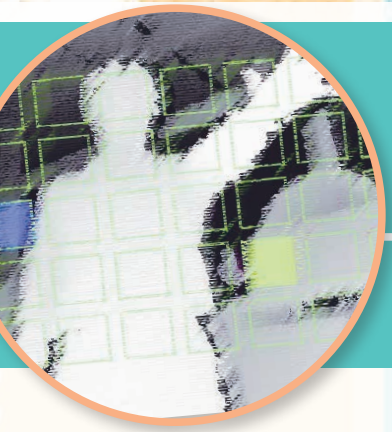

2. Processingではじめる Kinectプログラミング

応用 一般

第1回 Kinectプログラミングはじめの一歩

橋本 直（（独）科学技術振興機構）

Kinectとは

Kinectはマイクロソフト社が販売しているXbox 360向けのゲームデバイスです．全身を使ったジェスチャや音声認識によってゲームをプレイすることができます．Kinectは発売から60日間で800万台を販売し，「最も速く売れているコンシューマ向け電子機器」としてギネスブックにも認定されました．Kinectの発売後，すぐにハッカーたちによって解析が行われ，非公式にドライバやライブラリが作られました．これにより，Kinectを使ったアート作品や学術研究がまたたく間に生まれました．現在では，マイクロソフト社から公式の開発環境（Kinect for Windows SDK）も提供されています．インターネットの動画投稿サイトで検索すれば，たくさんのKinect作品が出てきます．ゲームやエンタテインメントだけでなく，医療現場やファッションなどでの応用事例も見つけることができます．いまやKinectを使った開発ブームの波は最高潮に達していると言っても過言ではないでしょう．

本稿ではKinectを使ったプログラミングについて2回にわたって解説します．プログラミング経験があまりない人でも手軽に楽しんでいただけるよう，プログラミング言語としてProcessingを用いました．第1回ではKinectが提供する機能について簡単に説明した後，プログラミング環境のセットアップ方法と基本的なデータの取得方法について解説します．第2回では骨格認識について解説し，ジェスチャ操作のアプリケーションの作り方を紹介します．

Kinectのハードウェア構成

プログラミングの話に入る前にKinectの仕組みについて簡単に説明します．Kinectは普通のRGBカメラに加え，赤外線カメラと赤外線レーザプロジェクタを搭載しています（図-1）．赤外線レーザプロジェクタからは人間の目には見えない赤外線のドットパターンが投影されます．これを赤外線カメラで読み取ることによって空間方向の奥行きを計測しています．この手法は「Light Coding」と呼ばれています．

Kinectはカメラ以外にもいろいろな入出力デバイスを搭載しています．音声認識と音源の位置推定を行うために4個のマイクを内蔵しています．また，本体の傾きを検知し，自動的に姿勢を修正するための仕組みとして，加速度センサとモータを備えています．本体の動作状態を表示するためにLEDが付いており，これもPC側のソフトウェアから制御できる仕様になっています．これらのハードウェア機能の利用の可否は，プログラミングに用いるライブラリの対応状況に依存します．

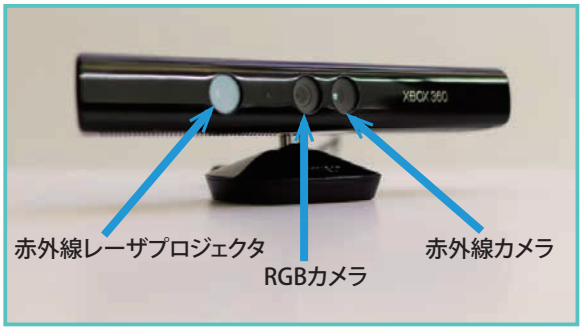

図-1 Kinectの外観

Kinect から得られるデータ

Kinect から得られるデータを以下に示します．ライブラリによって未サポートのものもありますが，基本的にカラー画像とデプスマップはどのライブラリでもサポートされています．

