



# メディアプログラミング演習

第5回

# 本日はビデオを使ったアプリの作成



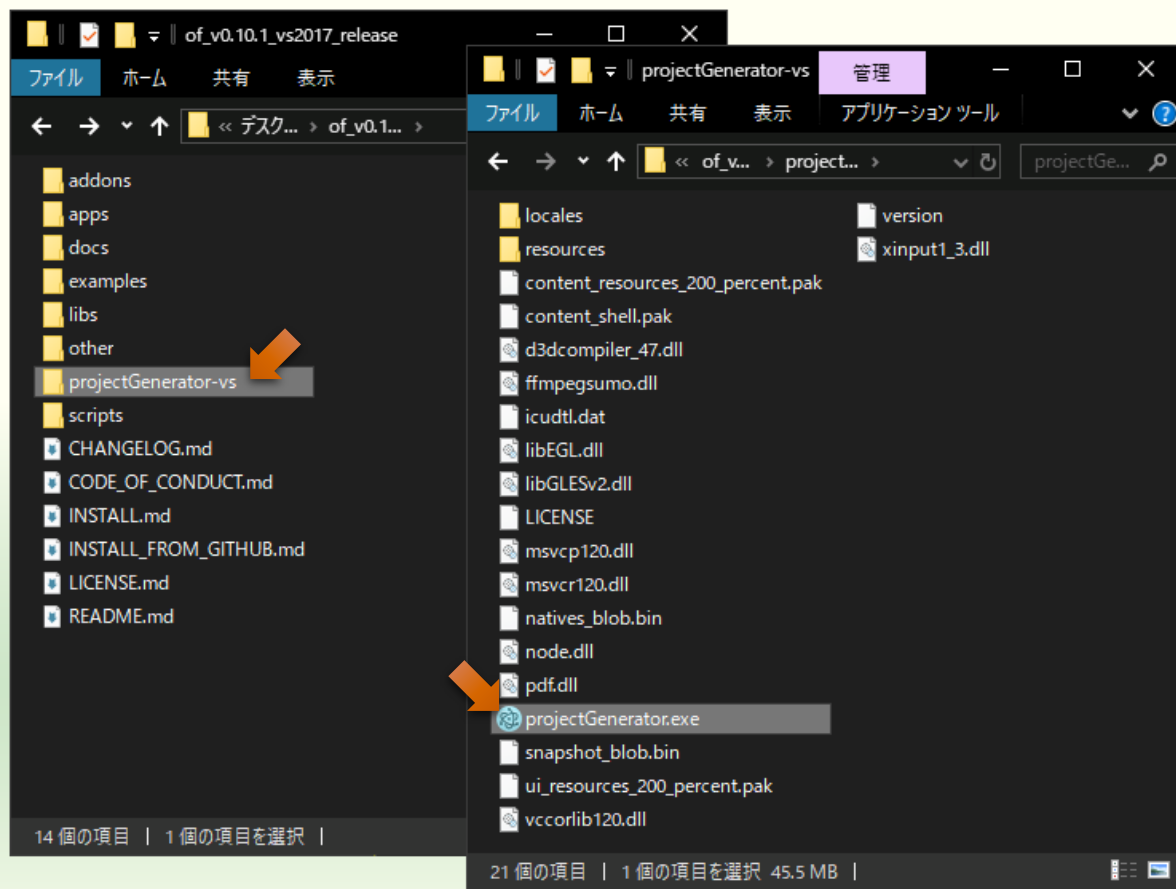


# 準備

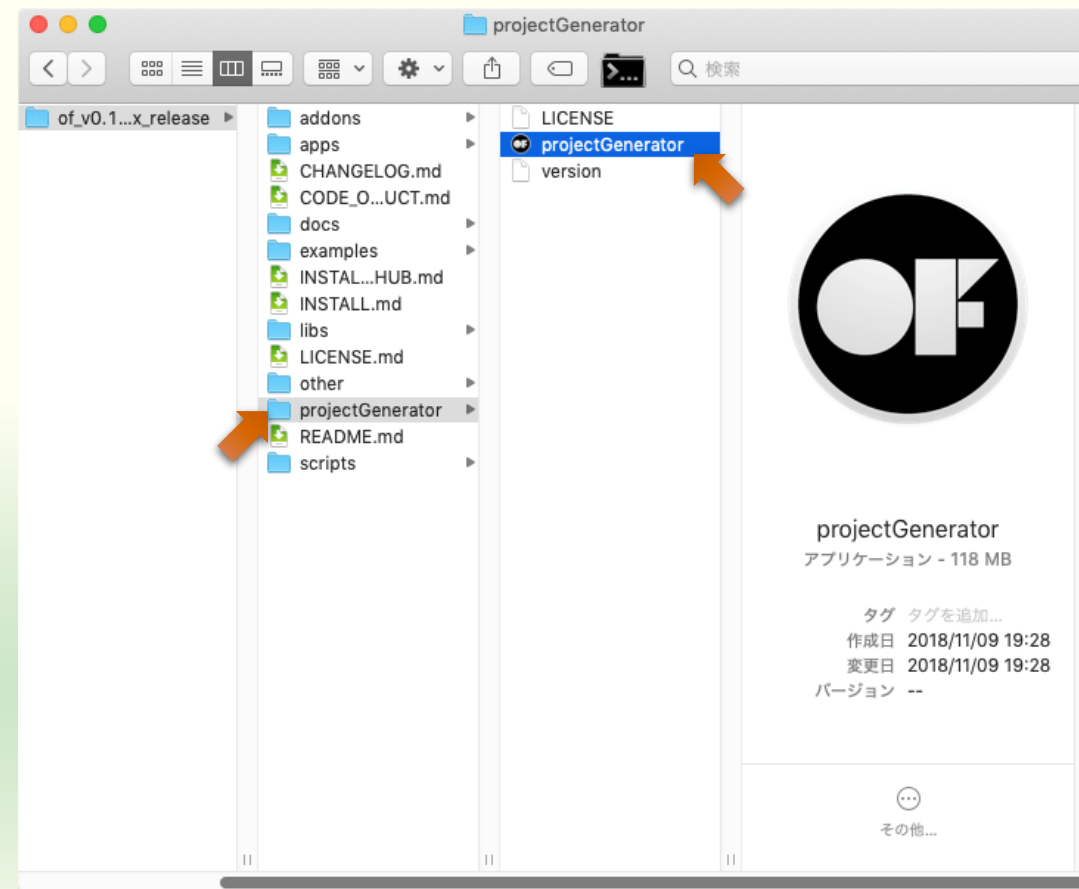
プロジェクトの作成

# projectGenerator を起動する

## windows 版のパッケージ



## macOS 版のパッケージ



# 空のプロジェクトの作成



The screenshot shows a web interface for creating a project. At the top, a dark bar contains a close button (x) and the text "create / update". Below this, the "Project name:" field contains "myGoodSketch" and an "import" button. The "Project path:" field contains "<openFrameworksの展開場所>%apps%myApps". The "Addons:" field is empty. The "Platforms:" field contains "Windows (Visual Studio 2017)". A green "Generate" button is at the bottom. Annotations in Japanese are present: a green speech bubble points to the "Project name:" field with the text "Project name はプロジェクトを作るたびに変わる (自分で設定しても可)"; an orange arrow points to the "Project path:" field with the text "そのまま"; another orange arrow points to the "Addons:" field with the text "空欄のまま"; a third orange arrow points to the "Platforms:" field with the text "そのまま"; and a fourth orange arrow points to the "Generate" button with the text "プロジェクト作成".

Project name: myGoodSketch import

Project path: <openFrameworksの展開場所>%apps%myApps

Addons: Addons...

Platforms: Windows (Visual Studio 2017) x

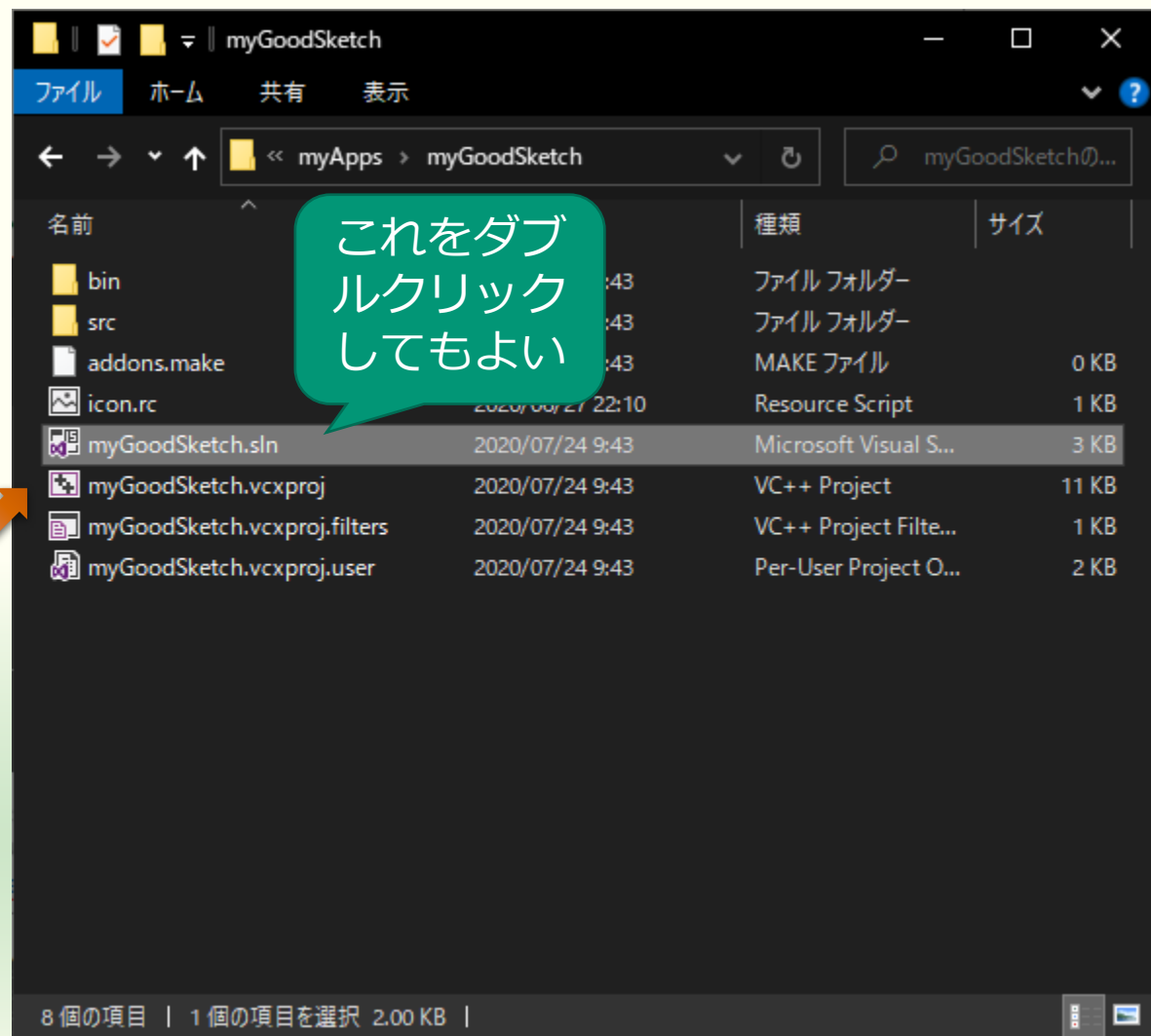
Generate

プロジェクト作成

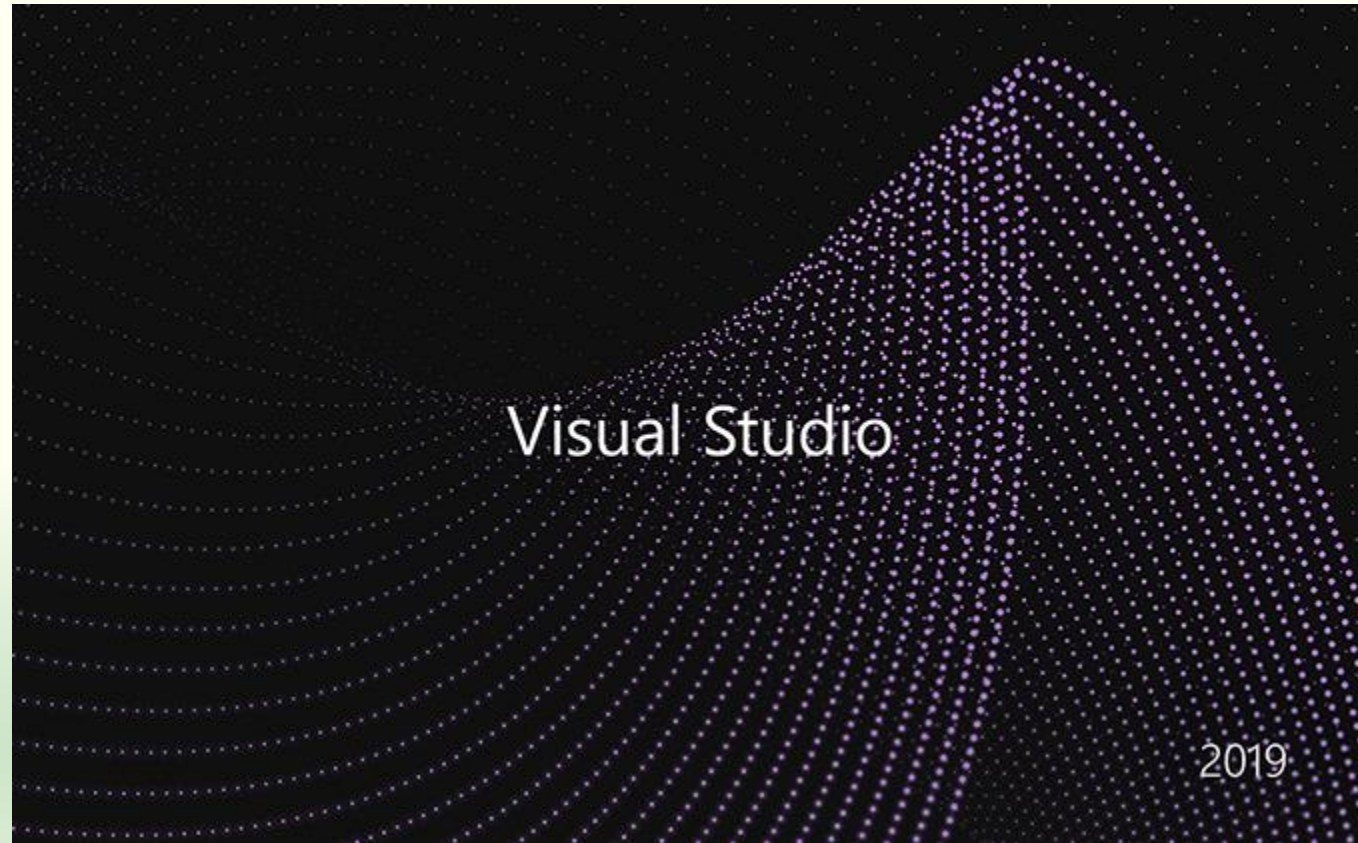
- Project name:
  - 作成するプロジェクト（プログラム）の名前
- Project path:
  - 作成するプロジェクトのファイルを置く場所
  - openFrameworks のパッケージを展開した場所の中の apps¥myApps



