The Death of a Mouse

Geert Bevin - XRebel Product Manager

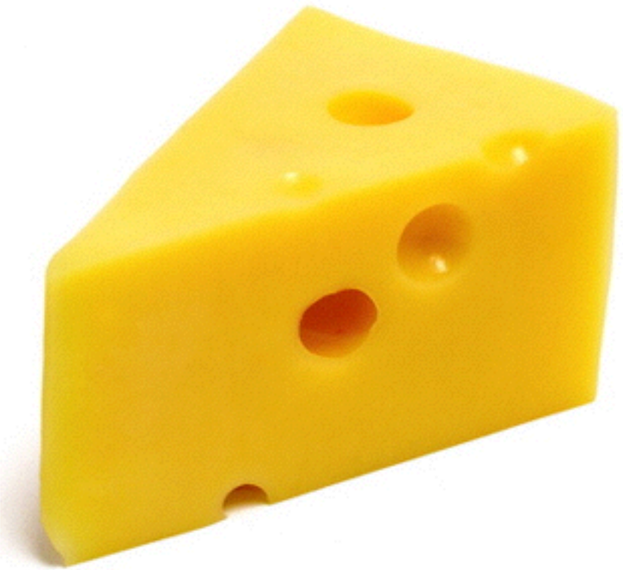
ZEROTURNAROUND

Who am I?

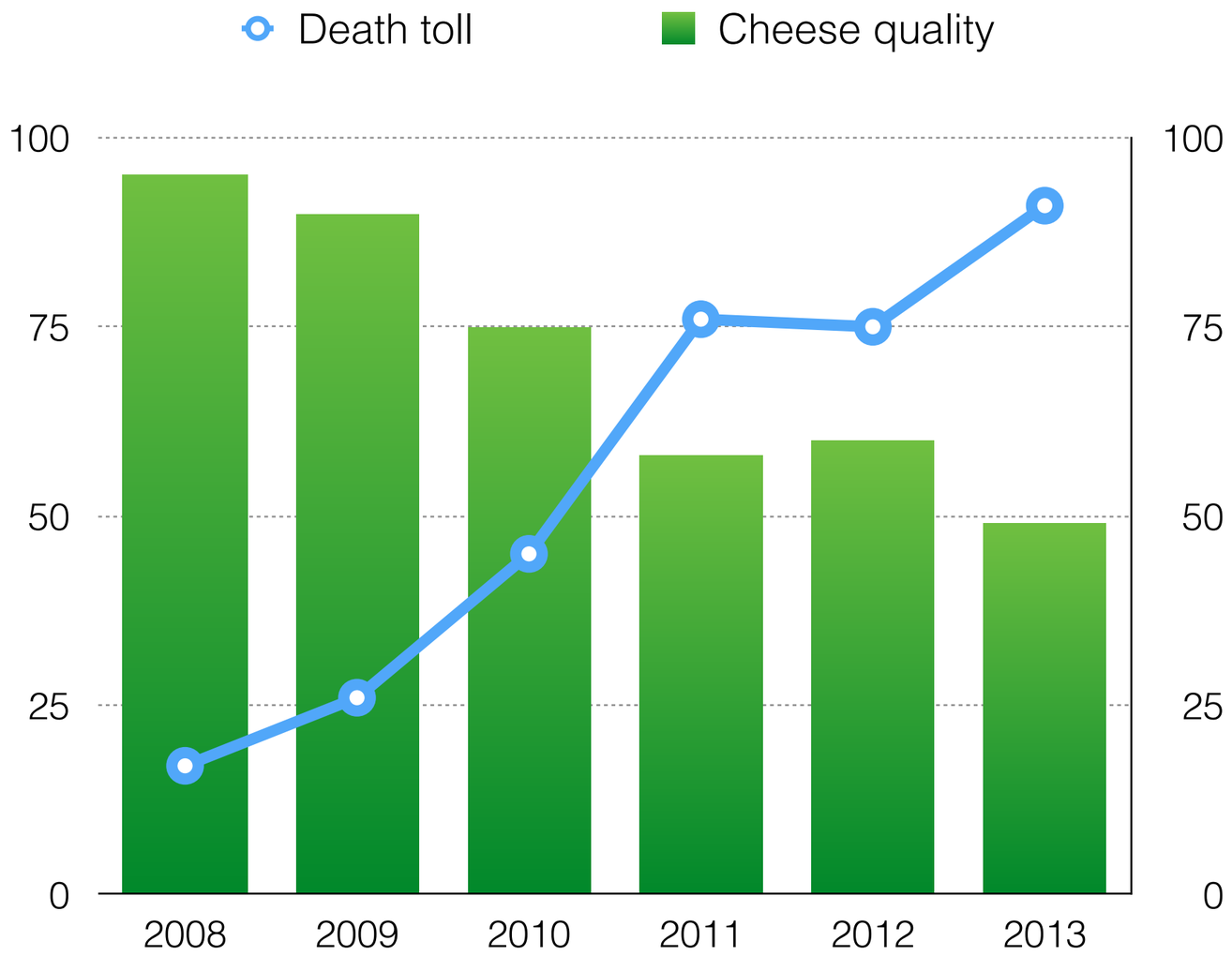
- Geert Bevin
- XRebel Product Manager at ZeroTurnaround
- Java Champion
- Creator of RIFE framework, native Java continuations
- Many other open-source projects

The decline of cheese

2008



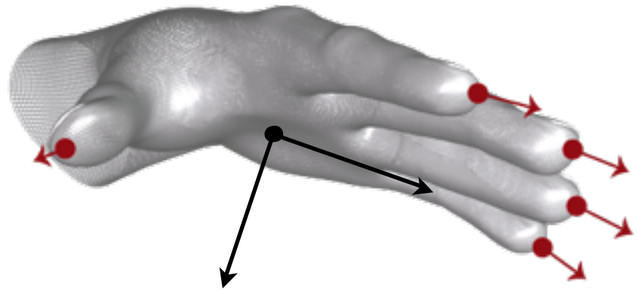
2013


Death toll
Cheese quality
100
75
50
25
0
100
75
50
25
0
2008
2009
2010
2011
2012
2013

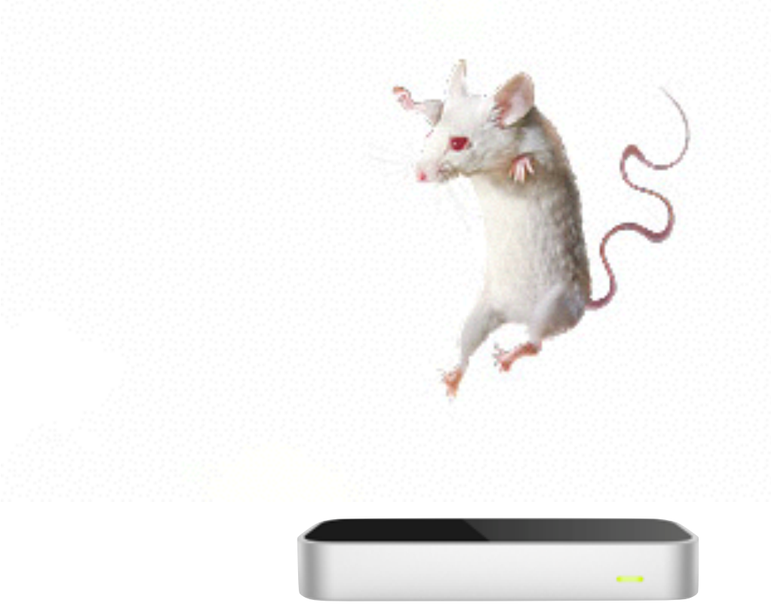
What qualifies me?

- 3 of the most popular applications in the LeapMotion Store
- GameWAVE : play existing keyboard-mouse games
- GECO : make music with hand gestures
- HandWAVE : simple computer control
- Total of more than 40000 users
- Alpha developer for Thalmic Myo, iOS music app on the way

Leap Motion Controller



Leap Motion Controller

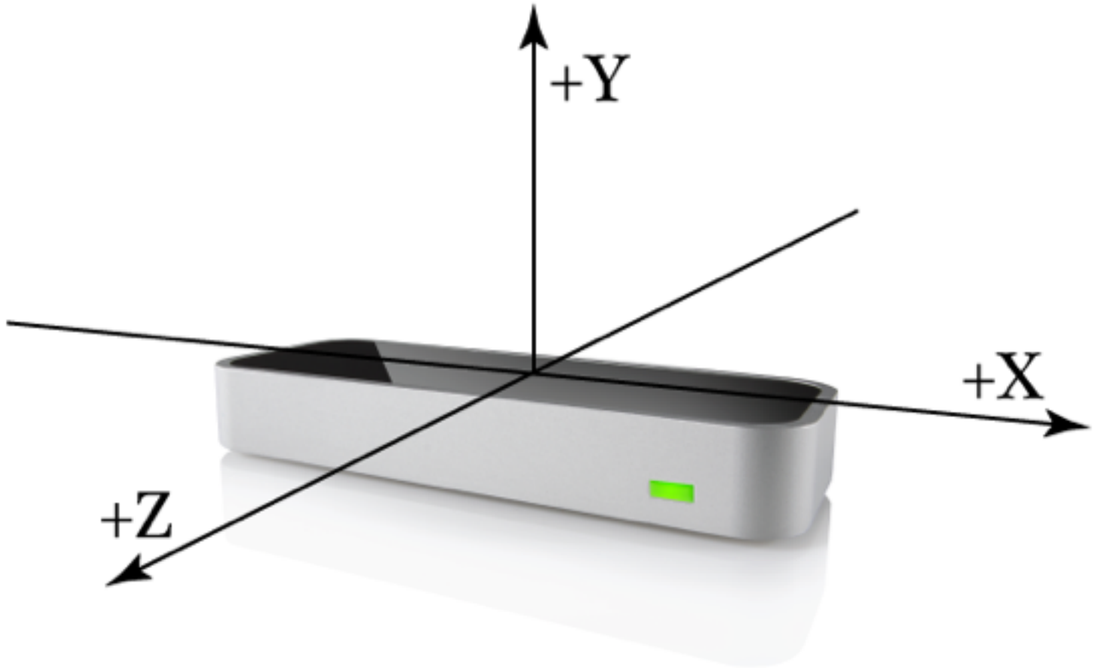


Tracking demo

Technical overview

Coordinate system

right-handed Cartesian
25 to 600 millimeters above device



Motion tracking data

- Unique ID for each visible entity

- Frames
- Hands
- Fingers
- Bones

- Gestures
- Tools
- Images

Very precise

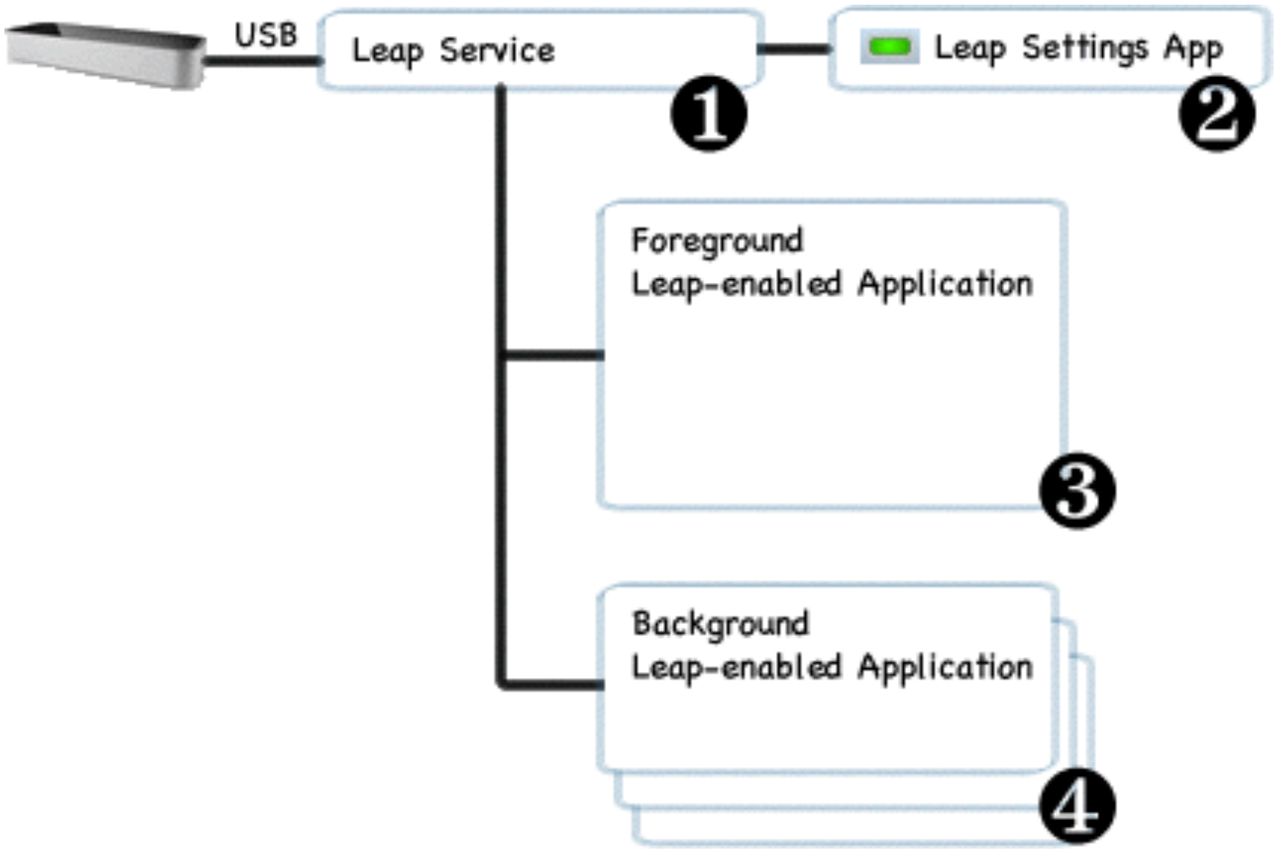
- 10µm of real resolution (1/100th of a mm)
- 200x more accurate than Kinect
- Adaptive sample rates: 60-300 hz, default 120 hz
- Data rate is preserved from end to end

Many supported languages

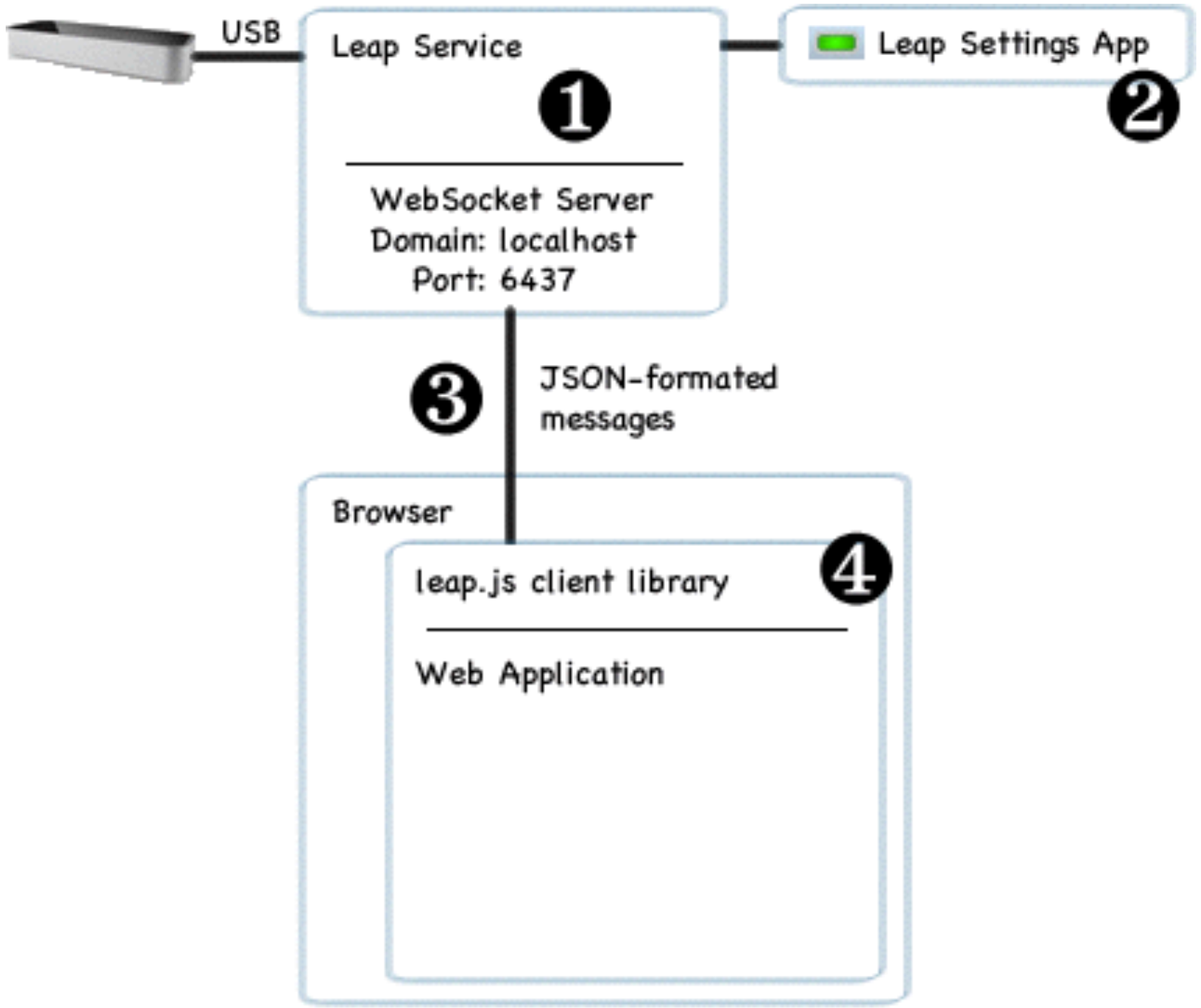
- C++
- Objective-C
- C#
- Java
- Python
- JavaScript

Architecture

Application Interface



WebSocket Interface



Basic Java example

Java example : part 1

```
import java.io.IOException;
import com.leapmotion.leap.*;

class Sample {
    public static void main(String[] args) {
        SampleListener listener = new SampleListener();
        Controller controller = new Controller();

        controller.addListener(listener);

        System.out.println("Press Enter to quit...");
        try { System.in.read(); }
        catch (IOException e) { e.printStackTrace(); }

        controller.removeListener(listener);
    }
}
```

Java example : part 2

```
class SampleListener extends Listener {
    public void onInit(Controller c)       {}
    public void onDisconnect(Controller c) {}
    public void onExit(Controller c)       {}

    public void onConnect(Controller controller) {
       controller.enableGesture(Gesture.Type.TYPE_SWIPE);
    }

    public void onFrame(Controller controller) {
        Frame frame = controller.frame();

        // Your processing logic
        System.out.println("Frame id: " + frame.id()
            + ", timestamp: " + frame.timestamp()
            + ", hands: " + frame.hands().count()
            + ", gestures: " + frame.gestures().count());
    }
}
```

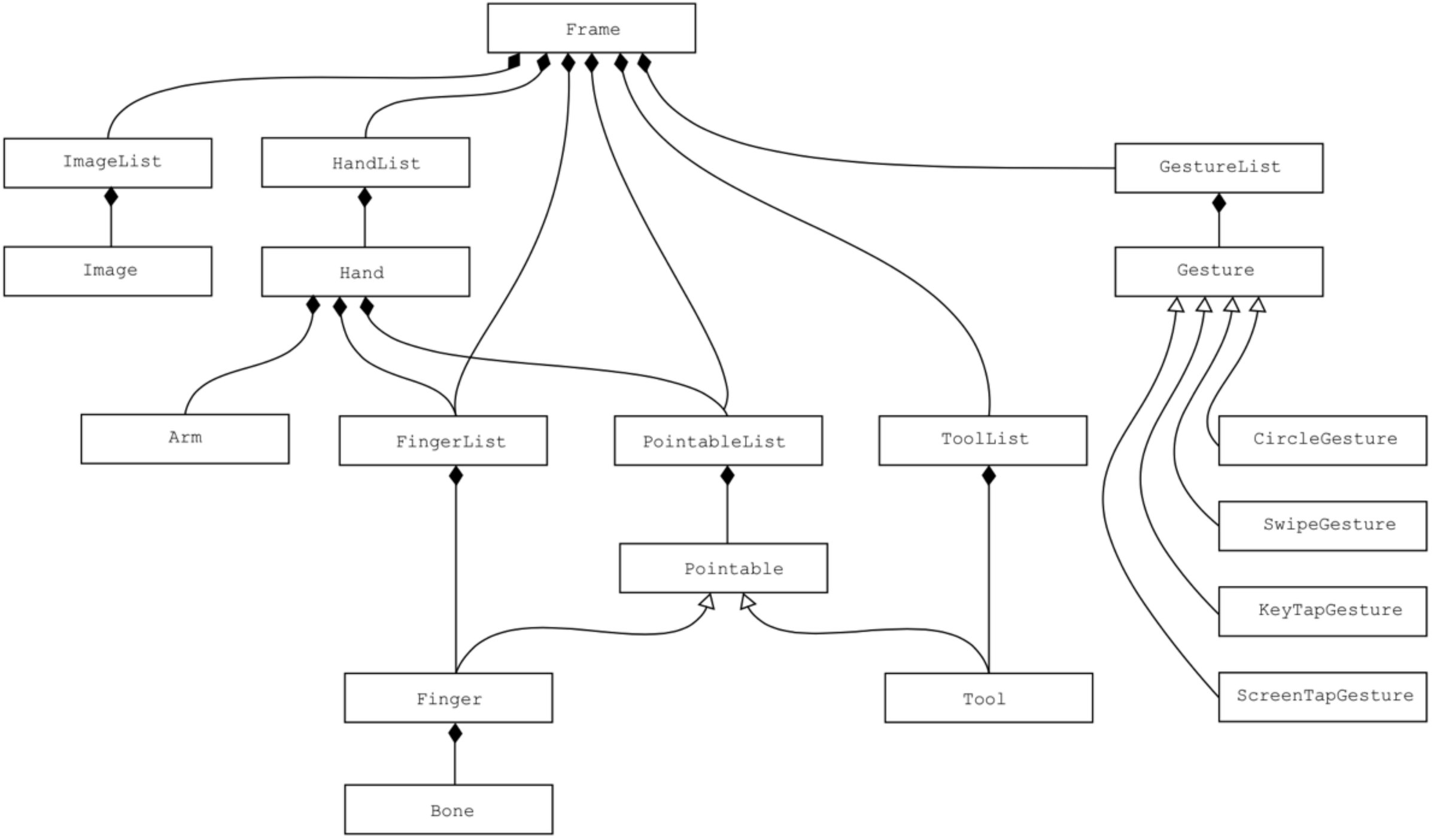
Compile and Run

```
javac -classpath <LeapSDK>/lib/LeapJava.jar Leap.java
```

```
java -classpath <LeapSDK>/lib/LeapJava.jar:.
     -Djava.library.path=<LeapSDK>/lib
     Leap
```

GameWAVE demo

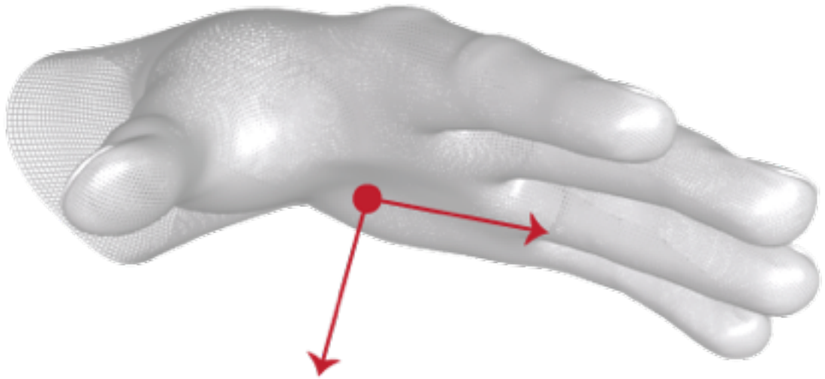
Leap data models


Frame
ImageList
HandList
GestureList
Image
Hand
Gesture
Arm
FingerList
PointableList
ToolList
CircleGesture
SwipeGesture
Pointable
KeyTapGesture
Finger
Tool
ScreenTapGesture
Bone

Frame Motion Analysis

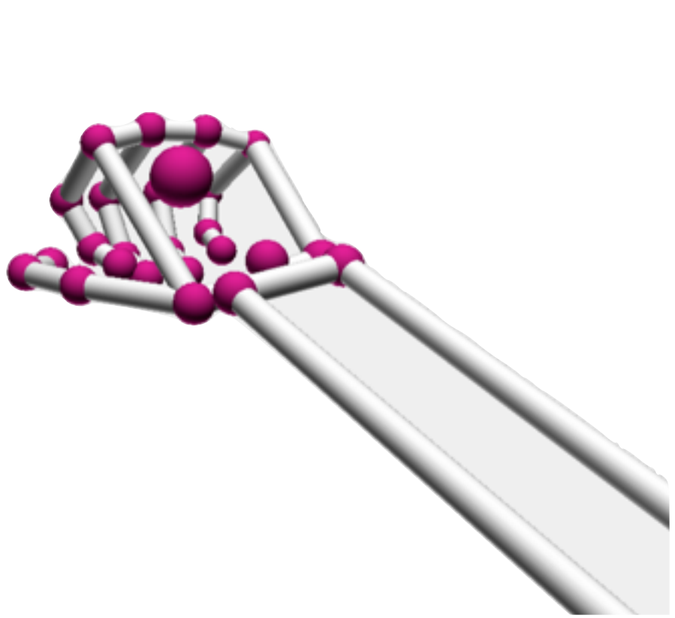
- Compare the current frame with a specified earlier frame
- **rotationAxis**: direction vector expressing the axis of rotation
- **rotationAngle**: angle of rotation clockwise around the rotation axis (using the right-hand rule)
- **rotationMatrix**: transform matrix expressing the rotation
- **scaleFactor**: factor expressing expansion or contraction
- **translation**: vector expressing the linear movement

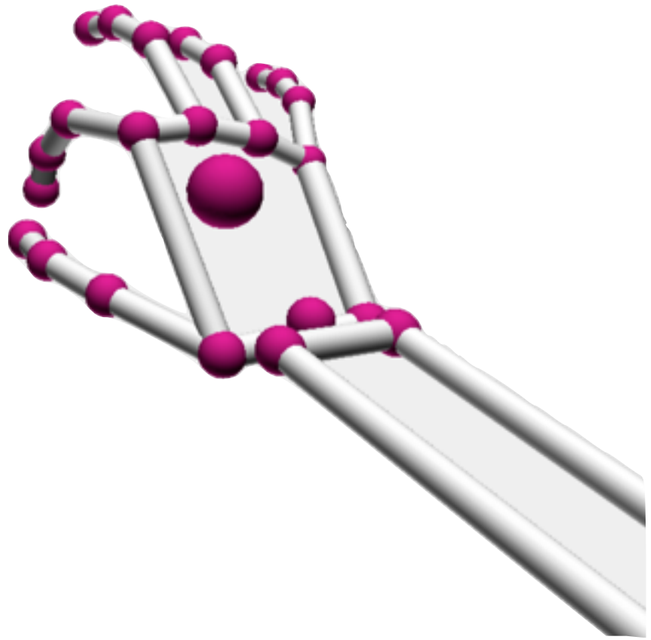
Hand Attributes 1/3



- **palmPosition**: center of the palm from Leap origin
- **palmVelocity**: speed of palm in millimeters per second
- **palmNormal**: vector perpendicular to the palm plane
- **direction**: vector pointing from center toward the fingers

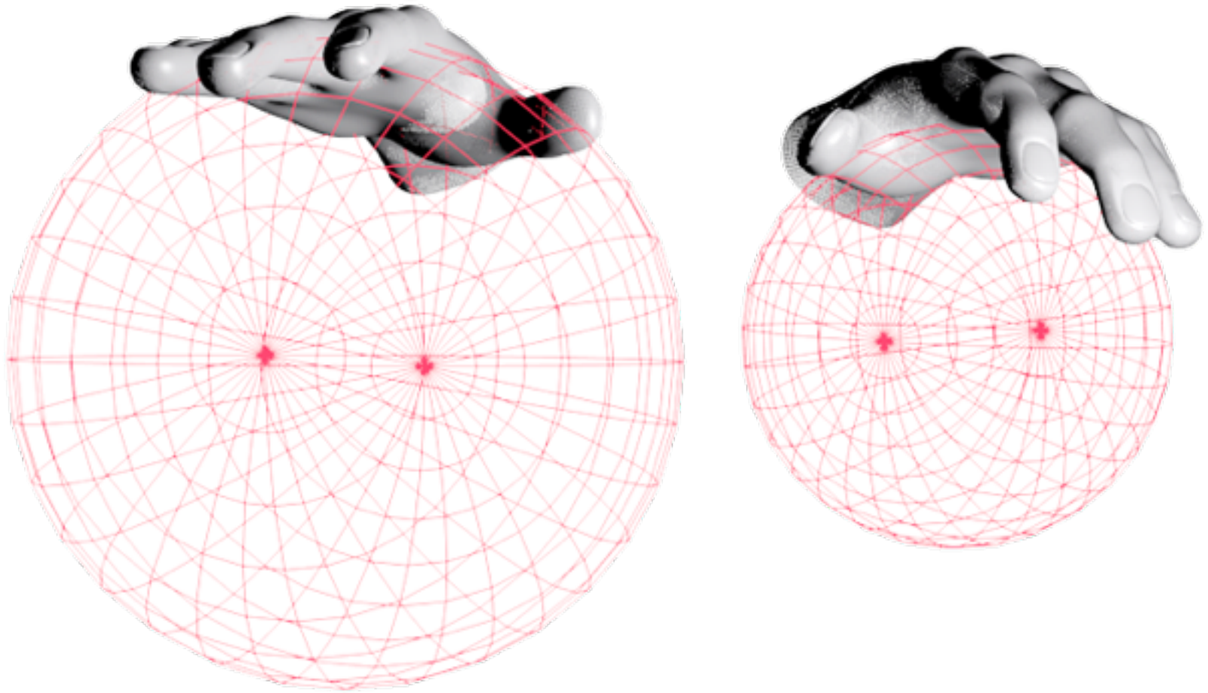
Hand Attributes 2/3





- **grabStrength**: conformity with grab posture
- **pinchStrength**: conformity with pinch posture

Hand Attributes 3/3



- **sphereCenter**: center of a sphere fit to curvature of hand
- **sphereRadius**: radius of a sphere fit to curvature of hand

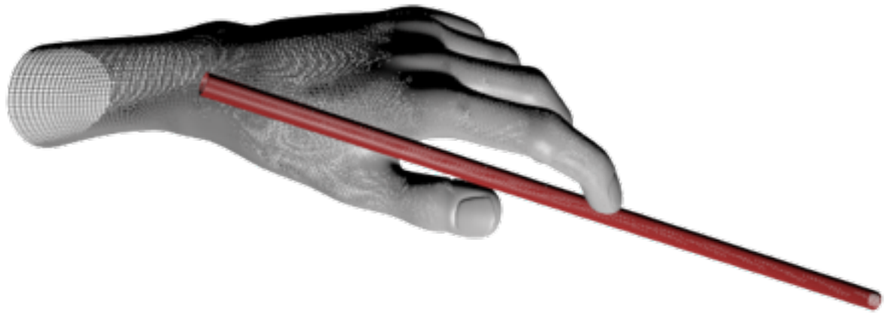
Hand Motion Analysis

- Compare hand in current frame with a specified earlier frame
- rotationAxis
- rotationAngle
- rotationMatrix
- scaleFactor
- translation

Hand Finger and Tool lists

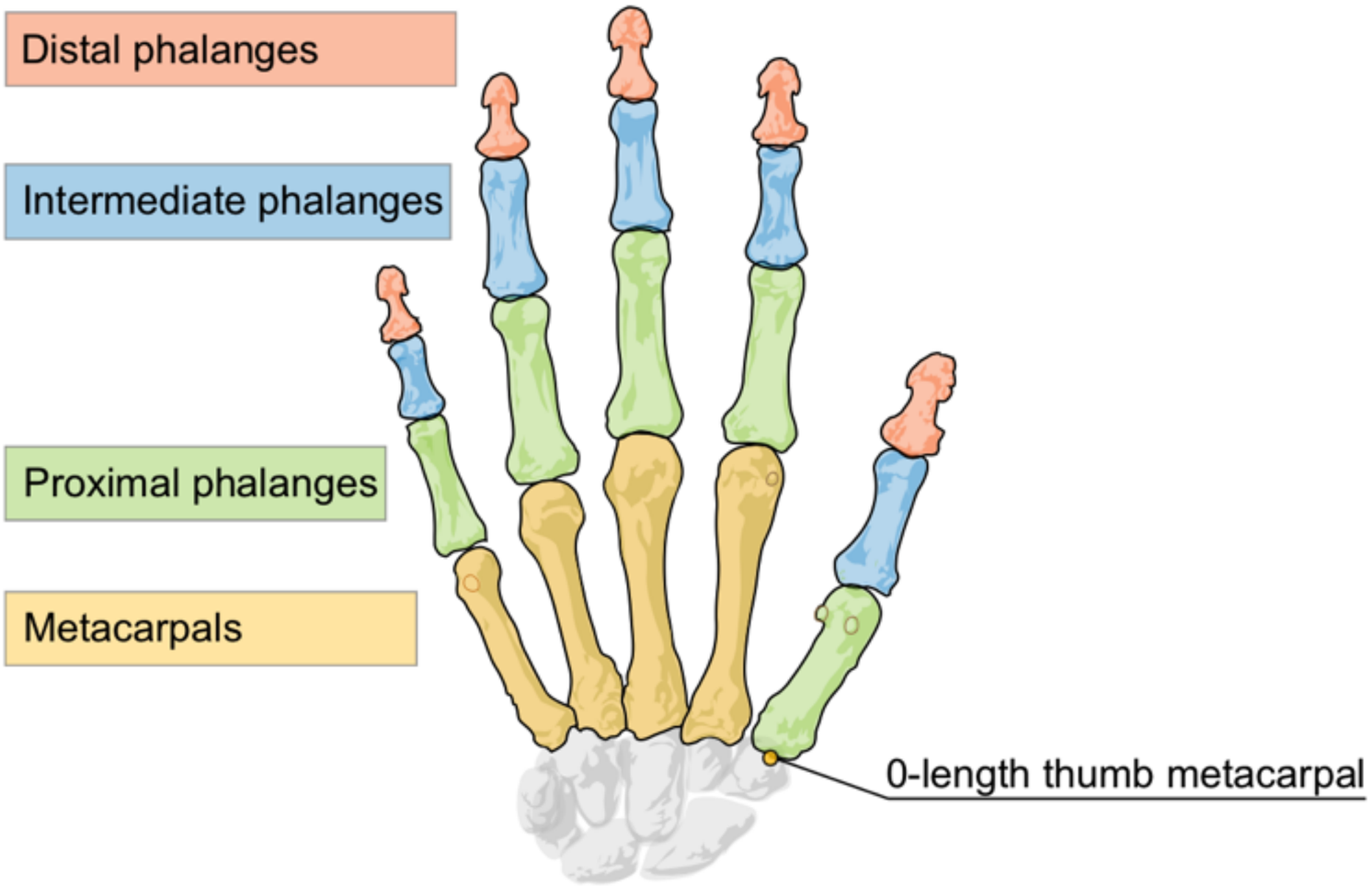
- **pointables**: both fingers and tools
- **fingers**: just the fingers
- **tools**: just the tools

Finger and Tool models



- **length**: length of the visible portion of the object
- **width**: average width of the visible portion
- **direction**: direction vector pointing same as object
- **tipPosition**: position of the tip
- **tipVelocity**: speed of the tip

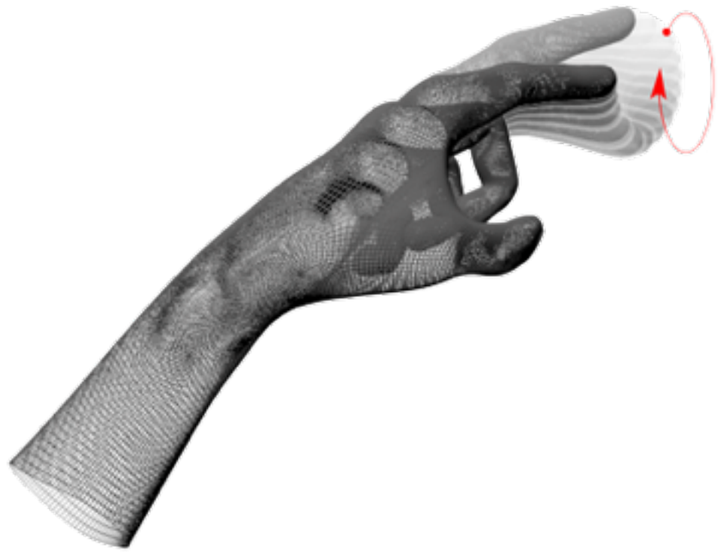
Finger bone model



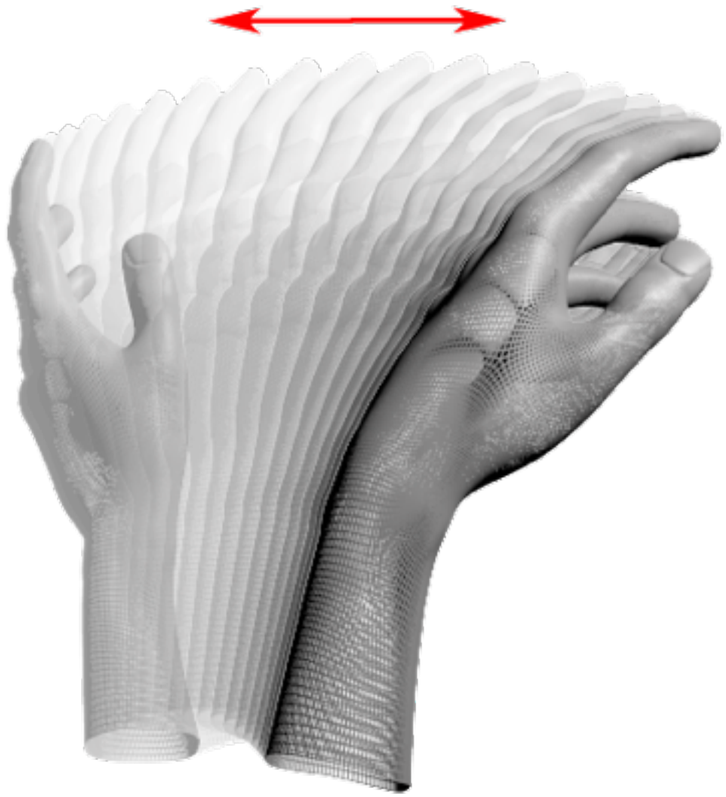

Leap Motion close to the Op site

Gestures

Circle gesture

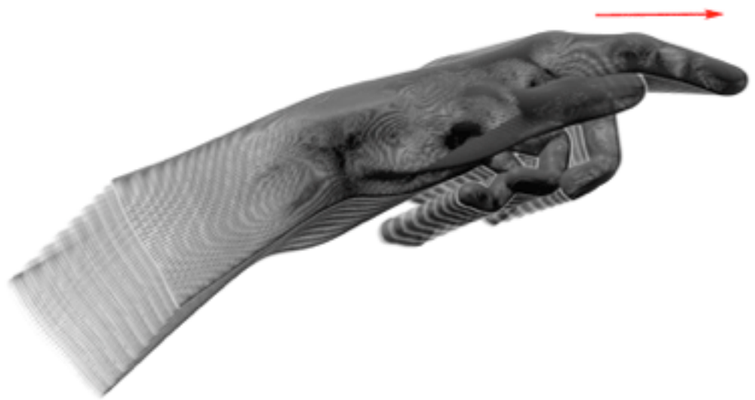


Swipe gesture



Key tap gesture

Screen tap gesture



Basic JavaScript example

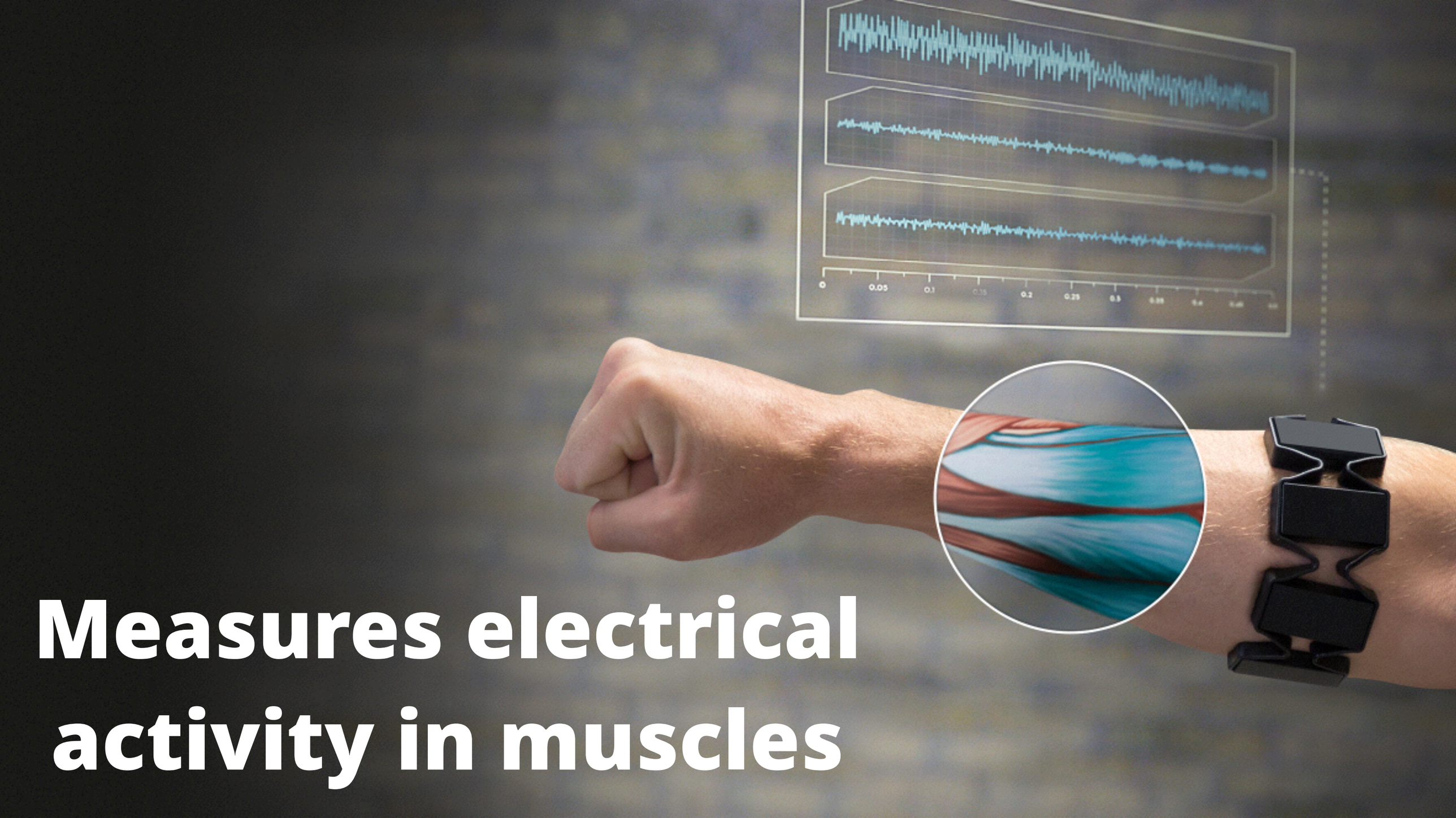
```
<html>
  <head>
    <title>Leap JavaScript Sample</title>
    <script src="//js.leapmotion.com/leap-0.6.2.js"></script>
    <script>
      var controllerOptions = {enableGestures: true};
      Leap.loop(controllerOptions, function(frame) {
        var frameOutput = document.getElementById("frameData");
        frameOutput.innerHTML = "Frame ID: " + frame.id  + "<br />"
          + "Timestamp: " + frame.timestamp + " &micro;s<br />"
          + "Hands: " + frame.hands.length + "<br />"
          + "Fingers: " + frame.fingers.length + "<br />"
          + "Tools: " + frame.tools.length + "<br />"
          + "Gestures: " + frame.gestures.length + "<br />";
      })
    </script>
  </head>
  <body>
    <h3>Frame data:</h3>
    <div id="frameData"></div>
  </body>
</html>
```