▶ カラー画像

RGB カメラから得られる画像です．画素ごとに RGB（赤・緑・青）の色情報を持っています．

▶ デプスマップ，深度画像

デプスマップは画素ごとに奥行きの距離情報を持つ 2 次元のマップデータです．値の範囲を 0 ～ 255 に正規化し，画像にしたものが深度画像です．この情報をもとに，物体までの距離や物体の形状を知ることができます．

▶ 赤外線画像

赤外線カメラで撮影されたグレイスケールの画像です．赤外線レーザプロジェクタから投影されたドットパターンを見ることができます．

▶ 人物領域情報

深度画像から人物領域のみを抽出して作成した画像データです．画素値がユーザ ID になっています．検出可能なユーザ数はライブラリに依存しますが，今回使用する simple-openni では，最大 15 人のユーザを検出・トラッキングすることができます．

▶ 骨格情報

人物領域の情報をもとに計算された関節の位置情報です．頭・手・肩・胴・脚などの関節について，画像上の座標値（x，y）と，空間中の座標値（X，Y，Z）の両方を得ることができます．

Processing のインストール

Processing はメディアアートとビジュアルデザインのために開発されたオープンソースのプログラミング言語です． マサチューセッツ工科大学のメディアラボに所属していた Casey Reas と Ben Fry によって開発されました．ラピッドプロトタイピングを意識して作られた言語なので，やりたいことを非常に短いコードで実現することができます．Processing は公式サイトのダウンロードページ（http://processing.org/download/）から無償でダウンロードすることができます．Windows 版は無印と [Without Java] の 2 種類がありますが，JDK（Java Development Kit）のバージョンについて特にケアしない場合は前者をダウンロードしてください．インストールは Windows，Mac ともに，ダウンロードした ZIP ファイルを解凍して出てきたものを任意のディレクトリに置くだけです．

simple-openni のインストール

Processing で Kienct を使ったプログラミングを行うためのライブラリは複数ありますが，今回は多くの機能が利用できる simple-openni というライブラリを使います．simple-openni は，PrimeSense 社が開発している OpenNI というライブラリを Processing で扱えるようにしたラッパーです．OpenNI の NI は Natural Interaction（自然なインタラクション）の略で，ジェスチャや姿勢認識を使ったインタラクションを意味します．OpenNI を使ったプログラミングでは C++ や C# についてある程度の技量が要求されますが，simple-openni は Processing 向けにカスタマイズされているため，非常に簡単に扱うことができるようになっています．simple-openni は http://code.google.com/p/simple-openni/ で配布されています．以下，執筆時点の最新版である Ver.0.26 のインストール方法について説明しますが，以降のバージョンでインストール方法が変わる場合がありますので，公式ページの「Installation」の説明を適宜参照してください．

Windows でのインストール

simple-openni のダウンロードページから OpenNI_NITE_Installer-Win32-0.26.zip をダウンロードします．このファイルを解凍すると，OpenNI，NITE，Kinect のドライバ，Kinect 互換製品のドライバの 4 つのインストーラが出てきますので，そ