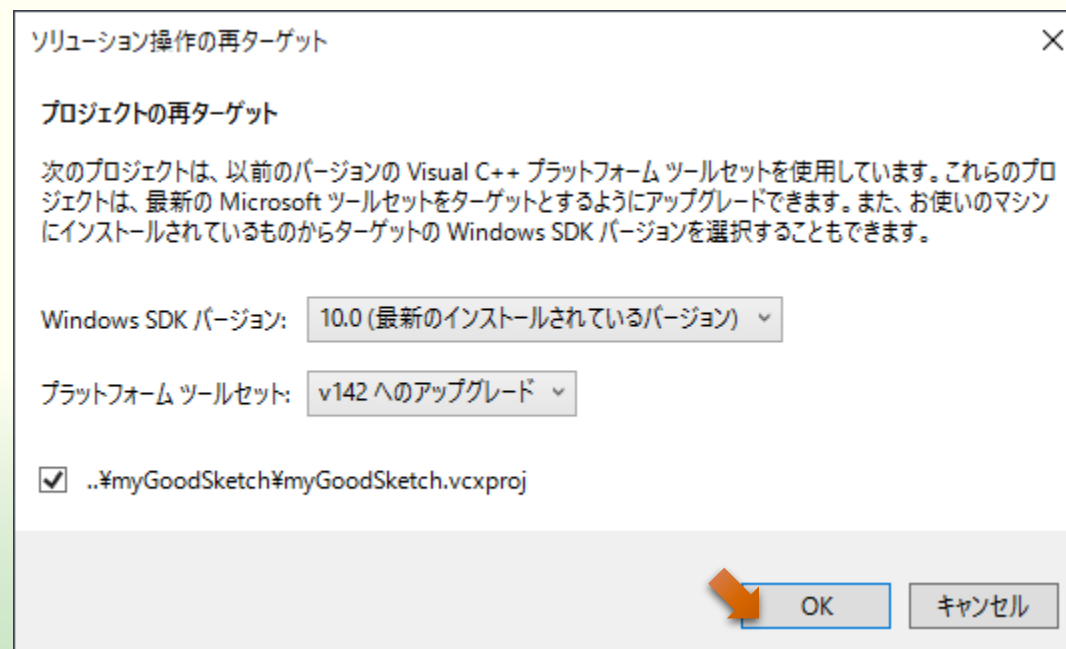
# プロジェクトの作成成功



# Visual Studio 2019 が起動する



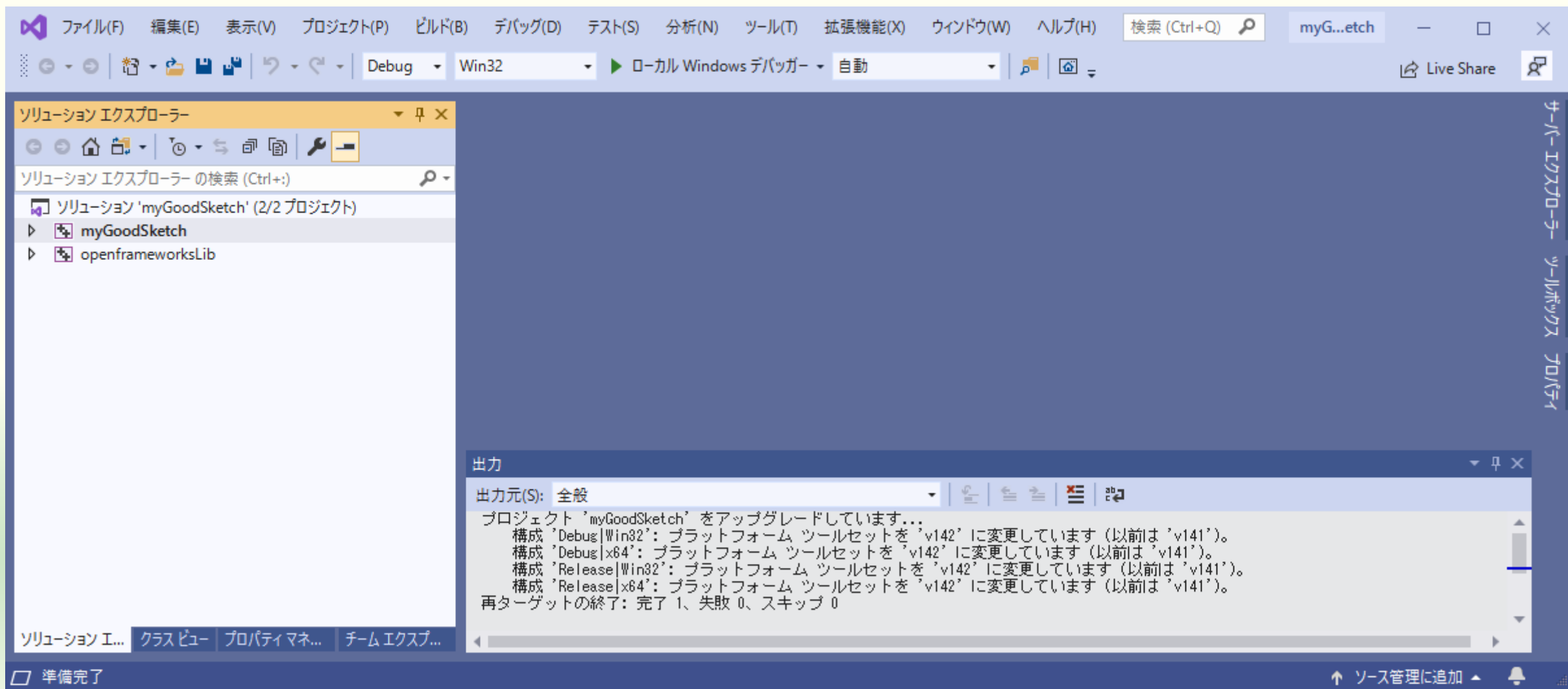
# ソリューションの再ターゲット



Visual Studio は頻繁に更新しているので皆さんがお使いの Visual Studio SDK のバージョンと合わない場合がある



# Visual Studio 起動

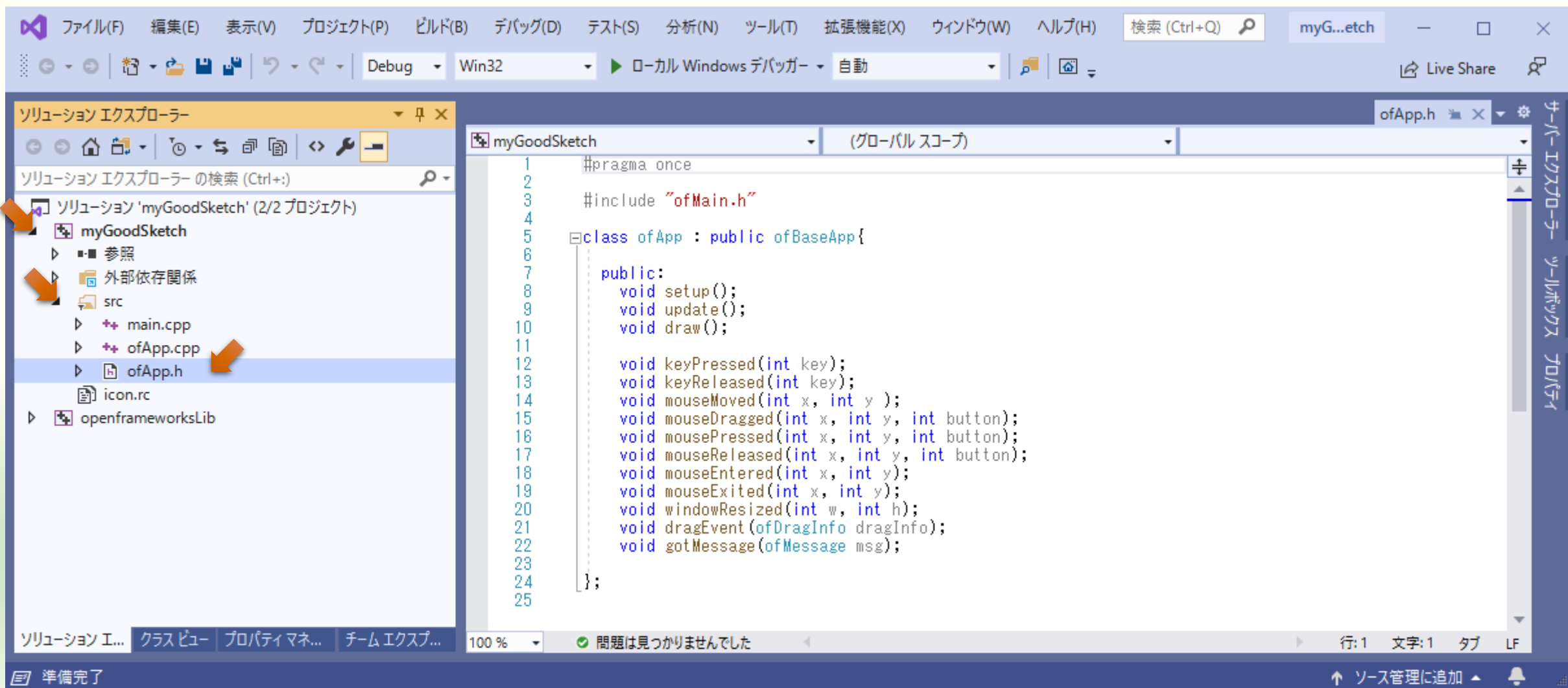




# 映像の入力

ビデオデータの取り扱い

# ofApp.h を開く



# ofApp クラスに映像入力のメンバ変数を追加する

```
#pragma once

#include "ofMain.h"

class ofApp : public ofBaseApp{
    ofVideoGrabber video;

public:
    void setup();
    void update();
    void draw();

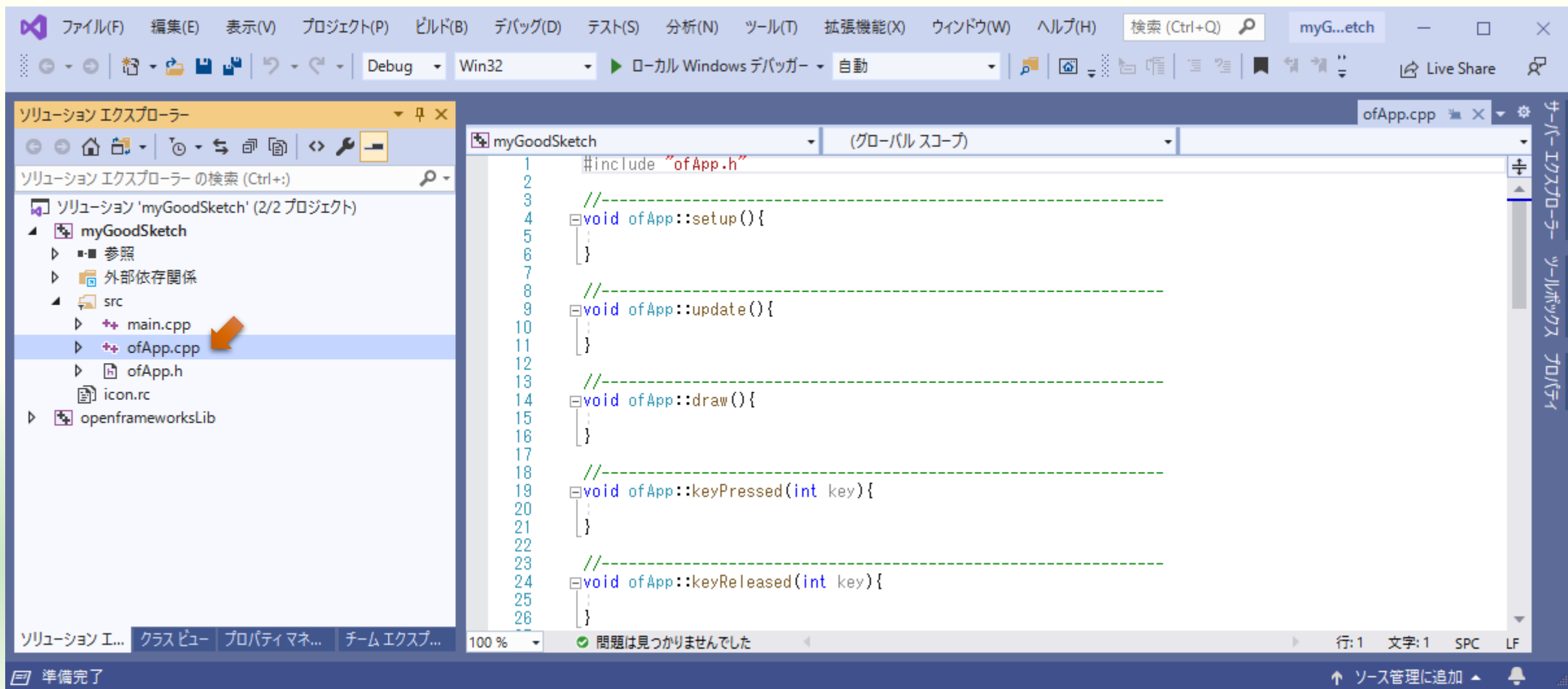
    void keyPressed(int key);
    void keyReleased(int key);
    void mouseMoved(int x, int y);
    (以下略)
```

