

IDD Media lab. 2015

openFrameworks Addonとは?

ofxOpenCv、ofxCv

多摩美術大学情報デザイン学科メディア芸術コース

2015年5月19日

田所淳

Addonとは?

Addonとは?

- ▶ Addonとは、oepnFrameworksの機能を何らかの方法で拡張するコード
- ▶ 「ofx」 という接頭辞で始まる

Addonとは?

- ▶ openFrameworksに機能を拡張するためのライブラリー
 - ▶ processingのLibrariesのような存在
 - ▶ openFrameworks単体ではできなかった様々な機能を実現
 - ▶ oF本体の開発者以外でも独自に開発して追加することが可能
-
- ▶ アドオン情報
 - ▶ ofxAddons
 - ▶ <http://ofxaddons.com/>

Addonとは?

▶ ofxAddons.com

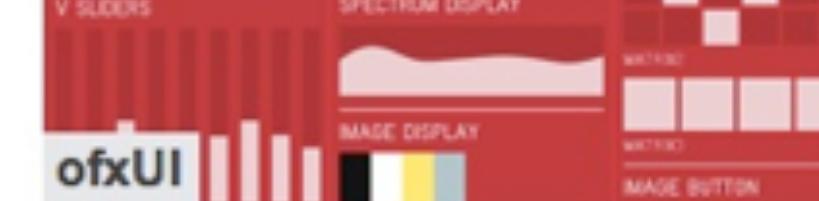
 ofxAddons is a directory of extensions and libraries for the **OpenFrameworks** creative coding toolkit. Compiled fresh from **Github** daily.

addons // freshest // unsorted // contributors // make your own!

category	animation	bridges	computer vision	graphics	gui	etc	stars	updated since	found 687 addons
clear									
all	hardware	interface	ios	machine learning	physics	sound	config file	example	
	typography	utilities	video/camera	web/networking					


ofxKinect
[video/camera] legacy openFrameworks wrapper for the xbox kinect (OF pre-0.8.0+ only) - ofxKinect is now included and is being maintained in OF releases
config file 1 example
created by  ofTheo
updated about a month ago (~0.7.4) ★ 470


ofxFaceTracker
[computer vision] ASM face tracking addon based on Jason Saragih's FaceTracker.
8 examples
created by  kylemcdonald
updated about a month ago (~0.7.4) ★ 372


ofxUI
[gui] A User Interface Library/Addon for openFrameworks
19 examples
created by  rezaali
updated 6 days ago (~0.8.0) ★ 274
fork by  Noura
updated 3 days ago (~0.8.0)


ofxOpenNI
[computer vision] Wrapper for OpenNI, NITE and SensorKinect
1 example
created by  gameoverhack
updated 4 days ago (~0.8.0) ★ 174
fork by  Lyptik


ofxCv
[computer vision] Alternative approach to interfacing with OpenCv from openFrameworks.
27 examples
created by  kylemcdonald

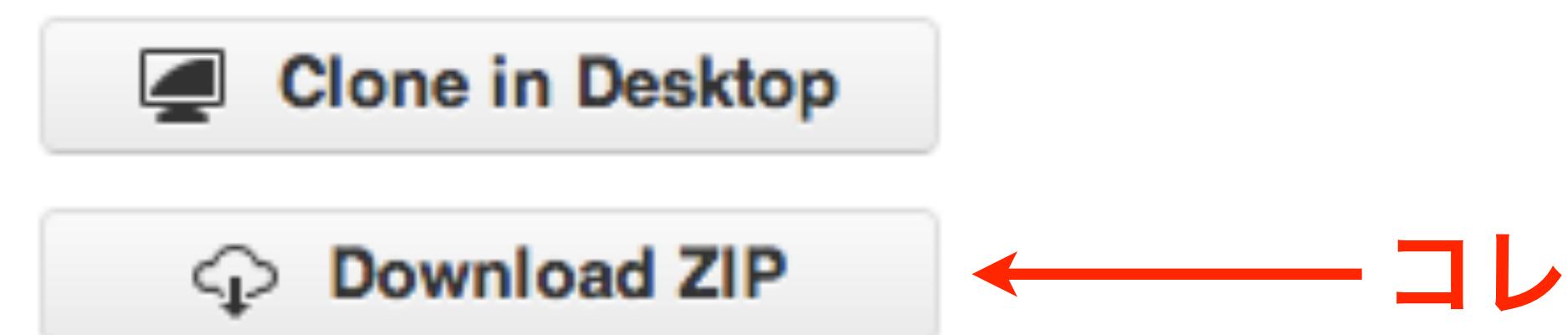

ofxBox2d
[physics] Openframework wrapper for box2d
11 examples
created by  vanderlin
updated about a month ago (~0.7.4) ★ 115

Addonとは?

- ▶ どうやってofxAddons.comのアドオンをインストールすのか?
- ▶ 方法1: git コマンドをつかって、コードの複製をつくる

```
$cd ofx_preRelease/addons/  
$git clone https://github.com/obviousjim/ofxSomeAddon
```

- ▶ 方法2: WebサイトからZip形式ダウンロード



Addonとは?

- ▶ Addonは何のためにあるの?
- ▶ 外部ライブラリやフレームワークをoFに統合
- ▶ ofxKinect, ofxMidi etc..
- ▶ 定型の作業、複雑な操作を単純化
- ▶ ofxQuadWrap、ofxControlPanel etc...

Addonsとは?

コンピュータ・ビジョン / AR
ofxOpenCv, ofxARToolkitPlus

シミュレーション / 物理計算 / 流体力学
ofxBox2d, ofxMSAFluid, ofxMSAPhysics, ofxRuiPhusics2d

音響
ofxMidi, ofxOsc, ofxSoundObj, ofxSuperCollider

デバイス / ハードウェア
ofxiPhone, ofxSuddenMotion

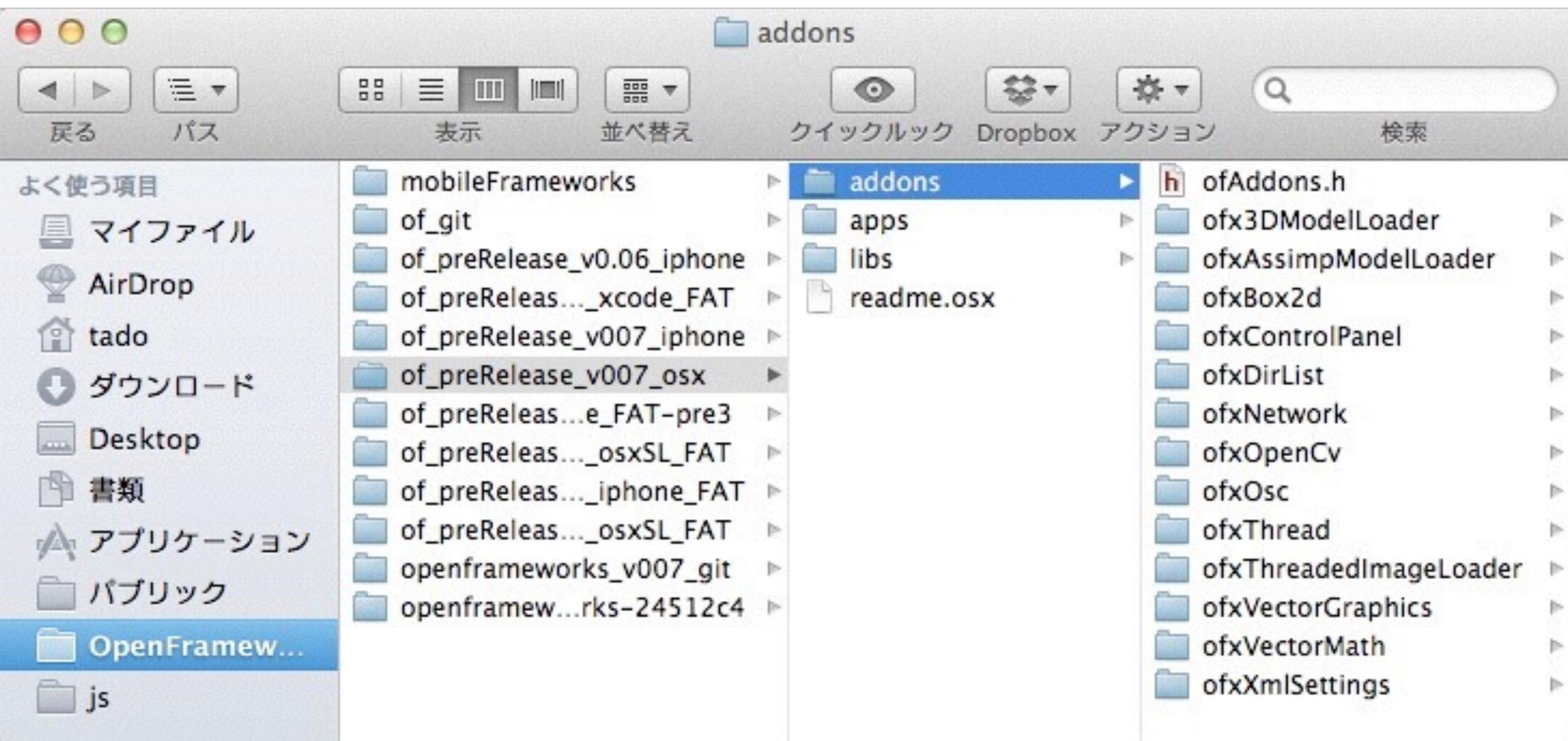
Addonsとは？

- ▶ 最初からいくつかのアドオンが付属してくる

アドオンの名称	説明
ofxDirList	ディレクトリの項目の一覧を生成
ofxXmlSettings	アプリケーションの設定をXML形式で保存、読み込み
ofxOsc	Open Sound ControlをOpenFrameworksで使用する
ofxOpenCv	画像処理・画像認識用のC言語ライブラリOpenCVを使用できるようにする
ofxNetwork	ネットワーク通信のプロトコル、TCPとUDPを使用可能にする、マルチキャストにも対応
ofxThred	クロスプラットフォームでスレッドの管理を実現
ofxVectorGraphics	OpenFrameworksからPostscriptを生成し出力する
ofx3dModelLoader	3ds形式の3DモデルをOpenFrameworksに読みこむ

