



Hände, Finger & Gesten

Vortrag von Hendrik Lichtenberg

Inhalt des Vortrags

1. Hände

- 1.1 Hände - Allgemeines
- 1.2 Hände im LM System
- 1.3 Aufbau
- 1.4 Verwendung in Java

3. Gesten

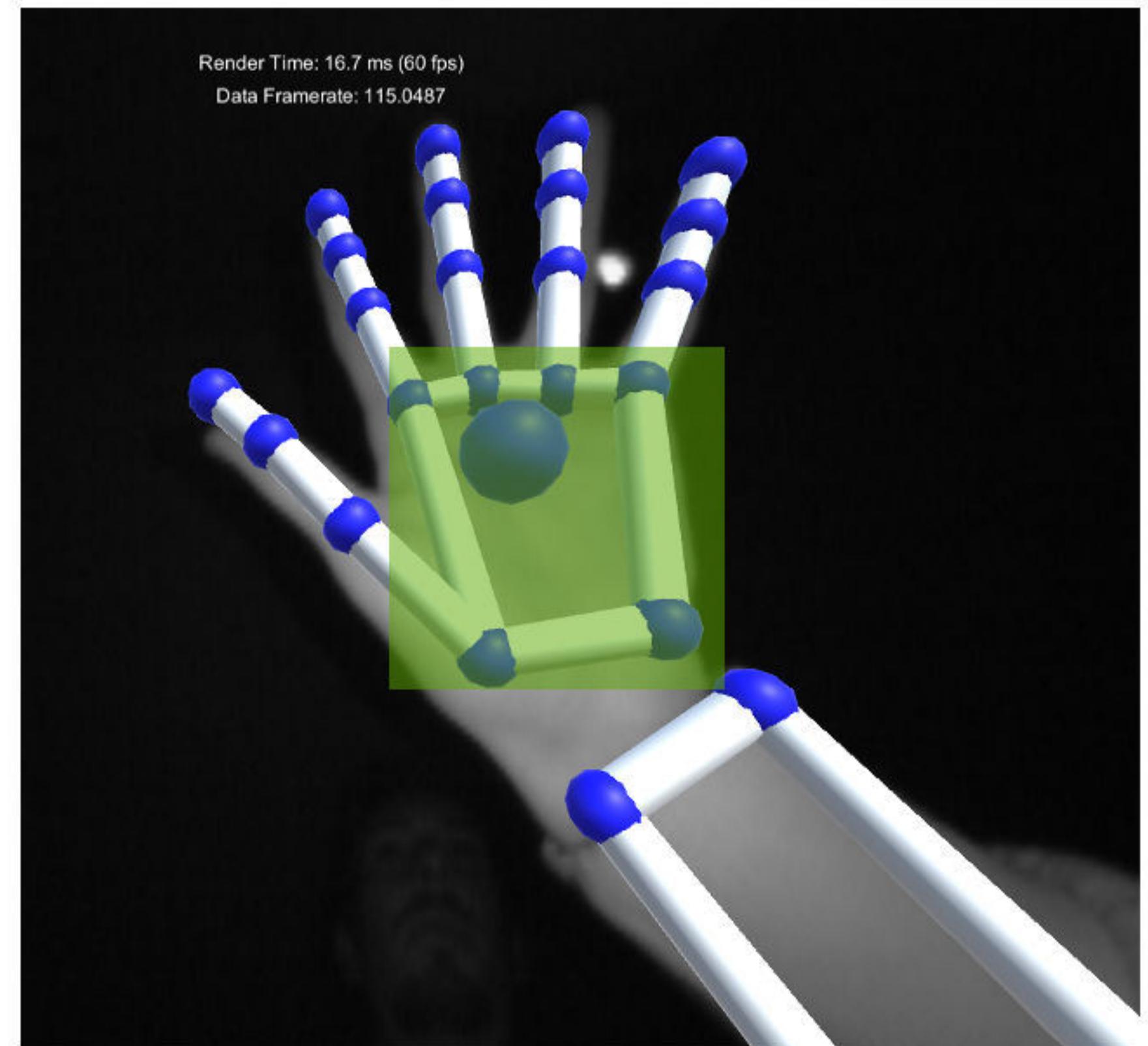
- 3.1 Gesten - Allgemeines
- 3.2 API 2.x vs Orion
- 3.3 Gestentracking in V2.0
- 3.4 Gestentracking in Orion

2. Finger

- 2.1 Finger - Allgemeines
- 2.2 Finger im LM System
- 2.3 Aufbau
- 2.4 Verwendung in Java

Hände - Allgemeines

- Hände stellen das primär getrackte Objekt dar.
- Der Controller generiert ein Modell der Hand und gleicht dieses mit dem Sensor Input ab.
- Der Controller registriert automatisch fehlerhafte Erkennung und korrigiert das Modell, jedoch kann es leicht zu Verwechslungen (Rechts/Links) kommen.