## ■ ofVideoGrabber

- カメラからの映像入力（ビデオキャプチャ）を行うクラス



# ofApp.cpp を開く



# ofApp.cpp で入力した映像を表示する

```
#include "ofApp.h"

//-----
void ofApp::setup(){
    video.setup(320, 240);
}

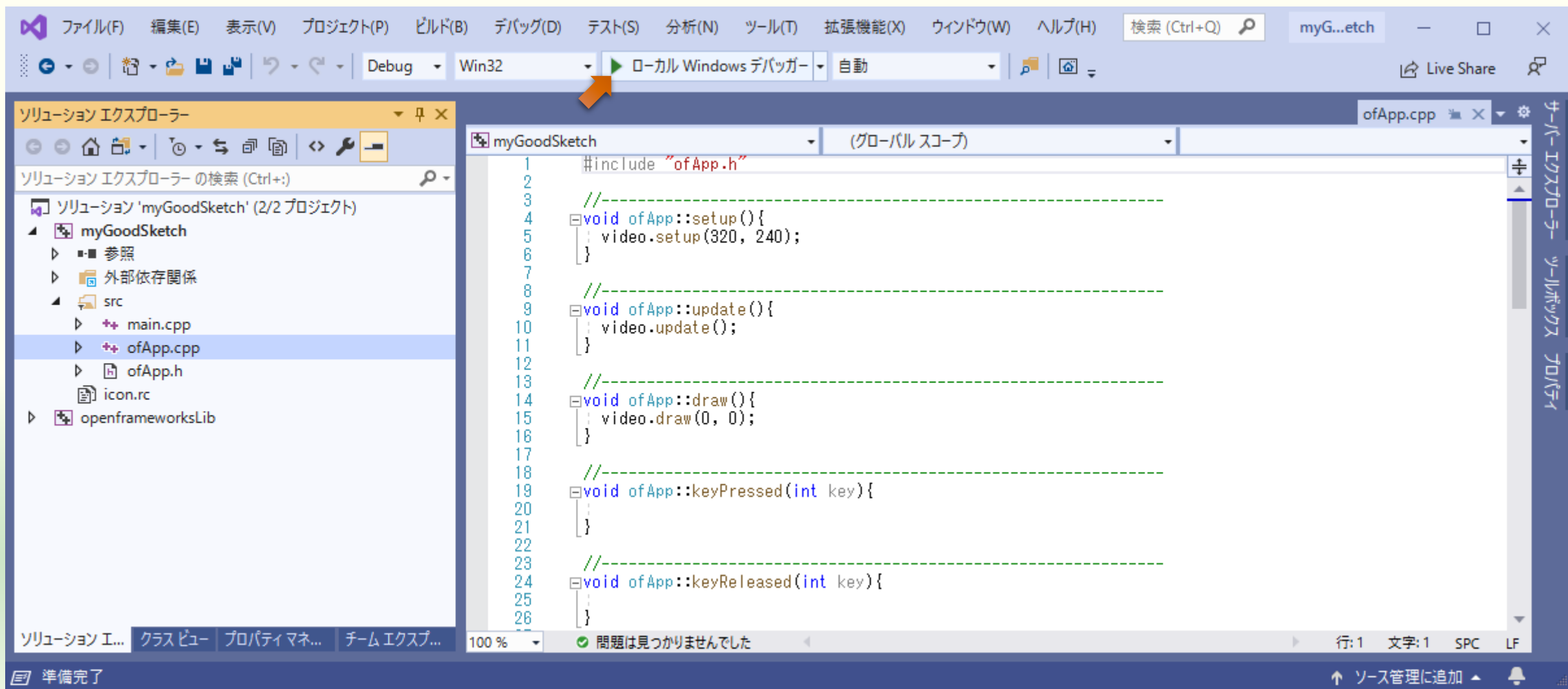
//-----
void ofApp::update(){
    video.update();
}

//-----
void ofApp::draw(){
    video.draw(0, 0);
}
```

(以下略)

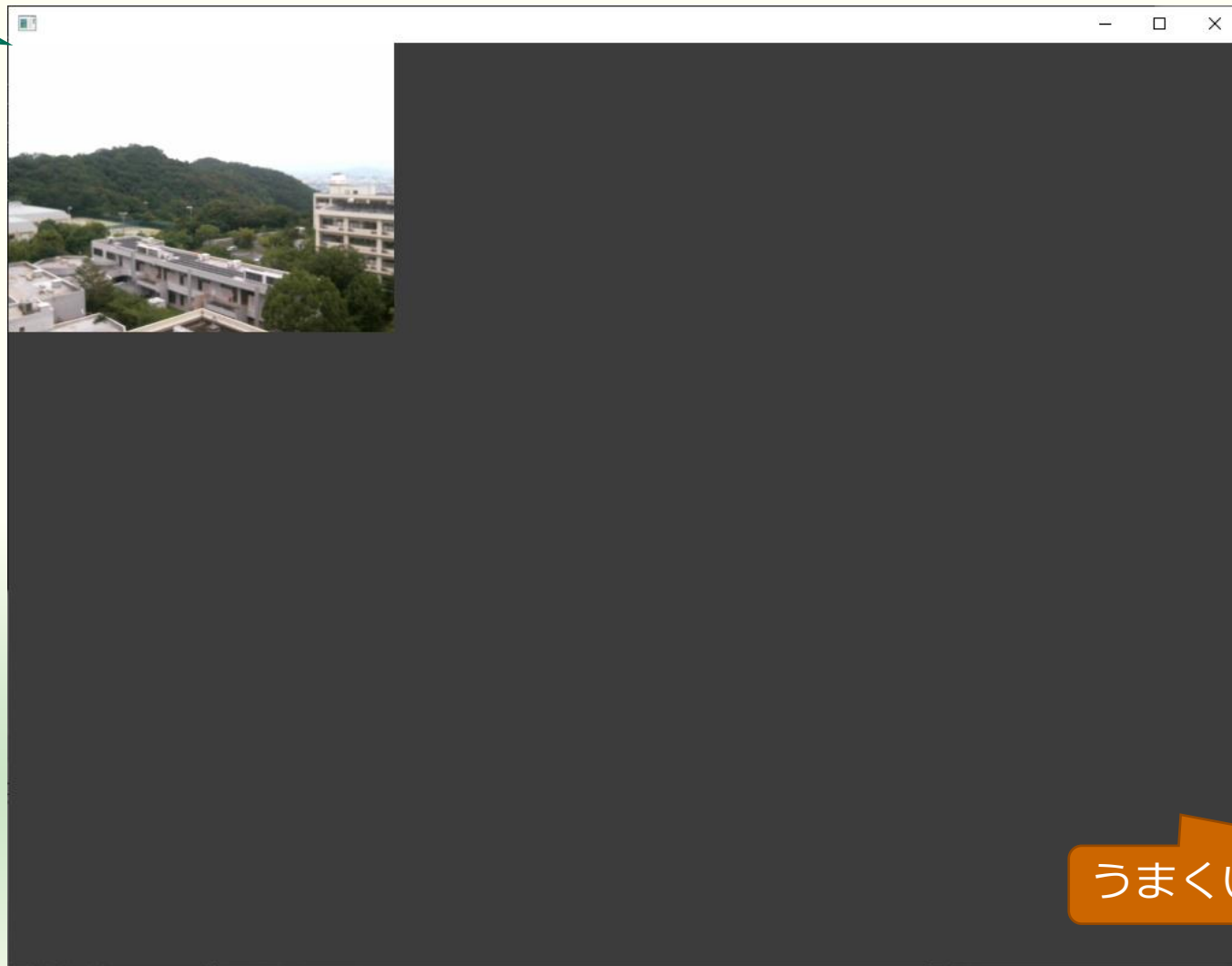
- video.setup(320, 240);
  - 映像入力の初期設定
  - 320, 240 は入力する映像の解像度
    - 640, 480 や 1280, 720 に対応するかどうかはカメラ次第なので試して
- video.update();
  - 映像入力から 1 フレーム取り込む
- video.draw(0, 0);
  - 入力した映像の 1 フレームをウィンドウの原点 (0, 0、左上隅) から描画する

# ビルドと実行



# 左上の原点を左上隅にして映像が表示される

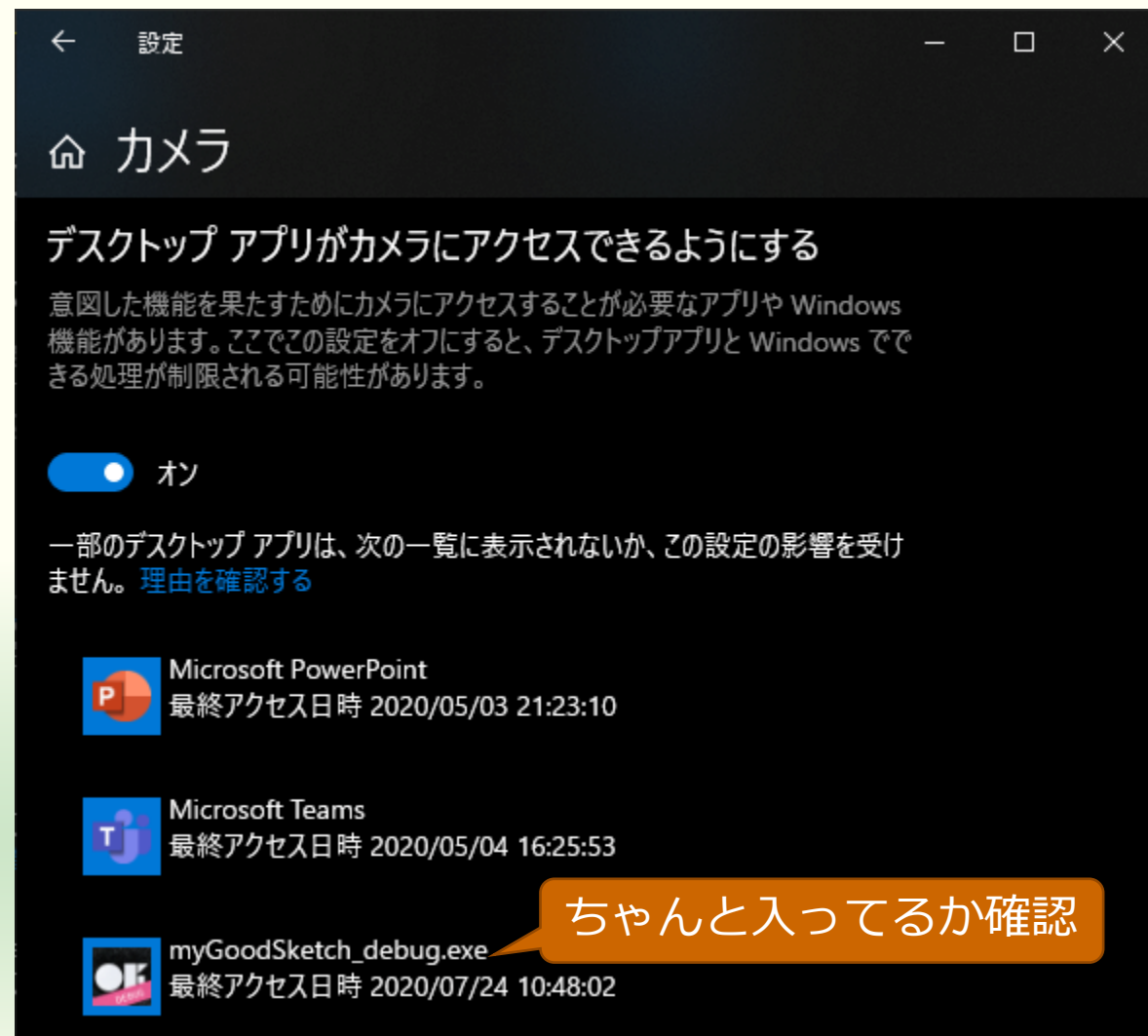
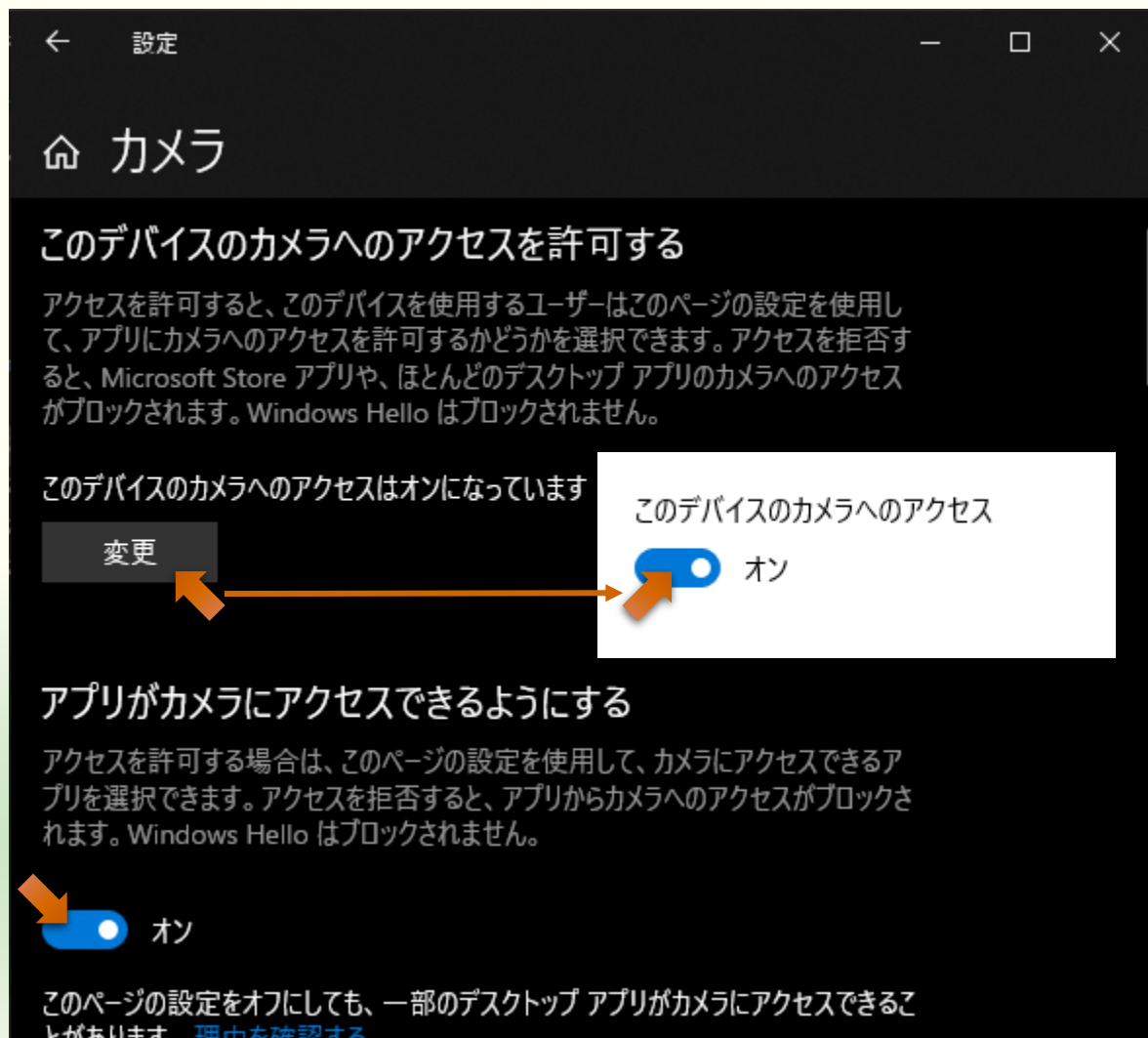
原点 (0, 0)