GECO showcase

VR applications


ightLine: The Chair
Will you take a seat?
SightLine:

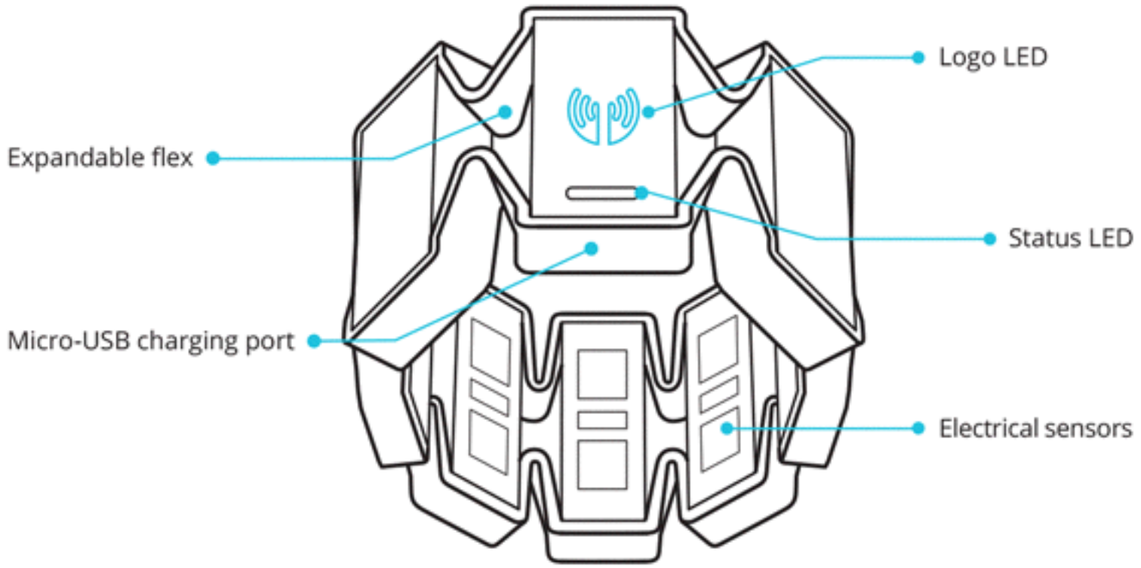
Myo



Measures electrical activity in muscles

Myo

- wireless over Bluetooth 4.0 LE
- wearable
- vibration feedback
- status LED



Myo data models

- orientation data stream
- acceleration data stream
- gyroscope data stream
- poses : wave in, wave out, spread fingers, fist, thumb to pinky

Myo architecture

- per-application HUB to access Myo instances
- listener interface
- limited language bindings atm:
  C++, Objective-C, Android Java, Lua

```
private DeviceListener listener = new AbstractDeviceListener() {
  public void onConnect(Myo myo, long t) { }
  public void onDisconnect(Myo myo, long t) { }
  public void onArmRecognized(Myo myo, long t, Arm arm, XDirection xdir) { }
  public void onArmLost(Myo myo, long timestamp) { }
  public void onOrientationData(Myo myo, long timestamp, Quaternion rotation) {
    float roll = (float) Math.toDegrees(Quaternion.roll(rotation));
    float pitch = (float) Math.toDegrees(Quaternion.pitch(rotation));
    float yaw = (float) Math.toDegrees(Quaternion.yaw(rotation));
    // use roll, pitch and yaw
  }
  public void onPose(Myo myo, long timestamp, Pose pose) {
    if (pose == FIST) { // use fist pose }
  }
};

// initialize the Hub singleton with an application identifier.
Hub hub = Hub.getInstance();
if (!hub.init(this, getPackageName())) { // handle init error }
// register for DeviceListener callbacks.
hub.addListener(listener);
// pair with nearby Myo
hub.pairWithAdjacentMyo();
```

MyoMIDI prototype demo

Some personal findings

Embrace precision

> filter only when really necessary

> avoid machine learning

> enable users to learn how to best use your interactions

> self-rewarding UI is very powerful

Carefully pick your gestures

> tempting to do cool and unique interactions

> users only occasionally use your application

> they don’t want to learn and re-learn

Consider screen-less

> keyboard-mouse has us always staring at the screen

> gesture control turns ourselves into our own reference point

> auditory and vibration cues can be good alternatives

Latency matters

> quickly hand off operations to independent processing queue

> make sure you test in sub-optimal environments

> consider battery and cpu usage

> sometimes less ‘correct’ algorithms might actually feel better

Leap Motion Controller

http://www.leapmotion.com

Thalmic Myo

http://www.thalmic.com

Thanks to