Addonsとは？

- ▶ アドオンがインストールされている場所
- ▶ 《oFをインストールしたフォルダ》 /addons/



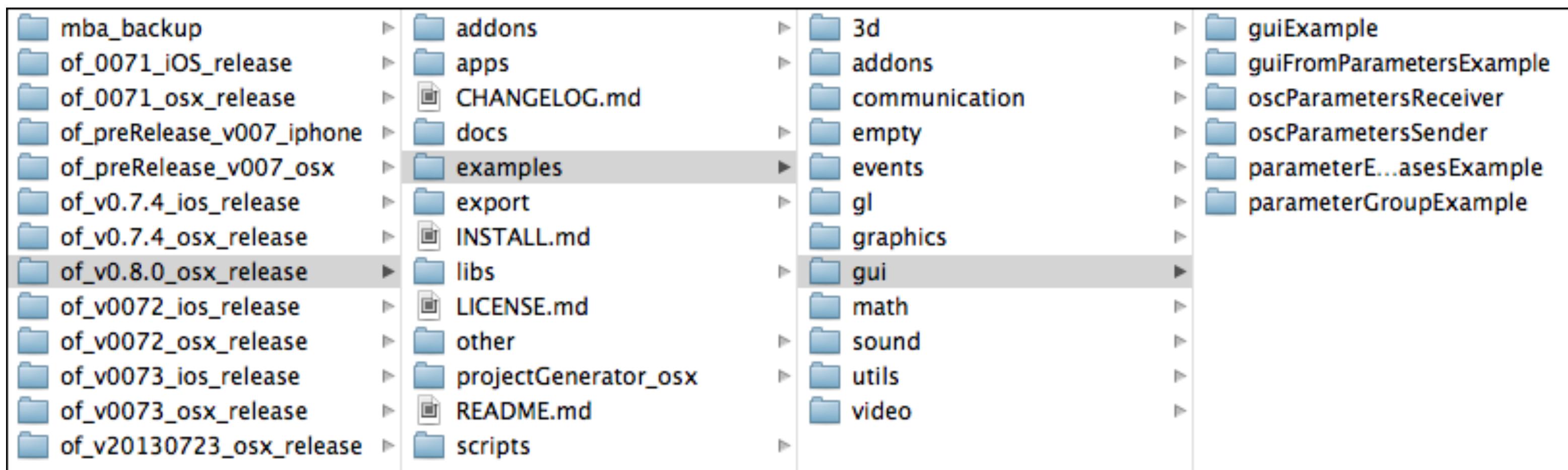
ofxGui

ofxGui

- ▶ まずは手始めに簡単なaddonから
- ▶ 簡単にプロジェクトにGUIを追加可能

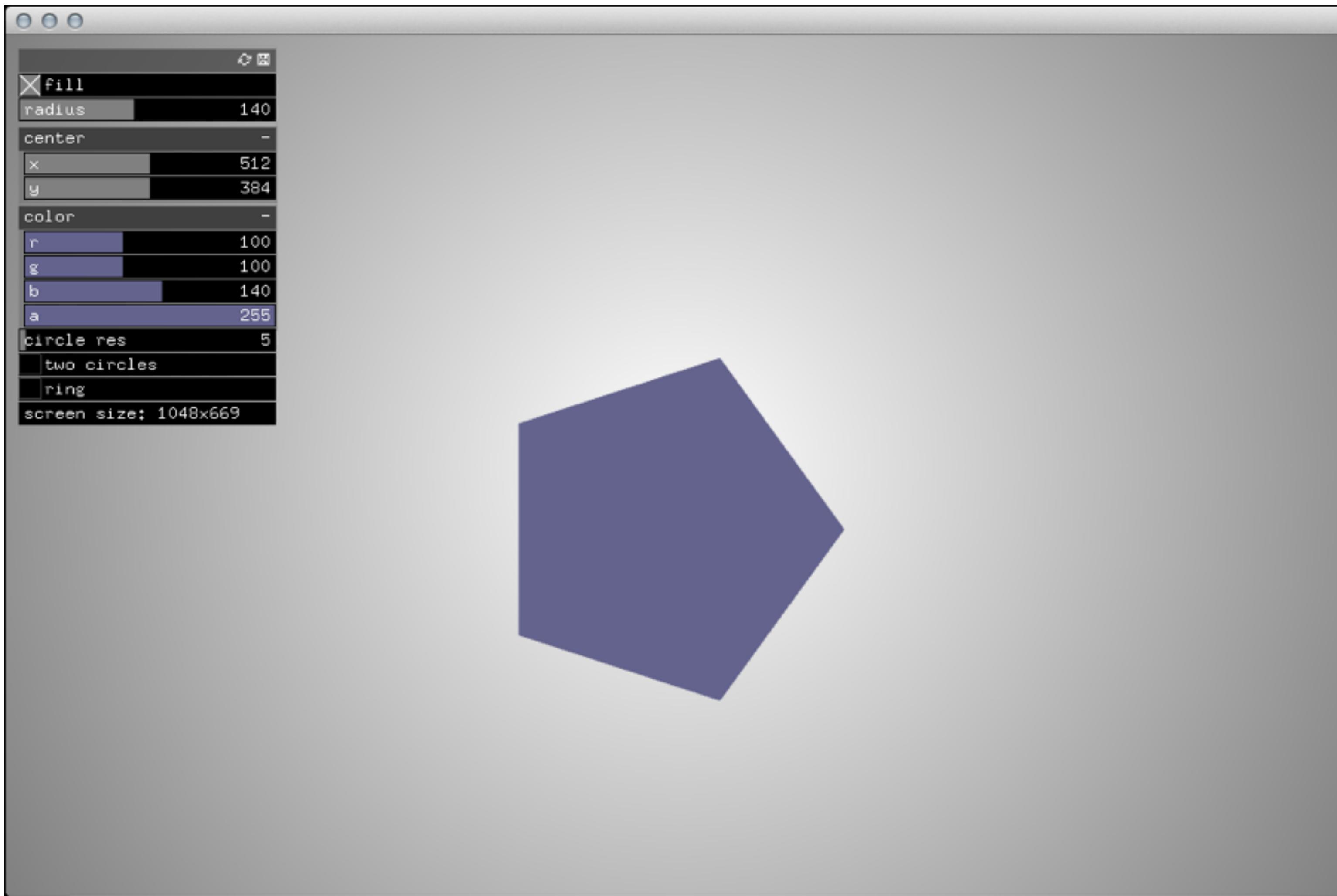
ofxGui

- ▶ Addon の使用法を知るにはサンプルをみるのが一番!
- ▶ Addon の多くにはサンプルが付属している
- ▶ ofxGuiの場合は 「of_v0.8.0_osx_release/examples/gui/」 に大量に掲載 (オフィシャル感!!)



ofxGui

- ▶ まずは基本の「guiExample」をみてみる



ofxGui

- ▶ 「guiExample」に含まれる様々なGUIの機能
- ▶ 数値のスライダー (int, float)
- ▶ カラー設定スライダー
- ▶ 位置 (ofVec2f)
- ▶ トグルボタン
- ▶ ボタン
- ▶ ... etc.

ofxGui

▶ ofxGuiで使用できる機能いろいろ

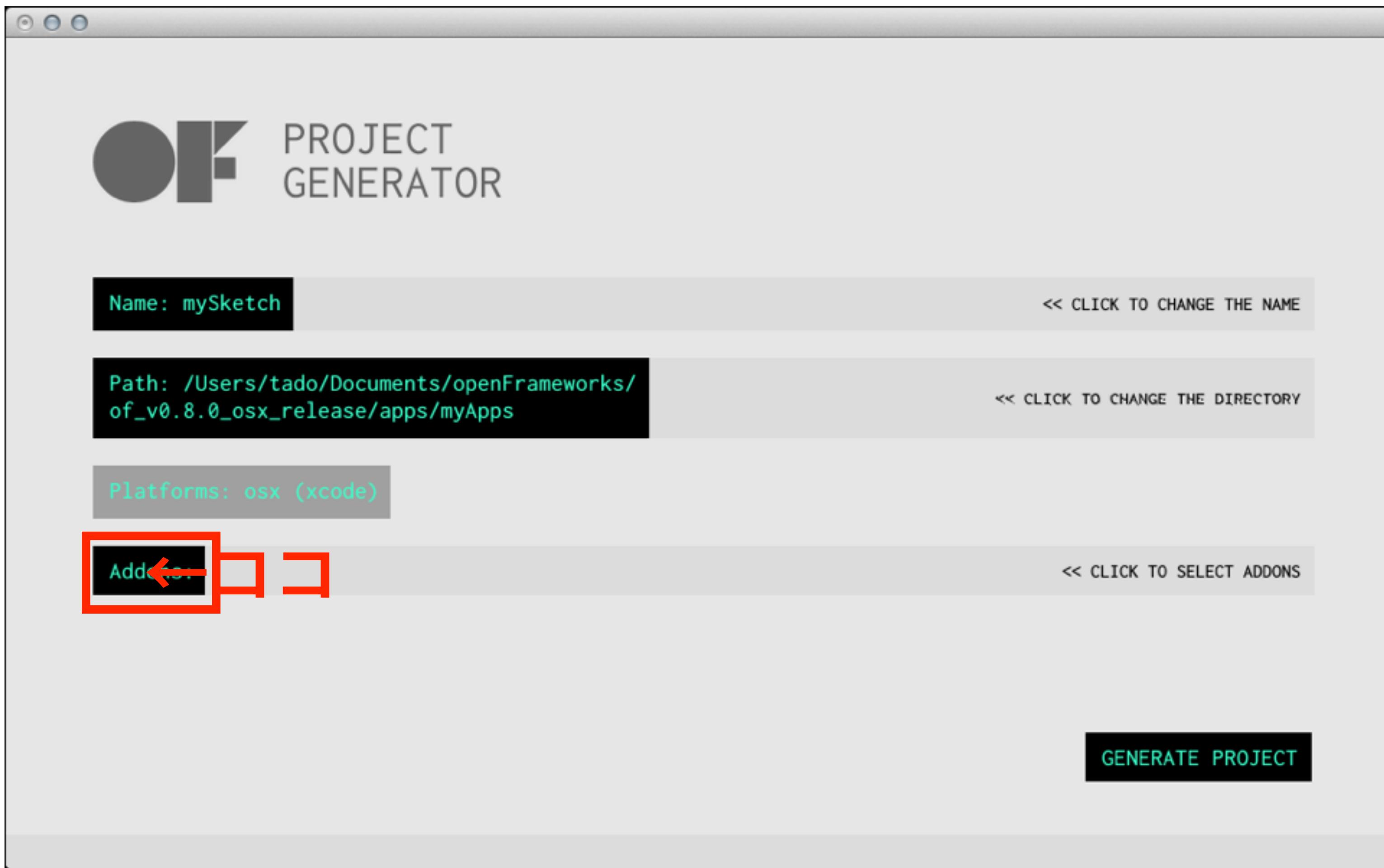
アドオンの名称	説明
ofxIntSlider	整数型 (int) のスライダー
ofxFLOATSlider	浮動小数点型 (float) のスライダー
ofxVec2Slider	2次元ベクトルのスライダー
ofxColorSlider	カラー生成スライダー
ofxButton	ボタン
ofxToggle	トグルスイッチ
ofxLabel	ラベル (テキスト表示)
ofxPanel	GUIの外枠

