

IDD Media lab. 2015

openFrameworks - Addon 2

センサーを使う

ofxKinect、ofxOpenNI、ofxLeapMotion

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26日

田所淳

## 今日の内容

---

- ▶ Addonをつかって、各種センサーからの入力を処理
- ▶ 今回はKinectとLeapMotionを扱います



# Kinectとは？

# Kinectとは？

- ▶ マイクロソフト「Xbox 360」用に開発されたゲーム用コントローラ

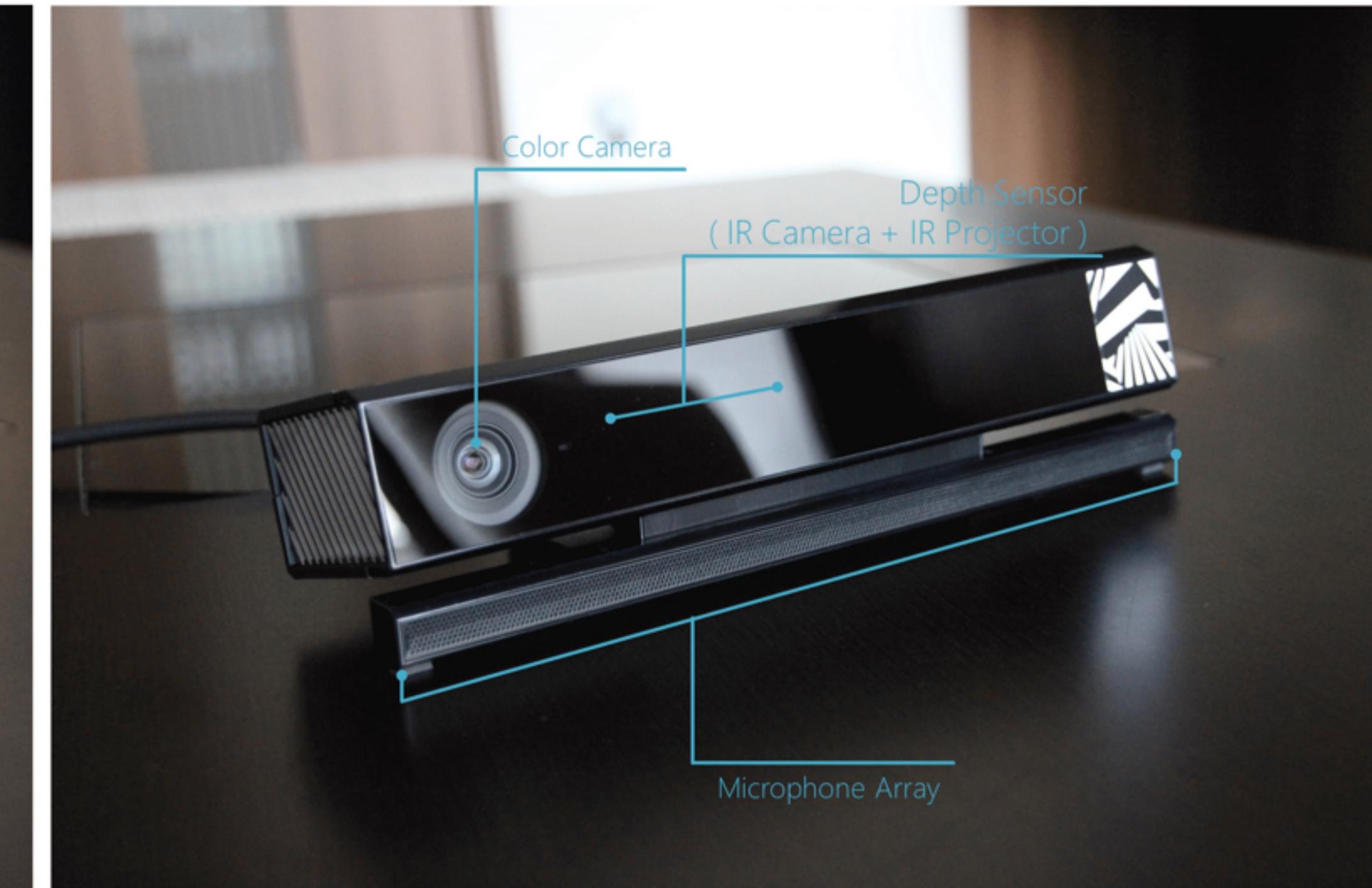


# Kinectとは？

- ▶ Kinect v1 (2012) と Kinect v2 (2014)



Kinect v1



Kinect v2

# Kinectとは?

---

- ▶ Kinect v1
- ▶ Light Coding 方式
- ▶ 投光した赤外線パターンを読み取り、パターンのゆがみからDepth情報を得る
  
- ▶ Kinect v2
- ▶ Time of Flight (TOF) 方式
- ▶ 投光した赤外線が反射して戻ってくる時間からDepth情報を得る
  
- ▶ 参考: Kinect v1とKinect v2の徹底比較 [C++] - Build Insider
- ▶ <http://www.buildinsider.net/small/kinectv2cpp/01>

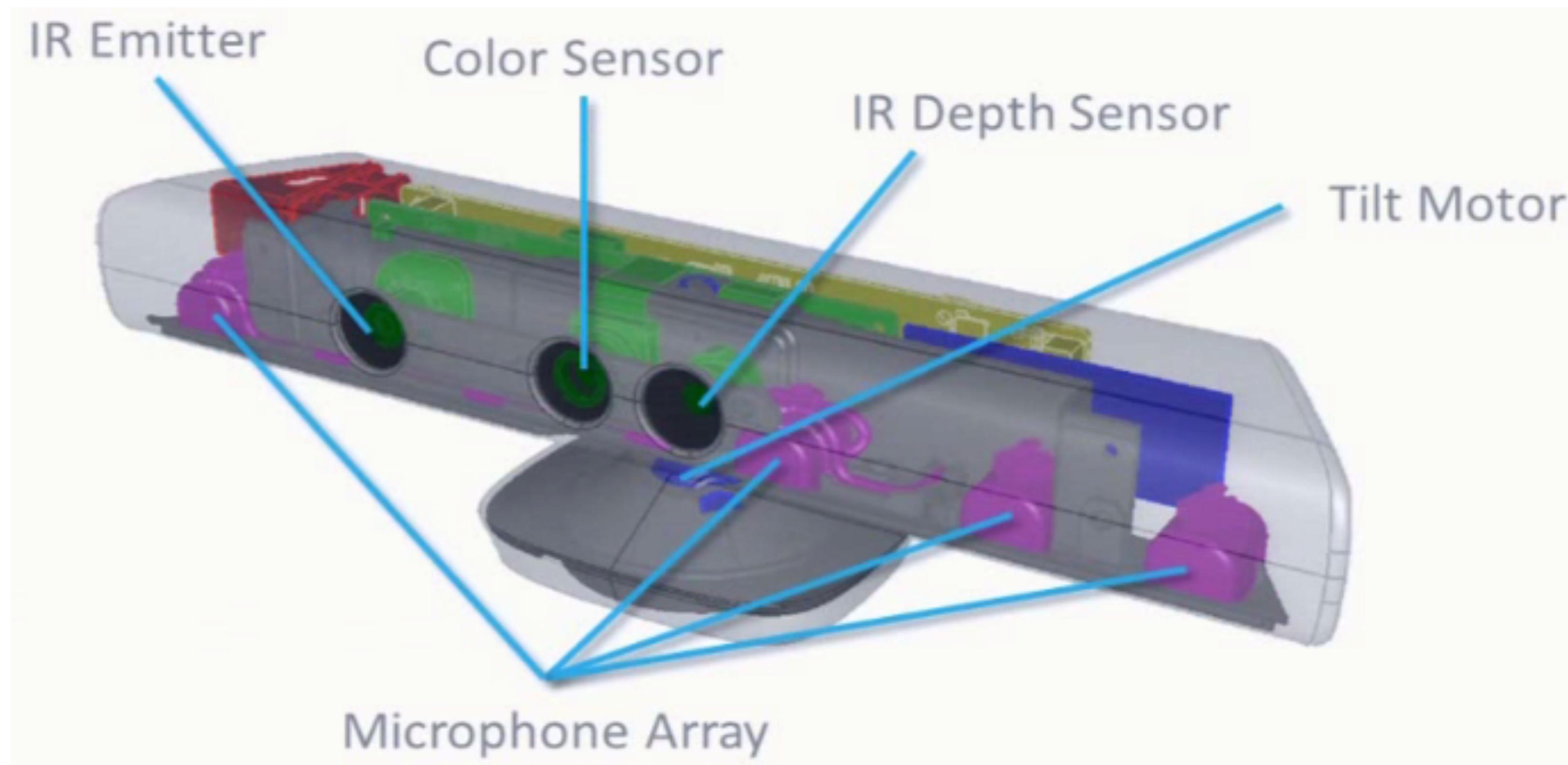
# Kinectとは?

---

- ▶ 今回は、Kinect v1を使用して解説します!

# Kinectとは？

- ▶ Kinect = センサーの集合体



# Kinectとは?

---

- ▶ Kinect にとりつけられたセンサー
- ▶ RGB Camera : 普通のカメラ
- ▶ Multi Array Mic:マイク → 音声入力・認識
- ▶ Motorized Tilt:カメラの向きをソフトウェアからコントロール
- ▶ 3D Depth Sensors:3次元深度センサー ← 注目

# Kinectとは?

---

- ▶ Light Coding
- ▶ 赤外線のパターンを照射している → その歪みから深度と形状を認識
- ▶ <https://youtu.be/1XMHkdGdQv8>



# Kinectとは?

---

- ▶ Kinect Hackというムーブメント
- ▶ Kinect と Xbox360 の接続 → USB2.0
- ▶ 一般的のPCやMacと接続して、信号をやりとりすることが可能
- ▶ これを解析したら、PCやMacからKinectをつかえるのでは...?
- ▶ Kinect Hack!

# Kinectとは?

---

- ▶ Kinect Hackというムーブメント
- ▶ オープンソースのドライバの開発 (libfreenect)
- ▶ なんと、Kinectの発売3時間後に配布!!
- ▶ <https://github.com/OpenKinect/libfreenect>
- ▶ その後数日で、様々な開発環境に移植される
- ▶ openFrameworks、Processing、Cinder、vvvv、Max/MSP、Quartz Composer...etc.

ofxKinect

## ofxKinect

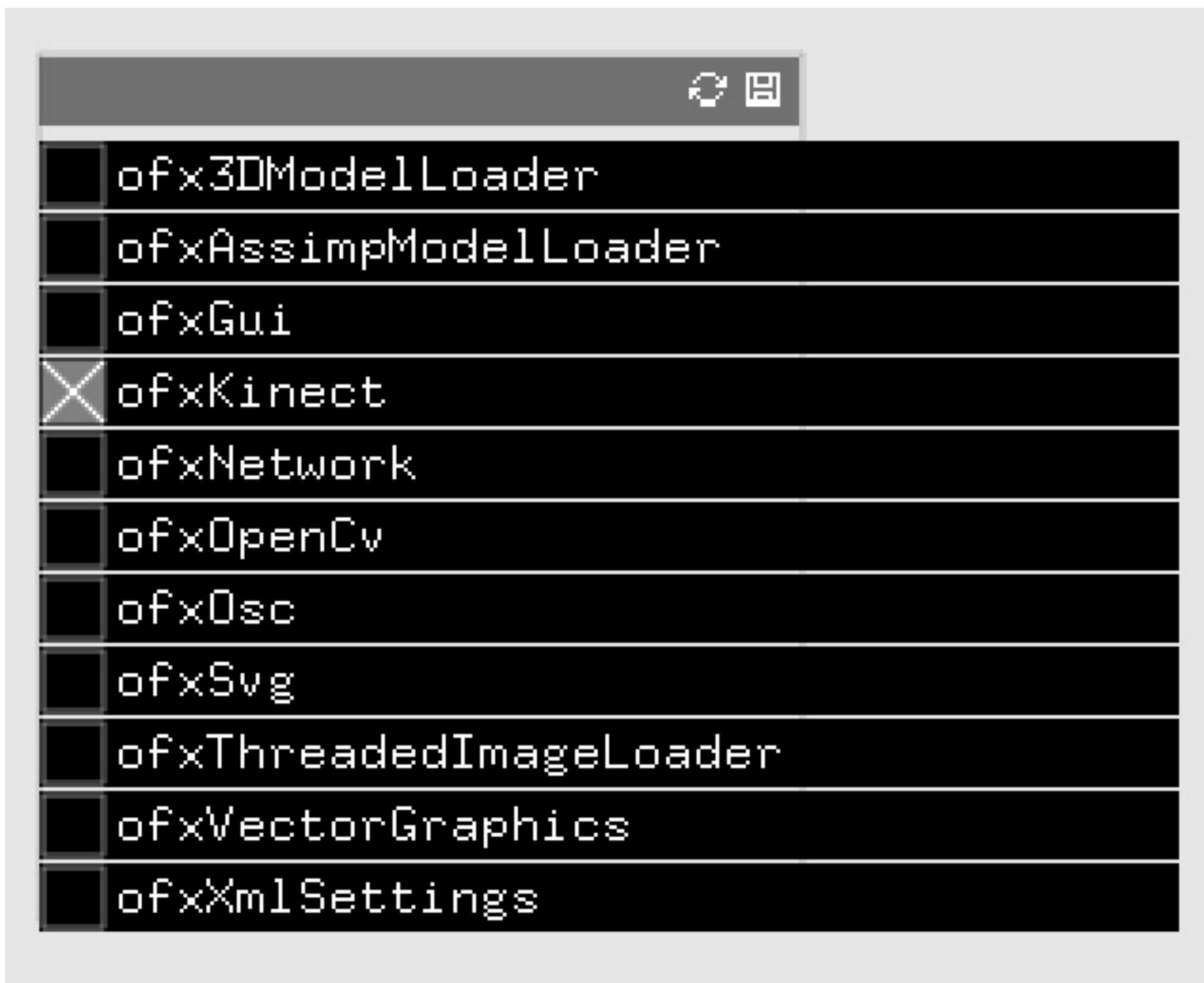
---

- ▶ openFrameworksからKinectを使うには、ofxKinectをAddonとして追加する
- ▶ 使用法はとても簡単!
- ▶ まずは、Kinectからの深度情報つき画像を取得してみる

# ofxKinect

---

- ▶ ProjectGeneratorからプロジェクトを生成
- ▶ Addon選択画面で、ofxKinectをチェックする



# ofxKinect

## ▶ ofApp.h

```
#pragma once

#include "ofMain.h"
#include "ofxKinect.h"

class ofApp : 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);

    ofxKinect kinect;      //Kinectインスタンス
    ofImage kinectImage;   // Kinectカメラ映像
    ofImage depthImage;    // Kinect深度映像
};

};
```

# ofxKinect

## ▶ ofApp.cpp

```
#include "ofApp.h"

void ofApp::setup(){
    //画面設定
    ofSetFrameRate(60);
    // Kinect初期化
    kinect.init();
    kinect.open();
}

void ofApp::update(){
    // Kinect状態更新
    kinect.update();
}

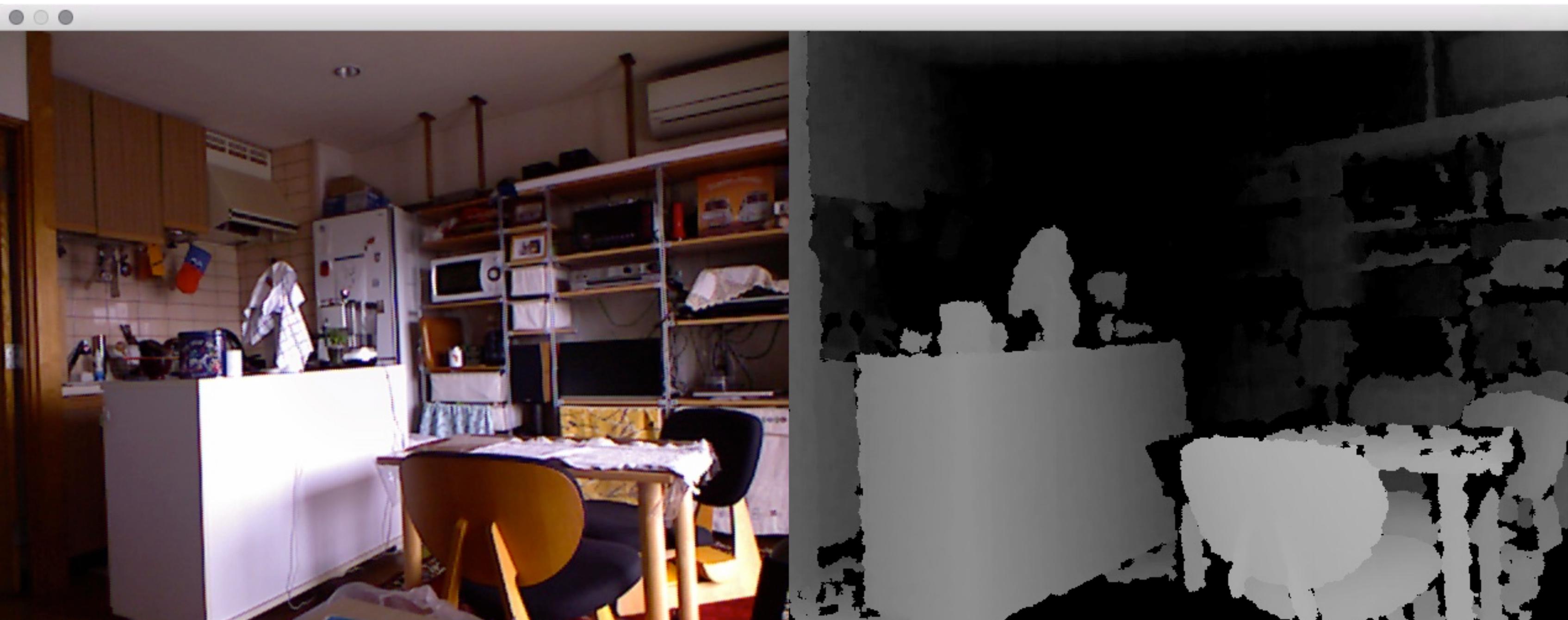
void ofApp::draw(){
    // Kinectカメラから撮影した映像
    kinect.draw(0, 0, kinect.width, kinect.height);
    // Kinect深度情報付き映像
    kinect.drawDepth(kinect.width, 0, kinect.width, kinect.height);
}
```

(後略)

# ofxKinect

---

- ▶ 左: RGBカメラの映像、右: 深度カメラの映像
- ▶ グレースケールが深度（奥行）を表現している



Kinectの深度センサーを活用した物体認識

## Kinectの深度センサーを活用した物体認識

---

- ▶ 深度情報をグレースケールで表現した画像を、閾値を設定して2値化する
- ▶ ある特定の深度の範囲のみを抽出することができる
- ▶ 映像の中の物体を抽出することが可能
- ▶ 通常のOpenCVで物体を認識する際に必要な、背景画像の登録がない
- ▶ 精度もとても高い

# Kinectの深度センサーを活用した物体認識

---

- ▶ Projectの生成
- ▶ 以下の4つのAddonをチェック
- ▶ ofxKinect
- ▶ ofxOpenCV
- ▶ ofxCv
- ▶ ofxGui

