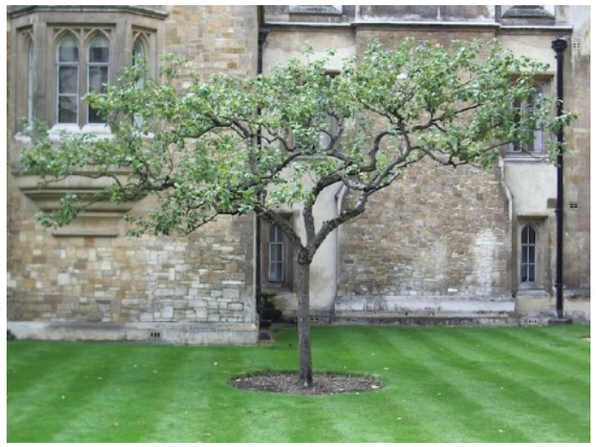
グレースケール変換プログラム

1. プロジェクト名: grayImg
2. src/imgProc/第3回/grayImgフォルダの下にgrayImg.cppを作成
3. 同フォルダ内にCMakeLists.txtを作成
   - その後　cmake ./ -G Xcode
4. moodleから画像をダウンロード
   - apple_tree.jpg
   - 同フォルダ内のDebugフォルダに格納
     （一度 Xcode で Run/Build 実行後）

プログラムの流れ

- 内容

1. 変数の宣言
2. カラー画像の読み込み
3. **グレースケール変換**
4. 表示 (カラー，**グレー**)
5. キー入力待ち



grayImg.cpp （1/2)

```
#include <iostream>
#include <opencv2/opencv.hpp>
#define FILE_NAME “../Debug/apple_tree.jpg”

//ウィンドウ名
#define WINDOW_NAME_INPUT "input"
#define WINDOW_NAME_OUTPUT "output"
```

```
int main(int argc, const char * argv[]) {
    //画像の入力
    cv::Mat src_img = cv::imread(FILE_NAME);
    if (src_img.empty()) { //入力失敗の場合
        fprintf(stderr, "Cannot read image file: %s.\n", FILE_NAME);
        return (-1);
    }

    //出力画像のメモリ確保
    cv::Mat gray_img = cv::Mat(src_img.size(), CV_8UC1);
```

grayImg.cpp (2/2)

```
    //画像の走査
    for (int y=0; y<src_img.rows; y++) {
        for (int x=0; x<src_img.cols; x++) {
            //画素値の取得
            cv::Vec3b s = src_img.at<cv::Vec3b>(y, x);
            uchar val = 0.11448*s[0] //B
            + 0.58661*s[1] //G
            + 0.29891*s[2];//R
            gray_img.at<uchar>(y, x) = val;
        }
    }
    /**
     */
    cv::imshow(WINDOW_NAME_INPUT, src_img);      //入力画像の表示
    cv::imshow(WINDOW_NAME_OUTPUT, gray_img);    //出力画像の表示
    cv::waitKey(); //キー入力待ち（止める）

    return 0;
}
```

処理結果