うまくいかなかったら次ページ



# 「キー」 → 「設定」 → 「プライバシー」



# 試しにデバイス番号を変えてみる

```
#include "ofApp.h"

//-----
void ofApp::setup(){
    video.setDeviceID(1);
    video.setup(320, 240);
}
```

(以下略)

デフォルトは0なので  
1 とかに変えてみる

1 が使えないと  
こうなる

C:\of\_v0.11.0\_vs2017\_release\apps\myApps\myGoodSketch\bin\myGo

\*\*\*\*\* VIDEOINPUT LIBRARY - 0.2000 - TFW2013 \*\*\*\*\*

SETUP: device[1] not found - you have 1 devices available

SETUP: this means that the last device you can use is device[0]

1 は使えない  
というエラー

0 までしか使えないと言っている

# 使えるデバイスを調べる

```
#include "ofApp.h"

//-----
void ofApp::setup(){
    for (auto device : video.listDevices()){
        cout << device.id << ": " << device.deviceName;
        if (device.bAvailable){
            // 使えるデバイスの後ろに "available" を付ける
            cout << " - available¥n";
        }
        else{
            // 使えないデバイスの後ろに "unavailable" を付ける
            cout << " - unavailable¥n";
        }
    }
    cout << endl;
    video.setDeviceID(0);
    video.setup(320, 240);
}
```

(以下略)

0 番を使うことにする

```
C:\of_v0.11.0_vs2017_release\apps\myApps\myGoodSketch\bin\myGoodSke...
***** VIDEOINPUT LIBRARY - 0.2000 - TFW2013 *****

VIDEOINPUT SPY MODE!

SETUP: Looking For Capture Devices
SETUP: 0) Logicoool Qcam Pro 9000
SETUP: 1) Leap Motion Controller
SETUP: 2) THETA UVC FullHD Blender
SETUP: 3) StUSBCam
SETUP: 4) THETA UVC HD Blender
SETUP: 5 Device(s) found

0: Logicoool Qcam Pro 9000 - available
1: Leap Motion Controller - available
2: THETA UVC FullHD Blender - available
3: StUSBCam - available
4: THETA UVC HD Blender - available

SETUP: Setting up device 0
SETUP: Logicoool Qcam Pro 9000
SETUP: Couldn't find preview pin using SmartTee
SETUP: Default Format is set to 640 by 480
SETUP: trying requested format RGB24 @ 320 by 240
SETUP: Capture callback set
SETUP: Device is setup and ready to capture.
```

video.listDevices() がデバッグ用に出力している

この PC には映像入力デバイスが 5 つ付いている

0 番の解像度のデフォルトは 640, 480 やで

でも 320, 240 でやれいうからやってみる

# 描画サイズをウィンドウサイズに合わせる

```
#include " ofApp.h"

//-----
void ofApp::setup(){
    (途中略)
}

//-----
void ofApp::update(){
    video.update();
}

//-----
void ofApp::draw(){
    video.draw(0, 0, ofGetWidth(), ofGetHeight());
}
```

(以下略)

- void ofApp::draw(float x, float y)
  - ofApp クラスの内部画像データの左上隅がウィンドウの (x, y) の位置になるように描画する
  - 内部画像データの画素数がそのまま反映される
- void ofApp::draw(float x, float y, float w, float h)
  - 内部画像データの画素数を幅 w、高さ h に変換して描画する

# 解像度は低いが全画面に表示される



# デバイスの設定パネルを呼び出せるようにする

```
#include " ofApp.h"
```

(途中略)

```
//-----
```

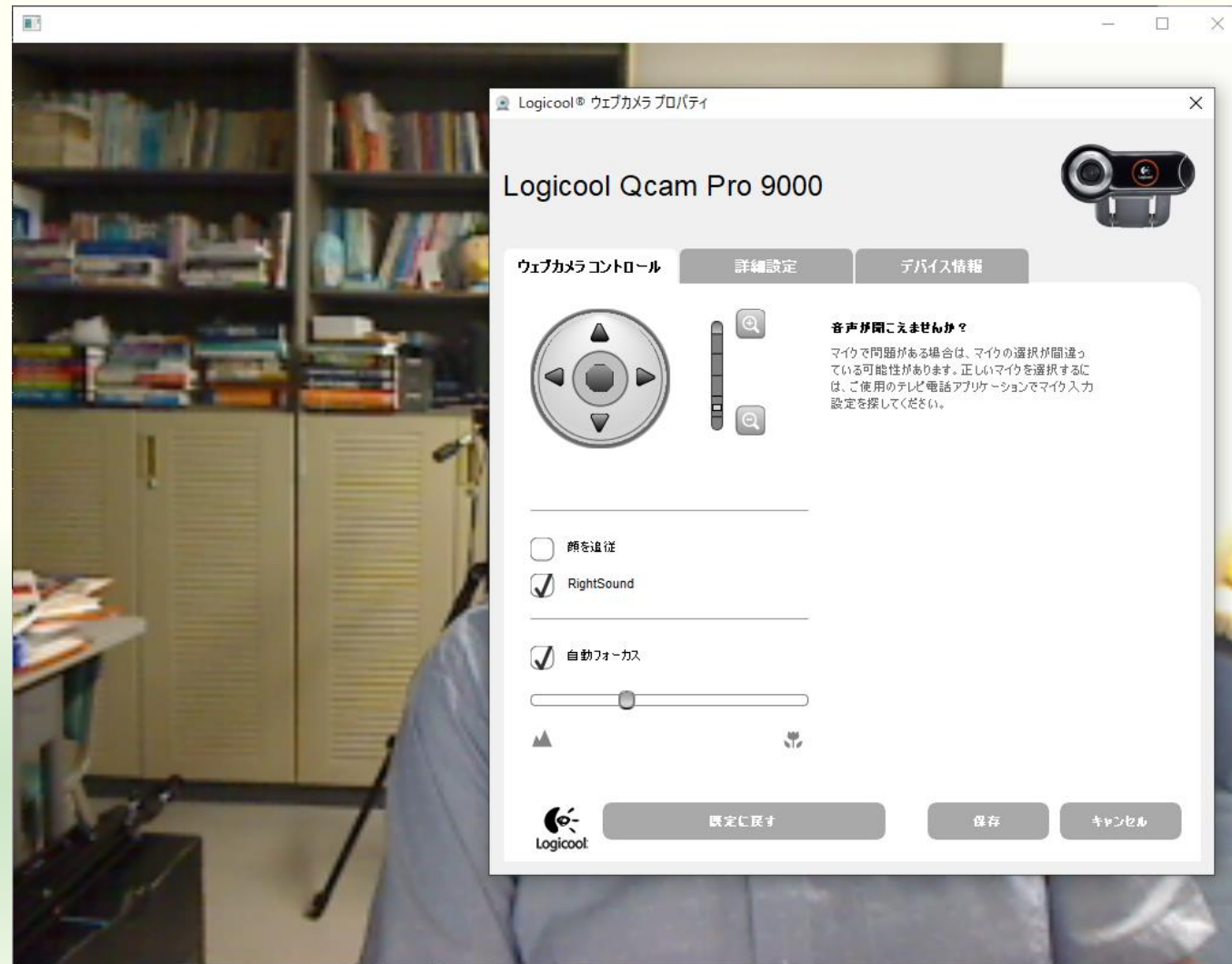
```
void ofApp::keyPressed(int key){  
    if (key == 's' || key == 'S'){  
        video.videoSettings();  
    }  
}
```

(以下略)

- 's' キーか 'S' キーをタイプしたら映像入力デバイスの設定パネルを開く
  - 映像入力デバイスが対応している場合



# 制御パネルを呼び出す





# 映像の加工

画素データの変更



# ofApp クラスに画素データとテクスチャのメンバ変数を追加する

```
#pragma once

#include "ofMain.h"

class ofApp : public ofBaseApp{
    ofVideoGrabber video;
    ofPixels color;
    ofTexture texture;

public:
    void setup();
    void update();
    void draw();

    void keyPressed(int key);
    void keyReleased(int key);
    void mouseMoved(int x, int y);
    (以下略)
```

- ofPixels
  - 画像の画素データのクラス
- ofTexture
  - 描画する画像を保持するクラス
  - テクスチャマッピング用のデータ



# ofApp.cpp で画素データ用のメモリを確保する

```
#include "ofApp.h"

//-----
void ofApp::setup(){
    (途中略)
    video.setDeviceID(0);
    video.setup(320, 240);
    color = video.getPixels();
}

//-----
void ofApp::update(){
    video.update();
}

(以下略)
```

- 画素データの表示に用いる変数 color のサイズやチャンネル数を 入力映像と同じにするために 入力画像 video の画素データ自体をコピーする



# フレームが更新されたか調べる

```
#include "ofApp.h"

//-----
void ofApp::setup(){
    (途中略)
}

//-----
void ofApp::update(){
    video.update();
    if (video.isFrameNew()){
        (次ページに続く)
```

- video.update();
  - 映像入力からフレームを取り込む
- if (video.isFrameNew()){
  - もしフレームが更新されていたら {} 内の処理をする
  - ofApp::update() や ofApp::draw() は画面表示のタイミングで実行されるが映像入力のタイミングは必ずしもそれと一致しない
  - そのため video.update(); しても前のフレームのまま更新されていないことがある

# 更新されたフレームの画素データをコピーする

(前ページからの続き)

```
ofPixels &input{ video.getPixels() };  
for (size_t i = 0; i < input.size(); ++i){  
    color[i] = input[i];  
}  
texture.loadData(color);  
}
```

(以下略)

- ofPixels &input{ video.getPixels() };
  - 更新されたフレームの画素データを input で**参照**できるようにする
- for (size\_t i = 0; i < input.size(); ++i){
  - 更新されたフレームの画素データ input の数だけ {} 内を繰り返す
  - color[i] = input[i];
    - 更新されたフレームの画素データ input を表示用画素データ color にコピーする
- texture.loadData(color);
  - コピーした画素データを描画に使うテクスチャデータに転送する

# 画素データを描画する

```
#include " ofApp.h"

//-----
void ofApp::setup(){
    (途中略)
}

//-----
void ofApp::update(){
    (途中略)
}

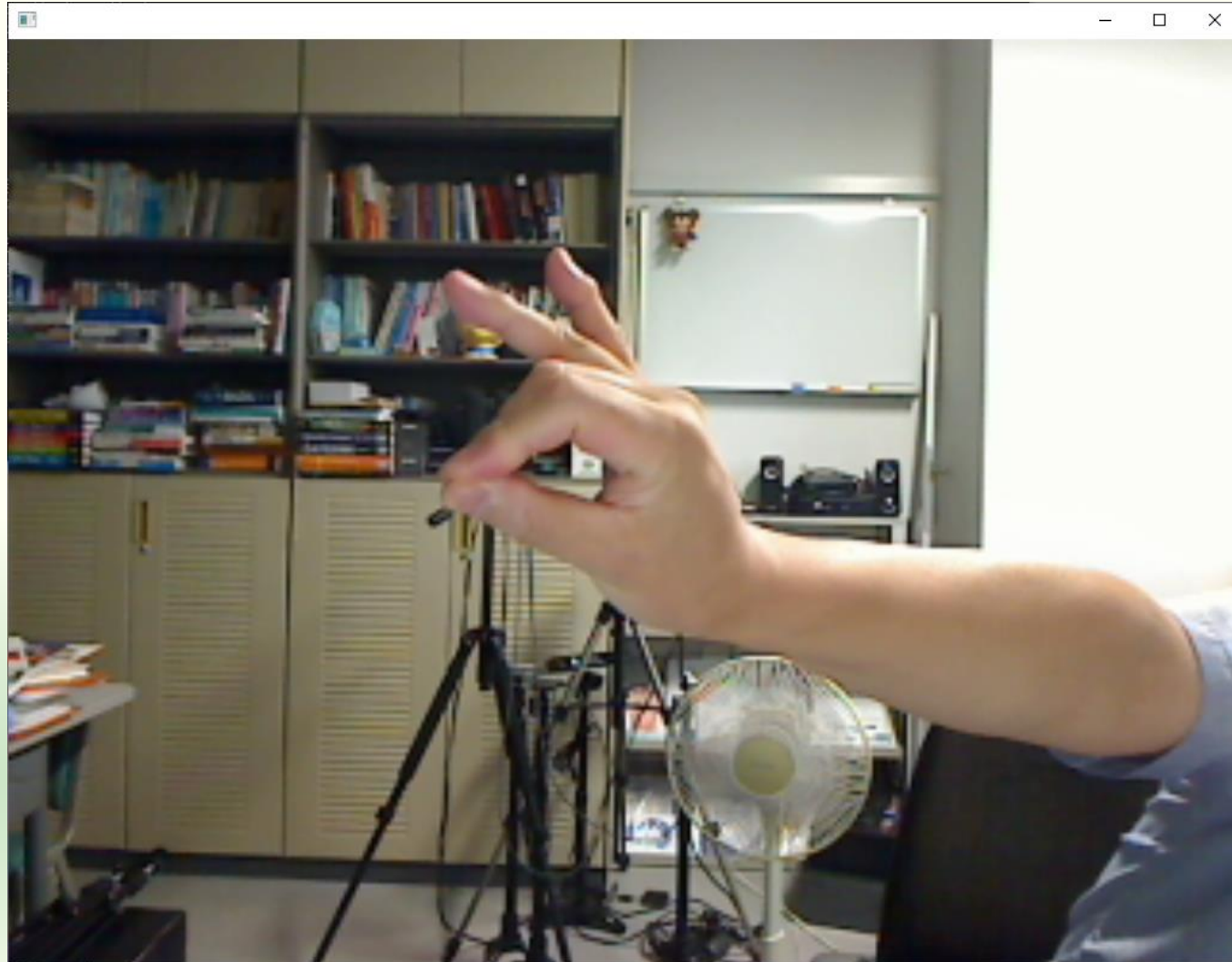
//-----
void ofApp::draw(){
    texture.draw(0, 0, ofGetWidth(), ofGetHeight());
}

(以下略)
```