# Kinectの深度センサーを活用した物体認識

## ▶ ofApp.h

```
#pragma once
#include "ofMain.h"
#include "ofxKinect.h"
#include "ofxGui.h"
#include "ofxCv.h"

class ofApp : 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);

    ofxKinect kinect;           //Kinectインスタンス
    ofImage kinectImage;        // Kinectカメラ映像
    ofImage depthImage;         // Kinect深度映像
    ofxCv::ContourFinder contourFinder; //CV輪郭抽出
    // GUI
    ofxPanel gui;
    ofxFooter footer;
    ofxFooter footer2;
    ofxFooter footer3;
    ofxFooter footer4;
    ofxFooter footer5;
    ofxFooter footer6;
    ofxFooter footer7;
    ofxFooter footer8;
    ofxFooter footer9;
    ofxFooter footer10;
    ofxFooter footer11;
    ofxFooter footer12;
    ofxFooter footer13;
    ofxFooter footer14;
    ofxFooter footer15;
    ofxFooter footer16;
    ofxFooter footer17;
    ofxFooter footer18;
    ofxFooter footer19;
    ofxFooter footer20;
    ofxFooter footer21;
    ofxFooter footer22;
    ofxFooter footer23;
    ofxFooter footer24;
    ofxFooter footer25;
    ofxFooter footer26;
    ofxFooter footer27;
    ofxFooter footer28;
    ofxFooter footer29;
    ofxFooter footer30;
    ofxFooter footer31;
    ofxFooter footer32;
    ofxFooter footer33;
    ofxFooter footer34;
    ofxFooter footer35;
    ofxFooter footer36;
    ofxFooter footer37;
    ofxFooter footer38;
    ofxFooter footer39;
    ofxFooter footer40;
    ofxFooter footer41;
    ofxFooter footer42;
    ofxFooter footer43;
    ofxFooter footer44;
    ofxFooter footer45;
    ofxFooter footer46;
    ofxFooter footer47;
    ofxFooter footer48;
    ofxFooter footer49;
    ofxFooter footer50;
    ofxFooter footer51;
    ofxFooter footer52;
    ofxFooter footer53;
    ofxFooter footer54;
    ofxFooter footer55;
    ofxFooter footer56;
    ofxFooter footer57;
    ofxFooter footer58;
    ofxFooter footer59;
    ofxFooter footer60;
    ofxFooter footer61;
    ofxFooter footer62;
    ofxFooter footer63;
    ofxFooter footer64;
    ofxFooter footer65;
    ofxFooter footer66;
    ofxFooter footer67;
    ofxFooter footer68;
    ofxFooter footer69;
    ofxFooter footer70;
    ofxFooter footer71;
    ofxFooter footer72;
    ofxFooter footer73;
    ofxFooter footer74;
    ofxFooter footer75;
    ofxFooter footer76;
    ofxFooter footer77;
    ofxFooter footer78;
    ofxFooter footer79;
    ofxFooter footer80;
    ofxFooter footer81;
    ofxFooter footer82;
    ofxFooter footer83;
    ofxFooter footer84;
    ofxFooter footer85;
    ofxFooter footer86;
    ofxFooter footer87;
    ofxFooter footer88;
    ofxFooter footer89;
    ofxFooter footer90;
    ofxFooter footer91;
    ofxFooter footer92;
    ofxFooter footer93;
    ofxFooter footer94;
    ofxFooter footer95;
    ofxFooter footer96;
    ofxFooter footer97;
    ofxFooter footer98;
    ofxFooter footer99;
    ofxFooter footer100;
    ofxFooter footer101;
    ofxFooter footer102;
    ofxFooter footer103;
    ofxFooter footer104;
    ofxFooter footer105;
    ofxFooter footer106;
    ofxFooter footer107;
    ofxFooter footer108;
    ofxFooter footer109;
    ofxFooter footer110;
    ofxFooter footer111;
    ofxFooter footer112;
    ofxFooter footer113;
    ofxFooter footer114;
    ofxFooter footer115;
    ofxFooter footer116;
    ofxFooter footer117;
    ofxFooter footer118;
    ofxFooter footer119;
    ofxFooter footer120;
    ofxFooter footer121;
    ofxFooter footer122;
    ofxFooter footer123;
    ofxFooter footer124;
    ofxFooter footer125;
    ofxFooter footer126;
    ofxFooter footer127;
    ofxFooter footer128;
    ofxFooter footer129;
    ofxFooter footer130;
    ofxFooter footer131;
    ofxFooter footer132;
    ofxFooter footer133;
    ofxFooter footer134;
    ofxFooter footer135;
    ofxFooter footer136;
    ofxFooter footer137;
    ofxFooter footer138;
    ofxFooter footer139;
    ofxFooter footer140;
    ofxFooter footer141;
    ofxFooter footer142;
    ofxFooter footer143;
    ofxFooter footer144;
    ofxFooter footer145;
    ofxFooter footer146;
    ofxFooter footer147;
    ofxFooter footer148;
    ofxFooter footer149;
    ofxFooter footer150;
    ofxFooter footer151;
    ofxFooter footer152;
    ofxFooter footer153;
    ofxFooter footer154;
    ofxFooter footer155;
    ofxFooter footer156;
    ofxFooter footer157;
    ofxFooter footer158;
    ofxFooter footer159;
    ofxFooter footer160;
    ofxFooter footer161;
    ofxFooter footer162;
    ofxFooter footer163;
    ofxFooter footer164;
    ofxFooter footer165;
    ofxFooter footer166;
    ofxFooter footer167;
    ofxFooter footer168;
    ofxFooter footer169;
    ofxFooter footer170;
    ofxFooter footer171;
    ofxFooter footer172;
    ofxFooter footer173;
    ofxFooter footer174;
    ofxFooter footer175;
    ofxFooter footer176;
    ofxFooter footer177;
    ofxFooter footer178;
    ofxFooter footer179;
    ofxFooter footer180;
    ofxFooter footer181;
    ofxFooter footer182;
    ofxFooter footer183;
    ofxFooter footer184;
    ofxFooter footer185;
    ofxFooter footer186;
    ofxFooter footer187;
    ofxFooter footer188;
    ofxFooter footer189;
    ofxFooter footer190;
    ofxFooter footer191;
    ofxFooter footer192;
    ofxFooter footer193;
    ofxFooter footer194;
    ofxFooter footer195;
    ofxFooter footer196;
    ofxFooter footer197;
    ofxFooter footer198;
    ofxFooter footer199;
    ofxFooter footer200;
    ofxFooter footer201;
    ofxFooter footer202;
    ofxFooter footer203;
    ofxFooter footer204;
    ofxFooter footer205;
    ofxFooter footer206;
    ofxFooter footer207;
    ofxFooter footer208;
    ofxFooter footer209;
    ofxFooter footer210;
    ofxFooter footer211;
    ofxFooter footer212;
    ofxFooter footer213;
    ofxFooter footer214;
    ofxFooter footer215;
    ofxFooter footer216;
    ofxFooter footer217;
    ofxFooter footer218;
    ofxFooter footer219;
    ofxFooter footer220;
    ofxFooter footer221;
    ofxFooter footer222;
    ofxFooter footer223;
    ofxFooter footer224;
    ofxFooter footer225;
    ofxFooter footer226;
    ofxFooter footer227;
    ofxFooter footer228;
    ofxFooter footer229;
    ofxFooter footer230;
    ofxFooter footer231;
    ofxFooter footer232;
    ofxFooter footer233;
    ofxFooter footer234;
    ofxFooter footer235;
    ofxFooter footer236;
    ofxFooter footer237;
    ofxFooter footer238;
    ofxFooter footer239;
    ofxFooter footer240;
    ofxFooter footer241;
    ofxFooter footer242;
    ofxFooter footer243;
    ofxFooter footer244;
    ofxFooter footer245;
    ofxFooter footer246;
    ofxFooter footer247;
    ofxFooter footer248;
    ofxFooter footer249;
    ofxFooter footer250;
    ofxFooter footer251;
    ofxFooter footer252;
    ofxFooter footer253;
    ofxFooter footer254;
    ofxFooter footer255;
    ofxFooter footer256;
    ofxFooter footer257;
    ofxFooter footer258;
    ofxFooter footer259;
    ofxFooter footer260;
    ofxFooter footer261;
    ofxFooter footer262;
    ofxFooter footer263;
    ofxFooter footer264;
    ofxFooter footer265;
    ofxFooter footer266;
    ofxFooter footer267;
    ofxFooter footer268;
    ofxFooter footer269;
    ofxFooter footer270;
    ofxFooter footer271;
    ofxFooter footer272;
    ofxFooter footer273;
    ofxFooter footer274;
    ofxFooter footer275;
    ofxFooter footer276;
    ofxFooter footer277;
    ofxFooter footer278;
    ofxFooter footer279;
    ofxFooter footer280;
    ofxFooter footer281;
    ofxFooter footer282;
    ofxFooter footer283;
    ofxFooter footer284;
    ofxFooter footer285;
    ofxFooter footer286;
    ofxFooter footer287;
    ofxFooter footer288;
    ofxFooter footer289;
    ofxFooter footer290;
    ofxFooter footer291;
    ofxFooter footer292;
    ofxFooter footer293;
    ofxFooter footer294;
    ofxFooter footer295;
    ofxFooter footer296;
    ofxFooter footer297;
    ofxFooter footer298;
    ofxFooter footer299;
    ofxFooter footer299;
};
```

# Kinectの深度センサーを活用した物体認識

## ▶ ofApp.cpp

```
#include "ofApp.h"

void ofApp::setup(){
    //画面設定
    ofSetFrameRate(60);
    ofBackground(0);
    // Kinect初期化
    kinect.init();
    kinect.open();
    // GUI初期設定
    gui.setup();
    gui.add(thresh.setup("Threshold", 127, 0, 255));
    gui.add(minRadius.setup("Min Radius", 10, 0, 400));
    gui.add(maxRadius.setup("Max Radius", 200, 0, 400));
}

void ofApp::update(){
    // CV設定
    contourFinder.setMinAreaRadius(minRadius);
    contourFinder.setMaxAreaRadius(maxRadius);
    // Kinect状態更新
    kinect.update();
    if (kinect.isFrameNew()) {
        // ofxCvで輪郭抽出
        cv::Mat mat = cv::Mat(kinect.height, kinect.width, CV_8UC1, kinect.getDepthPixels(), 0);
        contourFinder.setThreshold(thresh);
        contourFinder.findContours(mat);
    }
}
```

# Kinectの深度センサーを活用した物体認識

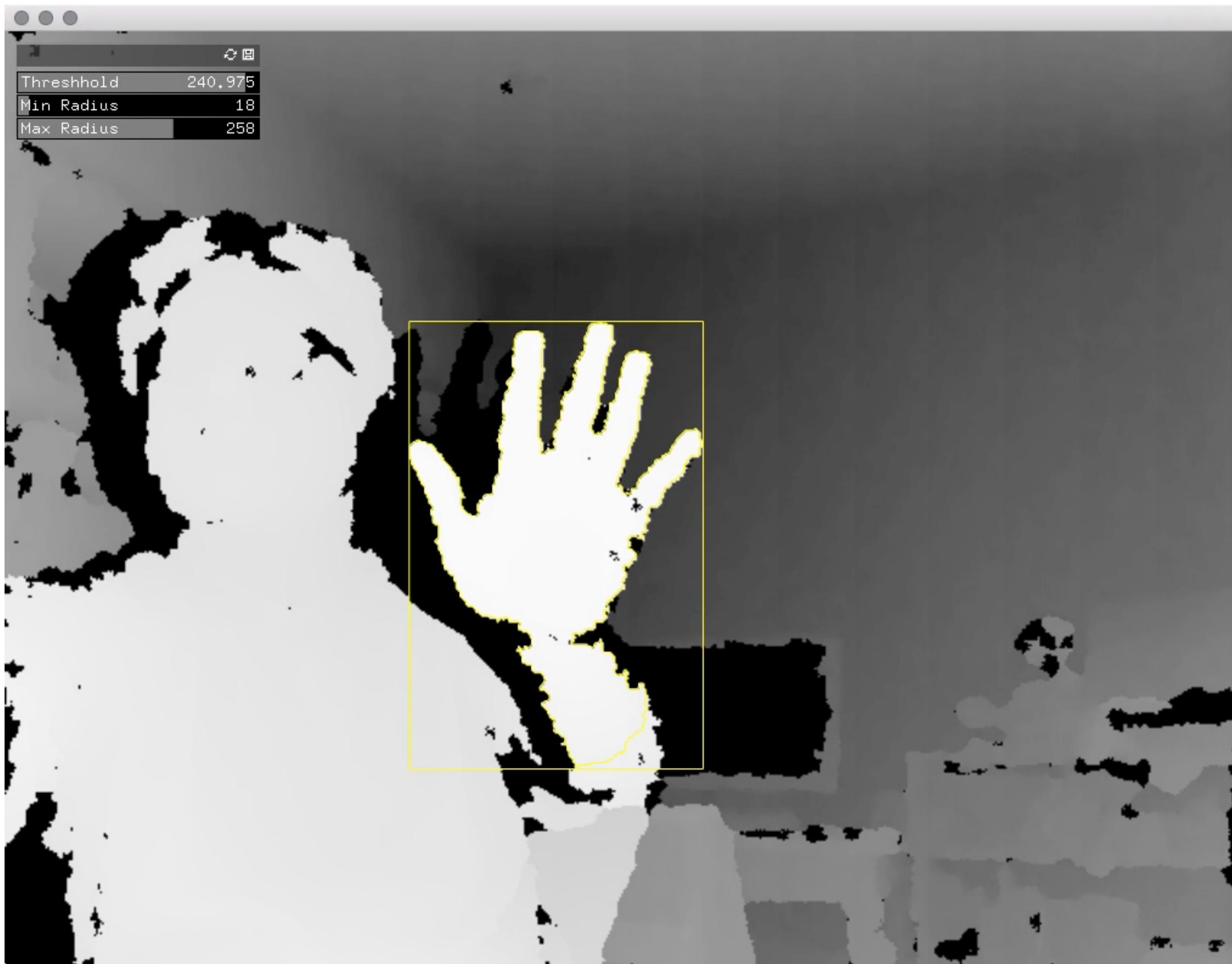
## ▶ ofApp.cpp

```
void ofApp::draw(){
    // 深度画像描画
    ofSetColor(255);
    kinect.drawDepth(0, 0, ofGetWidth(), ofGetHeight());
    // 輪郭抽出結果描画
    ofSetColor(255, 255, 0);
    ofPushMatrix();
    ofScale(ofGetWidth() / float(kinect.width), ofGetHeight() / float(kinect.height));
    contourFinder.draw();
    ofPopMatrix();
    // GUI描画
    gui.draw();
}
```

(後略)

# Kinectの深度センサーを活用した物体認識

- ▶ 特定の深度の物体だけを、きれいに認識できた!