ofxGui

- ▶ 簡単なプログラムで使い方をマスター
- ▶ まずは新規プロジェクトの生成方法から
- ▶ ProjectGeneratorの設定方法から

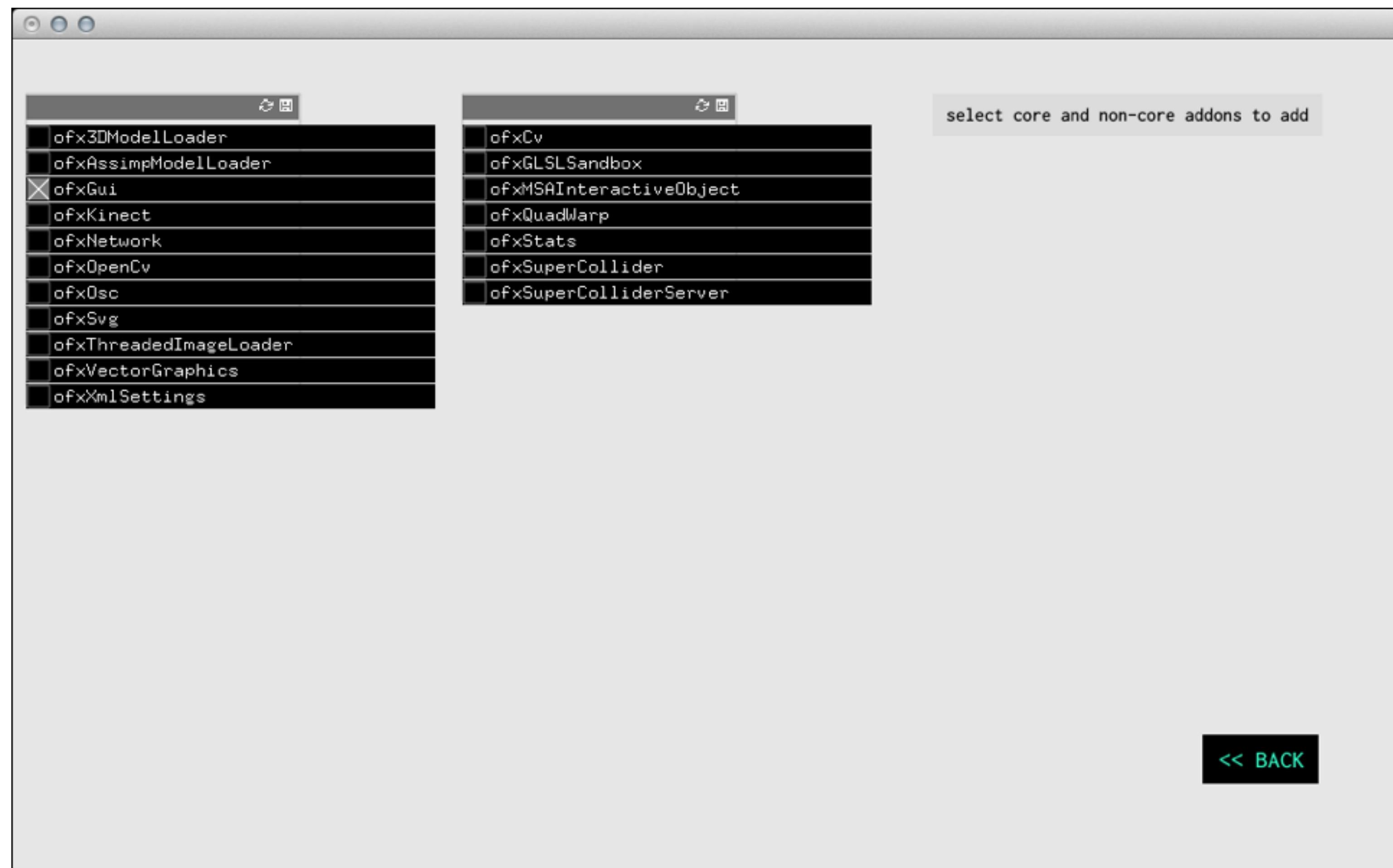
ofxGui

- ▶ プロジェクト名を設定したら、「Addons」ボタンを押す



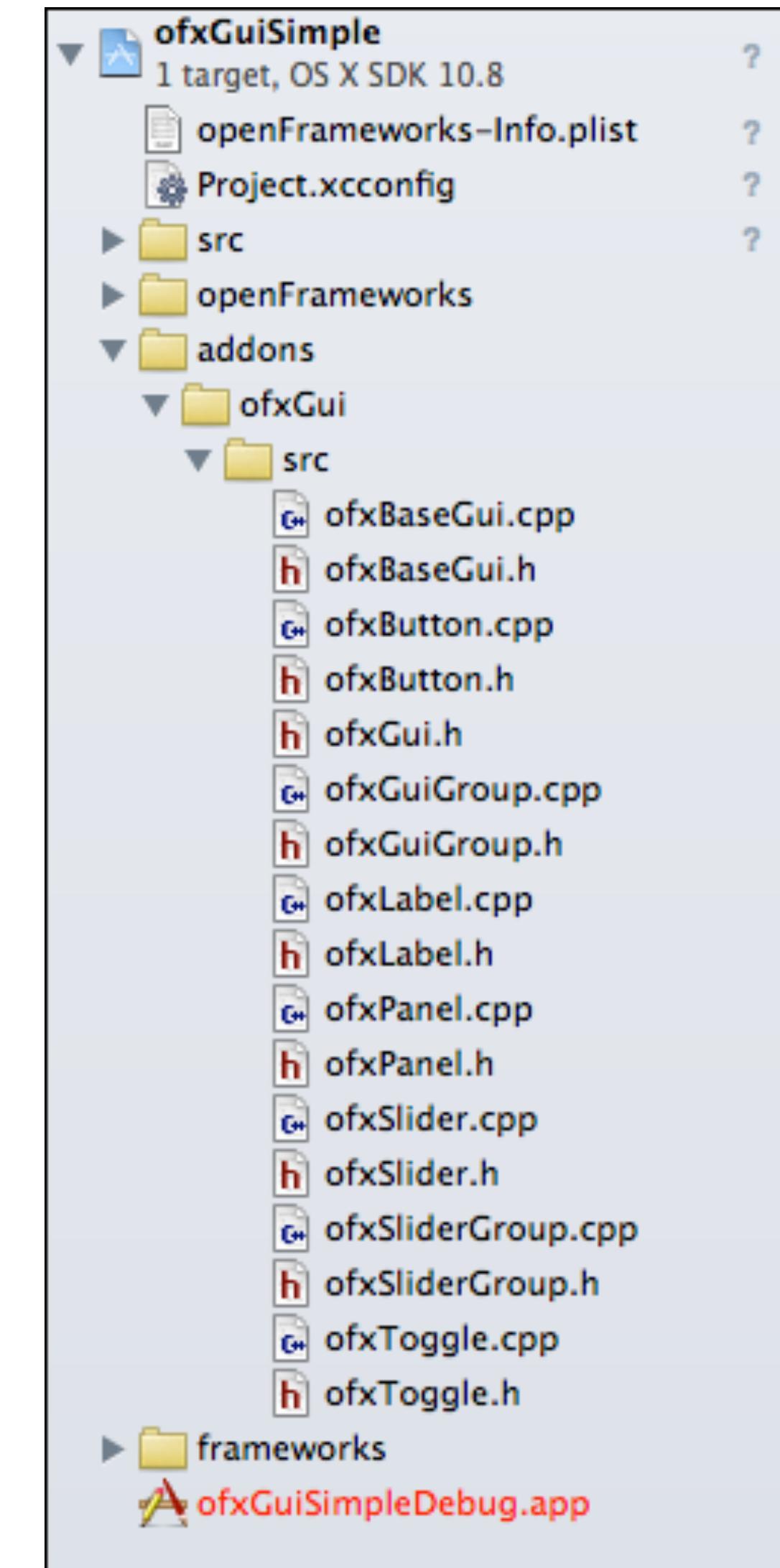
ofxGui

- ▶ ofxGuiのチェックボックスをチェックして戻る
- ▶ GENERATE PROJECT ボタンを押して完了



ofxGui

- ▶ プロジェクトを開いてみる
- ▶ addonsフォルダにAddonsが追加されているはず



ofxGui

- ▶ 以下のようなシンプルなGUI操作を実現したい
- ▶ 画面に円を描く
- ▶ 円の半径をスライダーで調整
- ▶ 円の色もスライダーで調整

ofxGui

- ▶ まず始めに、testApp.h の冒頭で ofxGui を読み込む

```
#pragma once

#include "ofMain.h"
#include "ofxGui.h" ←ココ

class testApp : public ofBaseApp{

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

    void keyPressed(int key);
    void keyReleased(int key);
    void mouseMoved(int x, int y );
    void mouseDragged(int x, int y, int button);
    void mousePressed(int x, int y, int button);
    void mouseReleased(int x, int y, int button);
    void windowResized(int w, int h);
    void dragEvent(ofDragInfo dragInfo);
    void gotMessage(ofMessage msg);

};
```

ofxGui

- ▶ 必要なパートを全てインスタンス化

```
#pragma once

#include "ofMain.h"
#include "ofxGui.h"

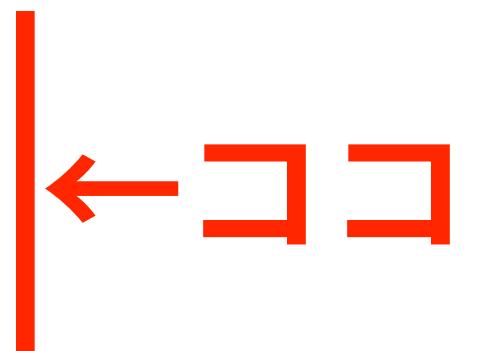
class testApp : public ofBaseApp{

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

    ...

    ofxPanel gui;
    ofxFLOATSlider radius;
    ofxCOLORSlider color;
    ofxVec2Slider position;

};
```



ofxGui

- ▶ まず、`gui.setup()` でGUI生成
- ▶ そこへ、スライダーの初期設定を追加していく
- ▶ `gui.add(スライダー.setup("名前", 初期値, 最小値, 最大値));`

ofxGui

▶ testApp.cpp 必要なパートを全てインスタンス化

```
#include "testApp.h"

//-----
void testApp::setup(){
    ofSetFrameRate(60);
    ofBackground(127);
    ofSetCircleResolution(32);

    // colorの初期値、最小値、最大値を設定
    ofColor initColor = ofColor(0, 127, 255, 255);
    ofColor minColor = ofColor(0,0,0,0);
    ofColor maxColor = ofColor(255,255,255,255);

    // positionの初期値、最小値、最大値を設定
    ofVec2f initPos = ofVec2f(ofGetWidth()/2, ofGetHeight()/2);
    ofVec2f minPos = ofVec2f(0, 0);
    ofVec2f maxPos = ofVec2f(ofGetWidth(), ofGetHeight());

    gui.setup();
    gui.add(radius.setup("radius", 200, 0, 400));
    gui.add(color.setup("color", initColor, minColor, maxColor));
    gui.add(position.setup("position", initPos, minPos, maxPos));
}
```

ofxGui

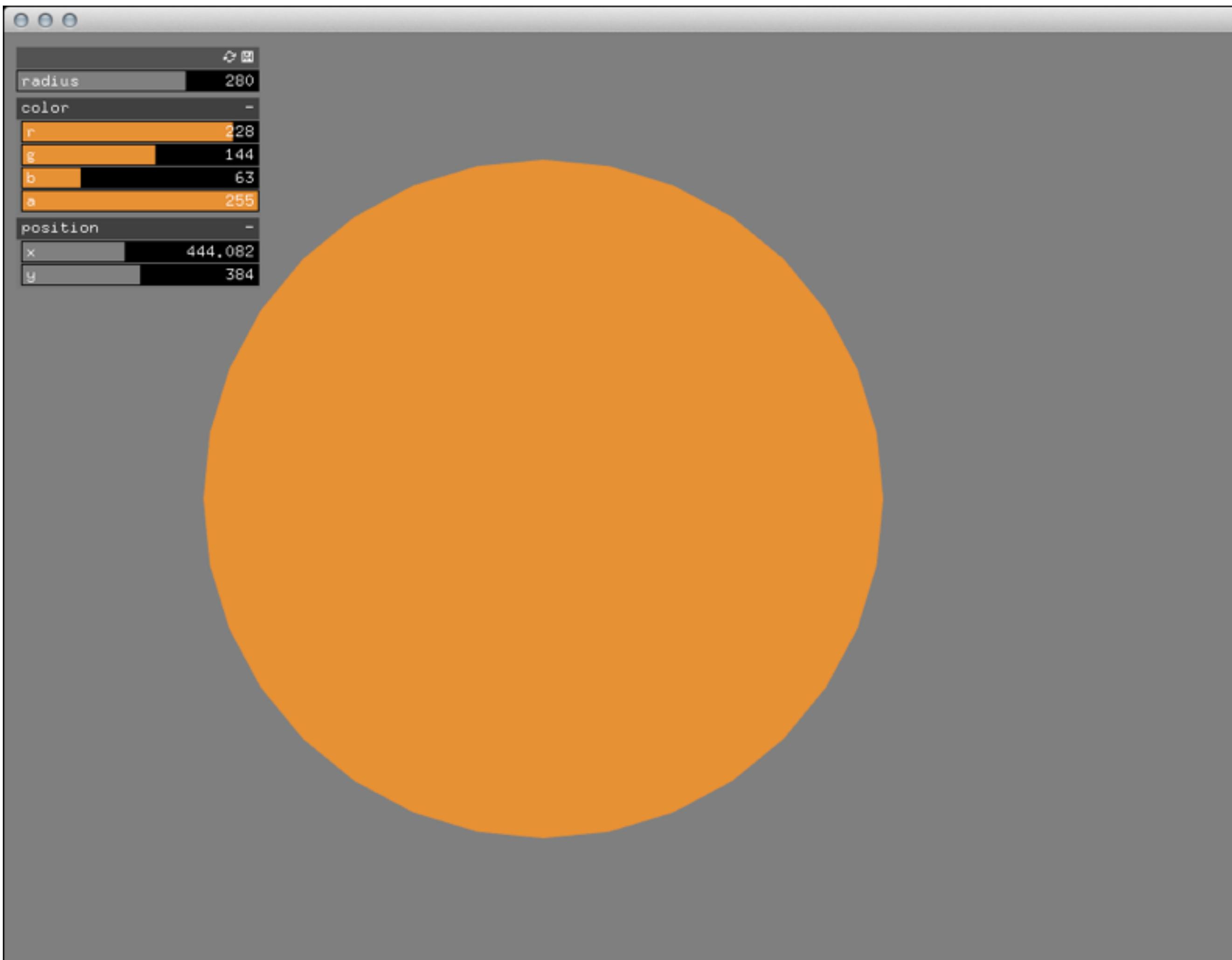
- ▶ あとは、その値を描画に適用するのみ(簡単!!)

```
void testApp::draw(){
    // パラメータを適用して円を描画
    ofSetColor(color);
    ofCircle(ofVec2f(position), radius);

    // GUIを表示
    gui.draw();
}
```