- ofPixels クラスの画素データは直接描画できない
  - ofApp::draw() などというメソッドはない



# 特に変わらない



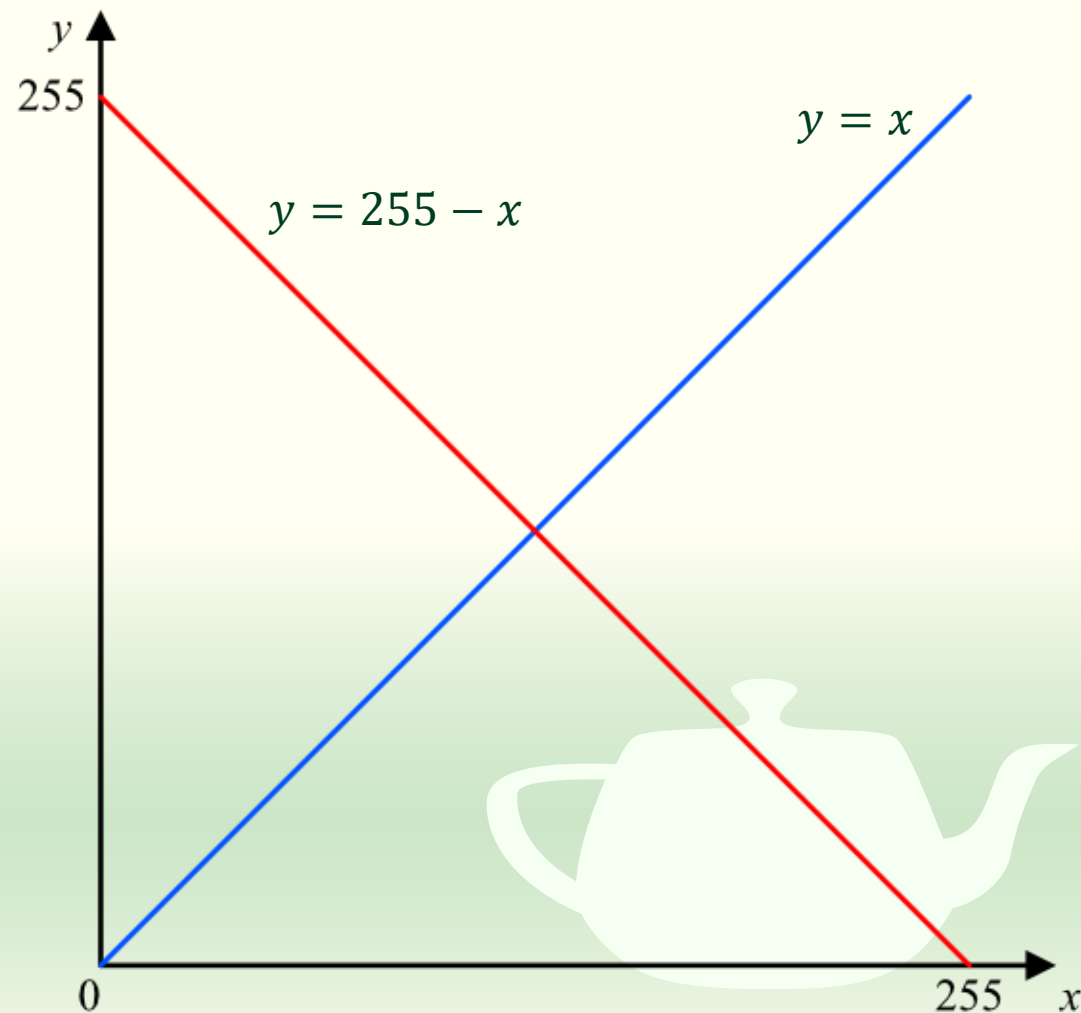
# 明るさを反転する

```
#include "ofApp.h"

//-----
void ofApp::setup(){
    (途中略)
}

//-----
void ofApp::update(){
    video.update();
    if (video.isFrameNew()){
        ofPixels &input{ video.getPixels() };
        for (size_t i = 0; i < input.size(); ++i){
            color[i] = 255 - input[i];
        }
        texture.loadData(color);
    }
}
```

(以下略)



# 階調の反転







# 加工方法の切り替え

キー操作で処理を切り替えられるようにする

# ofApp クラスにタイプしたキーを保持するメンバ変数を追加する

```
#pragma once

#include "ofMain.h"

class ofApp : public ofBaseApp{
    ofVideoGrabber video;
    ofPixels color;
    ofTexture texture;
    int select;

public:
    void setup();
    void update();
    void draw();

    void keyPressed(int key);
    void keyReleased(int key);
    void mouseMoved(int x, int y);
    (以下略)
```

- ofApp クラスに int 型の select というメンバ変数を追加する
  - これにキー操作の結果を入れる



# ofApp.cpp でキー操作の初期設定を行う

```
#include "ofApp.h"

//-----
void ofApp::setup(){
    (途中略)
    video.setDeviceID(0);
    video.setup(320, 240);
    color = video.getPixels();
    select = '0';
}
```

(以下略)

- select の初期値は文字定数の '0' にしておく
  - 実はこの後（次の次のページ）で select が '0' の時に color = input; という代入をして color に input のコピーを用意するので、ここでメモリを確保する必要はなくなる



# タイプしたキーを select に代入する

```
#include "ofApp.h"

//-----
void ofApp::draw(){
    texture.draw(0, 0, ofGetWidth(), ofGetHeight());
}

//-----
void ofApp::keyPressed(int key){
    if (key == 's' || key == 'S'){
        video.videoSettings();
    }
    else{
        select = key;
    }
}
```

- key を select に代入する
  - 's', 'S' 以外の**文字コード**が入る



# select の内容によって異なり処理をする

```
//-----  
void ofApp::update(){  
    video.update();  
    if (video.isFrameNew()){  
        ofPixels &input{ video.getPixels() };  
        switch (select){  
            case '0':  
                color = input;  
                break;  
            case '1':  
                for (size_t i = 0; i < input.size(); ++i){  
                    color[i] = 255 - input[i];  
                }  
                break;  
            default:  
                break;  
        }  
        texture.loadData(color);  
    }  
}
```

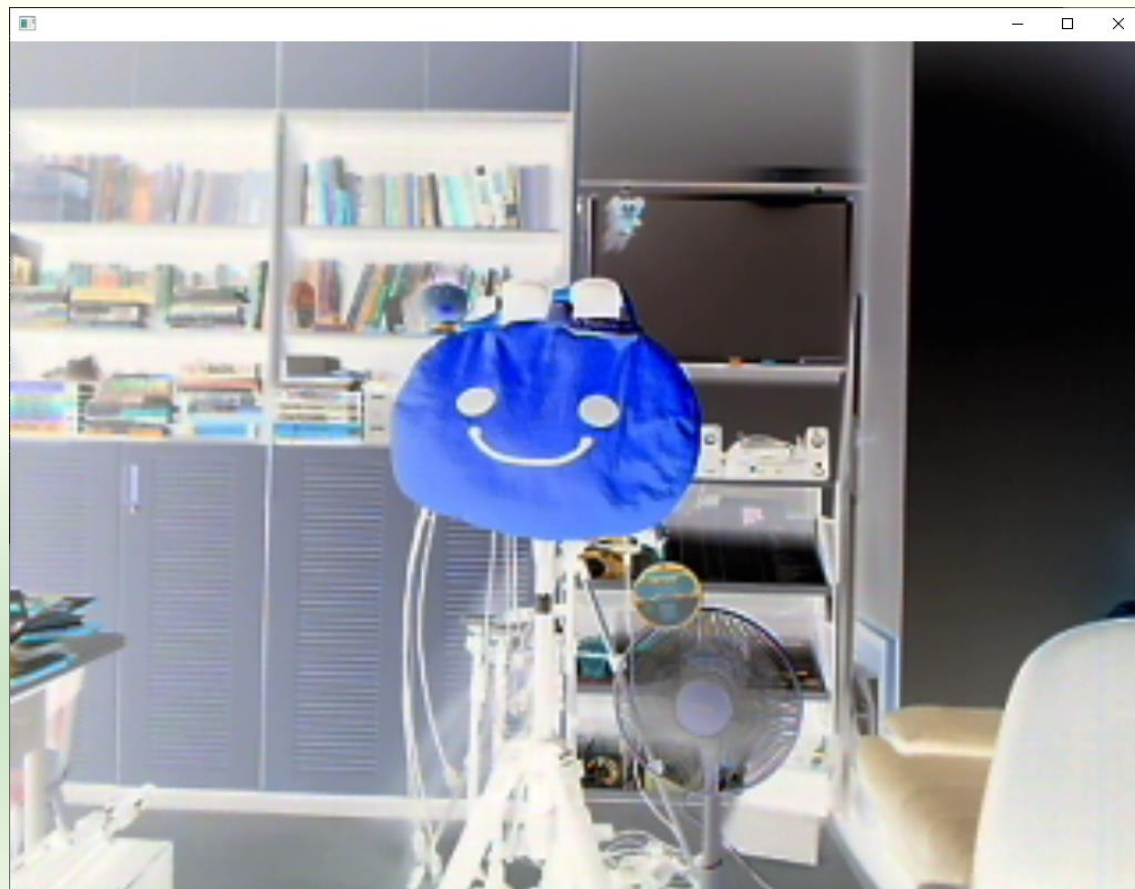
- swich (select){
  - select の内容に従って {} 内の case ラベルに分岐する
- case '0':
  - select の内容が '0' のとき、これ以降が実行される
- color = input;
  - input の全画素データを color にコピーする
- break;
  - swich の {} 内の処理から抜け出る
- default:
  - select がどの case ラベルとも一致しないとき、これ以降が実行される

# キーのタイプで結果が変わる

## ‘0’ のタイプ以降



## ‘1’ のタイプ以降



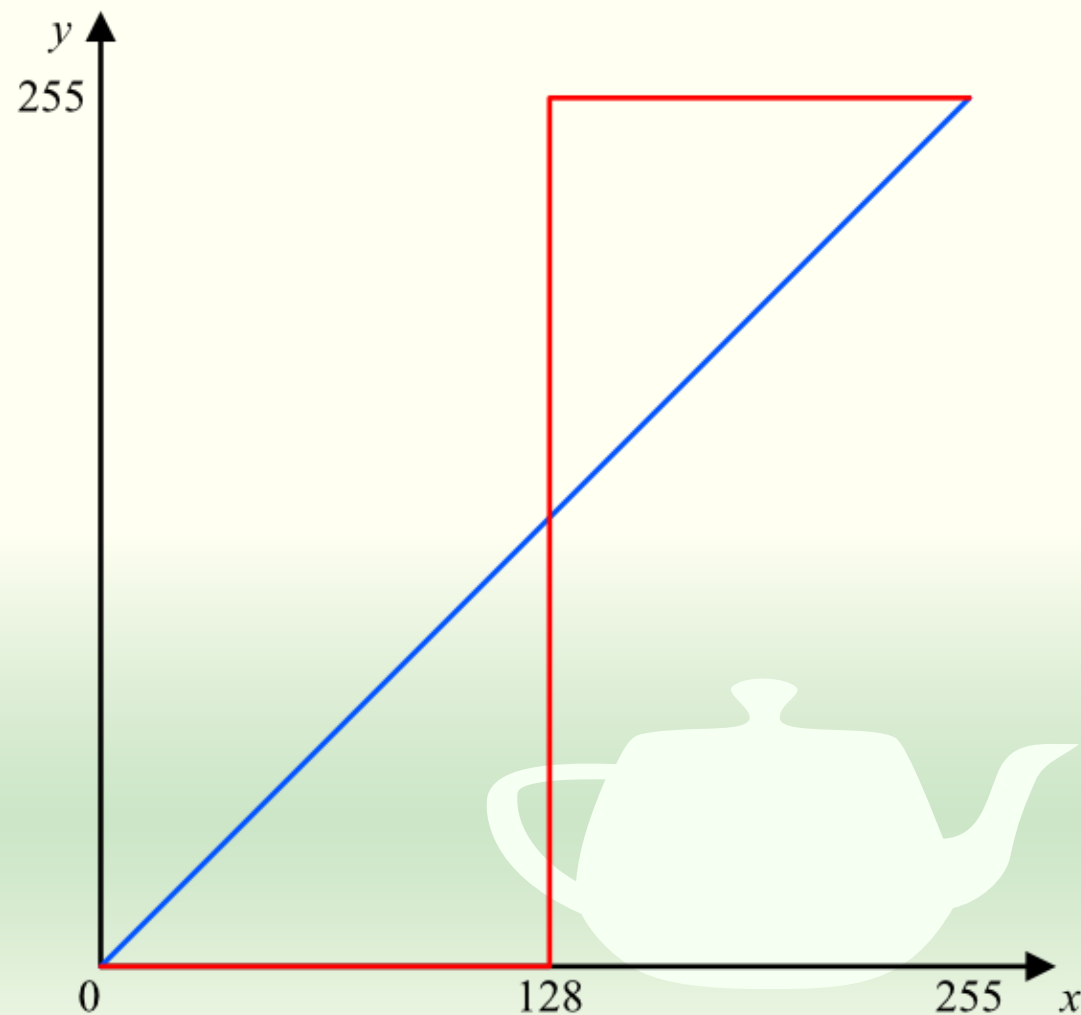


# 課題 5 - 1

2 値化

# 画像の各チャネルを二値化して表示する

- '2' キーをタイプしたら R, G, B のそれぞれのチャネルを 2 値化して表示するようにしなさい
- 2 値化
  - 入力データ  $x$  を閾値 (いきち)  $t$  と比較して、 $x < t$  なら値を最小値 (ここでは 0) に、それ以外なら最大値 (ここでは 255) にする
  - ここでは閾値  $t$  を 128 とする