Kinectで、ポイントクラウド!

# Kinectで、ポイントクラウド!

---

- ▶ ofxKinectのクラスは、カメラで映している範囲の全てのピクセルごとに、奥行の深度を取得することも可能
- ▶ カメラの映像と照合して、その位置の色も取得できる
- ▶ 下記のメソッドで取得する

```
kinect.getDistanceAt(x, y); // 深度情報を取得  
kinect.getColorAt(x, y); // 色を取得
```

- ▶ kinect - ofxKinectのインスタンス
- ▶ x, y - 取得する深度の位置

# Kinectで、ポイントクラウド!

---

- ▶ Kinectのカメラの解像度は、 $640 \times 480$  pixel
  - ▶ つまり、 $640 \times 480 = 307200$  ポイントの深度情報と色が得られる
  - ▶ 全ての地点に対して、実際に3D空間上に点を配置してみる
  - ▶ 3次元の点群で、立体が再現されるはず
- 
- ▶ この手法を「ポイントクラウド (point cloud)」と呼ぶ
  - ▶ openFrameworksで再現する際には、まず3Dのメッシュ (ofMesh)を定義して、頂点の座標として配置 → OpenGLで頂点のみを描画する

# Kinectで、ポイントクラウド!

## ▶ ofApp.h

```
#pragma once

#include "ofMain.h"
#include "ofxKinect.h"

class ofApp : 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);

    ofxKinect kinect;      // Kinectインスタンス
    ofImage kinectImage;   // Kinectカメラ映像
    ofImage depthImage;    // Kinect深度映像
    ofEasyCam cam;         // カメラ
};

};
```

# Kinectで、ポイントクラウド!

## ▶ ofApp.cpp

```
#include "ofApp.h"

void ofApp::setup(){
    ofSetFrameRate(60);
    ofBackground(0);
    kinect.setRegistration(true); // カメラ画像と深度カメラのズれを解消
    kinect.init(); // Kinect初期化
    kinect.open(); // Kinect開始
}

void ofApp::update(){
    kinect.update(); // Kinect更新
}

void ofApp::draw(){
    int w = 640;
    int h = 480;
    ofMesh mesh; // Meshを準備
    mesh.setMode(OF_PRIMITIVE_POINTS); // Meshの頂点を点で描画
    // 指定した間隔で全てのピクセルの位置と色を取得
    int step = 2;
    for(int y = 0; y < h; y += step) {
        for(int x = 0; x < w; x += step) {
            if(kinect.getDistanceAt(x, y) > 0) {
                mesh.addColor(kinect.getColorAt(x,y));
                mesh.addVertex(kinect.getWorldCoordinateAt(x, y));
            }
        }
    }
}
```

# Kinectで、ポイントクラウド!

## ▶ ofApp.cpp

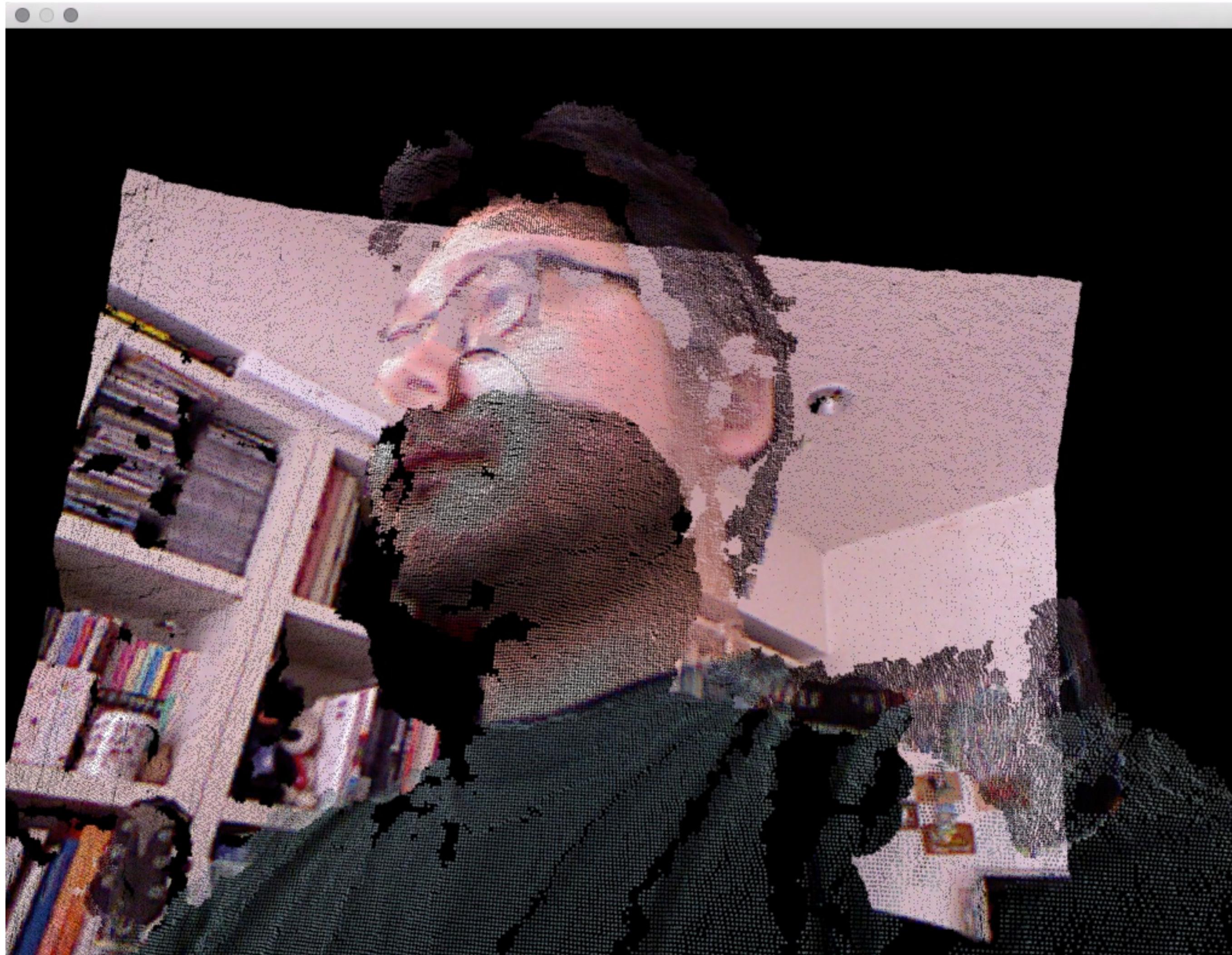
```
glPointSize(3); // 描画する点の大きさ

cam.begin(); // カメラ開始
ofPushMatrix();
ofScale(1, -1, -1); // 上下と前後を反転
ofTranslate(0, 0, -1000); // 全体位置を後ろへ
ofEnableDepthTest(); // 奥の物体を隠すように
mesh.draw(); // メッシュ描画
ofDisableDepthTest();
ofPopMatrix();
cam.end();
}
```

# Kinectで、ポイントクラウド!

---

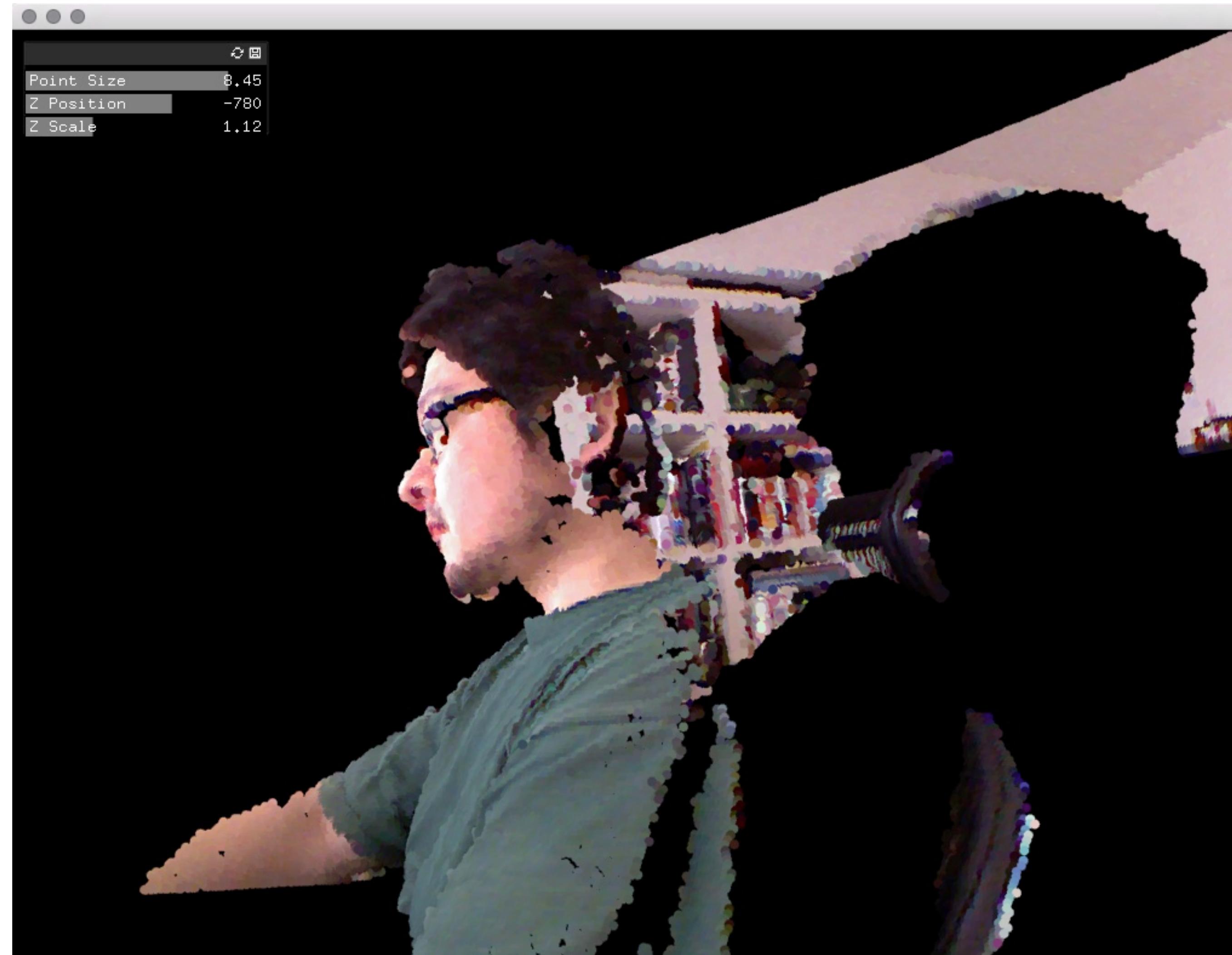
- ▶ ポイントクラウドが描けた!



