2013

Interacting with Leap motion with Processing Software

Author: Ram

Introduction:

This application note mainly focuses on getting access to the leap motion sensor data through a processing app as well as sending it through over the PC on various peripheral interfaces like the Serial port as well as TCP ports. This application may serve well as as good bridge in between connecting more external sources/sinks to the leap motion sensor.

Leap motion sensor

Leap motion sensor gives a whole new dimensio as well as a choice to explore and buld more promising systems that may explore gestures as an human interaction to systems. The leap motion captures more in details on what it offers so this application note is not focused on that and would lay more focus on the programming aspects on application development and possibly serve as a good starting point to connect it to embedded hardware like the Arduinos and the like !

For more info on the Leap motion device visit

https://www.leapmotion.com/

Required Software :

- Leap motion Software.
- Processing 2.0
- Leap Motion for Processing – Processing library .

Installation procedure:

Leap motion Software :

The Installation procedure is quite simple and is well explained at

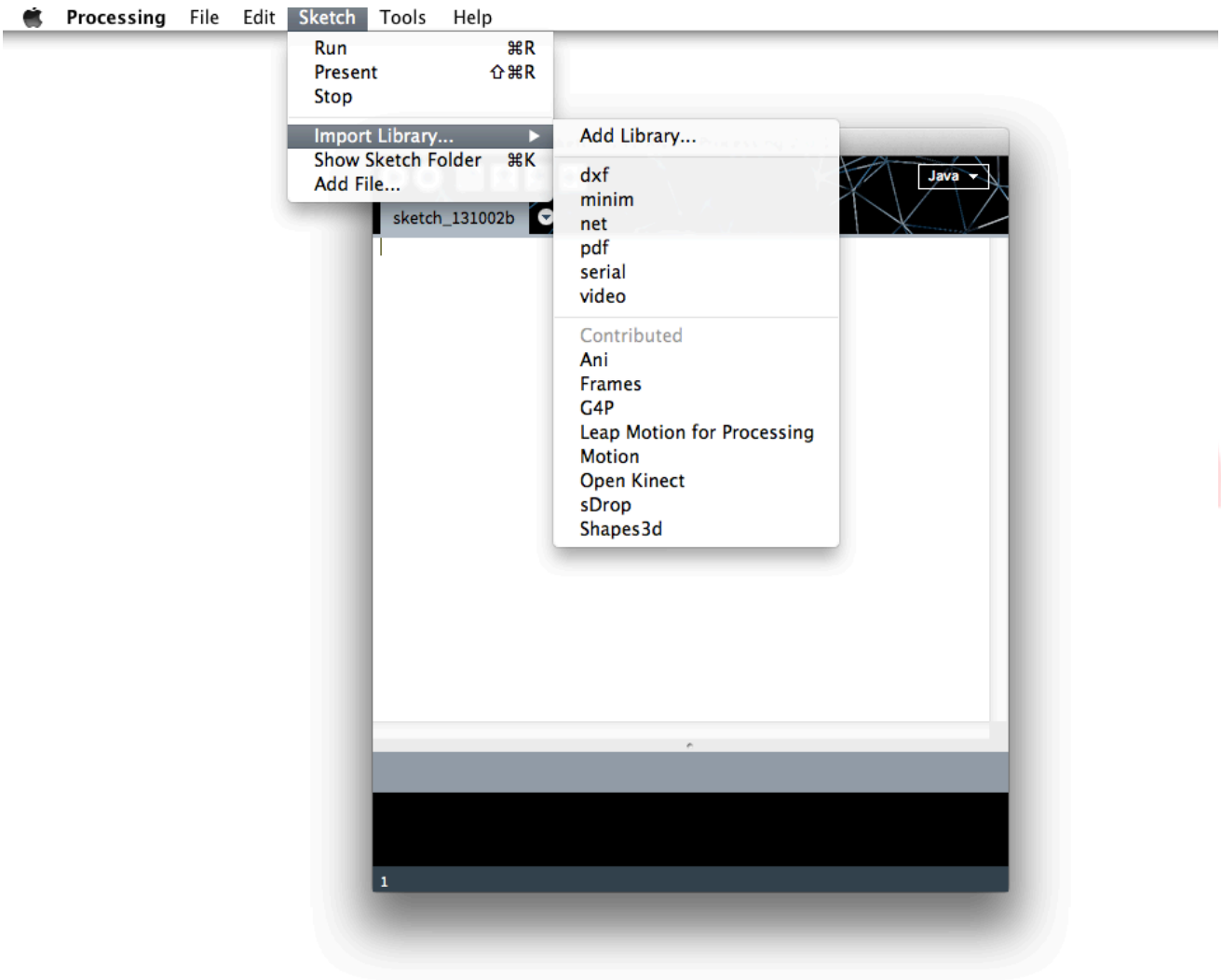
https://www.leapmotion.com/setup . Unfortunately the installation is still not available on the linux machines though so easily even though there are some posts on the web that show how to do that !

Processing 2.0 Software :

Processing software could be obtained from http://processing.org/ . Please make a note to download the appropriate executable(with/without java depending on whether you have java installed on your machine) .