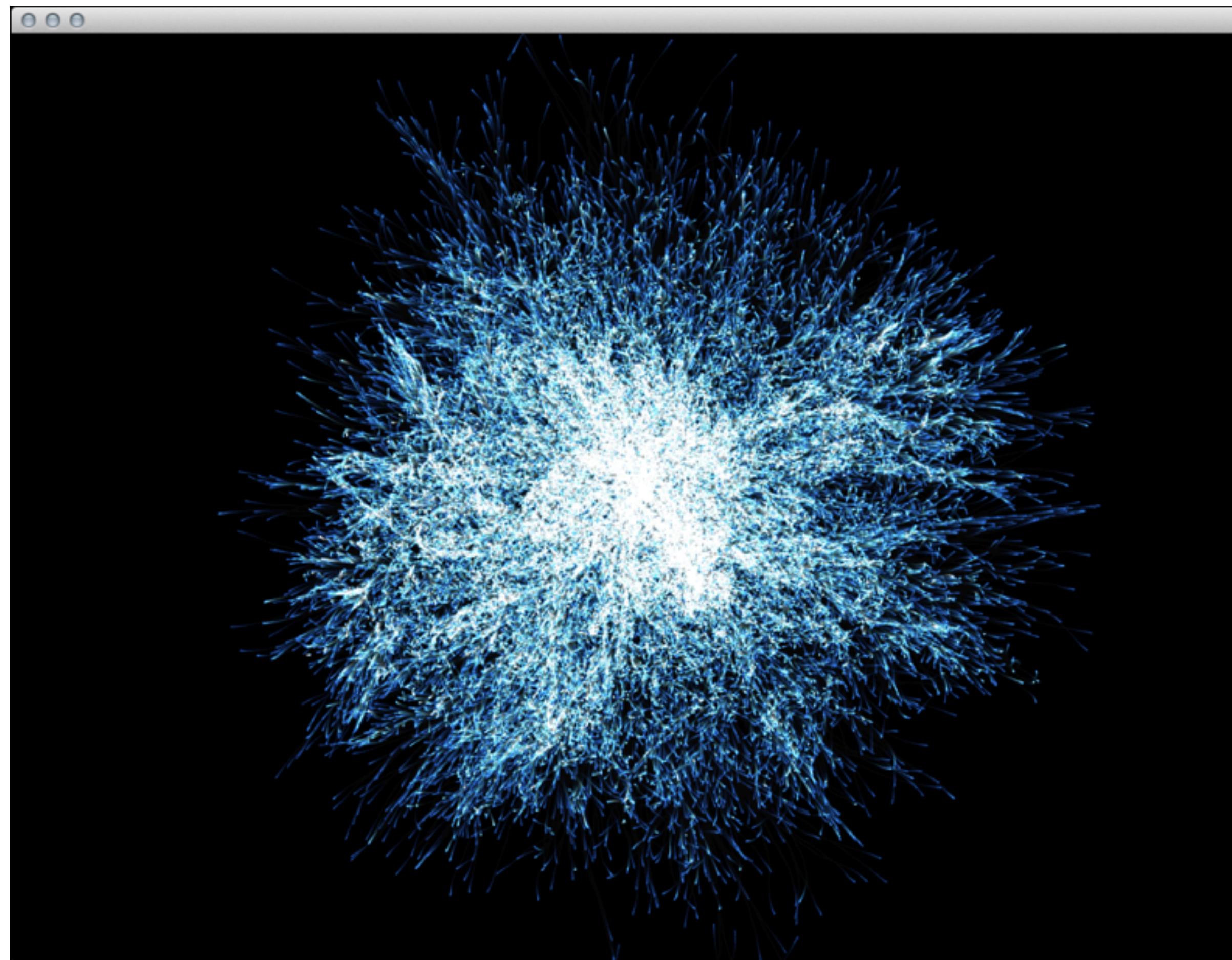
ofxGui

▶ 実行結果



ofxGui

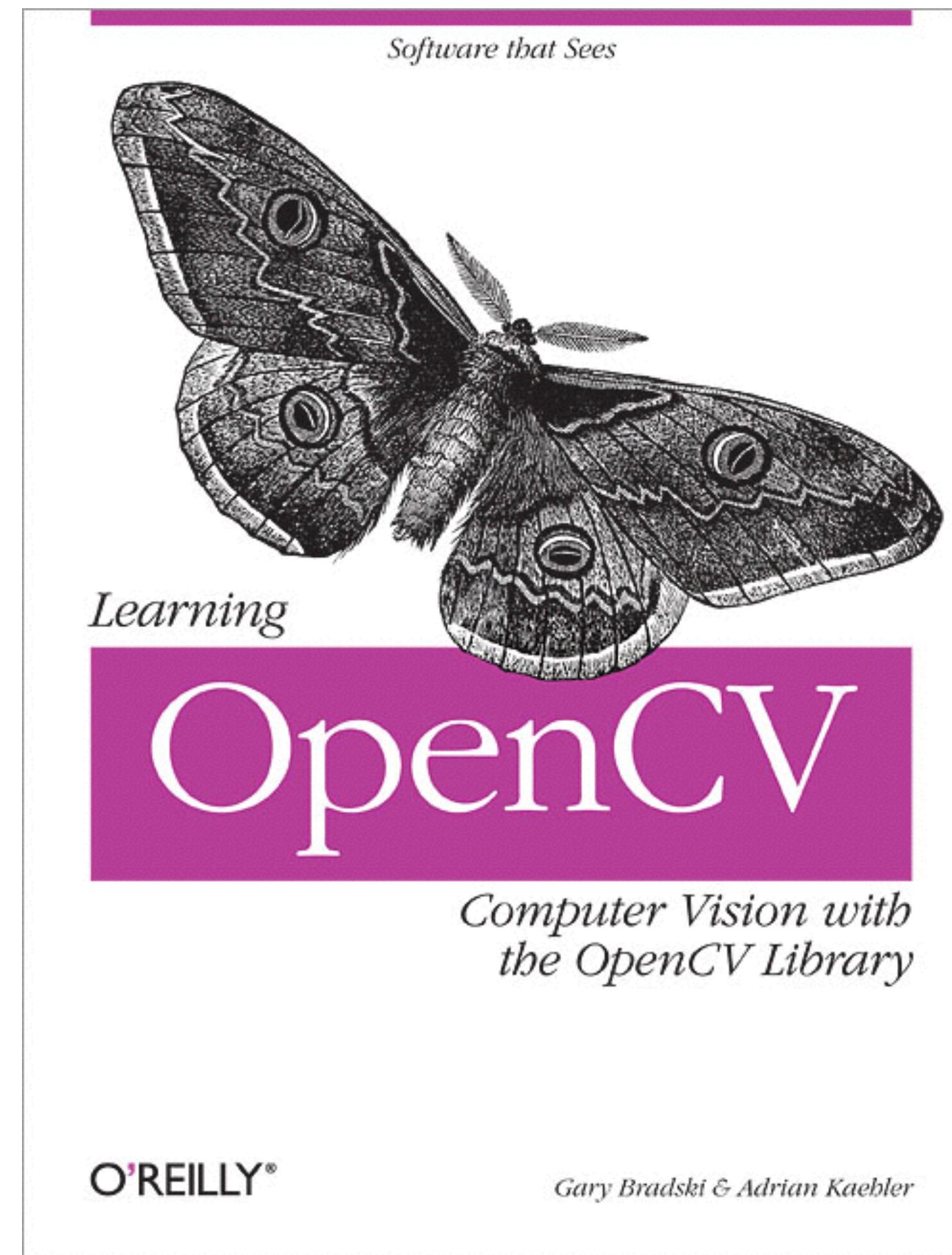
- ▶ ofxGuiのより実践的な練習
- ▶ 先週のVector FieldのサンプルのパラメータをGUIで操作



OpenCVとは
ofxOpenCVについて

OpenCVとは - ofxOpenCVについて

▶ OpenCV



OpenCVとは - ofxOpenCVについて

- ▶ OpenCVとは?
 - ▶ OpenCV = Open Computer Vision Library
 - ▶ 米 Intel 社で開発された画像処理・画像認識用のC言語ライブラリ
 - ▶ オープンソース、商用・非商用を問わず無料
 - ▶ 静止画にも動画にも対応
- ▶ ofxOpenCv
 - ▶ OpenCVをアドオンとして使用できるようにしたもの

OpenCVとは - ofxOpenCVについて

- ▶ openFrameworksとOpenCVを組みあわせた作品は、数多く存在する
- ▶ カメラの映像によるインタラクションは、市販の機材で構築可能で、かつメンテナンスも容易
- ▶ 個人制作する作品にも転用可能

OpenCVを活用した作品例

- ▶ YesYesNo “Night Lights”
- ▶ <http://yesyesno.com/night-lights>



OpenCVを活用した作品例

- ▶ Graffiti Research Lab. “L.A.S.E.R. Tag”
- ▶ <http://graffitiresearchlab.com/projects/laser-tag/>



OpenCVを活用した作品例

- ▶ The Manual Input Sessions, Tmema (Golan Levin & Zachary Lieberman) 2004
- ▶ <http://www.flong.com/projects/mis/>



OpenCVを活用した作品例

- ▶ The Eye Writer
- ▶ <http://www.eyewriter.org/>



OpenCVを活用した作品例

- ▶ Chris O'shea “Hand from Above”
- ▶ <http://www.chrisoshea.org/hand-from-above>



OpenCVを活用した作品例

- ▶ Chris O'shea “Audience”
- ▶ <http://www.chrisoshea.org/audience>



OpenCVを活用した作品例

- ▶ Golan Levin “Double-Taker (Snout)”
- ▶ <http://www.flong.com/projects/snout/>

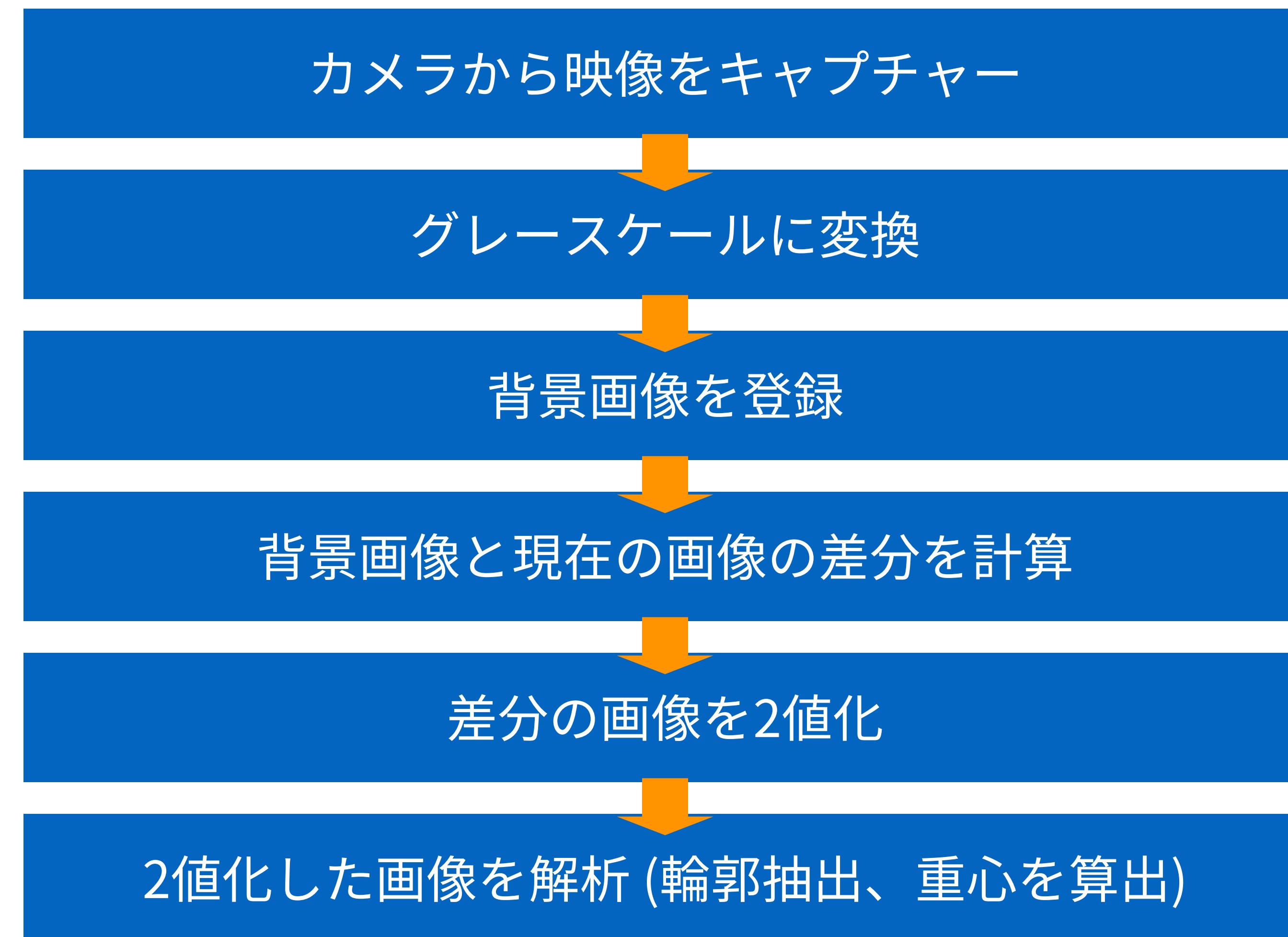


OpenCVで物体を認識する

- ▶ これらの作品に共通する技術
 - ▶ カメラで撮影された映像から、動いている物体を検出し、その位置や輪郭を抽出する
 - ▶ 「ロボットの眼」をプログラミングしている
-
- ▶ OpenCVを活用すると、この物体の認識が簡単に可能となる