# Kinectで、ポイントクラウド!

---

- ▶ ofxGUIを使用して、さらに細かい設定を可能に



# OpenNI

# OpenNI

---

- ▶ OpenNIとは?
- ▶ Open NI = Open Natural Interaction
- ▶ Kinectを使用して、ユーザのポーズやジェスチャーなどの自然な動きを使用して、インタラクションを行うためのオープンなライブラリ
  
- ▶ Kinectを開発した、PrimeSence社が主導
- ▶ しかし、2013年、PrimeSence社がAppleに買収されたためプロジェクト閉鎖
- ▶ OpenNIをForkした、OpenNI 2としてオープンソースプロジェクトとして継続
- ▶ <http://structure.io/openni>

## ofxOpenNI

---

- ▶ ofxOpenNI : OpenNIをopenFrameworksから使用するAddon
  - ▶ 古いAddonなので、ちょっとビルドが面倒
  - ▶ 0.8.4でサンプルがすぐに見られるバージョンを用意しました
  - ▶ <https://github.com/tado/ofxOpenNI>
- 
- ▶ いろいろサンプルを見てみましょう!

# ofxOpenNIサンプル

- ▶ OpenNI-ImageAndDepth-Simple
- ▶ シンプルな深度情報の表示



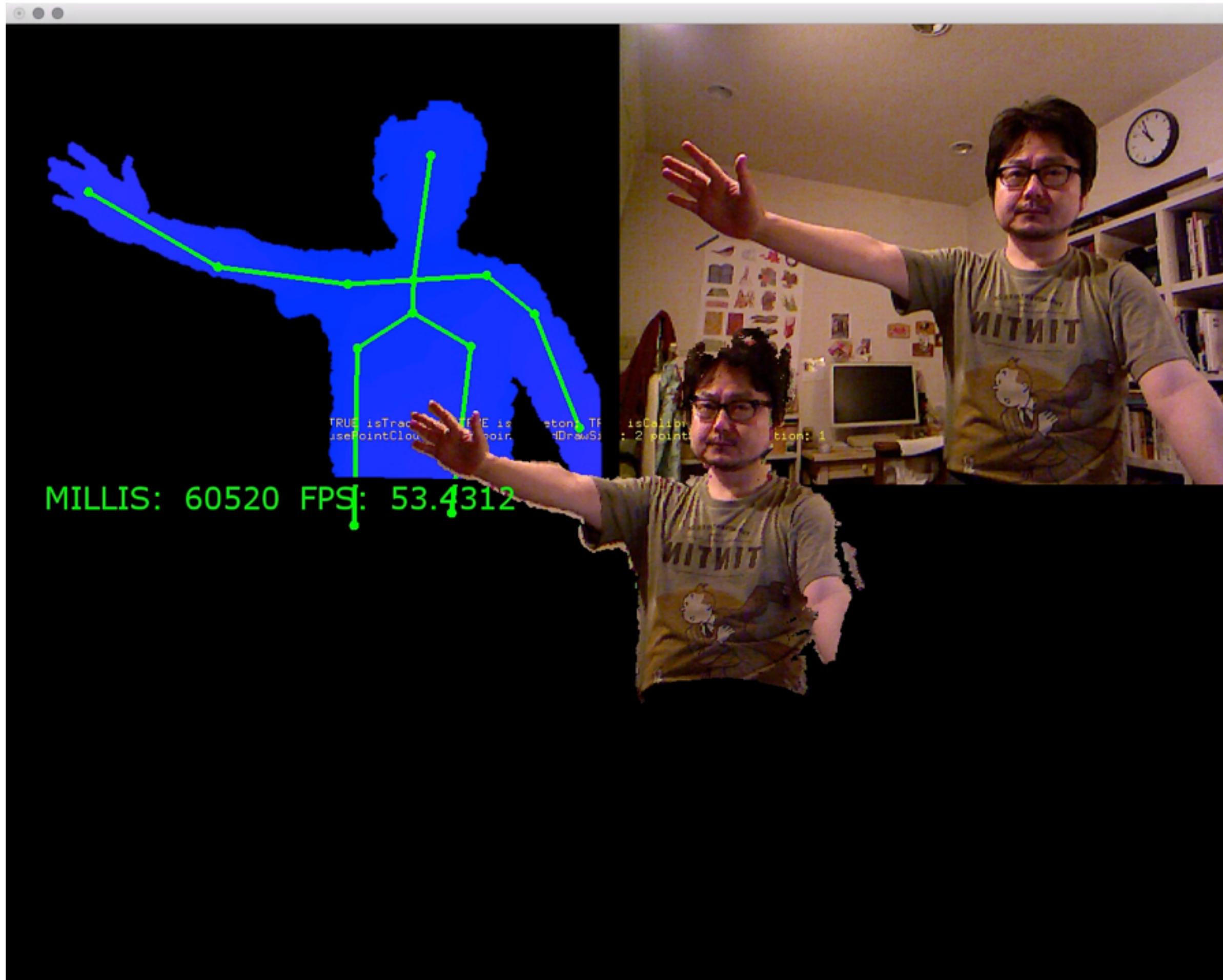
# ofxOpenNIサンプル

- ▶ OpenNI-UserAndCloud-Simple
- ▶ 人間の骨格情報の抽出



# ofxOpenNIサンプル

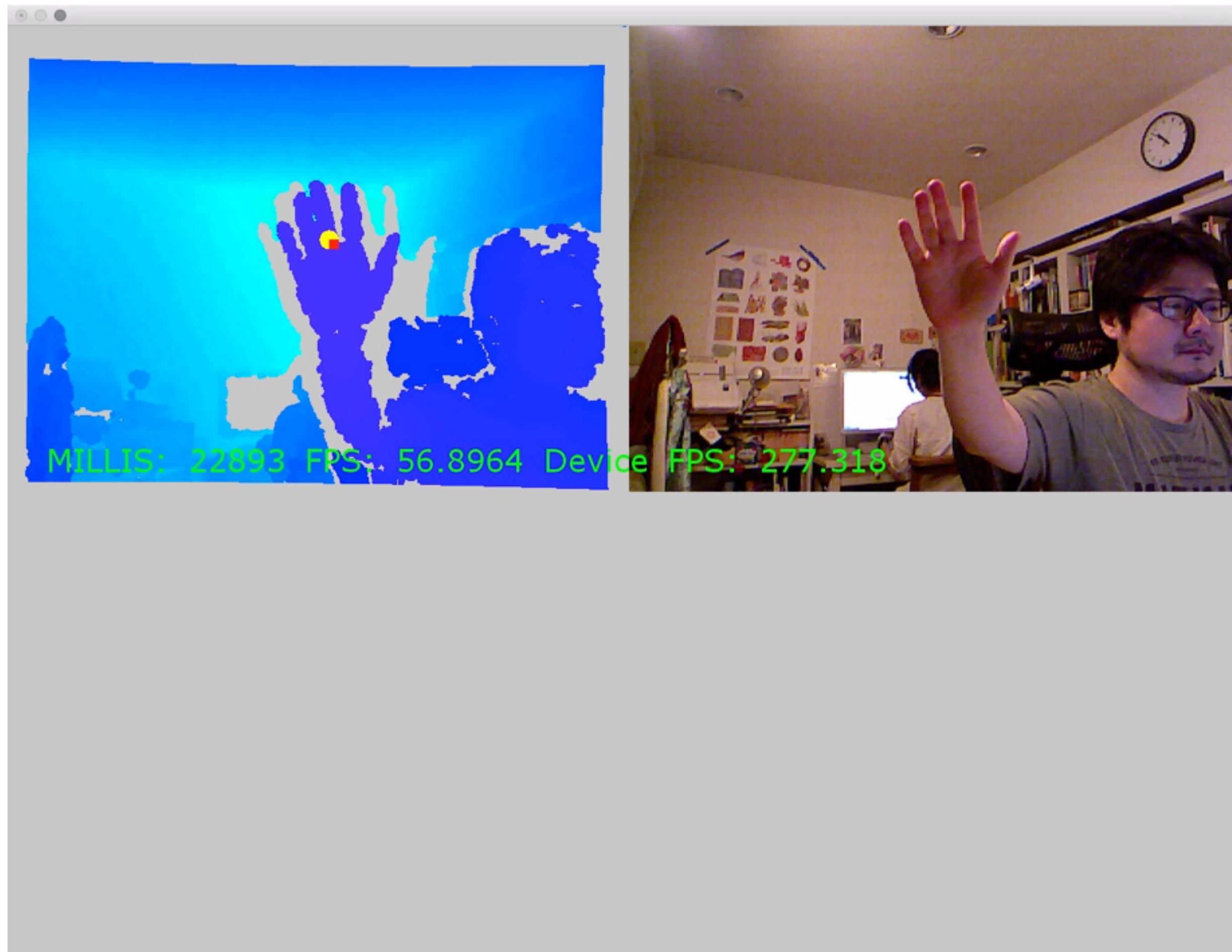
- ▶ OpenNI-UserAndCloud-Medium
- ▶ 人間の骨格情報の抽出 + ポイントクラウド