# 結果の例



# 課題のアップロード

- 作成したプログラムの実行結果のスクリーンショットを撮って **5-1.png** というファイル名で保存し、Moodle の第 5 回課題にアップロードしてください



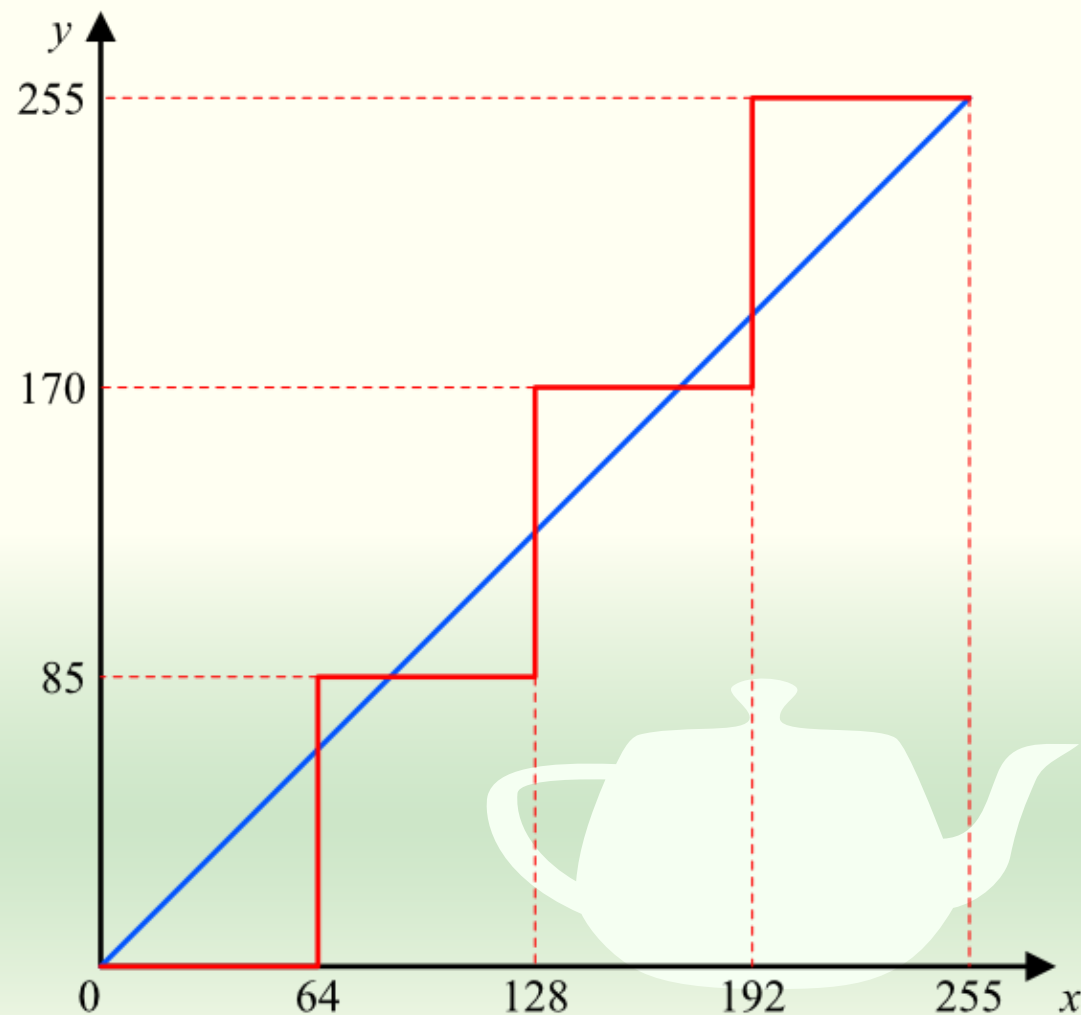


# 課題 5 - 2

ポスター化

# 画像の各チャネルを 4 階調化して表示する

- '3' キーをタイプしたら映像を 4 階調化（4 階調のポスター化）して表示するようにしなさい
- 画素値  $x$  は 0~255 の値を持つ
- $x$  を 64 で割り小数点以下を切り捨てれば 0~3 の値になる
  - 整数の除算をすれば小数点以下は切り捨てられる
- これに 85 を掛ければ 0, 85, 170, 255 のいずれかの値が得られる



# 結果の例



# 課題のアップロード

- 作成したプログラムの実行結果のスクリーンショットを撮って **5-2.png** というファイル名で保存し、Moodle の第 5 回課題にアップロードしてください





# チャネルの入れ替え

R と B を入れ替えて RGB を BGR にする

# ‘4’ キーで赤と青を入れ替えて表示する

```
//-----  
void ofApp::update(){  
    video.update();  
    if (video.isFrameNew()){  
        ofPixels &input{ video.getPixels() };  
        switch (select){  
            (途中略)  
            case '4':  
                for (size_t i = 0; i < input.size(); i += 3){  
                    int r{ input[i] };  
                    int g{ input[i + 1] };  
                    int b{ input[i + 2] };  
                    color[i] = b;  
                    color[i + 1] = g;  
                    color[i + 2] = r;  
                }  
                break;  
            default:  
                break;  
        }  
        (以下略)
```

- ofPixels &input{ video.getPixels() };
- input は 1 画素の各チャンネルが要素ごとに入っている
- RGB (アルファチャンネルなし)
  - input[0] → R, input[1] → G, input[2] → B
  - input[3] → R, input[4] → G, input[5] → B
- 従って要素の番号 i = 0 から始めて i < input.size() の間以下を繰り返す
  - r に input[i], g に input[i + 1], b に input[i + 2] を代入する
  - color[i] に b, color[i + 1] に g, color[i + 2] に r を代入する
  - i を 3 増やす (i += 3)



# 結果の例





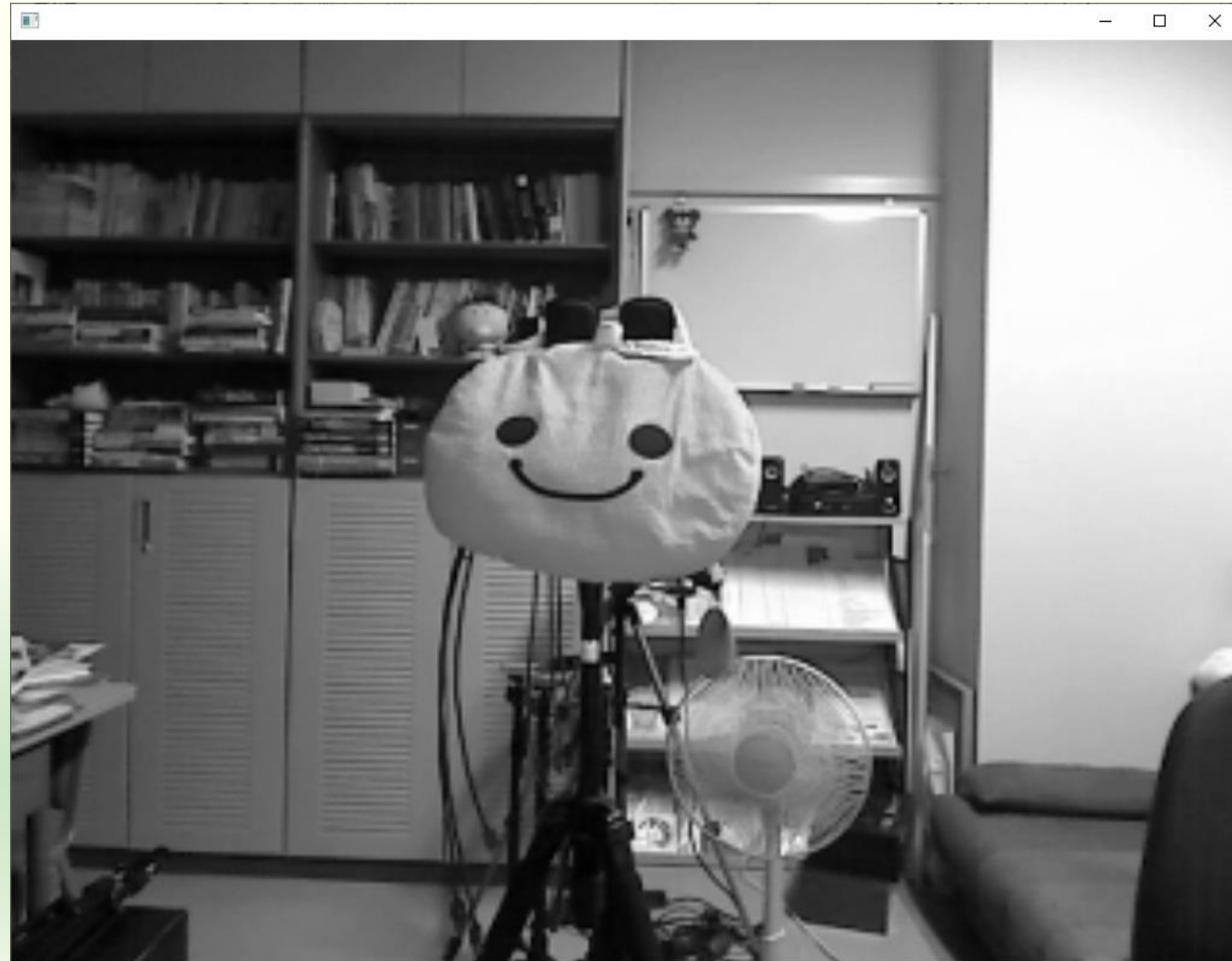
# 課題 5 – 3

グレースケール化

# 画像をグレースケール化して表示する

- '5' キーをタイプしたら映像をグレースケール化（白黒画像化）して表示するようにしなさい
- グレースケール化
  - $Y = (R + G + B)/3$ 
    - RGB 平均、または
  - $Y = 0.299R + 0.587G + 0.114B$ 
    - ITU-R Rec BT.601
    - 人間の目は G に対して感度が高く B に対して低いという特性があるため RGB 平均だと G が低めに出る
  - 他にも多くの変換式がある
- `ofPixels &input{ video.getPixels() };`
  - `input` は 1 画素の各チャンネルが要素ごとに入っている
  - RGB（アルファチャンネルなし）
    - `input[0] → R, input[1] → G, input[2] → B`
    - `input[3] → R, input[4] → G, input[5] → B`
  - 従って要素の番号 `i = 0` から始めて `i < input.size()` の間以下を繰り返す
    - `input[i], input[i + 1], input[i + 2]` をグレースケール化して `y` を求める
    - `color[i], color[i + 1], color[i + 2]` にいずれも `y` を代入する
    - `i` を 3 増やす (`i += 3`)

# 結果の例



# 課題のアップロード

- 作成したプログラムの実行結果のスクリーンショットを撮って **5-3.png** というファイル名で保存し、Moodle の第 5 回課題にアップロードしてください





# 背景差分法

背景を切り抜く

# 画像の差

入力映像

0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	1	1	1	1	1	1	1	1	1	0	0
0	1	2	2	2	2	2	2	2	2	1	0	0
0	1	2	3	3	3	3	3	3	2	1	0	0
0	1	2	3	4	4	4	4	3	2	1	0	0
0	1	2	3	4	5	5	4	3	2	1	0	0
0	1	2	3	4	5	5	4	3	2	1	0	0
0	1	2	3	4	5	5	4	3	2	1	0	0
0	1	2	3	4	4	4	4	3	2	1	0	0
0	1	2	3	3	3	3	3	3	2	1	0	0
0	1	2	2	2	2	2	2	2	2	1	0	0
0	1	1	1	1	1	1	1	1	1	1	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0

背景画像

0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	1	1	1	1	1	1	1	1	1	0	0
0	1	2	2	2	2	2	2	2	2	1	0	0
0	1	2	3	3	3	3	3	3	2	1	0	0
0	1	2	3	4	4	4	4	3	2	1	0	0
0	1	2	3	4	5	5	4	3	2	1	0	0
0	1	2	3	4	5	5	4	3	2	1	0	0
0	1	2	3	4	5	5	4	3	2	1	0	0
0	1	2	3	4	4	4	4	3	2	1	0	0
0	1	2	3	3	3	3	3	3	2	1	0	0
0	1	2	2	2	2	2	2	2	2	1	0	0
0	1	1	1	1	1	1	1	1	1	1	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0

—

=

差

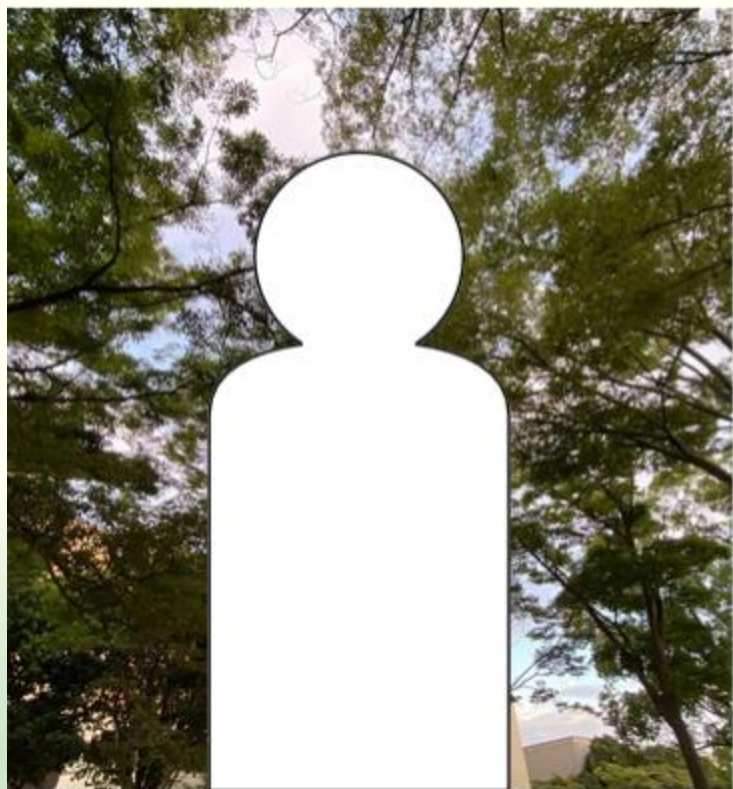
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0

画像は2次元の（画素の）データの配列⇒対応する画素同士の引き算が可能  
同じ画像同士を引き算すると、当然全部0すなわち黒



# 差異部分の抽出

入力映像



背景画像



差



したがって二つの画像に異なる部分があれば、その部分だけが0でなくなる  
逆に0に近い部分は内容が似ているので、その部分が背景だと判断できる



# ofApp.h で ofApp クラスに背景を保存するメンバ変数を追加する

```
#pragma once

#include "ofMain.h"

class ofApp : public ofBaseApp{
    ofVideoGrabber video;
    ofPixels color, saved;
    ofTexture texture;
    int select;

public:
    void setup();
    void update();
    void draw();

    void keyPressed(int key);
    void keyReleased(int key);
    void mouseMoved(int x, int y);
    (以下略)
```

- ofApp クラスのメンバ変数に ofPixels クラスの saved というインスタンスを追加する
  - saved に背景の画像を保存する



# ofApp.cpp で保存用のメモリを確保する

```
#include "ofApp.h"

//-----
void ofApp::setup(){
    (途中略)
    video.setDeviceID(0);
    video.setup(320, 240);
    saved = color = video.getPixels();
    select = '0';
}

//-----
void ofApp::update(){
    video.update();
}

(以下略)
```

- 画素データの保存に用いる変数  
saved のサイズとチャンネル数も  
入力映像と同じにする



# ofApp.cpp で '6' キーをタイプした時にフレームを保存する

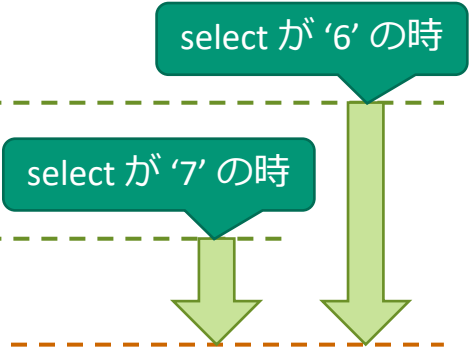
```
//-----  
void ofApp::update(){  
    video.update();  
    if (video.isFrameNew()){  
        ofPixels &input{ video.getPixels() };  
        switch (select){  
            case '0':  
                color = input;  
                break;  
            case '1':  
                for (size_t i = 0; i < color.size(); ++i){  
                    color[i] = 255 - input[i];  
                }  
                break;  
            (途中略)  
            case '6':  
                saved = input;  
                select = '7';  
                break;  
            default:  
                (以下略)
```

- select が '6' なら現在のフレームの画素データ input を saved にコピーして取っておく
- select に '7' を代入しておく
  - この後で select が '7' の時に saved の内容を表示するようにする



# ‘7’ キーをタイプしたら保存したフレームを表示する

```
//-----  
void ofApp::update(){  
    video.update();  
    if (video.isFrameNew()){  
        ofPixels &input{ video.getPixels() };  
        switch (select){  
            (途中略)  
            case '6': -----  
                saved = input;  
                select = '7';  
            case '7': -----  
                color = saved;  
                break; -----  
            default:  
                break;  
        }  
        (以下略)
```



- select が ‘7’ なら保存したフレーム saved を表示するフレーム color にコピーする
- この処理を case ‘6’ の break の前に置く
  - したがって case ‘6’ と次の case ‘7’ 間には break がない
  - そのため select が ‘6’ の時は case ‘6’ と case ‘7’ の両方の処理が実行され、select が ‘7’ の時は case ‘7’ 以降の処理だけが実行される

# ‘6’ キーをタイプして画像を保存していれば

‘0’ で現在の映像が表示される



‘7’ で保存した画像が表示される





# 課題 5 - 4

入力映像と背景画像の差の絶対値

# 入力映像と背景画像の差の絶対値を表示する

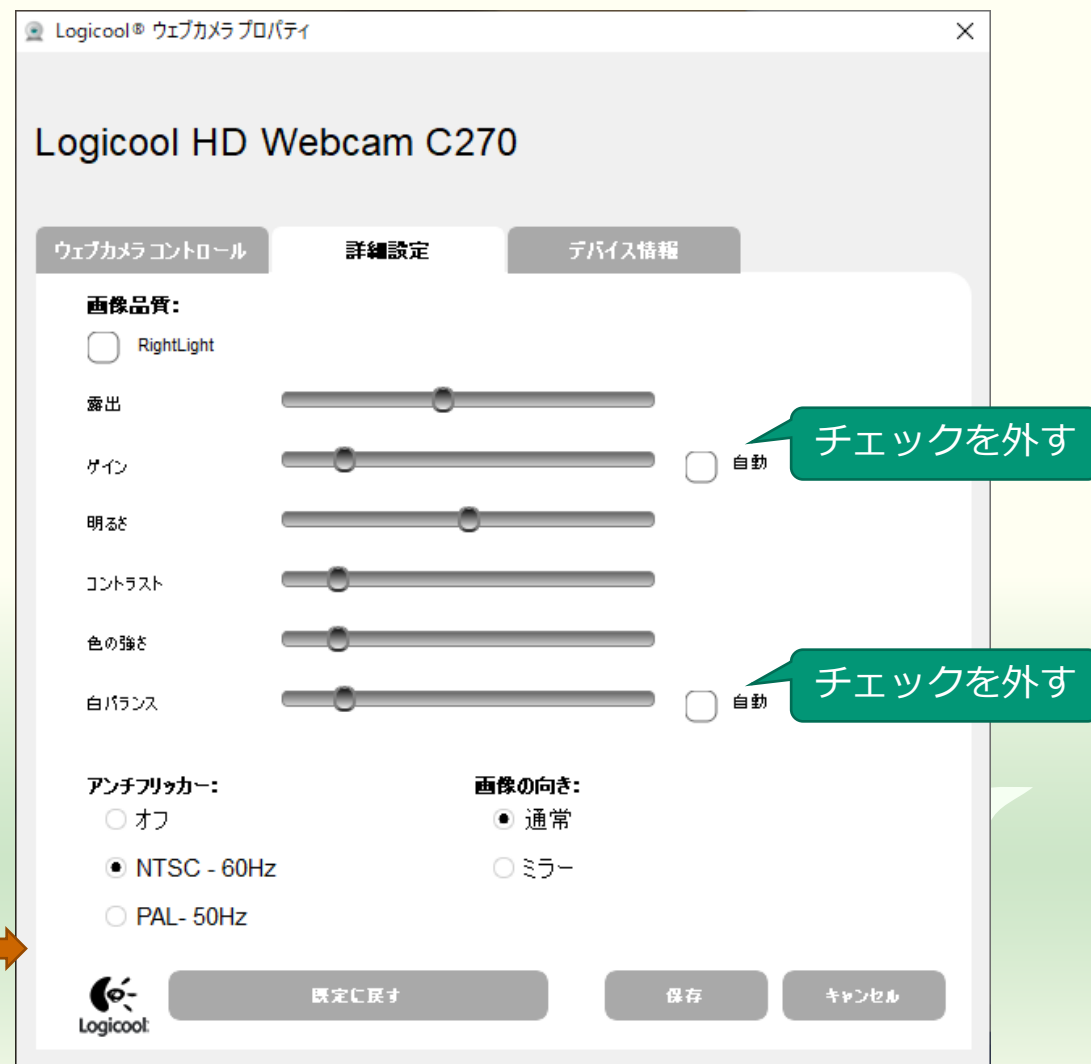
- '8' キーをタイプしたら入力映像と背景画像の差の絶対値を表示するようにしなさい
  - `input[i]` と `saved[i]` の差の絶対値を `color[i]` に代入する
- 絶対値を求める標準ライブラリ関数 `abs()`
  - `z = abs(x - y);`
    - `z` は `x` と `y` の差の絶対値
  - <https://cpprefjp.github.io/reference/cmath/abs.html>



# カメラの露出・利得を固定する

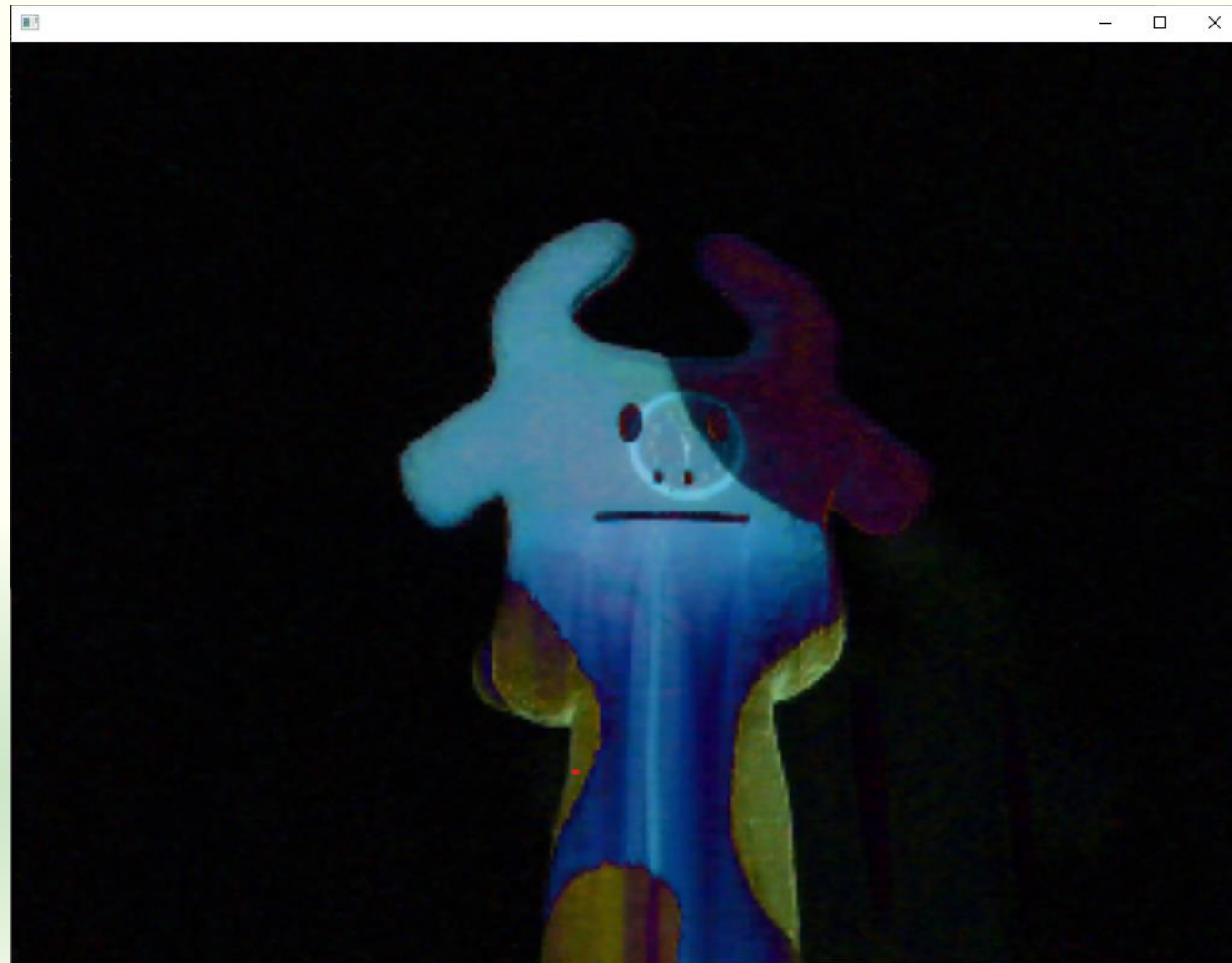
- この方法はカメラの露出や利得（ゲイン）を固定しておかないと背景がきれいに黒くならない
- ‘s’ または ‘S’ キーをタイプして制御パネルで調整する
  - カメラが制御パネルの呼び出しに対応していない場合はあらかじめご了承ください

これは一例 →





# 結果の例（入力映像と背景画像の差の絶対値）



# 課題のアップロード

- 作成したプログラムの実行結果のスクリーンショットを撮って **5-4.png** というファイル名で保存し、Moodle の第 5 回課題にアップロードしてください





## 課題 5 - 5

入力映像と背景画像の差の絶対値が閾値以下の画素を黒にする

# 差の絶対値が閾値以下の画素を黒で表示する

- '9' キーをタイプしたら入力映像と背景画像の差の絶対値が閾値以下の画素を黒で表示するようにしなさい
- 画素値の差の絶対値との比較
  - 入力映像と背景画像の R, G, B のチャンネルごとの差の絶対値のそれぞれと閾値を比較する
  - 全てのチャンネルの差の絶対値が閾値を超えていれば入力映像の画素を表示し、そうでなければ黒 ( $R = G = B = 0$ ) を表示する
- `ofPixels &input{ video.getPixels() };`
- 要素の番号  $i = 0$  から始めて  $i < \text{input.size}()$  の間以下を繰り返す
  - `input[i]` と `saved[i]`、`input[i + 1]` と `saved[i + 1]`、`input[i + 2]` と `saved[i + 2]` の差の絶対値をそれぞれ求める
  - これらの値のそれぞれと閾値を比較し、すべて閾値を超えていたら `color[i]`、`color[i + 1]`、`color[i + 2]` に `input[i]`、`input[i + 1]`、`input[i + 2]`、そうでなければ 0 を代入する
  - $i$  を 3 増やす ( $i += 3$ )