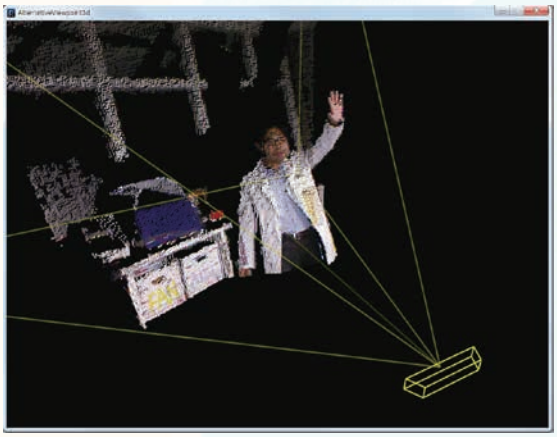

図-2　空間の奥行き情報を 3D 表示するデモ

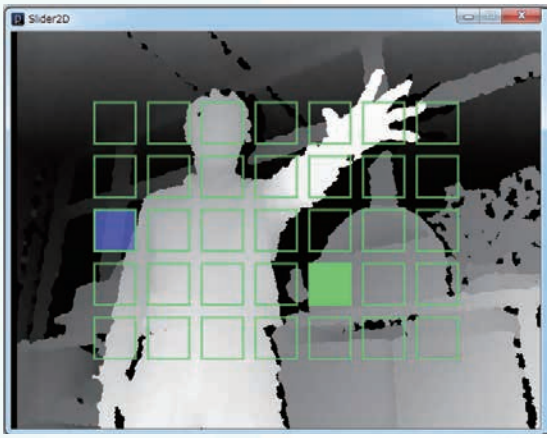

図-3　手の動きを使ってタイルを操作するデモ

れぞれ実行してください．64bit 版 Windows を使用している場合は，OpenNI_NITE_Installer-Win64-0.26.zip をダウンロードしてインストールを行ってください．また，Processing がインストールされているフォルダから「Java」というフォルダを削除した後，JDK の Web サイトから 64bit 版 JDK を入手してインストールしてください．

次に SimpleOpenNI-0.26.zip をダウンロードしてください．このファイルを解凍して出てきた「SimpleOpenNI」フォルダを「C:¥Users¥ ユーザ名 ¥Documents¥Processing¥libraries」に移動します．「Processing」というフォルダは Processing の初回起動時に「ドキュメント」フォルダの中に自動作成されますが，「libraries」というフォルダは初期状態では存在しないので作成してください．

■ Mac でのインストール

simple-openni のダウンロードページから OpenNI_NITE_Installer-OSX-0.24.zip をダウンロードします．ファイルを解凍したら，ターミナルを起動して解凍してできたフォルダに移動してください．

```
> cd ./OpenNI_NITE_Installer-OSX
```

以下のコマンドでインストールを実行します．

```
> sudo ./install.sh
```

次に，SimpleOpenNI-0.26.zip をダウンロードします．このファイルを解凍して出てきた「SimpleOpenNI」フォルダを「/Users/ ユーザ名 /Documents/Processing/libraries」に移動します．「Processing」というフォルダは Processing の初回起動時に「書類」フォルダの中に自動作成されますが，「libraries」というフォルダは初期状態では存在しないので作成してください．

サンプルを実行してみよう

simple-openni のインストールが済んだら，動作確認を兼ねてサンプルで遊んでみましょう．Kinect の USB ケーブルを PC に挿し，電源アダプタをコンセントに接続したら Processing を起動してください．メニューから「File」→「Examples...」を選択するとサンプル一覧が表示されます．ツリーの下の方にある「Contributed Libraries」の中に「SimpleOpenNI」のサンプルがあります．項目をダブルクリックするとサンプルが開きますので，Run ボタンを押して実行してください．空間の奥行きを 3D 表示するデモ（AlternativeViewpoint3d）や，手の動きを使ってタイルを操作するデモ（Slider2d）などが入っています（図-2，図-3）．

プログラムを書いてみよう

実際に自分の手でプログラムを書いてみましょう．今回はカラー画像，深度画像，赤外線画像の取得方法と，人物領域の抽出方法について紹介します．

図 -4　カラー画像の描画

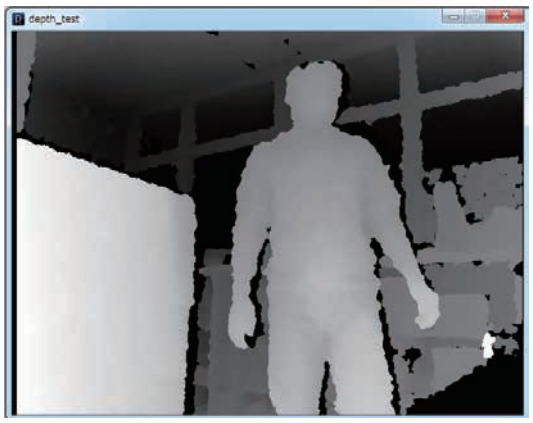

図 -5　深度画像の描画

■ カラー画像の取得

Kinect からカラー画像を取得して表示するプログラムをリスト 1 に示します．非常に短いコードで実現できていることがお分かりいただけると思います．これが Processing で書くことの魅力です．実行結果は図 -4 のようになります．