# ofxOpenNIサンプル

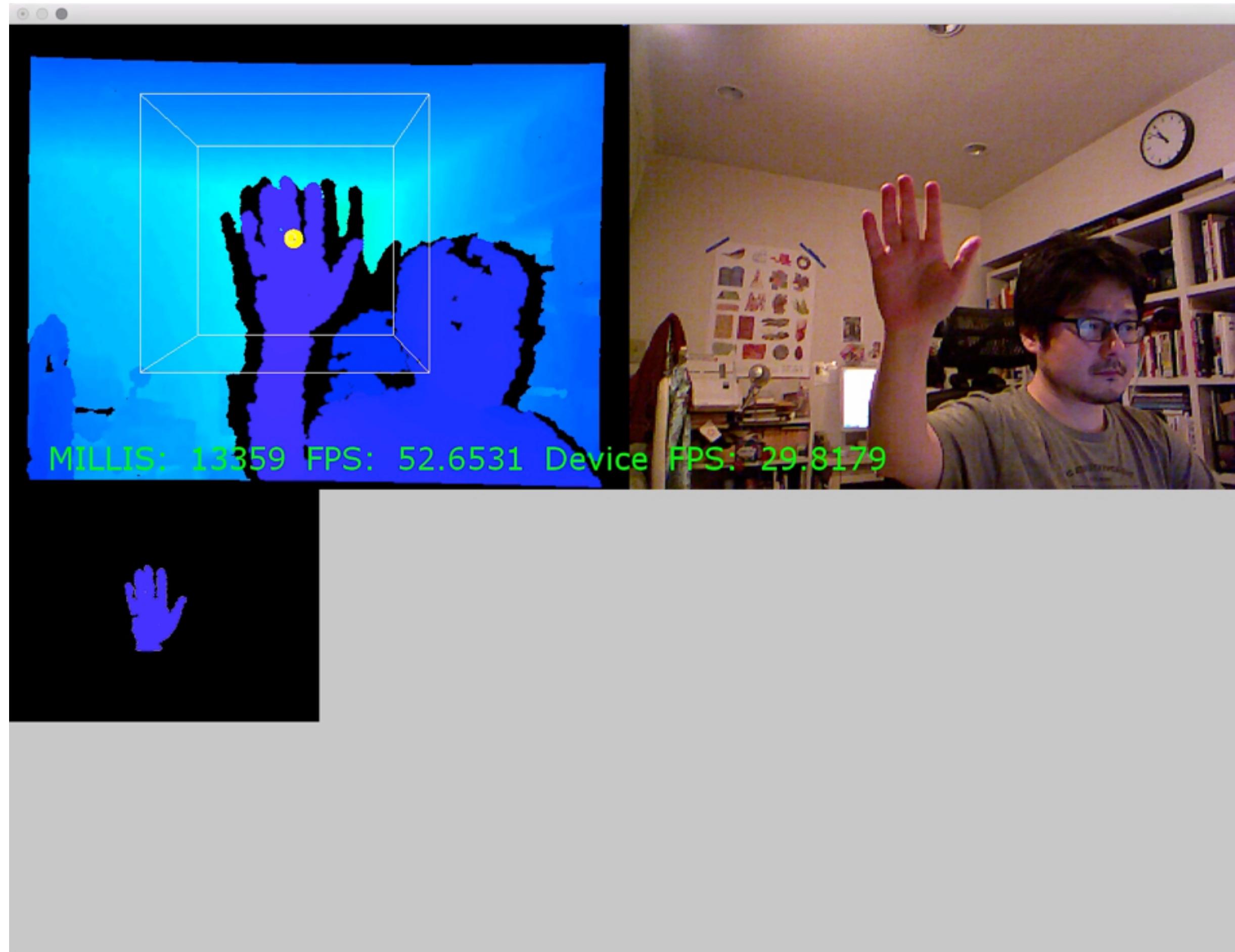
---

- ▶ OpenNI-HandTracking-Simple
- ▶ 手のトラッキング



# ofxOpenNIサンプル

- ▶ OpenNI-HandTracking-Medium
- ▶ 手のトラッキングと3D座標検出 + 抽出した手の輪郭表示



# Leap Motion

# Leap Motion

---

- ▶ 手のジェスチャーによってコンピュータを操作ができるデバイス
- ▶ 2基の赤外線カメラで撮影し、画像解析によって3D空間での手や指の位置を割り出す
- ▶ 検知できる範囲は半径50センチ程度、中心角110度の空間
- ▶ 0.01ミリの精度で認識



# Leap Motion

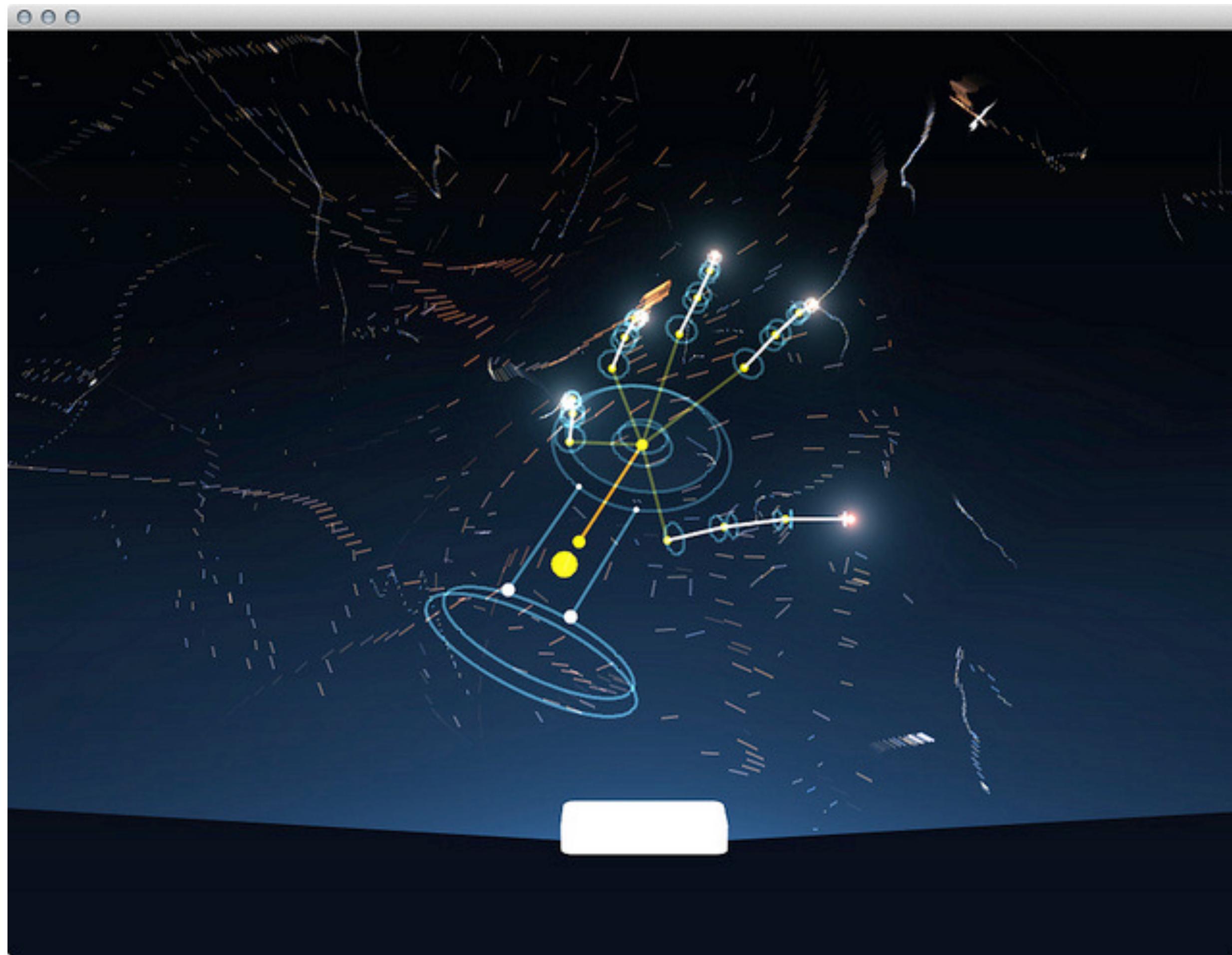
- ▶ まず始めにセットアップ
- ▶ Leap MotionのWebページからLeap Motion Controllerをダウンロード&インストール
- ▶ <https://www.leapmotion.com/setup>



# Leap Motion

---

- ▶ 付属のデモアプリで遊んでみる
- ▶ とても高精度の手のトラッキングを実感できる



# Leap Motion

- ▶ openFrameworksから、Leap Motionを使用するには、開発者登録が必要
- ▶ 以下のサイトから必要項目を入力して登録
- ▶ <https://developer.leapmotion.com/>

The screenshot shows the Leap Motion Developer Portal homepage. At the top, there's a navigation bar with the Leap Motion logo, "Developer Portal", and links for "Downloads", "Get Started", "Examples", "Documentation", "Community", and "Sign in". The main section features a large blue banner with the heading "Leap Motion SDK" in white. Below it, a message reads: "The next generation of tracking is here – but it's still a work in progress. Please consult the [known issues](#), and let us know what you think!". A green button says "Sign in to download SDK v2.2.5.26752 for OSX". It also notes that the software is available for "Windows and Linux". To the right, there's a graphic of a laptop screen displaying a 3D molecular model with purple and green spheres connected by white lines, set against a grid background. A play button icon is overlaid on the screen. At the bottom, there's a newsletter sign-up form with fields for email ("you@somewhere.com") and a "Subscribe" button.

# Leap Motion

- ▶ 開発者登録をすると、最新版のLeap Controllerがダウンロード可能
- ▶ ダウンロードしてインストール
- ▶ <https://beta.leapmotion.com/>

The screenshot shows the Leap Motion Developer Portal at <https://beta.leapmotion.com/>. The page title is "Leap Motion Software - Installer Preview". It includes a note about being an early preview of beta software, a list of improvements for version 2.2.6+29153, and a message about reporting issues through community forums. At the bottom, there are download links for Windows, OSX, and Linux.

LEAP MOTION Developer Portal

Downloads ▾ Get Started ▾ Examples ▾ Documentation ▾ Community ▾ Sign in

## Leap Motion Software - Installer Preview

This is an early preview of an upcoming release of the Leap Motion Software. This is beta software. Please proceed with caution.

This version, 2.2.6+29153, contains the following improvements:

- Adding an image.data\_pointer hook for faster image API access in Python
- Fix for device.serial\_number in Python
- Finger.Type is now a property in C#
- C++: Leap::FingerList and other list types now work with std::find\_if()
- Support for Unity 5 Editor (32-bit)
- Fixed crash on libxs library
- Improved error log reporting
- Improvements to Image while in Robust mode
- New Sign-Up button in App Home
- Apps that have not been installed are clearly marked in App Home
- Fix for App Home download failures

Since this is an early preview of the software it will not be supported through our regular support channels. If you experience any issues or have any feedback about this version of the software please make sure to report them in the [the community forums](#).

Windows

OSX

Linux

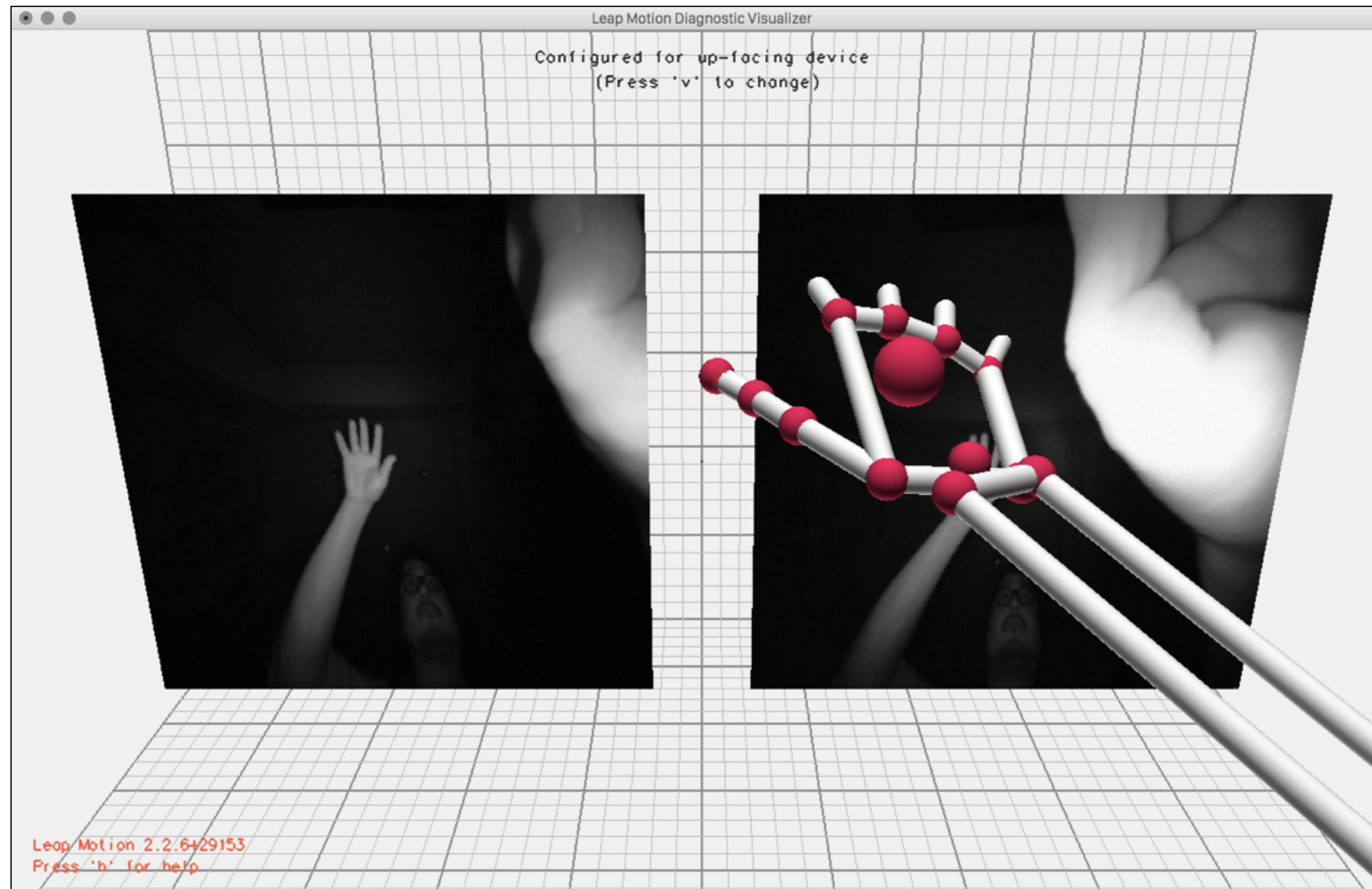
[Download v.2.2.6.29153 for Windows](#)