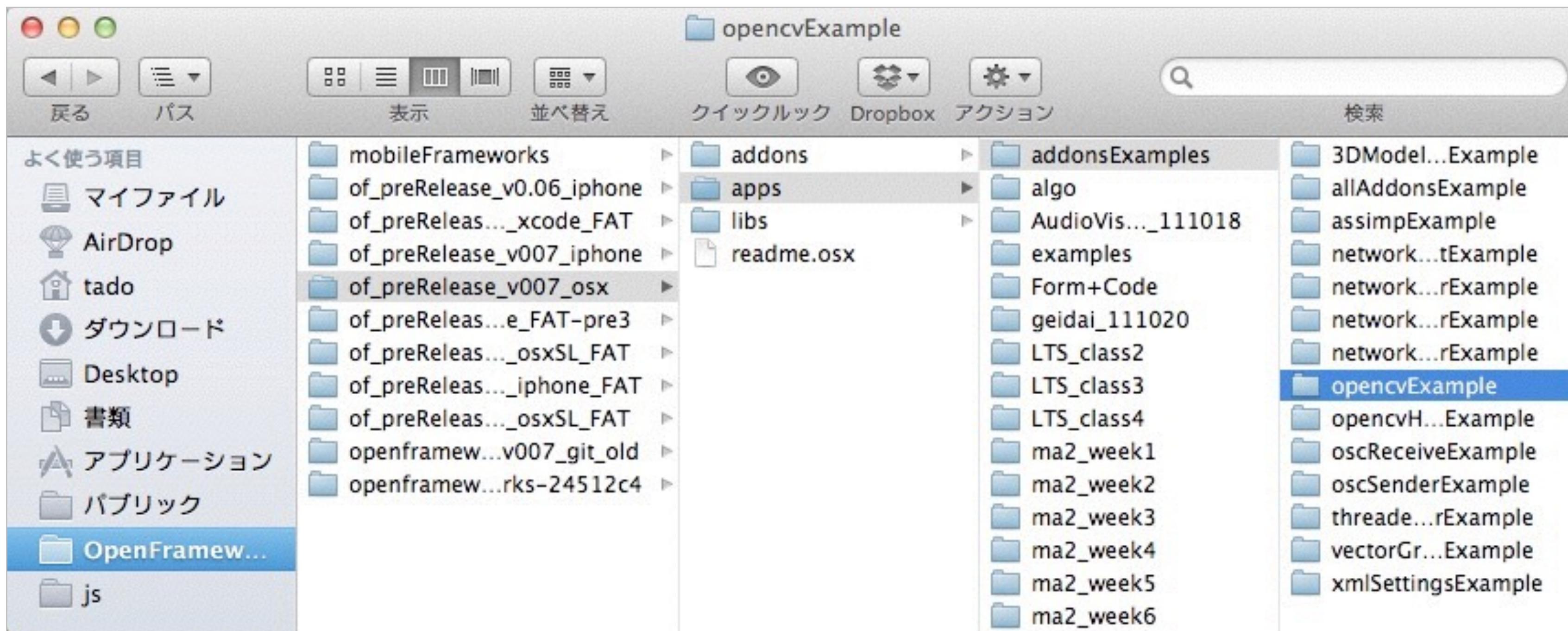
OpenCV で物体を認識する

- ▶ 映像から物体の形態を認識するアルゴリズム



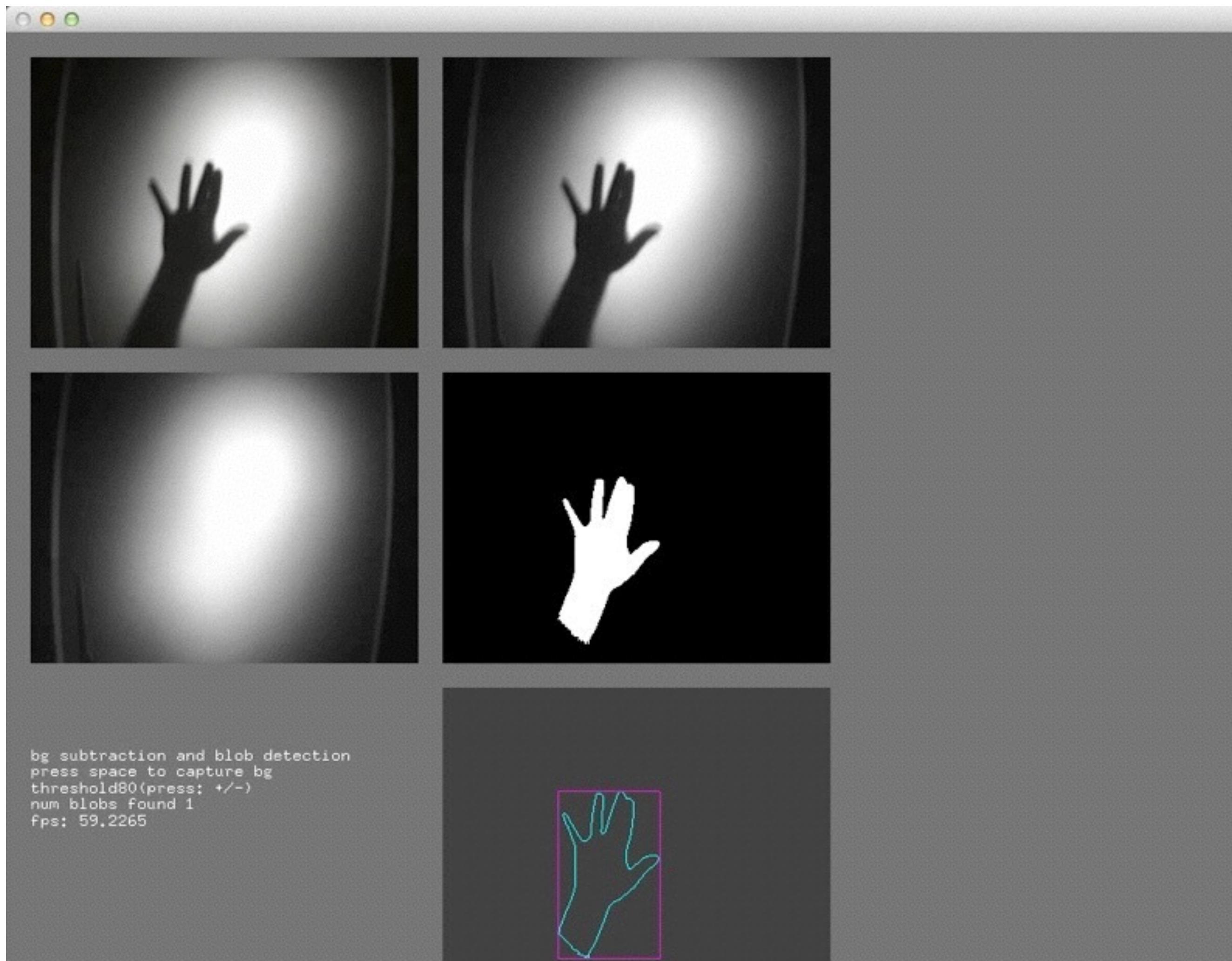
OpenCV で物体を認識する

- ▶ apps > addonsExamples > opencvExample に物体の輪郭を抽出するまでの、わかりやすいサンプルが掲載
- ▶ まずは、この基本プログラムを読み解いていきたい



ofxOpenCv 基本

- ▶ 実行結果：動いている物体の輪郭線が抽出される



OpenCVもう一つの実装 ofxCv

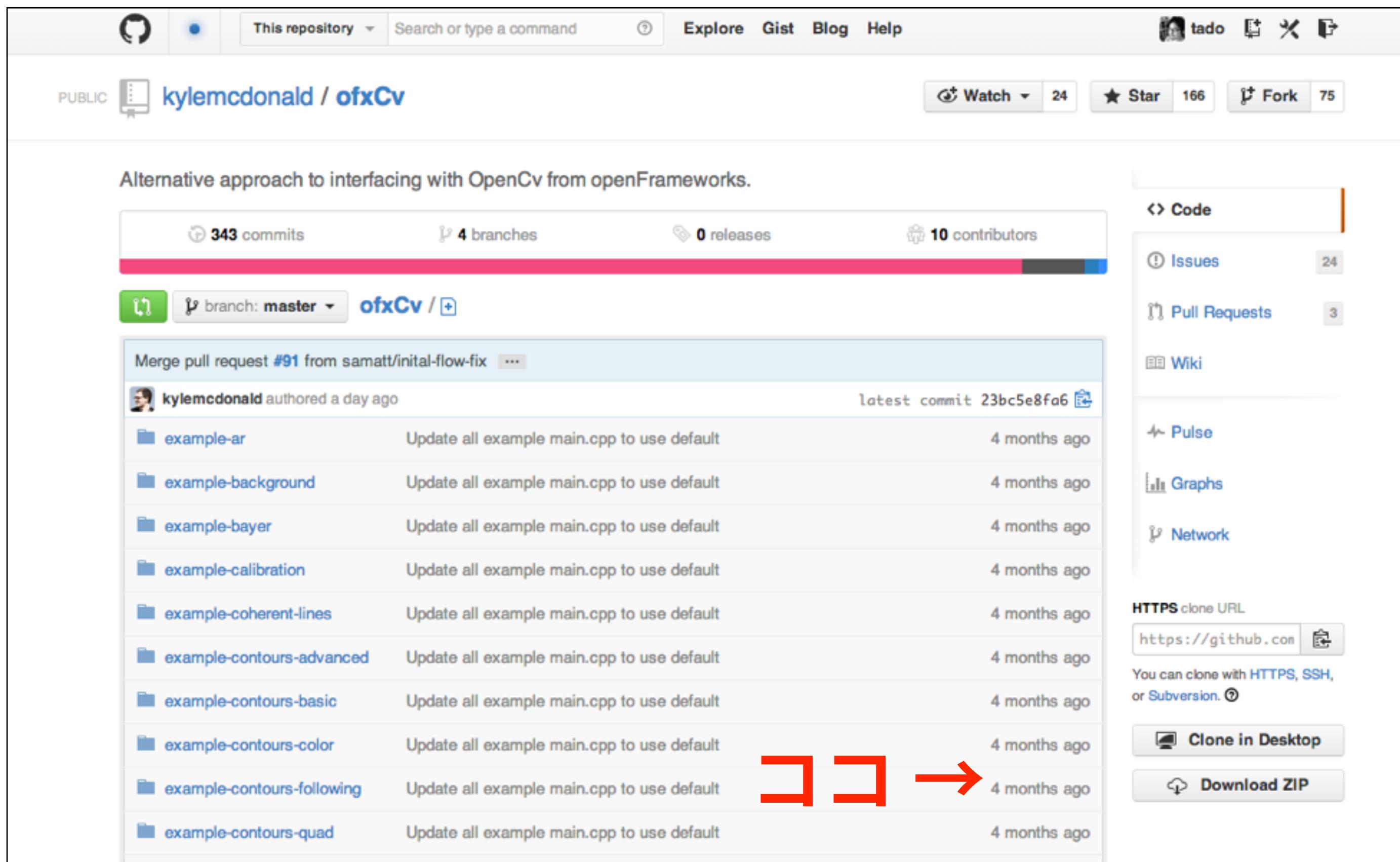
OpenCVもう一つの実装 - ofxCv

- ▶ openFrameworksには、ofxOpenCvの他に、OpenCVを使用するためのアドオンがもう一つ存在する
- ▶ → ofxCv

- ▶ ofxCv
- ▶ <https://github.com/kylemcdonald/ofxCv>
- ▶ Kyle McDonald氏によるアドオン「CVおじさん!」
- ▶ openFrameworksでOpenCVをより簡単に、よりパワフルに利用できる環境を目指している
- ▶ OpenCVとoFのより緊密な連携

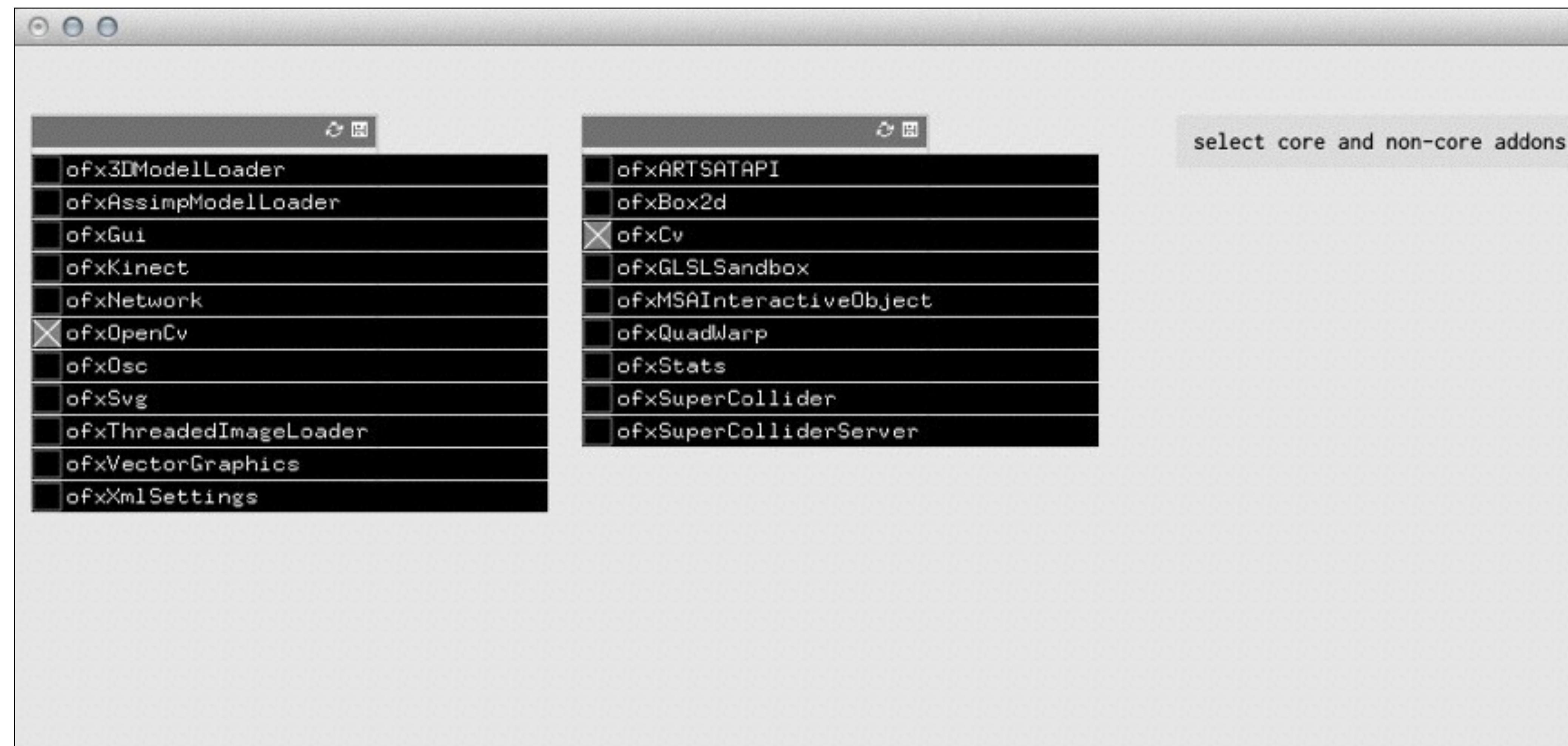
OpenCVもう一つの実装 - ofxCv

- ▶ ofxCvのインストール
- ▶ GitHubからZip形式でダウンロード
- ▶ addonsフォルダに格納



ofxCv 新規プロジェクトの作成

- ▶ ofxCvの新規プロジェクトを作成するには、ProjectGeneratorでofxOpenCVとofxCvを選択して、Generateする



輪郭抽出をofxCvで実装

- ▶ 先程、ofxOpenCVで実現した、映像の中の輪郭を抽出するプログラムを、ofxCvでも作成してみる
- ▶ ofxCvでは、とても短いプログラムで実現可能(エレガント!)

輪郭抽出をofxCvで実装

▶ testApp.h

```
#pragma once

#include "ofMain.h"
#include "ofxCv.h"

class testApp : public ofBaseApp {
public:
    void setup();
    void update();
    void draw();

    ofVideoGrabber cam;
    ofxCv::ContourFinder contourFinder;
};
```

輪郭抽出をofxCvで実装

▶ testApp.cpp

```
#include "testApp.h"

using namespace ofxCv;
using namespace cv;

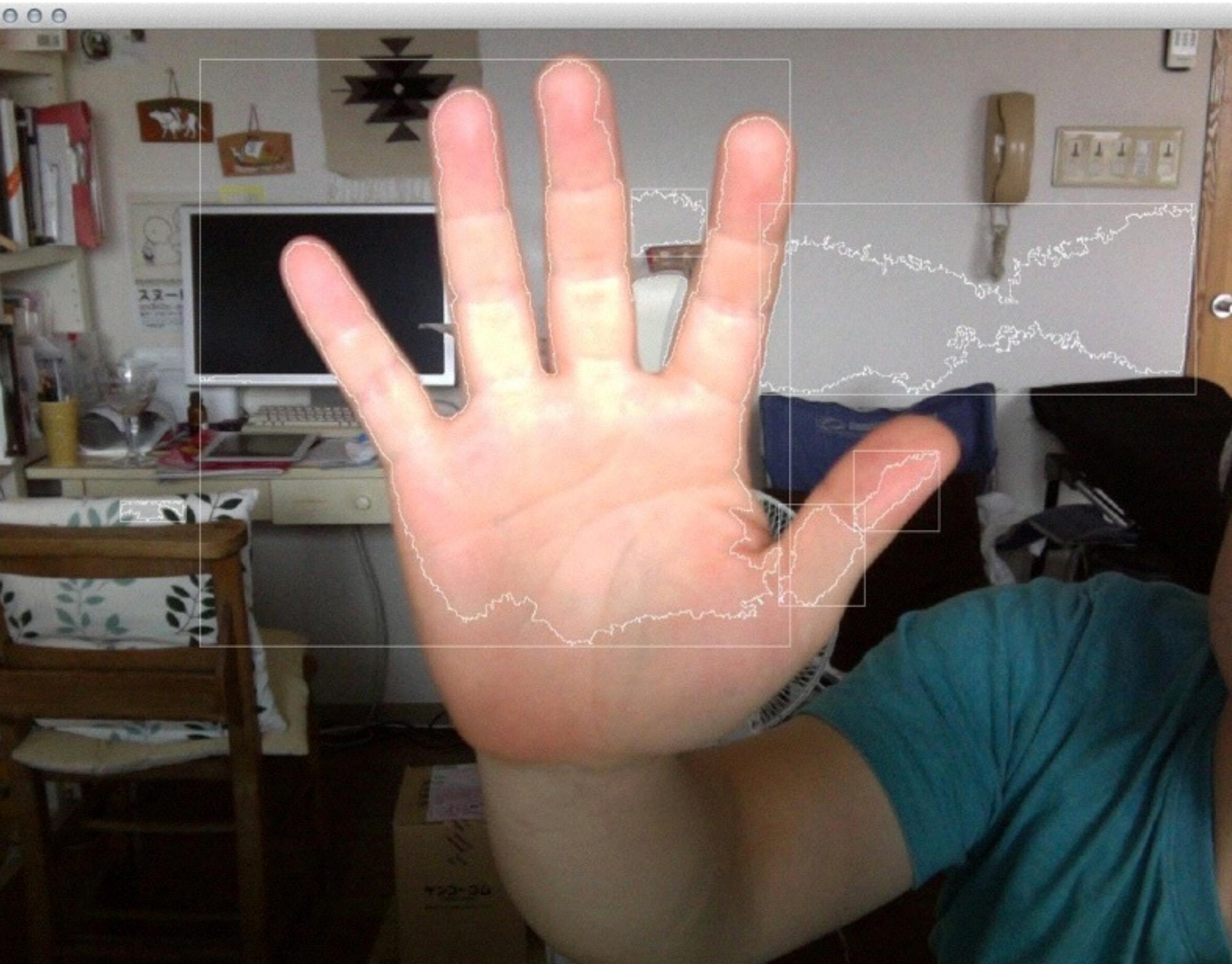
void testApp::setup() {
    cam.initGrabber(ofGetWidth(), ofGetHeight());
    contourFinder.setMinAreaRadius(10);
    contourFinder.setMaxAreaRadius(200);
}

void testApp::update() {
    cam.update();
    if(cam.isFrameNew()) {
        contourFinder.setThreshold(ofMap(mouseX, 0, ofGetWidth(), 0, 255));
        contourFinder.findContours(cam);
    }
}

void testApp::draw() {
    ofSetColor(255);
    cam.draw(0, 0);
    contourFinder.draw();
}
```

輪郭抽出をofxCvで実装

- ▶ 完成! マウスのX座標で、閾値を調整

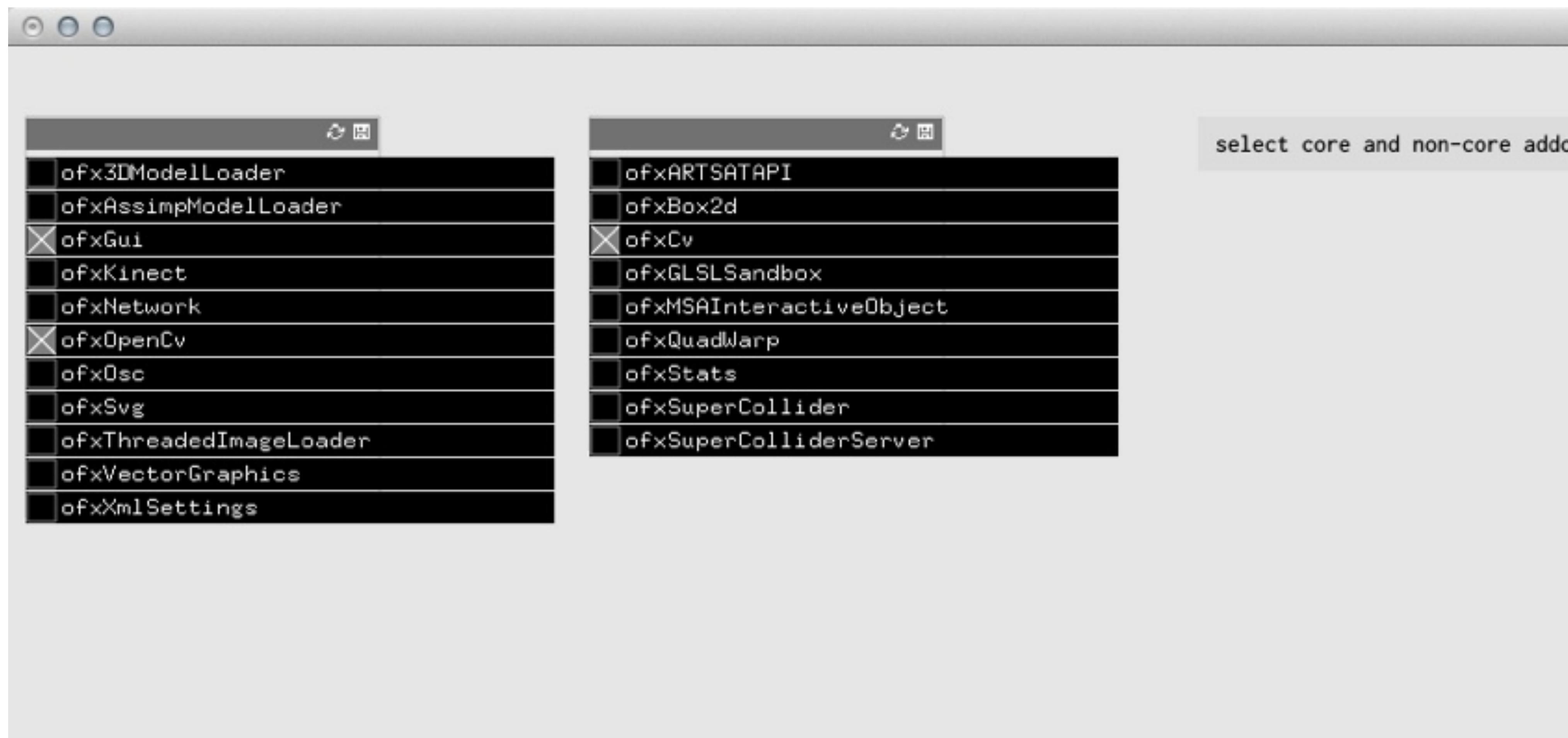


ofxCv + ofxGui

- ▶ プログラムをもっと実用的(作品に利用可能)なものに
 - ▶ 先週とりあげた、ofxGuiと組合せてみる
-
- ▶ 単純にカメラの画像の輪郭抽出ではなく、背景画像を登録して、その差分イメージの輪郭を抽出
 - ▶ 背景のリセットもGUIからできるように

ofxCv + ofxGui

- ▶ まずは、ofxOpenCV、ofxCv、ofxGuiを選択してプロジェクトを生成



ofxCv + ofxGui

▶ testApp.h

```
#pragma once

#include "ofMain.h"
#include "ofxCv.h"
#include "ofxGui.h"

class testApp : public ofBaseApp {
public:
    void setup();
    void update();
    void draw();
    void resetBackgroundPressed();

    ofVideoGrabber cam;
    ofxCv::ContourFinder contourFinder;
    ofxCv::RunningBackground background;
    ofImage thresholded;

    ofxPanel gui;
    ofxFloatSlider bgThresh; // 背景差分の閾値
    ofxFloatSlider contourThresh; // 輪郭抽出の閾値
    ofxButton resetBackgroundButton; // 背景リセットボタン
};
```

ofxCv + ofxGui

▶ testApp.cpp

```
#include "testApp.h"

using namespace ofxCv;
using namespace cv;

void testApp::setup() {
    // CV初期設定
    cam.initGrabber(ofGetWidth(), ofGetHeight());
    contourFinder.setMinAreaRadius(10);
    contourFinder.setMaxAreaRadius(200);
    background.setLearningTime(900);
    background.setThresholdValue(20);

    // GUI
    resetBackgroundButton.addListener(this, &testApp::resetBackgroundPressed);
    gui.setup();
    gui.add(bgThresh.setup("background thresh", 50, 0, 100));
    gui.add(contourThresh.setup("contour finder thresh", 127, 0, 255));
    gui.add(resetBackgroundButton.setup("reset background"));
}
```

ofxCv + ofxGui

▶ testApp.cpp

```
void testApp::update() {
    cam.update();
    if(cam.isFrameNew()) {
        // 背景差分画像を生成
        background.setThresholdValue(bgThresh);
        background.update(cam, thresholded);
        thresholded.update();

        // 背景差分画像の輪郭抽出
        contourFinder.setThreshold(contourThresh);
        contourFinder.findContours(thresholded);
    }
}

void testApp::draw() {
    // 差分画像を描画
    ofSetColor(255);
    thresholded.draw(0, 0);
    // 輪郭抽出結果を描画
    ofSetColor(255, 0, 0);
    contourFinder.draw();
    // GUI
    gui.draw();
}
```

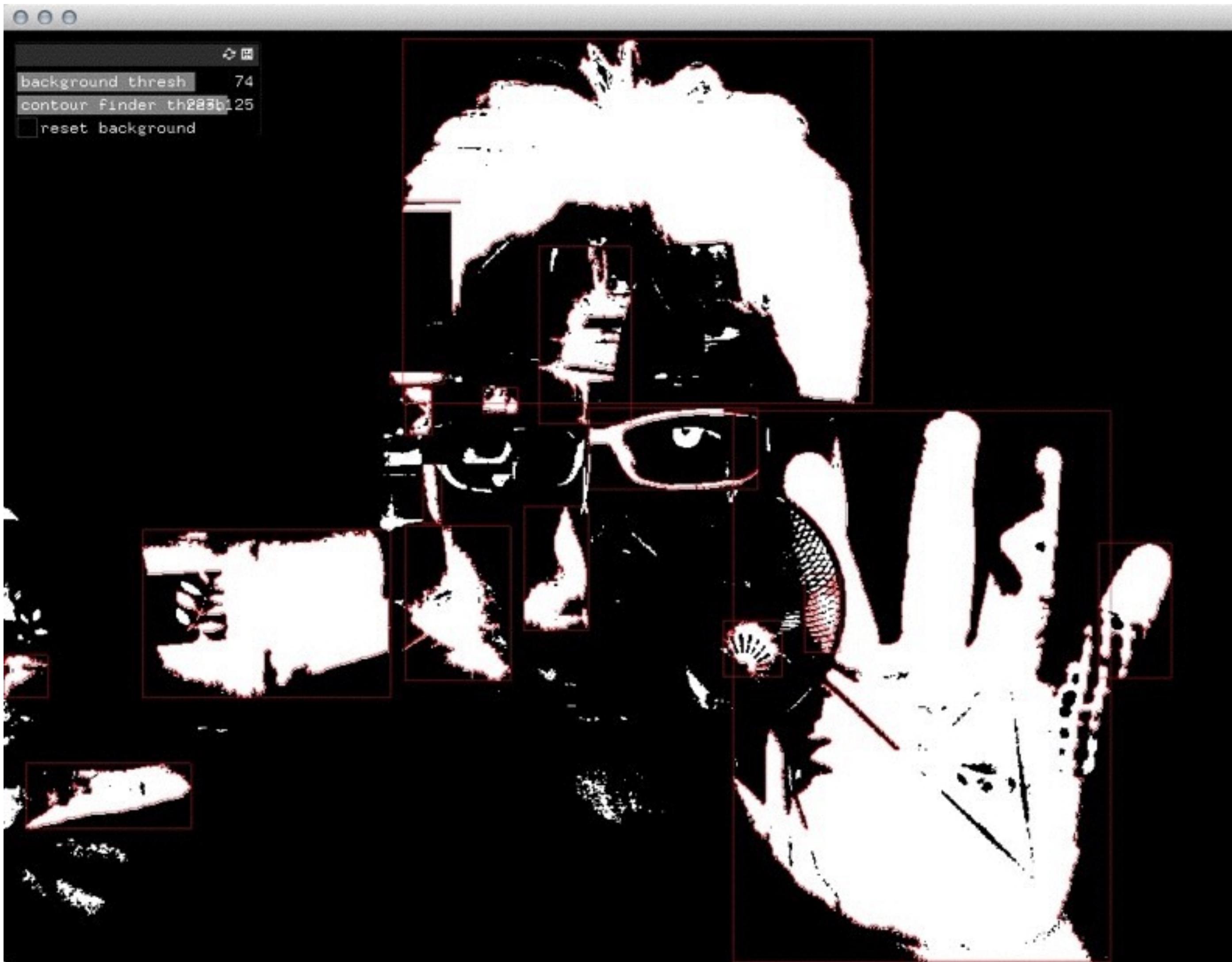
ofxCv + ofxGui

▶ testApp.cpp

```
void testApp::resetBackgroundPressed(){
    background.reset();
}
```

ofxCv + ofxGui

▶ 完成!!



オプティカル・フロー (Optical Flow)

オプティカル・フロー (Optical Flow)

- ▶ オプティカル・フロー
- ▶ 映像内での物体の動きを、ベクトル場で表したもの
- ▶ 参考: http://opencv.jp/sample/optical_flow.html

オプティカル・フロー (Optical Flow)

- ▶ ofxCvでは、オプティカル・フローのための2種類のクラスが用意されている
- ▶ `ofxCv::FlowFarneback`
- ▶ Gunnar Farneback のアルゴリズムを用いた、密なオプティカルフロー
- ▶ `ofxCv::FlowPyrLK`
- ▶ 疎な特徴集合に対するオプティカルフロー
- ▶ 参考: http://opencv.jp/opencv-2svn/cpp/motion_analysis_and_object_tracking.html

オプティカル・フロー (Optical Flow)

▶ testApp.h

```
#pragma once

#include "ofMain.h"
#include "ofxCv.h"
#include "ofxGui.h"

class testApp : public ofBaseApp {
public:
    void setup();
    void update();
    void draw();

    ofVideoGrabber camera;
    ofxCv::FlowFarneback farneback;
    ofxCv::FlowPyrLK pyrLk;
    ofxCv::Flow* curFlow;

    ofxPanel gui;
    ofxFLOATSlider pyrScale;
    ofxINTSlider levels;
    ofxINTSlider winsize;
    ofxINTSlider iterations;
    ofxINTSlider polyN;
    ofxFLOATSlider polySigma;
    ofxINTSlider winSize;
    ofxINTSlider maxLevel;
```

オプティカル・フロー (Optical Flow)

▶ testApp.h

```
ofxFLOATSlider maxFeatures;
ofxFLOATSlider qualityLevel;
ofxFLOATSlider minDistance;

ofxToggle OPTFLOW_FARNEBACK_GAUSSIAN;
ofxToggle useFarneback;

};
```

オプティカル・フロー (Optical Flow)

▶ testApp.cpp

```
#include "testApp.h"

using namespace ofxCv;
using namespace cv;

void testApp::setup() {
    // CV初期設定
    camera.initGrabber(320, 240);

    // GUI
    gui.setup();
    gui.add(pyrScale.setup("pyrScale", .5, 0, 1));
    gui.add(levels.setup("levels", 4, 1, 8));
    gui.add(winsize.setup("levels", 8, 4, 64));
    gui.add(iterations.setup("iterations", 2, 1, 8));
    gui.add(polyN.setup("polyN", 7, 5, 10));
    gui.add(polySigma.setup("polySigma", 1.5, 1.1, 2));
    gui.add(OPTFLOW_FARNEBACK_GAUSSIAN.setup("OPTFLOW_FARNEBACK_GAUSSIAN", false));

    gui.add(useFarneback.setup("useFarneback", true));
    gui.add(winSize.setup("winSize", 32, 4, 64));
    gui.add(maxLevel.setup("maxLevel", 3, 0, 8));

    gui.add(levels.setup("maxFeatures", 200, 1, 1000));
    gui.add(qualityLevel.setup("levels", 0.01, 0.001, .02));
```

オプティカル・フロー (Optical Flow)

▶ testApp.cpp

```
gui.add(minDistance.setup("minDistance", 4, 1, 16));  
  
curFlow = &farneback;  
}  
  
void testApp::update() {  
    camera.update();  
    if(camera.isFrameNew()) {  
        if(useFarneback) { // Farnebackの密なオプティカルフロー  
            curFlow = &farneback;  
            farneback.setPyramidScale(pyrScale);  
            farneback.setNumLevels(levels);  
            farneback.setWindowSize(winsize);  
            farneback.setNumIterations(iterations);  
            farneback.setPolyN(polyN);  
            farneback.setPolySigma(polySigma);  
            farneback.setUseGaussian(OPTFLOW_FARNEBACK_GAUSSIAN);  
        } else { // 画像ピラミッドを利用した、疎なオプティカルフロー  
            curFlow = &pyrLk;  
            pyrLk.setMaxFeatures(maxFeatures);  
            pyrLk.setQualityLevel(qualityLevel);  
            pyrLk.setMinDistance(minDistance);  
            pyrLk.setWindowSize(winSize);  
            pyrLk.setMaxLevel(maxLevel);  
        }  
    }  
}
```

オプティカル・フロー (Optical Flow)

▶ testApp.cpp

```
// オプティカルフローを計算
    curFlow->calcOpticalFlow(camera);
}

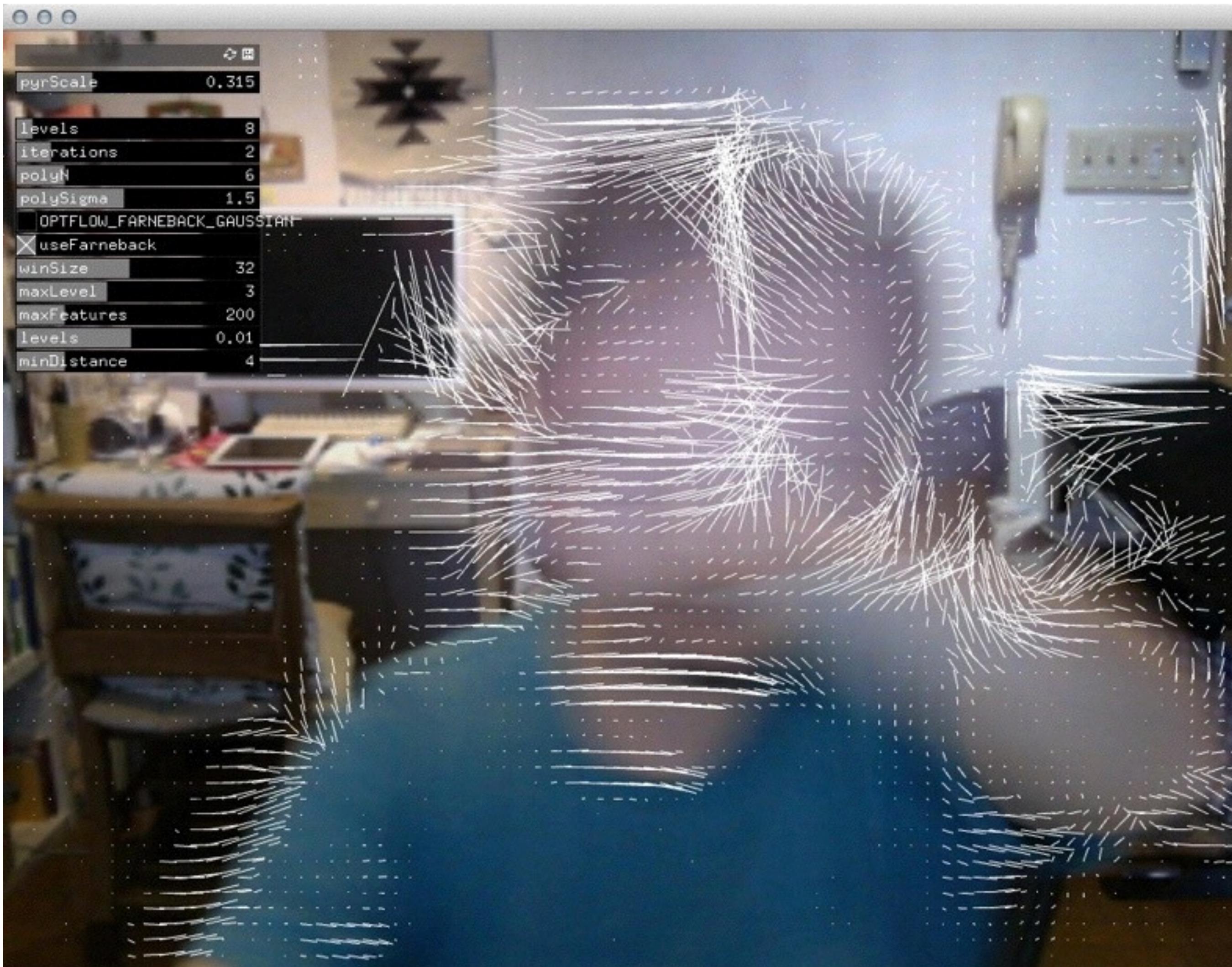
void testApp::draw() {
    ofBackground(0);

    ofSetColor(255);
    camera.draw(0,0,ofGetWidth(),ofGetHeight());
    curFlow->draw(0,0,ofGetWidth(),ofGetHeight());

    gui.draw();
}
```

オプティカル・フロー (Optical Flow)

▶ 完成!!