# 結果の例（差の絶対値が閾値以下の画素を黒）



# 課題のアップロード

- 作成したプログラムの実行結果のスクリーンショットを撮って **5-5.png** というファイル名で保存し、Moodle の第 5 回課題にアップロードしてください





## 課題 5 - 6

入力映像と背景画像の差の絶対値が閾値以下の領域に別の画像を表示する

# ofApp クラスに別の画像を保持するメンバ変数を追加する

```
#pragma once

#include "ofMain.h"

class ofApp : public ofBaseApp{
    ofVideoGrabber video;
    ofPixels color, saved;
    ofTexture texture;
    ofImage image;
    int select;

public:
    void setup();
    void update();
    void draw();

    void keyPressed(int key);
    void keyReleased(int key);
    void mouseMoved(int x, int y);
    (以下略)
```

- ofApp クラスに ofImage クラスの image というメンバ変数を追加する





# ofApp.cpp の setup() で image に画像を読み込む

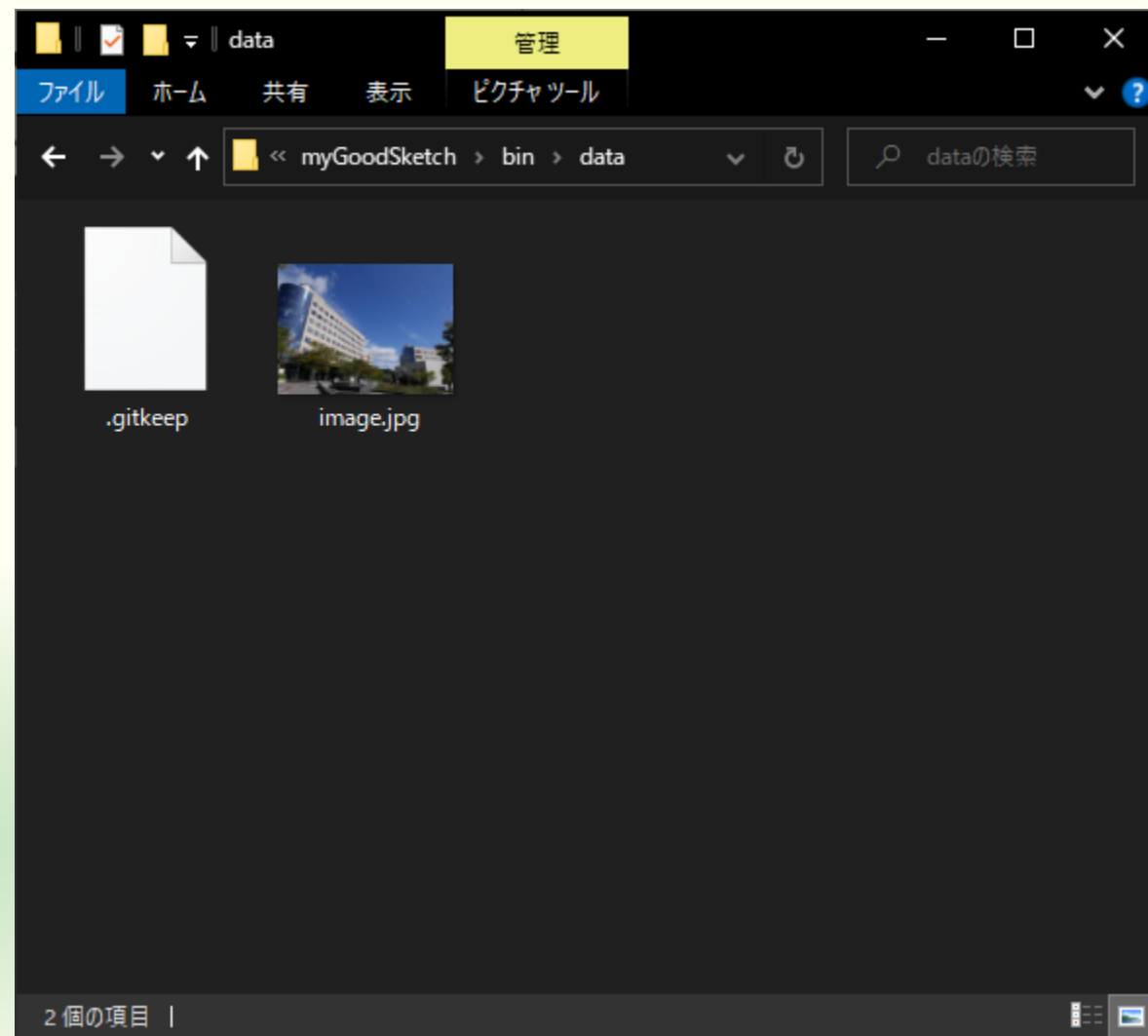
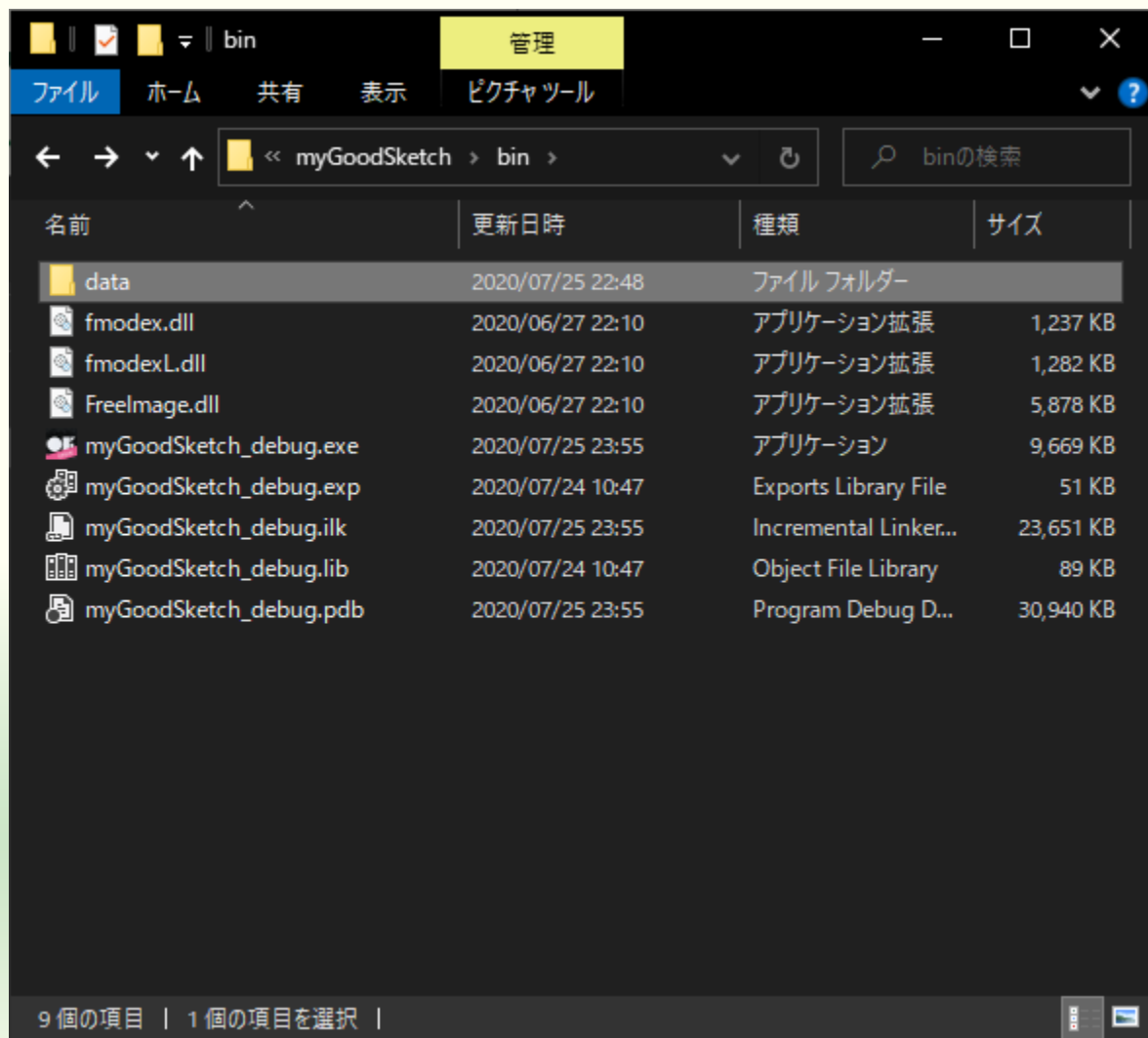
```
#include "ofApp.h"

//-----
void ofApp::setup(){
    (途中略)
    video.setDeviceID(0);
    video.setup(320, 240);
    saved = color = video.getPixels();
    select = '0';
    image.load("image.jpg");
    image.resize(video.getWidth(),
                video.getHeight());
}
```

(以下略)

- image.load("image.jpg");
  - プロジェクトのフォルダの bin の data の中にある image.jpg という画像ファイルを image に読み込む
- "image.jpg" は画像ファイル名
  - JPEG, PNG, GIF 画像が読み込める
- image.resize(video.getWidth(), video.getHeight());
  - 読み込んだ画像のサイズを入力映像のサイズに合わせる

# 画像は bin フォルダの中の data に配置する



# 入力映像と背景画像の差の絶対値が閾値以下の領域に別の画像を表示する

- 入力映像と背景画像の差の絶対値が閾値以下の画素を黒にする代わりに別の画像の同じ位置の画素の色を表示する
- ofImage の画像から画素データを取り出すには `getPixels()` メソッドを使う
  - `ofPixels &back = image.getPixels();`
    - `back` は入力映像と同じサイズにした画像 `image` の画素データを参照する



# 課題のアップロード

- 作成したプログラムの実行結果のスクリーンショットを撮って **5-6.png** というファイル名で保存し、Moodle の第 5 回課題にアップロードしてください





# テクスチャマッピング

入力映像を 3 D オブジェクトに貼り付ける

# ofApp クラスに箱とライト、カメラのメンバ変数を追加する

```
#pragma once

#include "ofMain.h"

class ofApp : public ofBaseApp{
    ofVideoGrabber video;
    ofPixels color, saved;
    ofTexture texture;
    ofImage image;
    ofBoxPrimitive box;
    ofLight light;
    ofEasyCam camera;
    int select;

public:
    void setup();
    void update();
    void draw();
```

(以下略)

- ofBoxPrimitive は箱のクラス
- ofLight はライトのクラス
- ofEasyCam はマウスで制御できるカメラのクラス



# カメラとライトの設定を行う

```
#include "ofApp.h"

//-----
void ofApp::setup(){
    (途中略)
    video.setDeviceID(0);
    video.setup(320, 240);
    saved = color = video.getPixels();
    select = '0';
    image.load("image.jpg");
    image.resize(video.getWidth(),
        video.getHeight());
    camera.setPosition(0.0f, 0.0f, 200.0f);
    light.setPosition(60.0f, 80.0f, 100.0f);
    light.enable();
}
```

- camera.setPosition(0.0f, 0.0f, 200.0f);
  - カメラ camera の位置を設定する
- light.setPosition(60.0f, 80.0f, 100.0f);
  - ライト light の位置を設定する
- light.enable();
  - ライト light を有効にする



# 表示する画像を箱の表面のサイズに合わせる

```
//-----  
void ofApp::update(){  
    video.update();  
    if (video.isFrameNew()){  
        ofPixels &input = video.getPixels();  
        switch (select) {  
            (途中略)  
        default:  
            break;  
        }  
        texture.loadData(color);  
        box.mapTexCoordsFromTexture(texture);  
    }  
}
```

- box.mapTexCoordsFromTexture(texture);
  - box の表面に texture をぴったり貼り付ける設定を行う





# 表示している画像に重ねて箱を描く

```
//-----  
void ofApp::draw(){  
    ofDisableLighting();  
    texture.draw(0, 0, ofGetWidth(), ofGetHeight());  
    ofEnableLighting();  
    camera.begin();  
    ofEnableDepthTest();  
    texture.bind();  
    box.draw();  
    texture.unbind();  
    ofDisableDepthTest();  
    camera.end();  
}
```

- ofEnableLighting();～  
ofDisableLighting();
  - この間の 3D CG の描画で陰影付けが有効になる
- ofEnableDepthTest();～  
ofDisableDepthTest();
  - この間の 3D CG の描画で隠面消去処理を有効にする
- texture.bind();～ texture.unbind();
  - この間に描画する 3D CG の図形で texture のマッピング（貼り付け）を有効にする

# 映像が箱の表面にマッピングされる





# 課題 5 – 7

箱の代わりに映像を球にマッピングする

# 球にマッピングした結果



# 箱の代わりに映像を球にマッピングしなさい

- ofBoxPrimitive を ofSpherePrimitive に替えるだけでできる
  - 球のサイズが小さいので setRadius() メソッドで設定してください
  - 光源の位置は変更したほうがいいかもしれません
- 余裕があれば ofConePrimitive, ofCylinderPrimitive, ofIcoSpherePrimitive, ofPlanePrimitive でも試してみてください



# 課題のアップロード

- 作成したプログラムの実行結果のスクリーンショットを撮って **5-7.png** というファイル名で保存し、Moodle の第 5 回課題にアップロードしてください
- ソースプログラム **ofApp.h** と **ofApp.cpp** を Moodle の第 5 回課題にアップロードしてください





# 時間の余った人向け課題

第3回の「階層構造」や「課題3－4」の球の部分に映像を貼り付けてみてください



# 補足

グレースケール化について



# グレースケール化の方法による違い (1)

RGB 平均



ITU-R Rec BT.601



# グレースケール化の方法による違い (2)

RGB 平均



ITU-R Rec BT.601