［リスト 1　カラー画像を表示するプログラム］

```
import SimpleOpenNI.*;
SimpleOpenNI kinect;

void setup () {
  size (640, 480) ; // 画面サイズの設定
  kinect = new SimpleOpenNI (this) ; // 初期化
  kinect.enableRGB () ; // カラー画像の有効化
}

void draw () {
  background (0) ; // 背景の初期化
  kinect.update () ; // データの更新
  image (kinect.rgbImage (), 0, 0) ; // 画像描画
}
```

■ 深度画像，デプスマップの取得

次は Kinect の主役である深度画像を表示させてみましょう．enalbeDepth() によって深度画像を有効にし，depthImage() によって深度画像を取得します．リスト 2 を実行すると，図 -5 のような画像が表示されます．

［リスト 2　深度画像を表示するプログラム］

```
import SimpleOpenNI.*;
SimpleOpenNI kinect;

void setup () {
  size (640, 480) ; // 画面サイズの設定
  kinect = new SimpleOpenNI (this) ; // 初期化
  kinect.enableDepth () ; // 深度画像の有効化
}

void draw () {
  background (0) ; // 背景の初期化
  kinect.update () ; // データの更新
  image (kinect.depthImage (), 0, 0) ; // 描画

  // マウスカーソルがある場所の距離を表示
  int[] depthMap = kinect.depthMap () ;
  int index = mouseX + mouseY * width;
  int d = depthMap[index];
  println (d + " mm") ;
}
```

このプログラムではデプスマップも取得し，マウスカーソルがある場所の距離情報をコンソールに表示しています．depthMap() から得られる距離データの単位は mm（ミリメートル）です．

■ 赤外線画像の取得

赤外線画像を表示させて，赤外線レーザプロジェクタから投影されている見えないドットパターンを見てみましょう．赤外線画像を有効にするには enableIR() を，赤外線画像を表示するには irImage() を使います．リスト 3 を実行すると，図 -6 のよう

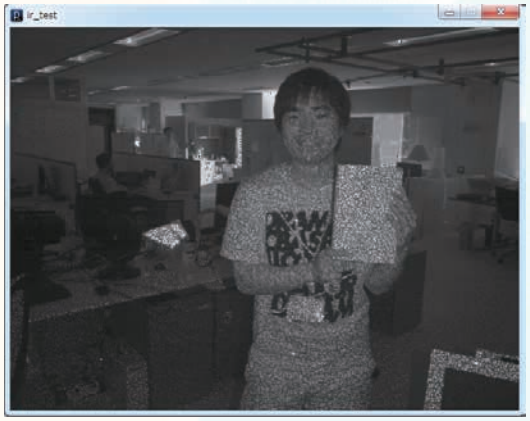

図 -6　赤外線画像の描画

な画像が表示されます．

［リスト 3　赤外線画像を表示するプログラム］

```
import SimpleOpenNI.*;
SimpleOpenNI kinect;

void setup () {
  size (640, 480) ; // 画面サイズの設定
  kinect = new SimpleOpenNI (this) ; // 初期化
  kinect.enableIR () ; // 赤外線画像の有効化
}

void draw () {
  background (0) ; // 背景の初期化
  kinect.update () ; // データの更新
  image (kinect.irImage () , 0, 0) ; // 描画
}
```

■ 人物領域の抽出

一般的なカメラを用いた人物領域の抽出では，緑色で塗られた背景の前に人物を立たせて画像中の緑色以外の領域を人物領域とする方法（クロマキー法）や，あらかじめ人物がいない状態で背景を撮影しておき，人物が立ったときに変化があった領域だけ抽出する方法（背景差分法）が用いられます．クロマキー法では緑色の服を着ることができない，背景差分法では照明条件が変わるとノイズが多くなるなどの問題がありますが，Kinect の深度データに基づく人物抽出ではそのような問題が起こらないのでとても有用です．