オプティカル・フロー + パーティクル

オプティカル・フロー + パーティクル

- ▶ オプティカル・フローで生成したベクトル場に、パーティクルを置いてみる
 - ▶ 映像を解析して、舞い散るパーティクルが実現できるはず!!
-
- ▶ Particleクラスは、以前作成したものをそのまま再利用

オプティカル・フロー + パーティクル

▶ testApp.h

```
#pragma once

#include "ofMain.h"
#include "ofxCv.h"
#include "ofxGui.h"
#include "Particle.h"

class testApp : public ofBaseApp {
public:
    void setup();
    void update();
    void draw();

    // Optical Flow
    ofVideoGrabber camera;
    ofxCv::FlowFarneback flow;

    // GUI
    ofxPanel gui;
    ofxFLOATSlider pyrScale;
    ofxINTSlider levels;
    ofxINTSlider winsize;
    ofxINTSlider iterations;
    ofxINTSlider polyN;
    ofxFLOATSlider polySigma;
    ofxFLOATSlider flowScale;
```

オプティカル・フロー + パーティクル

▶ testApp.h

```
ofxToggle drawFlow;
ofxToggle fullscreen;
ofxButton resetParticleButton;

void resetParticlePressed();

// Particle
vector<Particle> particles;
static const int NUM = 20000;
};
```

オプティカル・フロー + パーティクル

▶ testApp.cpp

```
#include "testApp.h"

using namespace ofxCv;
using namespace cv;

void testApp::setup() {
    // CV初期設定
    camera.initGrabber(320, 240);
    ofSetFrameRate(60);
    ofEnableBlendMode(OF_BLENDMODE_ADD);

    // GUI
    resetParticleButton.addListener(this, &testApp::resetParticlePressed);
    gui.setup();
    gui.add(pyrScale.setup("pyrScale", .5, 0, 1));
    gui.add(levels.setup("levels", 4, 1, 8));
    gui.add(winsize.setup("winsize", 8, 4, 64));
    gui.add(iterations.setup("iterations", 2, 1, 8));
    gui.add(polyN.setup("polyN", 7, 5, 10));
    gui.add(polySigma.setup("polySigma", 1.5, 1.1, 2));
    gui.add(flowScale.setup("flowScale", 0.05, 0.0, 0.2));
    gui.add(drawFlow.setup("draw opticalflow", true));
    gui.add(fullscreen.setup("fullscreen", false));
    gui.add(resetParticleButton.setup("reset particles"));

}
```

オプティカル・フロー + パーティクル

▶ testApp.cpp

```
//パーティクルを生成
for (int i = 0; i < NUM; i++) {
    Particle myParticle;
    myParticle.friction = 0.01;
    myParticle.radius = 2;
    myParticle.setup(ofVec2f(ofRandom(ofGetWidth()), ofRandom(ofGetHeight())), ofVec2f(0, 0));
    particles.push_back(myParticle);
}
}

void testApp::update() {
    camera.update();
    if(camera.isFrameNew()) {
        flow.setPyramidScale(pyrScale);
        flow.setNumLevels(levels);
        flow.setWindowSize(winsize);
        flow.setNumIterations(iterations);
        flow.setPolyN(polyN);
        flow.setPolySigma(polySigma);

        // オプティカルフローを計算
        flow.calcOpticalFlow(camera);
    }
}
```

オプティカル・フロー + パーティクル

▶ testApp.cpp

```
// Particleのアップデート
for (int i = 0; i < particles.size(); i++){
    //particleの力をリセット
    particles[i].resetForce();

    //オプティカルフローから、それぞれのparticleにかかる力を算出
    ofVec2f force;
    ofVec2f pos;
    pos.x = particles[i].position.x * float(camera.getWidth()) / float(ofGetWidth());
    pos.y = particles[i].position.y * float(camera.getHeight()) / float(ofGetHeight());
    force = flow.getFlowOffset(pos.x, pos.y) * flowScale;

    //Particleの状態を更新
    particles[i].addForce(ofVec2f(force.x, force.y));
    particles[i].updateForce();
    particles[i].bounceOffWalls();
    particles[i].update();
}
```

オプティカル・フロー + パーティクル

▶ testApp.cpp

```
void testApp::draw() {
    ofFullscreen(fullscreen);
    ofBackground(0);

    ofSetColor(255);
    camera.draw(0,0,ofGetWidth(),ofGetHeight());

    if (drawFlow) {
        ofSetColor(255, 0, 255);
        flow.draw(0,0,ofGetWidth(),ofGetHeight());
    }

    ofSetColor(0, 127, 255);
    for (int i = 0; i < particles.size(); i++){
        particles[i].draw();
    }

    gui.draw();
}
```

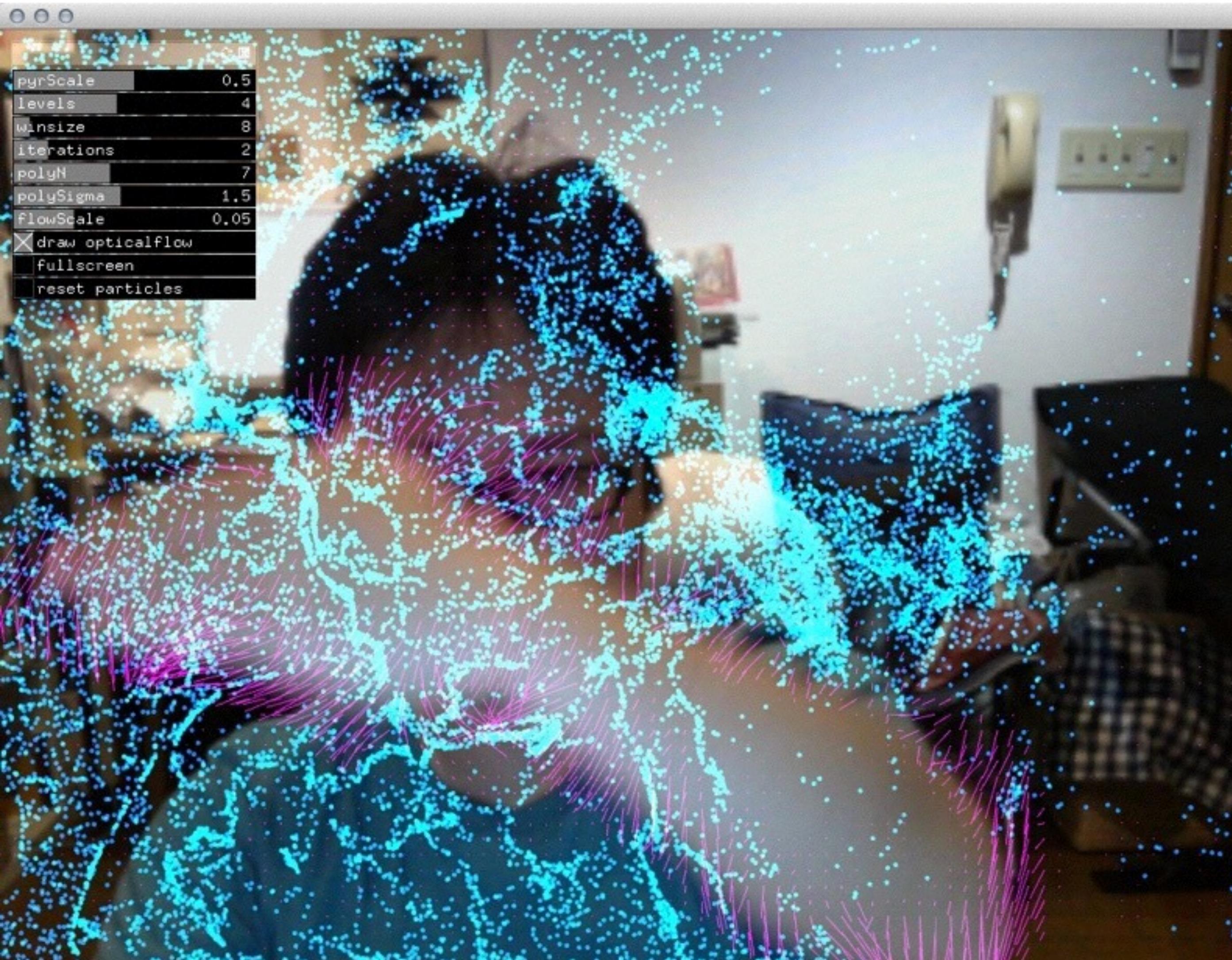
オプティカル・フロー + パーティクル

▶ testApp.cpp

```
void testApp::resetParticlePressed(){
    particles.clear();
    for (int i = 0; i < NUM; i++){
        Particle myParticle;
        myParticle.friction = 0.01;
        myParticle.radius = 2;
        myParticle.setup(ofVec2f(ofRandom(ofGetWidth()), ofRandom(ofGetHeight())),
ofVec2f(0, 0));
        particles.push_back(myParticle);
    }
}
```

オプティカル・フロー + パーティクル

▶ 完成!!



Tracking-Learning-Detection (TLD) ofxTldTracker

Tracking-Learning-Detection (TLD) ofxTldTracker

- ▶ Tracking-Learning-Detection (TLD)
- ▶ Zdenek Kalal氏による、学習しながら物体を認識する仕組み
- ▶ <http://personal.ee.surrey.ac.uk/Personal/Z.Kalal/tld.html>

Tracking-Learning-Detection (TLD)

TLD is an award-winning, real-time algorithm for tracking of unknown objects in video streams. The object of interest is defined by a bounding box in a single frame. TLD simultaneously tracks the object, learns its appearance and detects it whenever it appears in the video. The result is a real-time tracking that often improves over time.



Tracking-Learning-Detection (TLD) ofxTldTracker

- ▶ TLDをopenFrameworksのAddon化 by 登本悠介氏 (ライゾマティクス)
- ▶ <https://github.com/yusuketomoto/ofxTldTracker>

The screenshot shows the GitHub repository page for 'yusuketomoto / ofxTldTracker'. The page includes the repository name, a summary of activity (4 commits, 1 branch, 0 releases, 1 contributor), and a list of recent commits. The right sidebar provides links to Code, Issues, Pull requests, Wiki, Pulse, and Graphs.

Repository Summary:

- 4 commits
- 1 branch
- 0 releases
- 1 contributor

Recent Commits:

File	Message	Date
.gitignore	update example	7 months ago
src	update example	7 months ago
libs	remove	2 years ago
example_simple	update example	7 months ago
example_mosaic	update example	7 months ago
.gitignore	yusuketomoto authored on Oct 12, 2014	latest commit f4329aca2b

Right Sidebar:

- Code
- Issues (0)
- Pull requests (0)
- Wiki
- Pulse
- Graphs

HTTPS clone URL: <https://github.com/yusuketomoto/ofxTldTracker>

Tracking-Learning-Detection (TLD) ofxTldTracker

- ▶ 実行例 : 指先トラッキング!