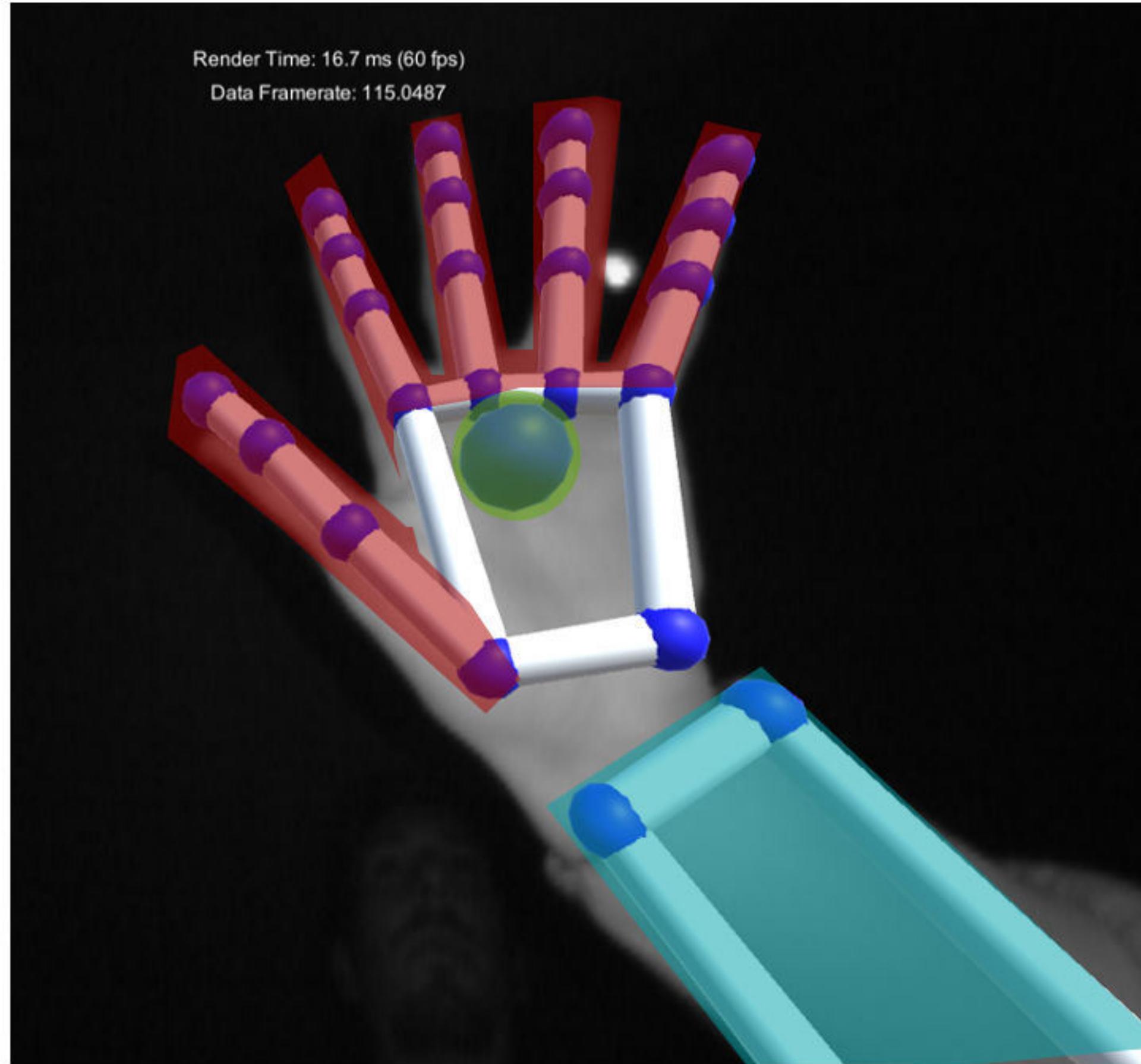


Hände im Leap Motion System



developer.leapmotion.com/documentation/java

Aufbau

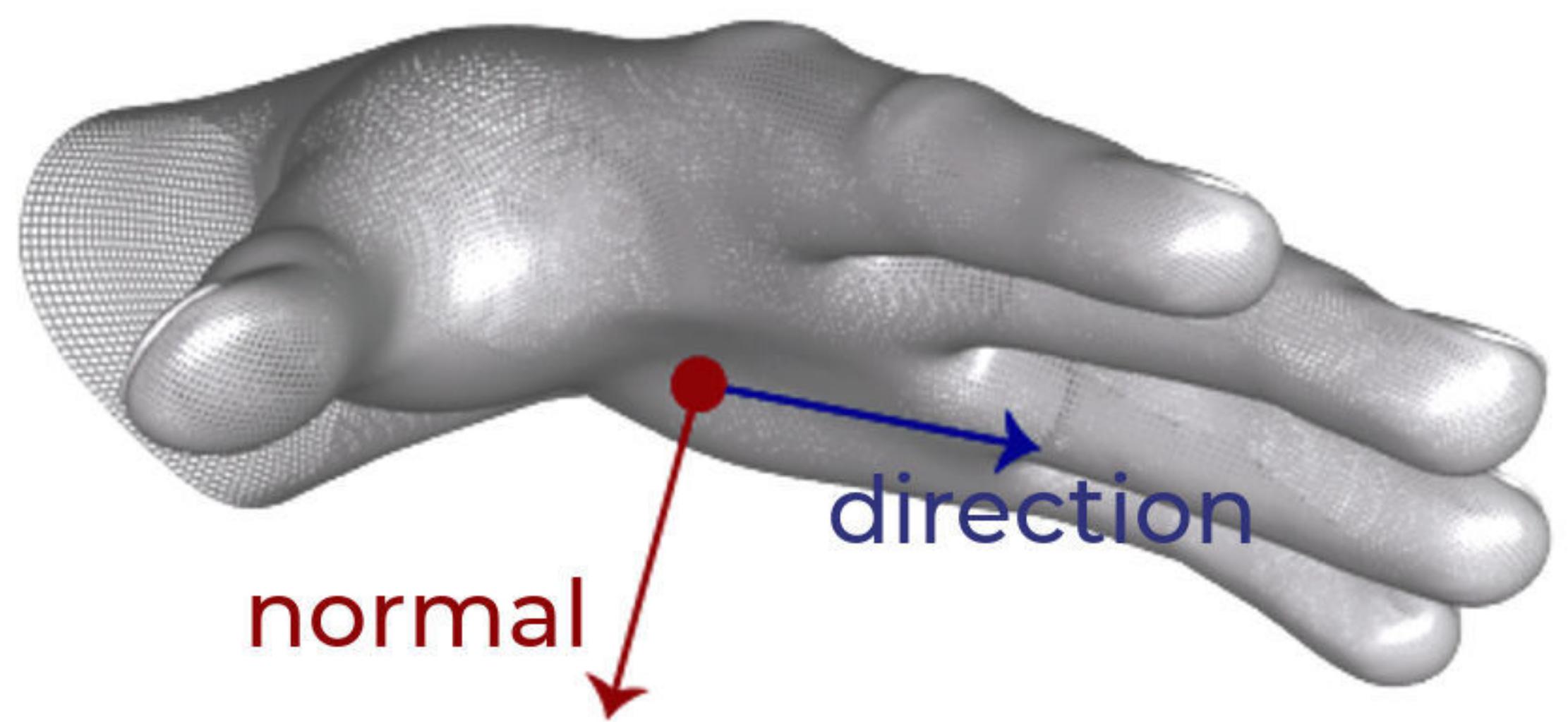


- █ Palm
- █ Fingers
- █ Arm

Hand Objekt enthält Informationen wie:

- Rotation
- Position der Handfläche (Palm)
- Links/ Rechts
- Beschleunigung der Handfläche (mm/s)

Aufbau - Vektoren

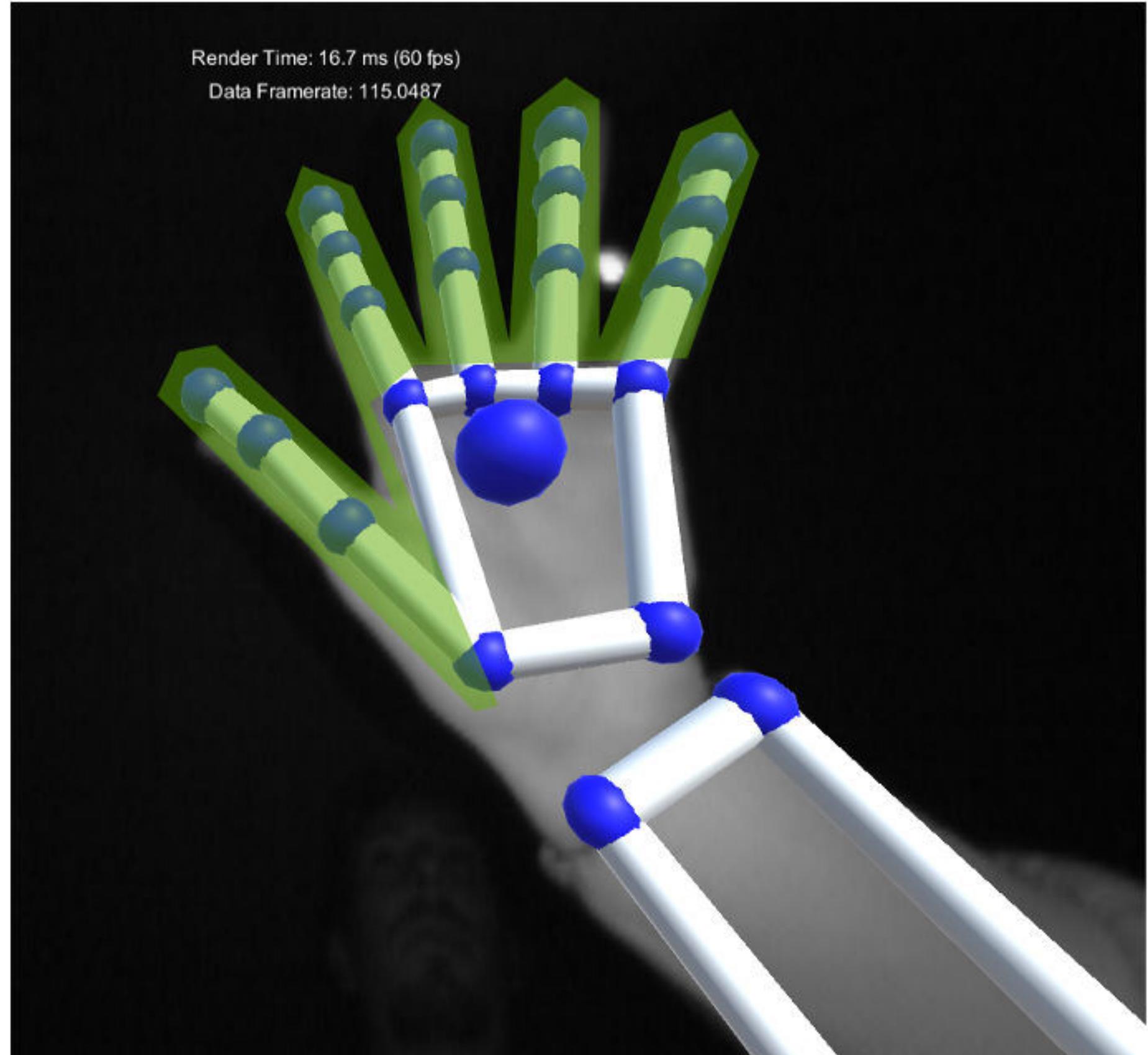


Hände - Java

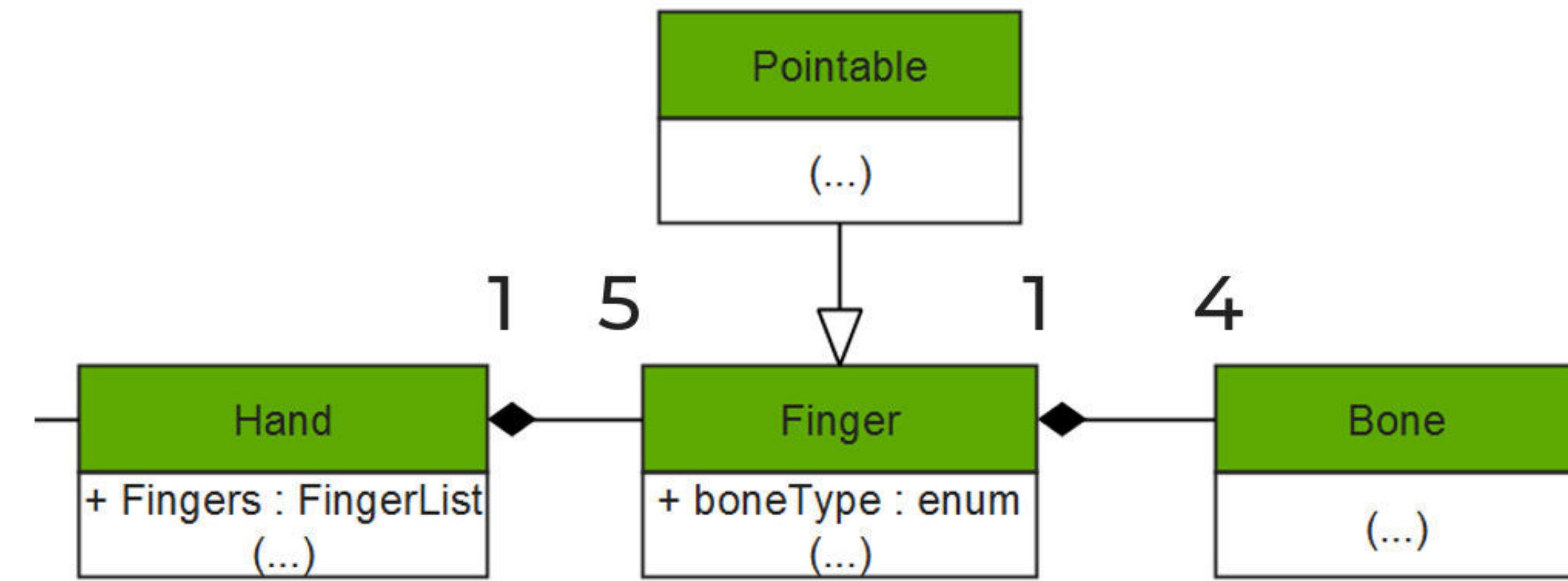
Siehe Java Codebeispiel
Debug.GetHandData(...)

Finger - Allgemeines

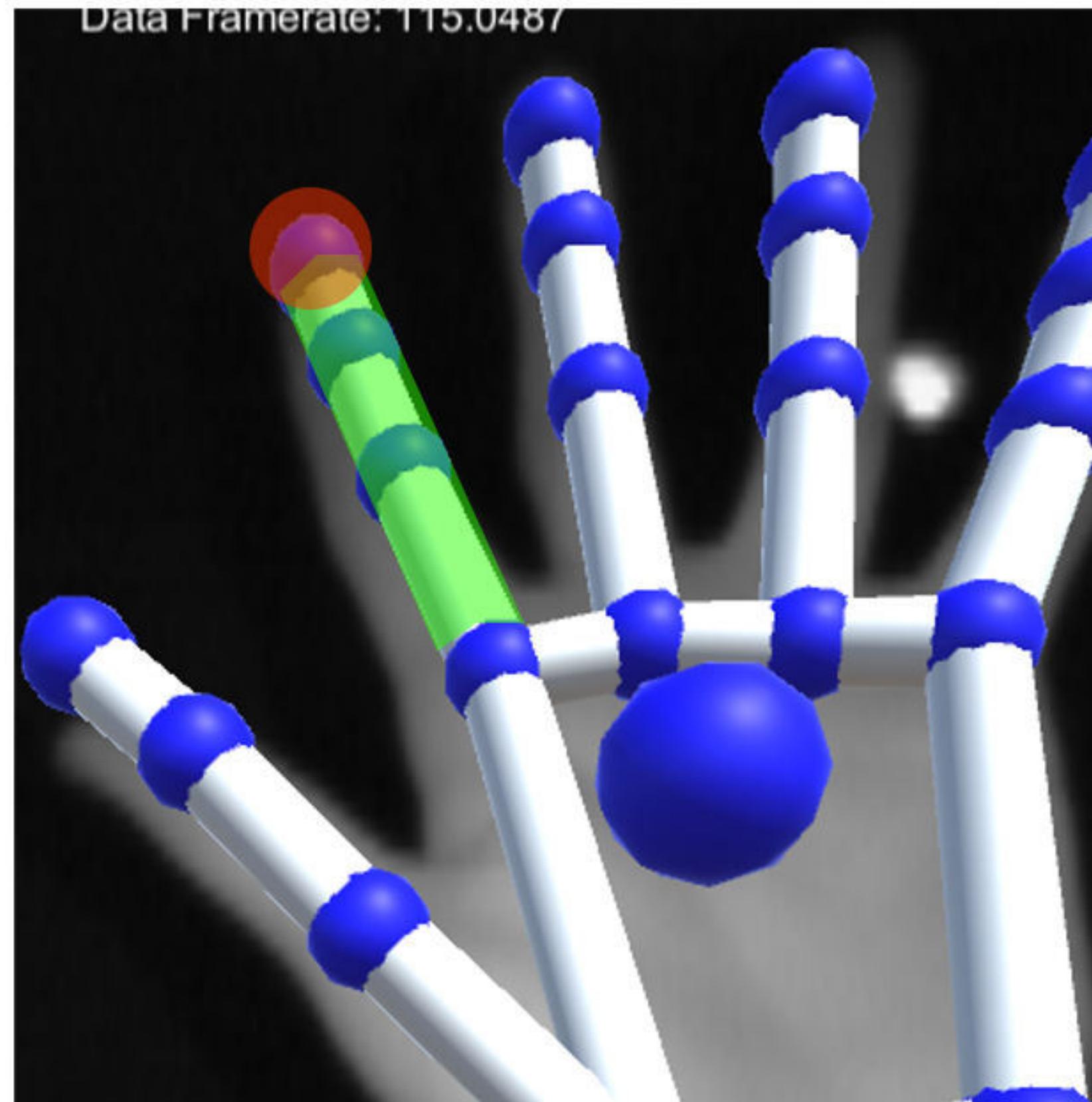
- Der Hand untergeordnetes Objekt
- Zwei Hände á fünf Finger (Big Surprise)
- Stellt u.a. Daten wie Position, Rotation, Geschwindigkeit, Richtung, Länge, Breite, Pos., Rot. & Geschwindigkeit der Fingerspitze zur Verfügung



Finger im Leap Motion System



Finger - Aufbau



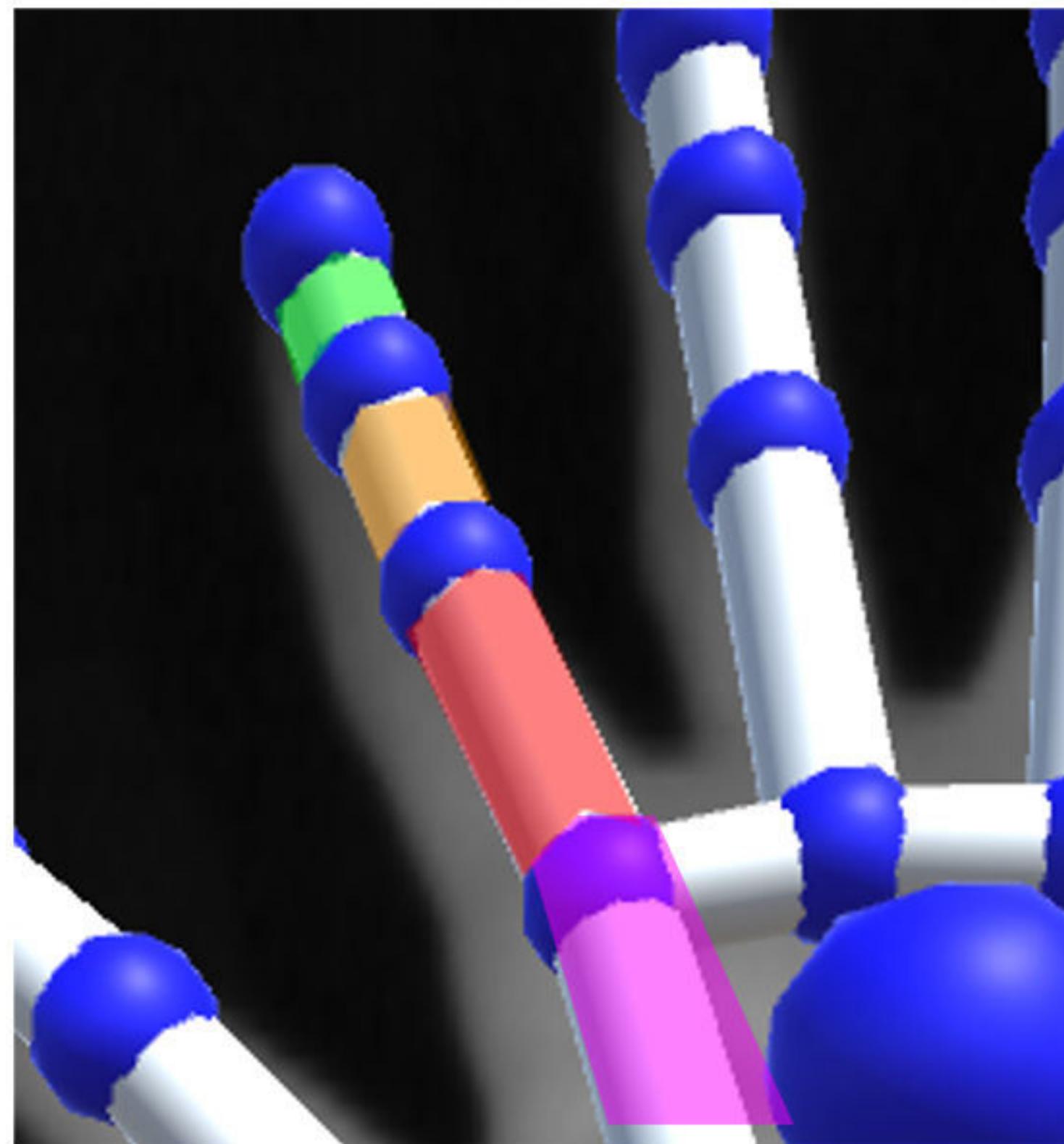
Tip



Finger

Finger Objekte enthalten Bones.
Während ein Finger Objekt bereits die meisten relevanten Daten enthält, lassen sich über Bone Objekte detailliertere Daten auslesen.

Finger - Aufbau Bones



TYPE_DISTAL



TYPE_INTERMEDIATE

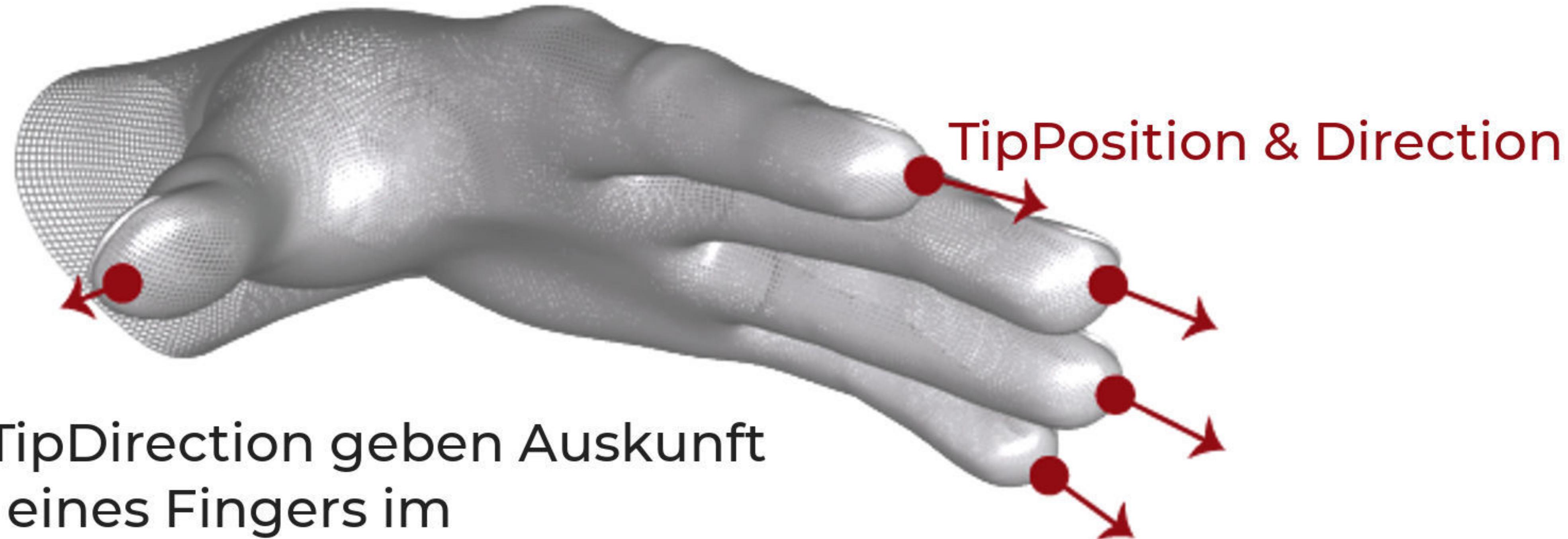


TYPE_PROXIMAL



TYPE_METACARPAL

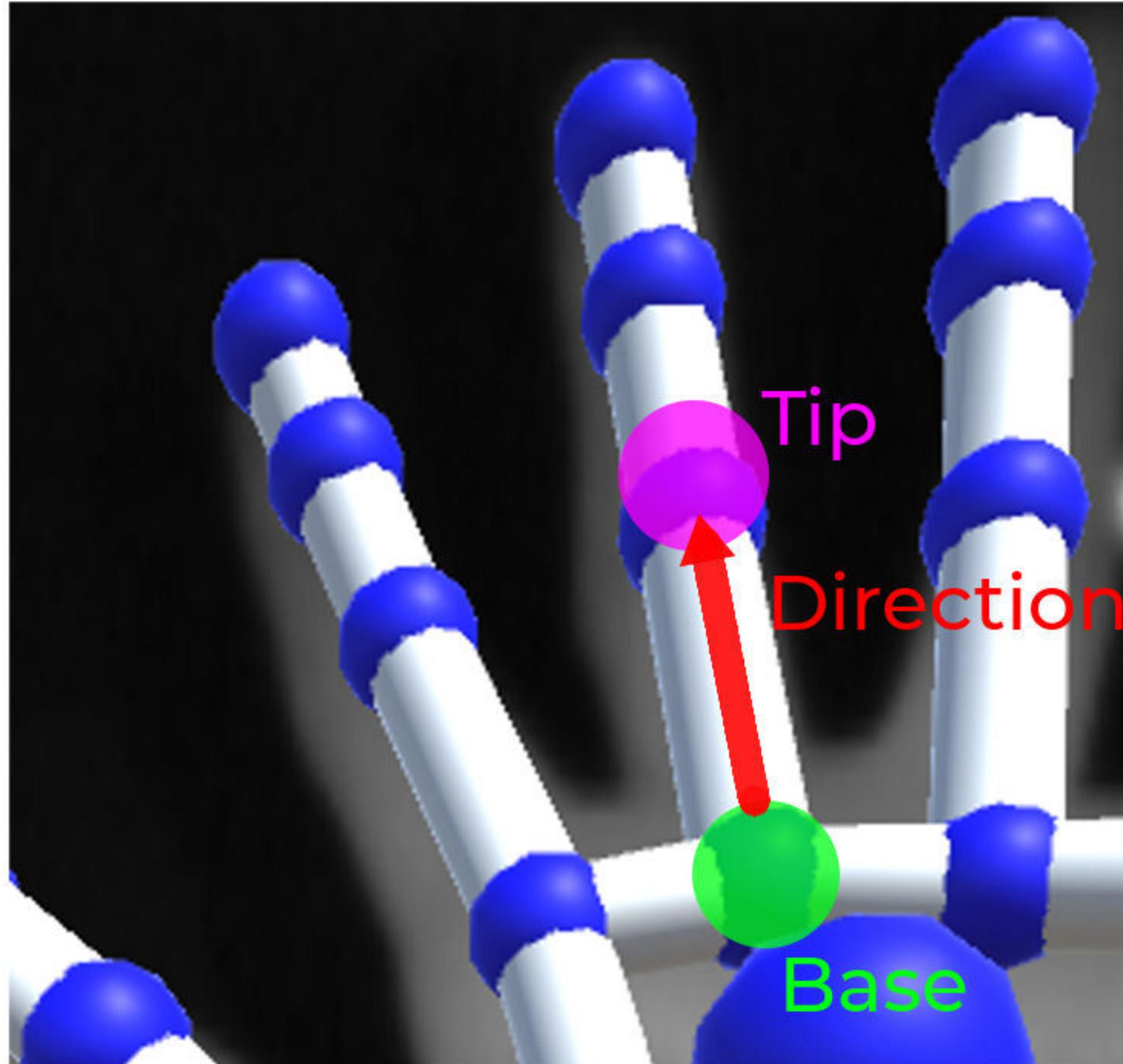
Finger/ Pointable - Vektoren



TipPosition & TipDirection geben Auskunft
über die Lage eines Fingers im
Koordinatensystem was bereits viele
Anwendungsmöglichkeiten bietet

developers.leapmotion.com

Finger - Vektoren Bones



- Base stellt den 'Anfang' eines Bones dar
- Tip stellt das 'Ende' eines Bones dar
- Direction stellt den Richtungsvektor eines Bones dar

Anm.: Base != Basis. Basis meint die Orientierung des Bones als 3D Matrix

Finger - Java

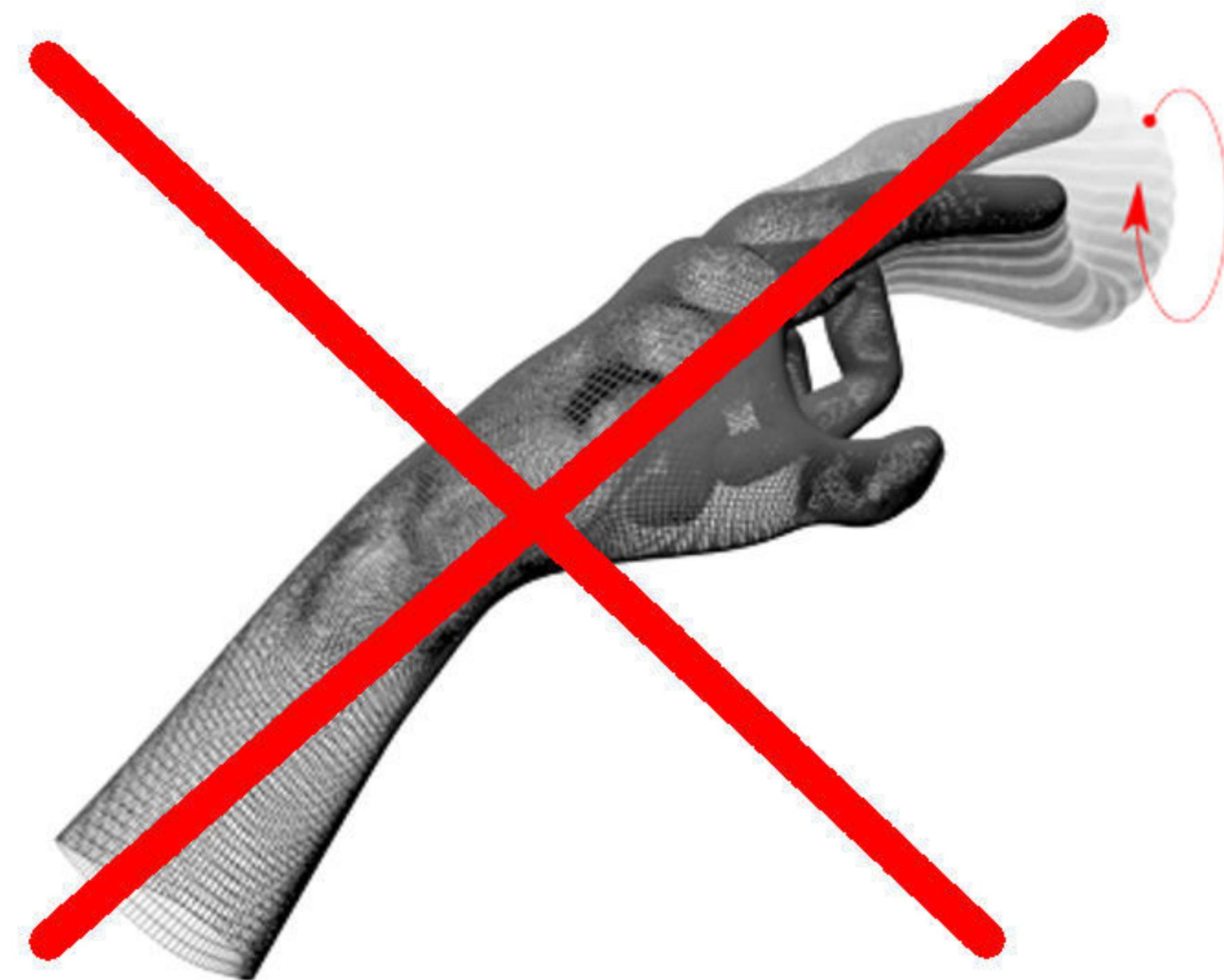
Siehe Java Codebeispiel
`Debug.GetFingerData(...)`

Gesten - Allgemein



- Gesten repräsentieren spezifische Bewegungsmuster, die vom Controller erkannt werden
- Die v2 API kennt folgende Gesten out-of-the-box:
 - Circle: (siehe Abb.)
 - Swipe: lange, lineare Bew. des Fingers
 - Key Tap: horizontales 'tippen'
 - Screen Tap: vertikales 'tippen'

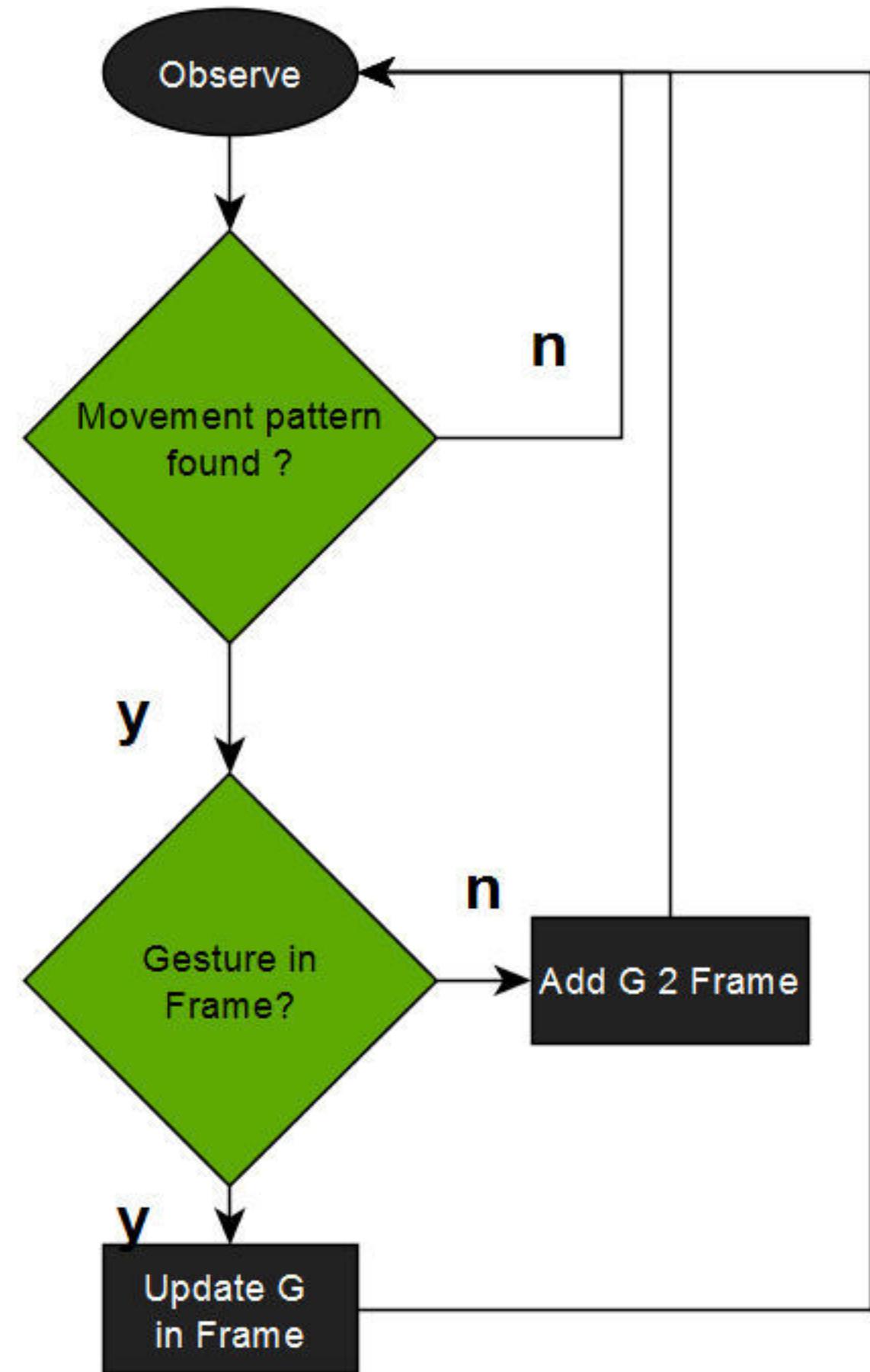
Gesten - V2 vs Orion



- Vorgefertigte Gesten wurden mit Version 3.0 der API deprecated
- Statt vorgefertigten Gesten sollen Entwickler diese selbst entwerfen

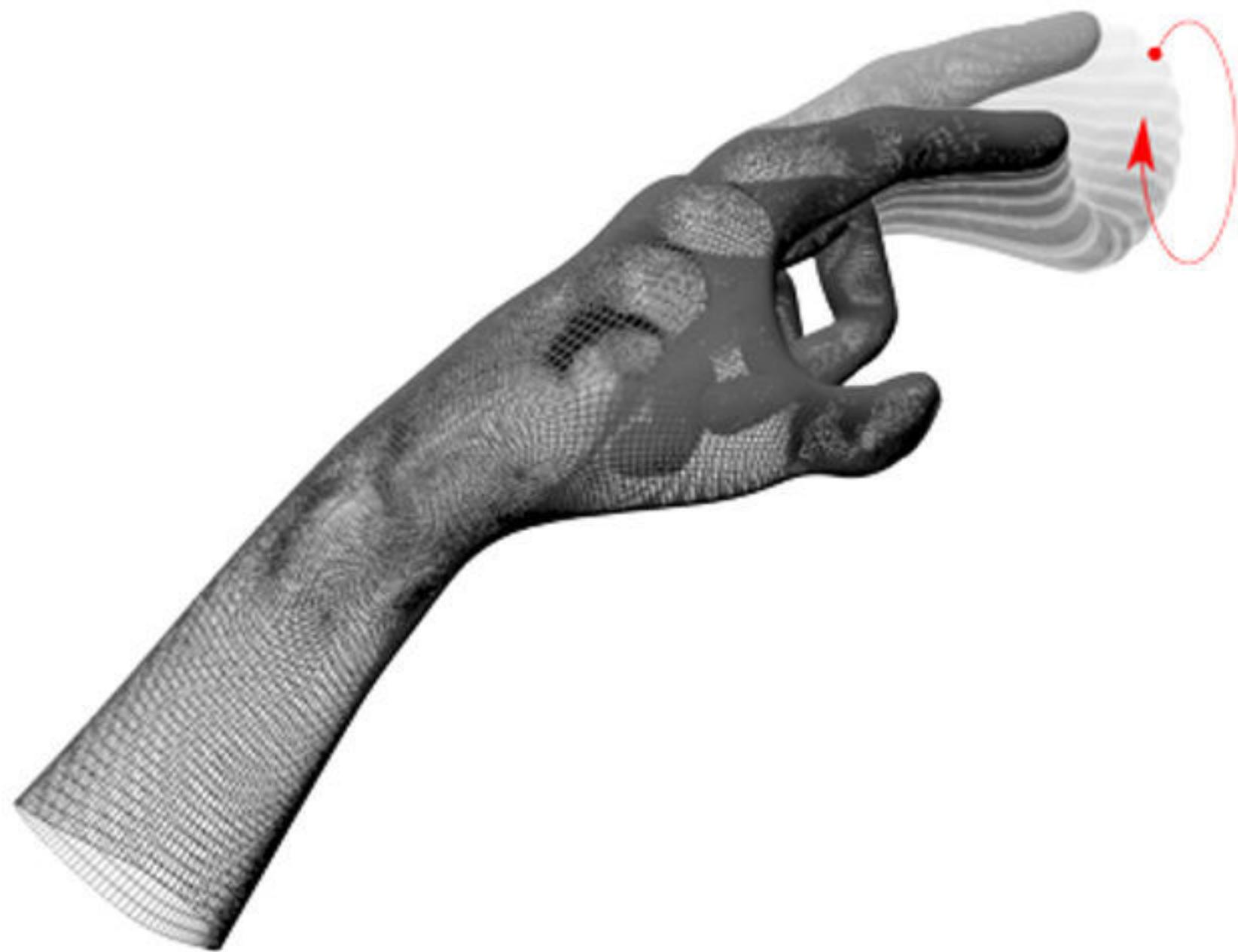
"We removed the four built-in gestures in Orion. This was for a variety of reasons, but mostly because they never really worked well enough and worked even less well in VR." - Joe_Ward

Gesten - Tracking in V2



1. Controller sucht nach spez. Bewegungsmustern
2. Wurde ein Muster gefunden wird geprüft ob es eine 'neue' Geste ist
3. Ist die Geste 'neu' wird sie der Gestenliste des Frames hinzugefügt
4. Ist die Geste 'continuous' wird sie in der Liste aktualisiert (z.B. Circles)

Gesten - Circle



Kreisförmige Bewegung eines Fingers

Erfasst folgende Daten:

- Zentrum des beschriebenen Kreises
- Ausführender Finger
- Anzahl der Kreisbewegungen
- Radius des beschriebenen Kreises

Gesten - Swipe



'Wischen' mit der Hand

Erfasst folgende Daten:

- Richtung der Bewegung
- Ausführende Finger
- Position im Koordinatensystem
- Geschwindigkeit in mm/s
- Startposition der Geste

Gesten - KeyTap

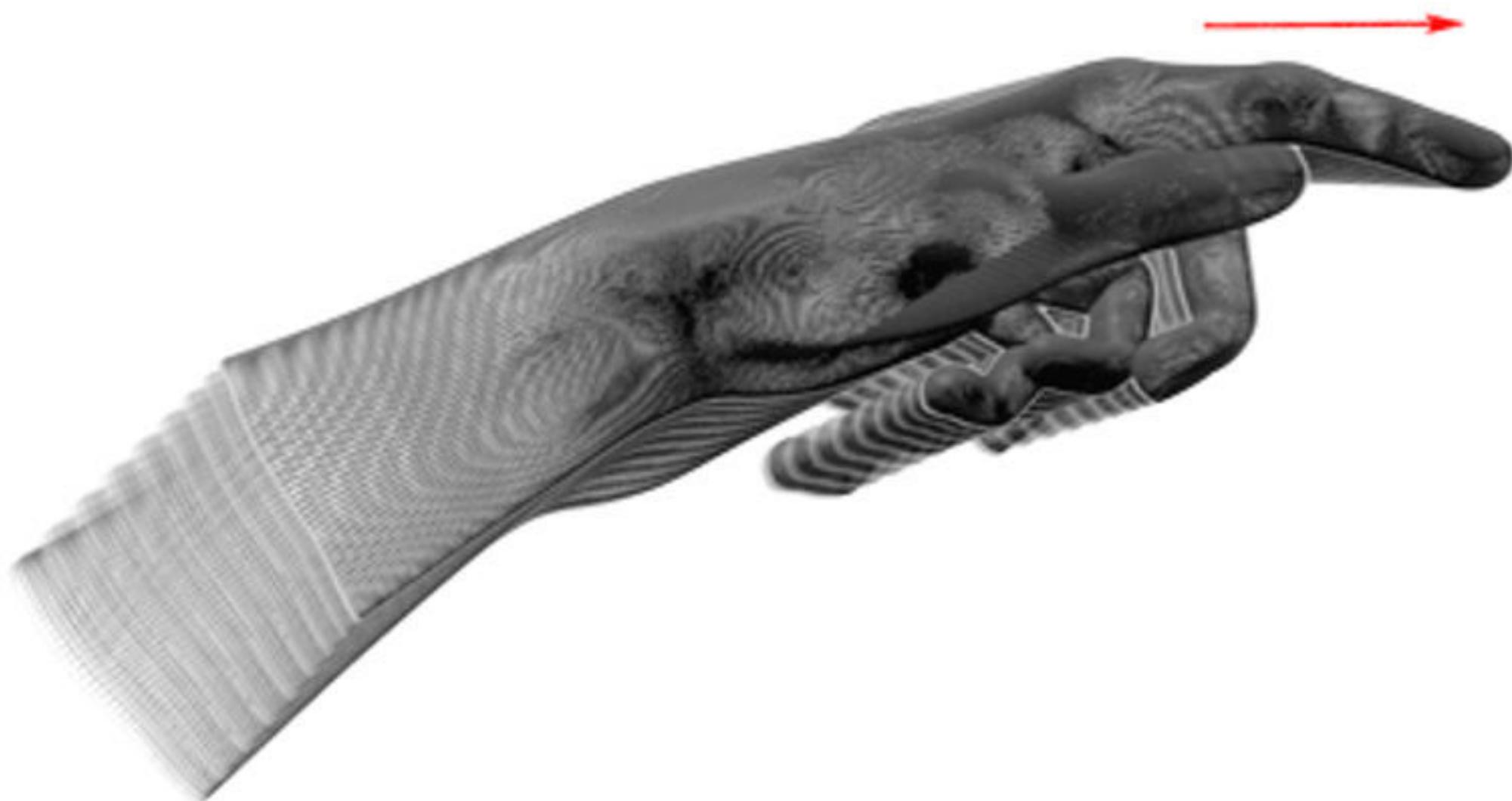
Vertikales Tippen wie auf einer Tastatur



Erfasst folgende Daten:

- Richtung der Bewegung
- Ausführende Finger
- Position im Koordinatensystem

Gesten - ScreenTap



Horizontales Tippen wie auf einem Touchdisplay

Erfasst folgende Daten:

- Richtung der Bewegung
- Ausführende Finger
- Position im Koordinatensystem

Gesten - Java

```
Frame frameV2 = controller.frame();
GestureList gestures = frameV2.gestures();
for(int n = 0; n < gestures.count(); n++)
{
    if(gestures.get(n).type().equals(Gesture.Type.TYPE_SCREEN_TAP) == true)
    {
        System.out.println("Gesture recognized: ScreenTap");
        ScreenTapGesture tap = new ScreenTapGesture(gestures.get(n));
        System.out.println("Tap direction" + tap.direction());
    }
    else if(gestures.get(n).type().equals(Gesture.Type.TYPE_CIRCLE) == true)
    {
        System.out.println("Gesture recognized: ScreenTap");
        CircleGesture circle = new CircleGesture(gestures.get(n));
        System.out.println("Circles drawn: " + circle.progress());
    }
}
```

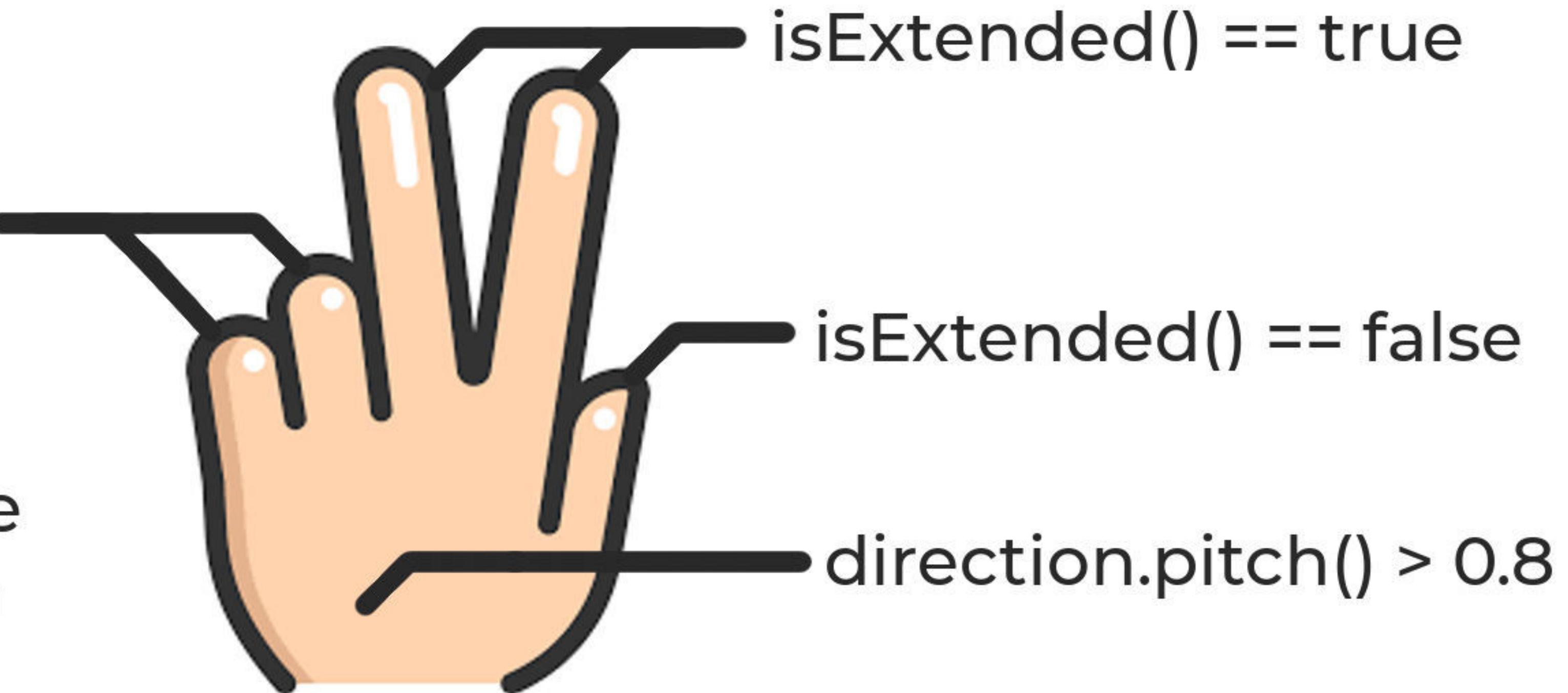
Gestentracking - Orion (V3)



- Klasse 'Gesture' wird ab V3.0 nicht mehr unterstützt.
- Gesten müssen selbst entwickelt werden
- Gestenerkennung über charakteristische Sensordaten (z.B. isExtended, Rotation...)
- Kreative Anwendung der Sensordaten ermöglicht die Implementierung nahezu jeder Geste

Gesten - Beispiel 'Peace'

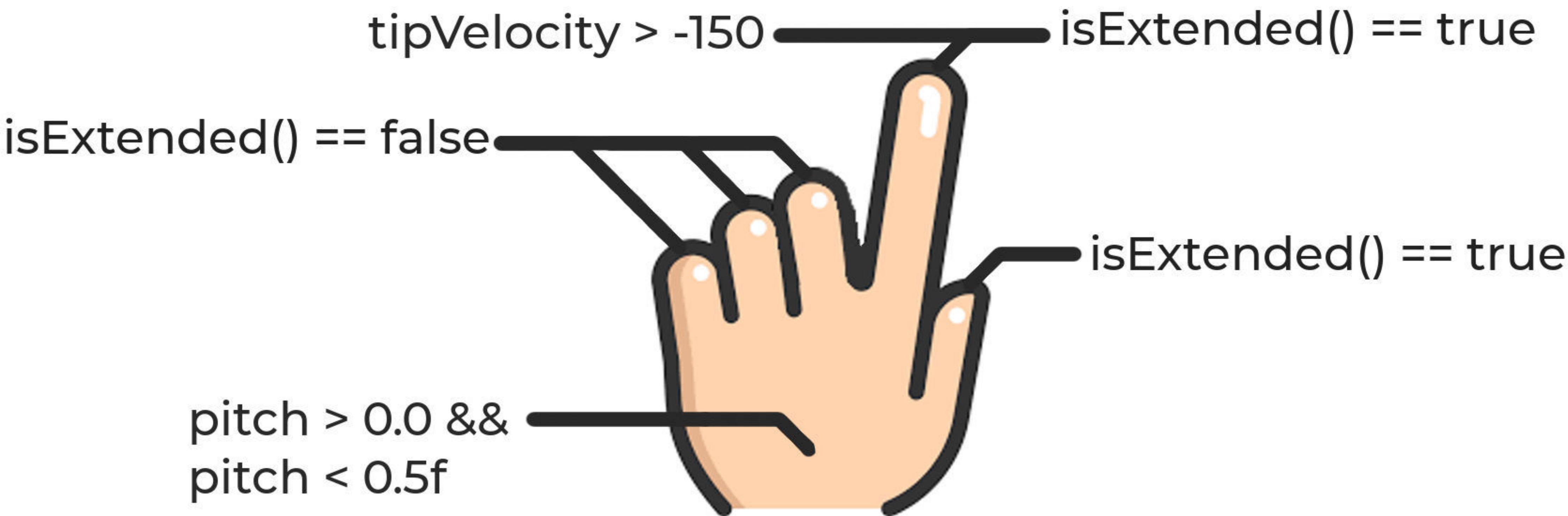
Wir wollen prüfen welche Bedingungen erfüllt sein müssen damit die Geste erkannt wird



Gesten - Beispiel 'Peace'

Siehe Java Codebeispiel
`Debug.doesPeaceSign(...)`

Gesten - Beispiel 'Point'



Gesten - Beispiel 'Point'

Siehe Java Codebeispiel
`Debug.doesPointGesture(...)`

Codebeispiel Repo



[https://github.com/Moimus/
LeapDebugTool/tree/master](https://github.com/Moimus/LeapDebugTool/tree/master)

Danke für Ihre Aufmerksamkeit.
Noch Fragen?