_img = cv::imread(FILE_NAME); //画像の読み込み
    if (src_img.empty()) { //入力失敗の場合
        fprintf(stderr, "読み込み失敗\n");
        return (-1);
    }
    cv::imshow(WINDOW_NAME, src_img); //画像の表示
    cv::waitKey(30); //キー入力待ち
```

```
    fprintf(stderr, "width=%d, height=%d\n",
            src_img.cols, src_img.rows);
    int x, y;
    while (1) {
        scanf("%d %d", &x, &y);
        if (x<0 || y<0) { //負の値の場合
            fprintf(stderr, "終了です\n");
            break; //終了
        }
        if (src_img.cols <= x ||
            src_img.rows <= y) { //大きさを超えた場合
            fprintf(stderr, "画面外です\n");
            continue;
        }
        cv::Vec3b val = src_img.at<cv::Vec3b>(y, x);
        printf("%d, %d: (%d, %d, %d)\n", x, y,
                val[2], val[1], val[0]);
    }
    return (0);
}
```

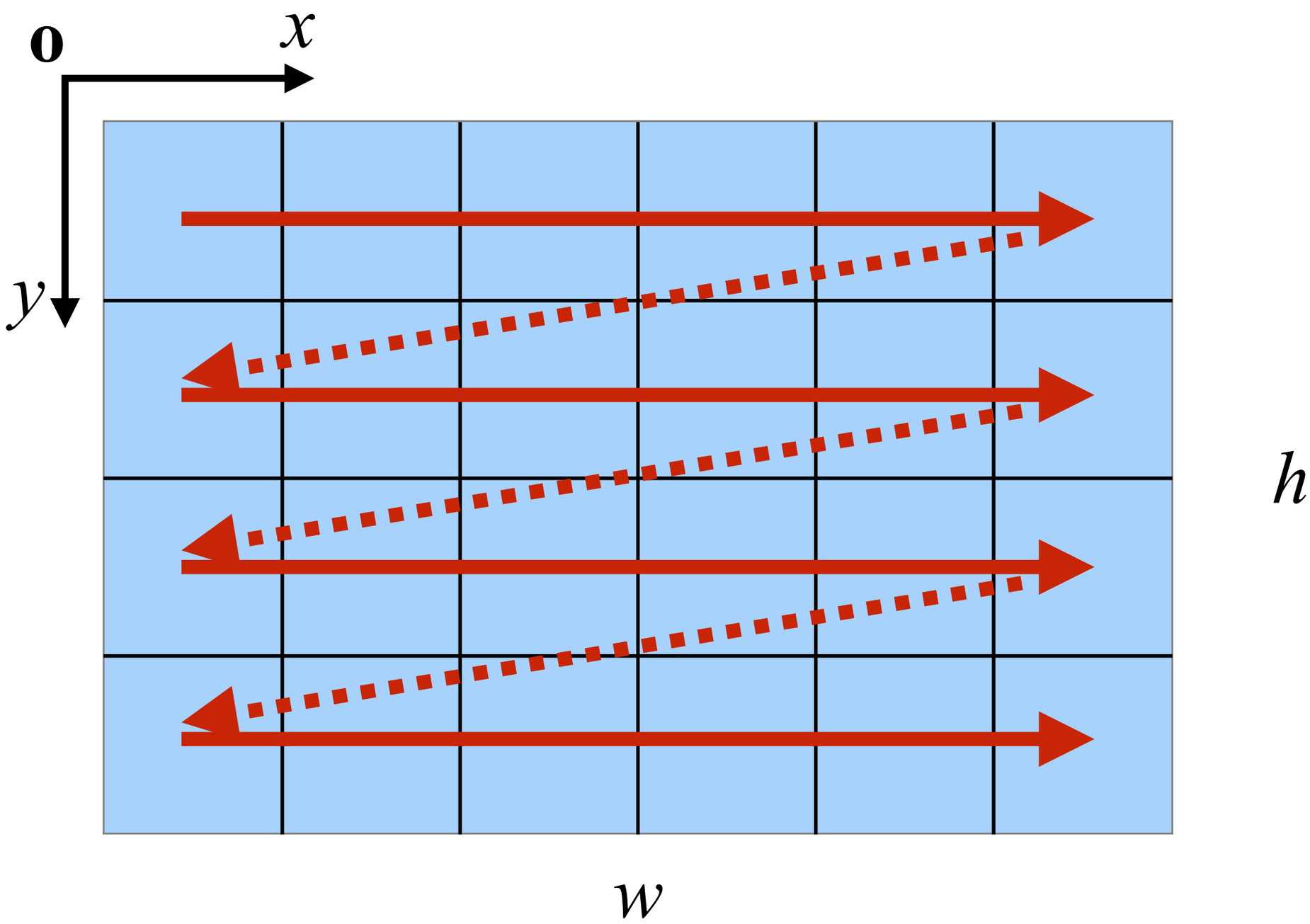
色値の取得方法