[Download v.2.2.6.29153 for OSX](#)

[Download v.2.2.6.29153 for Linux](#)

# Leap Motion

- ▶ 最新版のVisualizerでは、手の認識精度が向上
- ▶ さらに、カメラの画像を直接取得できるようになった



# Leap Motion

- ▶ openFrameworksから、LeapMotionを使うには
- ▶ ofxLeapMotionを使用する
- ▶ <https://github.com/ofTheo/ofxLeapMotion>

The screenshot shows the GitHub repository page for `ofTheo/ofxLeapMotion`. The repository is described as "A wrapper for the Leap SDK ( 0.8.1 and up ) - compatible with Leap 1.0 Release". It has 63 commits, 1 branch, 0 releases, and 8 contributors. The master branch is selected. The repository has 29 watchers, 116 stars, and 47 forks. The right sidebar includes links for Code, Issues (1), Pull requests (0), Wiki, Pulse, and Graphs. A HTTPS clone URL is provided at the bottom.

A wrapper for the Leap SDK ( 0.8.1 and up ) - compatible with Leap 1.0 Release

63 commits 1 branch 0 releases 8 contributors

branch: master / +

Update README.md

ofTheo authored on Oct 29, 2014 latest commit 1e1639afc6

| File             | Description   | Time          |
|------------------|---|---------------|
| example-gestures | Merge remote-tracking branch 'IxDS/linux32' into 2.0sdk | 11 months ago |
| example          | Merge remote-tracking branch 'IxDS/linux32' into 2.0sdk | 11 months ago |
| libs             | changes from golan for digital art project              | 8 months ago  |
| src              | moved initializers to constructor                       | 8 months ago  |
| .gitignore       | reverted to full gitignore                              | 11 months ago |
| README.md        | Update README.md  | 7 months ago  |
| addon_config.mk  | added Leap Linux SDK 2.0 (64bit)                        | 11 months ago |

HTTPS clone URL  
https://github.com/[Clone](#)

You can clone with [HTTPS](#), [SSH](#), or [Subversion](#).

Clone in Desktop

## Leap Motion

---

- ▶ まずは、シンプルに手の骨格をトラッキングしてみたい
- ▶ ProjectGeneratorで、ofxLeapMotionをチェックしてプロジェクト生成

# Leap Motion

## ▶ ofApp.h

```
#pragma once

#include "ofMain.h"
#include "ofxLeapMotion.h"

class ofApp : 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);

    ofxLeapMotion leap; // Leap Motionのメインクラスをインスタンス化
    vector <ofxLeapMotionSimpleHand> simpleHands; // シンプルな手のモデルのvector配列
    ofEasyCam cam; //カメラ
    ofLight light; //ライト
};
```

# Leap Motion

## ▶ ofApp.cpp

```
#include "ofApp.h"

void ofApp::setup(){
    // 画面設定
    ofSetFrameRate(60);
    ofSetVerticalSync(true);
    ofBackground(31);
    // 照明とカメラ
    ofEnableLighting();
    light.setPosition(200, 300, 50);
    light.enable();
    cam.setOrientation(ofPoint(-20, 0, 0));
    // GL設定
    glEnable(GL_DEPTH_TEST);
    glEnable(GL_NORMALIZE);
    // Leap Motion開始
    leap.open();
}

void ofApp::update(){
    // 検出された手の数だけ、ofxLeapMotionSimpleHandのvector配列に追加
    simpleHands = leap.getSimpleHands();

    // フレーム更新して、手が検出されたら
    if( leap.isFrameNew() && simpleHands.size() ){
        // 画面の大きさにあわせて、スケールをマッピング
        leap.setMappingX(-230, 230, -ofGetWidth()/2, ofGetWidth()/2);
        leap.setMappingY(90, 490, -ofGetHeight()/2, ofGetHeight()/2);
        leap.setMappingZ(-150, 150, -200, 200);
    }
}
```

# Leap Motion

## ▶ ofApp.cpp

```
}

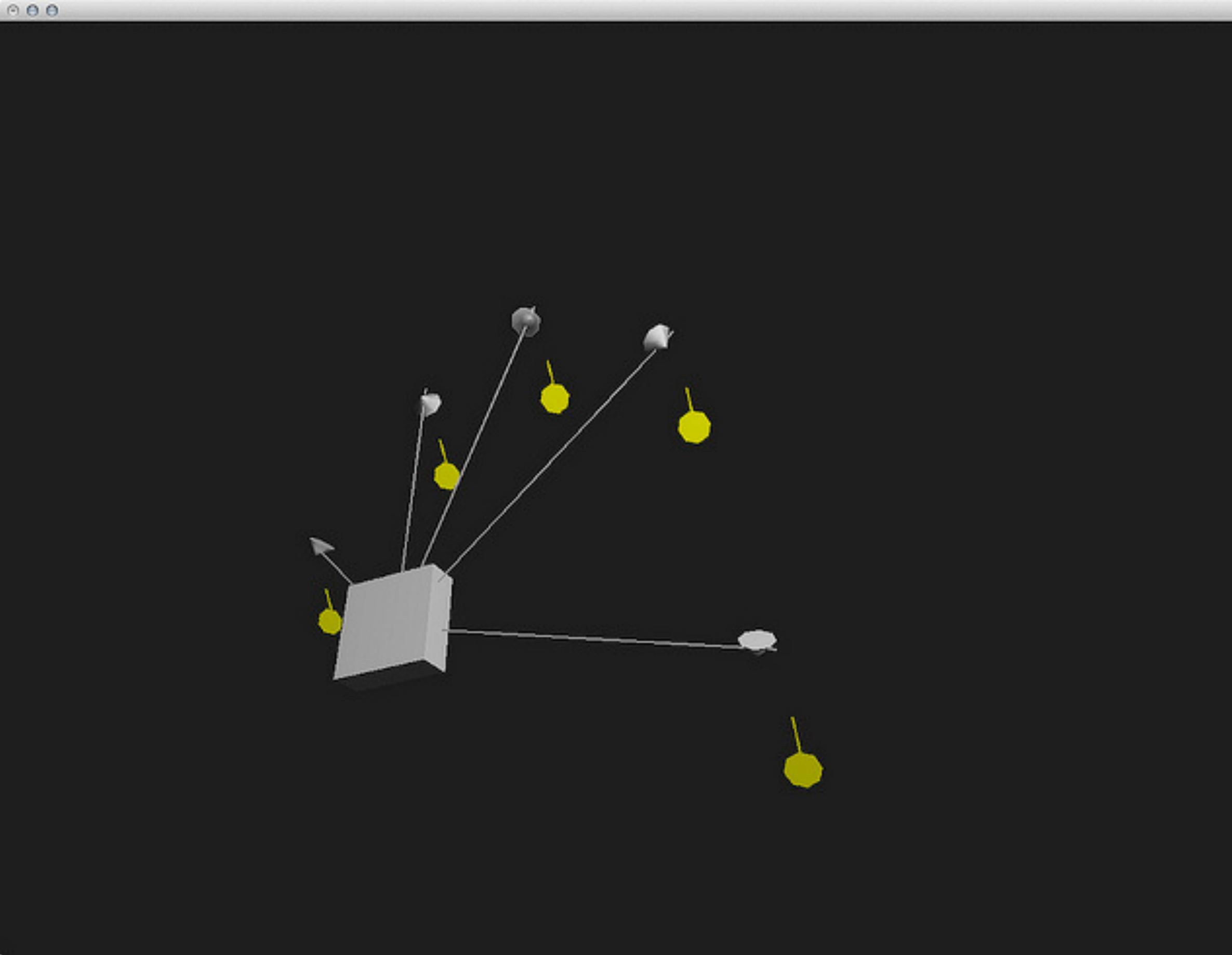
// ofxLeapMotionに現在のフレームは古くなったことを通知
leap.markFrameAsOld();
}

//-----
void ofApp::draw(){
    // 検出された数だけ、手を描画
    cam.begin();
    for(int i = 0; i < simpleHands.size(); i++){
        simpleHands[i].debugDraw();
    }
    cam.end();
}
```

# Leap Motion

---

- ▶ 手の骨格が描画できた!