カラー画像から人物領域だけを抽出するプログラムをリスト 4 に示します．実行結果は図 -7 のようになります．

［リスト 4　人物領域を抽出するプログラム］

```
import SimpleOpenNI.*;
SimpleOpenNI kinect;

void setup () {
  size (640, 480) ;
  kinect = new SimpleOpenNI (this) ;
  kinect.enableRGB () ;
  kinect.enableDepth () ;
  kinect.setMirror (true) ;
  kinect.alternativeViewPointDepthToImage () ;
  kinect.enableUser (SimpleOpenNI.SKEL_PROFIL
E_ALL) ;
}

void draw () {
  background (255) ;
  kinect.update () ;
  int[] userMap = null;
  int userCount = kinect.getNumberOfUsers () ;
  if (userCount > 0) {
    userMap = kinect.getUsersPixels (
                    SimpleOpenNI.USERS_ALL) ;
  }
  loadPixels () ;
  for (int y=0; y<kinect.rgbHeight () ; y++) {
    for (int x=0; x<kinect.rgbWidth () ; x++) {
      int i = x + y * kinect.rgbWidth () ;
      if (userMap != null && userMap[i] > 0)
      pixels[i]=kinect.rgbImage () .pixels[i];
    }
  }
  updatePixels () ;
}
```

simple-openni では，深度データから複数のユーザを検出し，トラッキングを行うことができます．ユーザのトラッキングを有効化するには，enableUser() というメソッドを使います．引数 に SimpleOpenNI.SKEL_PROFILE_ALL を 与 えたときは全身のすべての部位を追跡します．ユーザトラッキング処理によって検出されたユーザの数は getNumberOfUsers() で取得することができます．画面内にユーザが 1 人以上映っていれば，

getUsersPixels（SimpleOpenNI.USERS_ALL）によって人物領域情報（ユーザ ID を画素値とする画像データ）を得ることができます．このプログラムでは，画面上の画素ごとに人物領域情報（userMap）をチェックし，ユーザ ID が 0 よりも大きい画素（＝ユーザがいる画素）のときに，カラー画像を画面に描画しています．ここで特定のユーザ ID のときだけ描画するようにすれば，特定のユーザだけを表示させることができます．

人物領域情報は深度画像に基づいて作成されます．その深度画像を撮影している赤外線カメラと，カラー画像を撮影している RGB カメラは視点の位置が異なるため，単純に 2 つを重ねただけでは人物の領域が一致しません．これを補正するために，alternativeViewPointDepthToImage() というメソッドを setup() 関数内で実行しています．

setMirror() は画像のミラー反転（左右反転）を設定するためのメソッドです．引数が true のときに各種画像データが左右反転します．Kinect を使ったアプリケーションではカメラとユーザが向かい合わせになるため，ミラー反転をかけたほうが何かと都合が良いです．

映像から人物領域を抽出できるようになると，ゲーム画面の中に自分をそっくりそのまま登場させたり，遠隔地にいる人の姿を壁に投影して遠隔コミュニケーション（テレイグジスタンス）のアプリケーションを作ったりというように応用の幅が一気に広がります．

図 -7　人物領域の抽出結果

次回は

はじめての Kinect プログラミングはいかがでしたでしょうか？　今回は Kinect からいろいろな画像データを取得する方法を紹介しました．次回は骨格認識の方法について解説し，ジェスチャ認識を用いた応用作品を紹介します．どうぞお楽しみに．

（2012 年 4 月 17 日受付）

橋本 直（正会員）｜hashimoto@designinterface.jp

2009 年九州工業大学大学院工学研究科博士後期課程修了．博士（工学）．同年より（独）科学技術振興機構 ERATO 五十嵐デザインインタフェースプロジェクト研究員．人とロボットのインタフェースに関する研究に従事．