```
cv::Vec3b val = src_img.at<cv::Vec3b>(y, x);
printf("%d, %d: (%d, %d, %d)\n",
        x, y, val[2], val[1], val[0]);
```

- cv::Vec3b型
  - unsigned char (uchar) 型の3要素を持つベクトル
    - ➡ unsigned char: 1バイト (256段階)
  - 以下の様に格納される
    - ➡ R: val[2]
    - ➡ G: val[1]
    - ➡ B: val[0]

画像の走査方法

- ディジタル画像の画素にアクセスし，なんらかの処理を施すこと



画像の走査のプログラム

- プロジェクト名: pixelScan
- 内容

1.変数宣言 (入力画像, 出力画像など)

2.画像の走査

- 入力画像の画素値の取得
- 出力画像の画素値にコピー

3.画像の表示 (入力画像, 出力画像)

4.キー入力待ち

```
#include <iostream>
#include <opencv2/opencv.hpp>

//画像ファイル（小さめが良い）
#define FILE_NAME “../Debug/aquarium.jpg”
#define WINDOW_NAME_INPUT "input"
#define WINDOW_NAME_OUTPUT "output"

int main(int argc, const char * argv[]) {
    int x, y;

    //画像の入力
    cv::Mat src_img = cv::imread(FILE_NAME);
    if (src_img.empty()) { //入力失敗の場合
        return (-1);
    }

    //出力画像のメモリ確保
    cv::Mat dst_img = cv::Mat(src_img.size(), CV_8UC3);
```

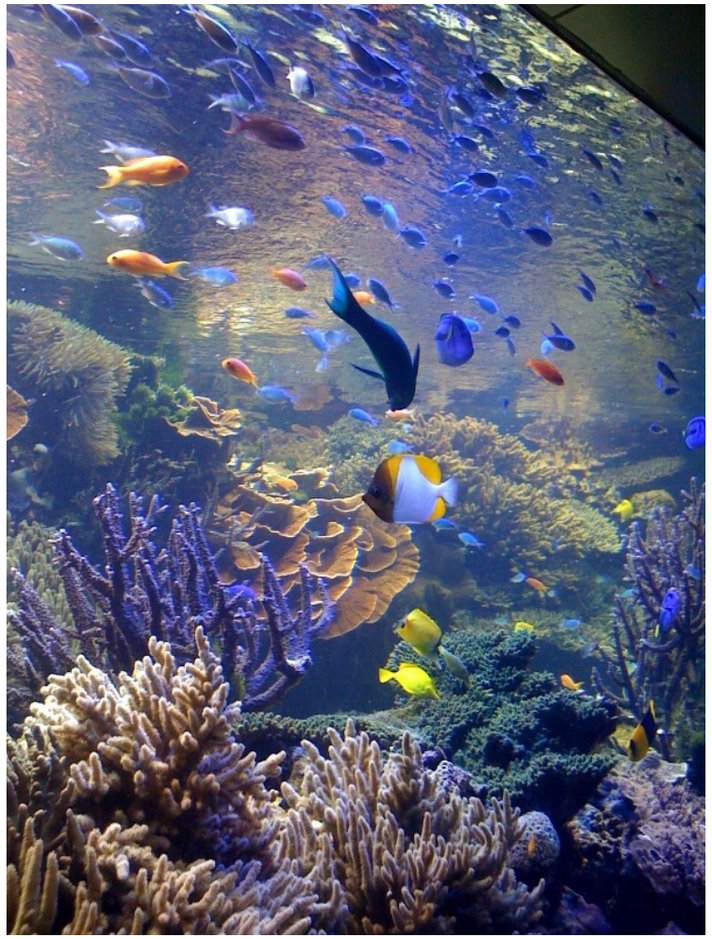
```
//画像の走査
for (y=0; y<src_img.rows; y++) {//縦方向
    for (x=0; x<src_img.cols; x++) {//横方向
        //画素値の取得
        cv::Vec3b s = src_img.at<cv::Vec3b>(y, x);
        s[0] = s[0]; //B
        s[1] = s[1]; //G
        s[2] = s[2]; //R
        dst_img.at<cv::Vec3b>(y, x) = s;

    }
}
cv::imshow(WINDOW_NAME_INPUT, src_img);//画像の表示
cv::imshow(WINDOW_NAME_OUTPUT, dst_img);//画像の表示
cv::waitKey(); //キー入力待ち（止める）
return 0;
}
```

画素値の変更

- 画素値を30だけ赤くする

```
s[2] = s[2] + 30; //R (赤)
```



画像に黒枠をつける

- プロジェクト名: imgWaku
- 内容
  - プロジェクト pixelScanの**内容をコピー**

```
if (x==0 || x==src_img.cols-1 ||
    y==0 || y==src_img.rows-1) {
       s[0] = 0; //B
       s[1] = 0; //G
       s[2] = 0; //R
}else{
       s[0] = s[0]; //B
       s[1] = s[1]; //G
       s[2] = s[2]; //R
}
```

練習

- imgWakuで黒枠を太くしよう
- 外枠10画素の太枠を描け

回答

```
if (x<10 || src_img.cols-10 <= x ||
    y<10 || src_img.rows-10 <= y) {
    s[0] = 0; //B
    s[1] = 0; //G
    s[2] = 0; //R
}else{
    s[0] = s[0]; //B
    s[1] = s[1]; //G
    s[2] = s[2]; //R
}
```