# Leap Motion

---

- ▶ もう少し詳細に手の情報を取得する
- ▶ ofxLeapMotionでは、あらかじめ用意されているofxLeapMotionSimpleHandクラスを使用すると、とても簡単に手の座標などを取得できる
- ▶ しかし、Leap MotionのSDKで用意されているフルの機能を活用
- ▶ ofxLeapMotionでは、Leap Motion SDKの情報を直接取得できる方法も用意している
- ▶ 詳細は、LeapMotionのDocument参照
- ▶ [https://developer.leapmotion.com/documentation/cpp/devguide/  
Leap\\_Overview.html](https://developer.leapmotion.com/documentation/cpp/devguide/Leap_Overview.html)
  
- ▶ 実際に使ってみる

# Leap Motion

## ▶ ofApp.h

```
#pragma once

#include "ofMain.h"
#include "ofxLeapMotion.h"

class ofApp : 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);

    ofxLeapMotion leap; // Leap Motionのメインクラスをインスタンス化
    vector <ofxLeapMotionSimpleHand> simpleHands; // シンプルな手のモデルのvector配列
    ofEasyCam cam; //カメラ
    ofLight light; //ライト
};
```

# Leap Motion

## ▶ ofApp.cpp

```
#include "ofApp.h"

void ofApp::setup(){
    // 画面設定
    ofSetFrameRate(60);
    ofSetVerticalSync(true);
    ofBackground(31);
    // 照明とカメラ
    ofEnableLighting();
    light.setPosition(200, 300, 50);
    light.enable();
    cam.setOrientation(ofPoint(-20, 0, 0));
    // Leap Motion開始
    leap.open();
}

void ofApp::update(){
    // Leap Motion SDKで用意されている手(Hand)のクラスを取得してvector配列へ
    vector <Hand> hands = leap.getLeapHands();

    // 手が検出されたら
    if( leap.isFrameNew() && hands.size() ){
        // vector配列に記憶した座標をクリア
        fingerPos.clear();
        spherePos.clear();
        sphereSize.clear();

        // 画面の大きさにあわせて、スケールをマッピング
        leap.setMappingX(-230, 230, -ofGetWidth()/2, ofGetWidth()/2);
    }
}
```

# Leap Motion

## ▶ ofApp.cpp

```
leap.setMappingY(90, 490, -ofGetHeight()/2, ofGetHeight()/2);
leap.setMappingZ(-150, 150, -200, 200);

for(int i = 0; i < hands.size(); i++){
    // 指の位置を取得
    for(int j = 0; j < hands[i].fingers().count(); j++){
        ofVec3f pt;
        const Finger & finger = hands[i].fingers()[j];
        pt = leap.getMappedofPoint( finger.tipPosition() );
        fingerPos.push_back(pt);
    }

    // 指がとりかこむ球体を取得
    ofVec3f sp = leap.getMappedofPoint(hands[i].sphereCenter());
    float r = hands[i].sphereRadius();
    spherePos.push_back(sp);
    sphereSize.push_back(r);
}

// ofxLeapMotionに現在のフレームは古くなったことを通知
leap.markFrameAsOld();
}
```

# Leap Motion

## ▶ ofApp.cpp

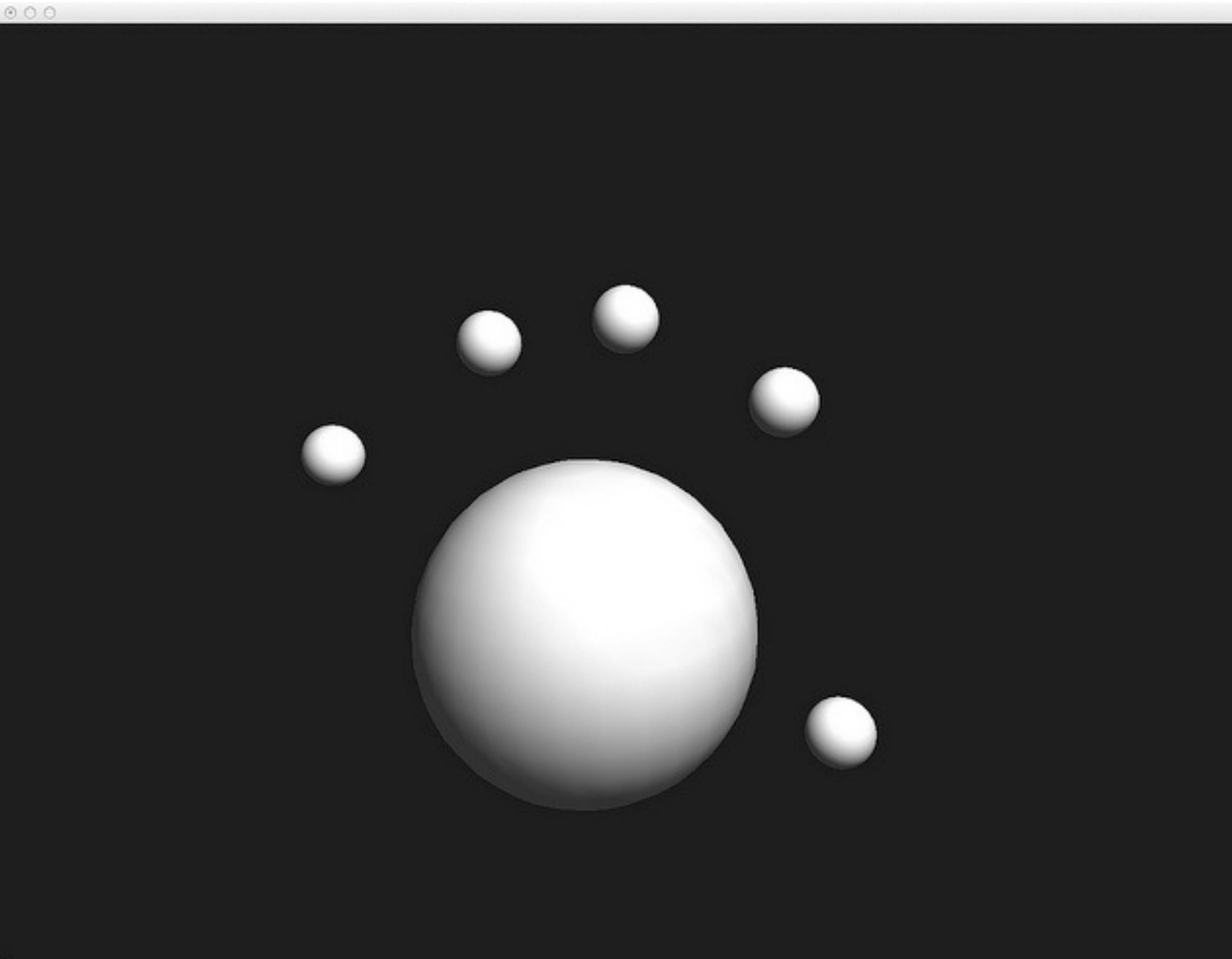
```
void ofApp::draw(){
    ofEnableDepthTest();
    cam.begin();
    // 検出された指の数だけくりかえし
    for(int i = 0; i < fingerPos.size(); i++){
        // 検出された位置に球を描画
        ofSpherePrimitive sphere;
        sphere.setPosition(fingerPos[i].x, fingerPos[i].y, fingerPos[i].z);
        sphere.draw();
    }

    // 検出された指がとりかこむ球
    for(int i = 0; i < spherePos.size(); i++){
        // 検出された位置に球を描画
        ofSpherePrimitive sphere;
        sphere.setPosition(spherePos[i].x, spherePos[i].y, spherePos[i].z);
        sphere.setRadius(sphereSize[i]*1.5); //このスケールは今は目分量...
        sphere.draw();
    }
    cam.end();
}
```

# Leap Motion

---

- ▶ 指先の1本1本を独立して検知



## Leap Motion

---

- ▶ 検出した指の座標をつかって、仮想空間とインタラクションに挑戦
- ▶ 以前作成した、3D空間のパーティクル“ParticleVec3”クラスを流用

# Leap Motion

## ▶ ofApp.h

```
#pragma once

#include "ofMain.h"
#include "ofxLeapMotion.h"
#include "ParticleVec3.h"

class ofApp : public ofBaseApp{

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

(中略)

    ofxLeapMotion leap; // Leap Motionのインスタンス
    ofEasyCam cam; //カメラ
    ofLight light; //ライト
    vector <ofVec3f> fingerPos; // 指の位置の配列
    vector <ofVec3f> spherePos; // 手が取り囲む球体の位置の配列
    vector <float> sphereSize; // 手が取り囲む球体の大きさの配列
    ofVboMesh mesh; // メッシュ
    static const int NUM = 50000; // パーティクルの数
    ParticleVec3 particles[NUM]; // パーティクル配列
};
```

# Leap Motion

## ▶ ofApp.cpp

```
#include "ofApp.h"

void ofApp::setup(){
    // 画面設定
    ofSetFrameRate(60);
    ofSetVerticalSync(true);
    ofBackground(31);
    // 照明とカメラ
    ofEnableLighting();
    light.setPosition(200, 300, 50);
    light.enable();
    cam.setOrientation(ofPoint(-20, 0, 0));
    // Leap Motion開始
    leap.open();
    // パーティクル初期化
    for (int i = 0; i < NUM; i++) {
        particles[i].position.set(ofRandom(-ofGetWidth()/2.0, ofGetWidth()/2.0),
                                  ofRandom(-ofGetHeight()/2.0, ofGetHeight()/2.0),
                                  ofRandom(-ofGetHeight()/2.0, ofGetHeight()/2.0));
        particles[i].friction = 0.005;
        particles[i].minx = -ofGetWidth()/2.0;
        particles[i].maxx = ofGetWidth()/2.0;
        particles[i].miny = -ofGetWidth()/2.0;
        particles[i].maxy = ofGetWidth()/2.0;
        particles[i].minz = -ofGetWidth()/2.0;
        particles[i].maxz = ofGetWidth()/2.0;
    }
    // メッシュ初期化
    mesh.setMode(OF_PRIMITIVE_POINTS);
    static GLfloat distance[] = { 0.0, 0.0, 1.0 };
    glPointParameterfv(GL_POINT_DISTANCE_ATTENUATION, distance);
    glPointSize(3000);
}
```

# Leap Motion

## ▶ ofApp.cpp

```
void ofApp::update(){
    // Leap Motion SDKで用意されている手(Hand)のクラスを取得してvector配列へ
    vector <Hand> hands = leap.getLeapHands();
    // 手が検出されたら
    if( leap.isFrameNew() && hands.size() ){
        // vector配列に記憶した座標をクリア
        fingerPos.clear();
        spherePos.clear();
        sphereSize.clear();
        // 画面の大きさにあわせて、スケールをマッピング
        leap.setMappingX(-230, 230, -ofGetWidth()/2, ofGetWidth()/2);
        leap.setMappingY(90, 490, -ofGetHeight()/2, ofGetHeight()/2);
        leap.setMappingZ(-150, 150, -200, 200);
        for(int i = 0; i < hands.size(); i++){
            // 指の位置を取得
            for(int j = 0; j < hands[i].fingers().count(); j++){
                ofVec3f pt;
                const Finger & finger = hands[i].fingers()[j];
                pt = leap.getMappedofPoint( finger.tipPosition() );
                fingerPos.push_back(pt);
            }
        }
    }
    // ofxLeapMotionに現在のフレームは古くなったことを通知
    leap.markFrameAsOld();
    // メッシュとパーティクル更新
    mesh.clear();
    for (int i = 0; i < NUM; i++) {
        for (int j = 0; j < fingerPos.size(); j++) {
            particles[i].addAttractionForce(fingerPos[j].x, fingerPos[j].y, fingerPos[j].z, ofGetWidth(), 0.1);
        }
        particles[i].update();
        particles[i].throughOffWalls();
        mesh.addVertex(particles[i].position);
    }
}
```

# Leap Motion

## ▶ ofApp.cpp

```
void ofApp::draw(){
    ofEnableDepthTest();
    // カメラ開始
    cam.begin();
    // 検出された指の数だけくりかえし
    ofSetColor(255, 127, 0, 127);
    for(int i = 0; i < fingerPos.size(); i++){
        // 検出された位置に球を描画
        ofSpherePrimitive sphere;
        sphere.setPosition(fingerPos[i].x, fingerPos[i].y, fingerPos[i].z);
        sphere.draw();
    }
    // メッシュ描画
    ofEnableBlendMode(OF_BLENDMODE_ADD);
    ofSetColor(255);
    mesh.draw();
    ofDisableBlendMode();
    // カメラ終了
    cam.end();
}
```

# Leap Motion

---

- ▶ 3D空間のパーティクルと直接触れあうことが可能!!