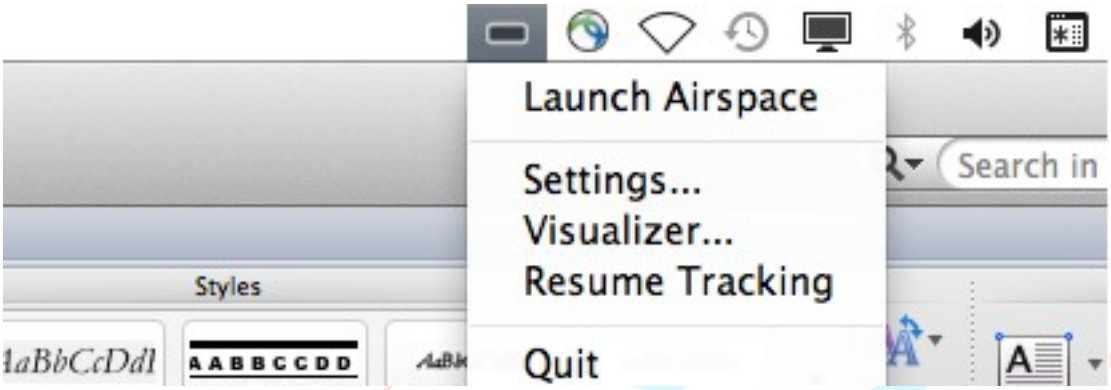
Installing the Leap Motion for Processing library in processing

Step 1: Open the Processing software.
Step 2: From the **Sketch -> Import Library** menu click on **Add Library ...**



Step 3: Search for the “leap Motion for Processing “ Library and click on install
Step 4: Restart processing so that the library installation is picked up.

Start the leap motion tracking by opening up the leap motion app and resume the tracking.



Processing Code for the project

```
//All credits to developers that enabled easy access to leap motion sdk via this simple processing library.
import de.voidplus.leapmotion.*;
////////////////////////////////////////////

import processing.serial.*;
import processing.net.*;

int enable_serial=0; //Set enable_serial to 1 if you plan to send the signals through a serial port.
int enable_tcp=1;

Server s;
Serial myPort;
PFont f;
LeapMotion leap;



void setup(){
  size(800, 500, P3D);
  background(255);
  f = createFont("Georgia", 24);
  textFont(f);
  textAlign(CENTER, CENTER);
  leap = new LeapMotion(this).withGestures();
  background(0);
  text("Leap Demo App created by Tenet Technetronics. \n \n Sends Stream on Tcp port 56565 \n (data sent for gestures
in braces): \n Swipe(1) Circle(2) ScreenTap (3) KeyTap (4) ",400,250);

  // TO DO ask the user to put the right comport number into this !!!!
  if(enable_serial==1)
  {
  String portName = Serial.list()[0]; //Change the index 0 to any other if the serial port on your pc is different.
  myPort = new Serial(this, portName, 9600);
  }

  //TCP code begins
  if(enable_tcp==1)
  {
  s = new Server(this, 56565); //Change the port number as per your needs.
  }
}

void draw(){

}

// SWIPE GESTURE
void leapOnSwipeGesture(SwipeGesture g, int state){
  int      id             = g.getId();
  Finger   finger         = g.getFinger();
  PVector  position       = g.getPosition();
  PVector  position_start  = g.getStartPosition();
  PVector  direction      = g.getDirection();
  float    speed          = g.getSpeed();
  long     duration       = g.getDuration();
  float    duration_seconds = g.getDurationInSeconds();


  switch(state){
    case 1: // Start
      break;
    case 2: // Update
      break;
    case 3: // Stop
      background(0);
     text("Swipe !!!",450,250);
     if(enable_serial==1)
     {
     myPort.write(1);
     }
```

```
    if(enable_tcp==1)
     {
    s.write(1);
     }
     break;
  }
}

// CIRCLE GESTURE
void leapOnCircleGesture(CircleGesture g, int state){
  int      id              = g.getId();
  Finger   finger          = g.getFinger();
  PVector  position_center  = g.getCenter();
  float    radius          = g.getRadius();
  float    progress        = g.getProgress();
  long     duration        = g.getDuration();
  float    duration_seconds = g.getDurationInSeconds();

  switch(state){
    case 1: // Start
      break;
    case 2: // Update
      break;
    case 3: // Stop
      background(0);
      text("Circle !!! ",450,250);
    if(enable_serial==1)
    {
     myPort.write(2);
    }
     if(enable_tcp==1)
  {
    s.write(2);
  }
     break;
  }
}

// SCREEN TAP GESTURE
void leapOnScreenTapGesture(ScreenTapGesture g){
  int      id              = g.getId();
  Finger   finger          = g.getFinger();
  PVector  position        = g.getPosition();
  PVector  direction       = g.getDirection();
  long     duration        = g.getDuration();
  float    duration_seconds = g.getDurationInSeconds();
  background(0);
  text("Screen Tap !!! ",450,250);
   if(enable_serial==1)
   {
  myPort.write(3);

   }
    if(enable_tcp==1)
  {
   s.write(3);
  }

}

// KEY TAP GESTURE
void leapOnKeyTapGesture(KeyTapGesture g){
  int      id              = g.getId();
  Finger   finger          = g.getFinger();
  PVector  position        = g.getPosition();
  PVector  direction       = g.getDirection();
  long     duration        = g.getDuration();
  float    duration_seconds = g.getDurationInSeconds();
  background(0);
  text("Key Tap !!! ",450,250);
   if(enable_serial==1)
   {
  myPort.write(4);
```

```
  }
   if(enable_tcp==1)
 {
 s.write(4);

  }
}
```

Usage of the code:

With the leap motion hardware connected to the host system & running the processing code, appropriate gesture should be reported on screen as well as a corresponding letter is sent on the TCP port. If some one needs to control the hardware say an Arduino with this code. One could enable the serial port code and send the commands over serial to the Arduino as well.

Some Snap shots of the processing app in action is shown below: