Leap Motion Version 2

Michaël Van Canneyt

September 6, 2014

Abstract

The Leap Motion team has released version 2 of their API. The Object Pascal code to connect to the Leap Motion has been adapted to integrate the new features. An overview.

1 Introduction

The Leap Motion is a small device to track movement of human hands, which can be plugged in a USB port of your PC or Mac. More info about this device can be found on

`http://www.leapmotion.com/`

Object Pascal support for this device has been discussed in a previous contribution. Recently, the Leap Motion team released version 2 of their API, and the Object Pascal interface to the Leap Motion has been adapted to integrate the new features in the API.

The new features of the API include the following:

- Skeletal tracking: Detection of the various parts of the fingers, and the arm. These are represented in the data model.
- Focus: The leap motion controller application keeps a list of connected clients, i.e. programs to which it sends data. By default, only the 'focused' application receives leap events. The application is required to keep track of whether it has focus or not.
- Background: If an application that does not have focus (i.e. is in the background) wishes to receive events anyway, it must request this from the leap controller.
- Image acquisition. The new API allows to retrieve the image of the space above the leap as the leap sees it. Unfortunately, these images are not available through the Javascript API, and hence not in the Object Pascal API.

In the following sections, we'll discuss the new possibilities.

2 Focus and Background

If an application connected to a V2 leap controller wishes to receive frames, by default, it needs to have focus. This is controlled by the `Focus` property of the LeapController component. Setting the property to `True` or `False` will report the current state of the application to the leap controller. The property can be set before a connection is made: the state will be reported as soon as connection is made.

For compatibility with version 1 of the API, the `Focus` property is set to `True` by default.

If an application wishes to receive events ven if it does not have focus, the `Background` property can be used to request events even if the application does not have focus. If set to `True`, the application will receive events even if it is in the background. Like the `Focus` property, setting the property will report the current state to the leap controller.

For compatibility with version 1 of the API, the `Background` property is set to `True` by default.

The use of these properties is precarious: Some experimenting reveals that the stack of connected applications as maintained by the leap controller sometimes gets messed up, causing a connected client not to receive events even if it has reported to have focus. Setting the `Background` property to `True` solves this.

3 Skeletal tracking

Version 2 of the Leap Controller introduces skeletal tracking: the detected bone structure of the hand and the arm is reported as well. The following additional properties have been introduced in the `Hand` structure:

PalmWidth The width of the palm.

Confidence A measure of the accuracy of the hand detection. This is a value between 0 and 1, where 1 is the 'best' value.

GrabStrength A measure of a 'grabbing' hand position detection. This is a value between 0 and 1, where 0 means no grabbing (open palm) and 1 means a grabbing position (almost closed fingers).

PinchStrength A measure of a 'pinching' hand position detection: gives a measure of how close the thumb is to another finger (not necessarily an index finger). This is a value between 0 and 1, where 0 means no pinching and 1 means a pinching position (almost closed thumb and finger).

TimeVisible An indication of how long the hand has been detected by the Leap motion device (in seconds).

HandType Indicates whether this hand is a left or a right hand.

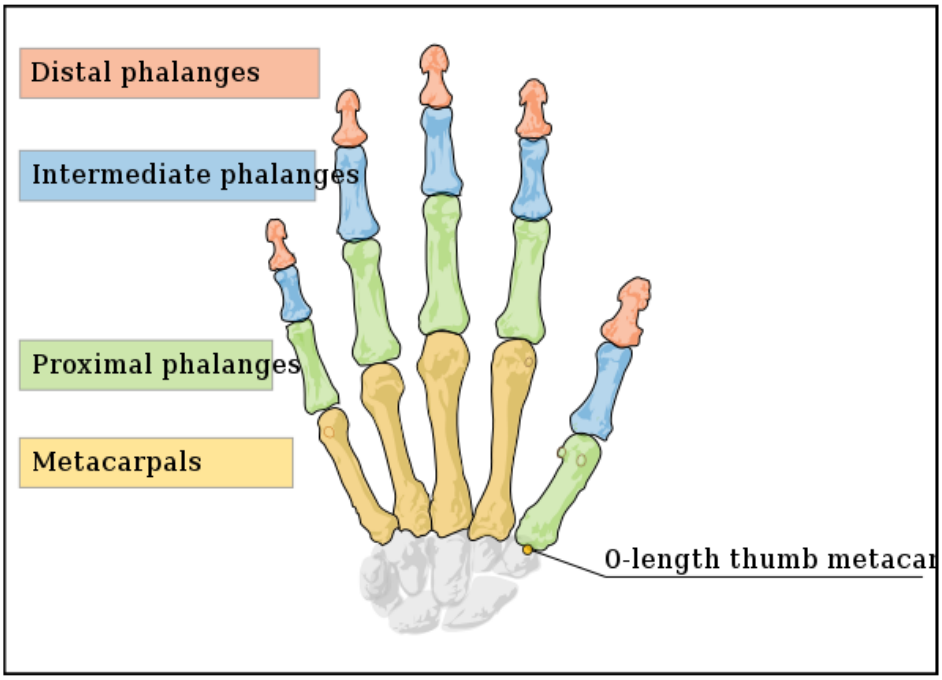
In addition, data about the arm to which the hand is connected is available in the varArm property of the `Thand` class. The `Arm` property itself is a class: `TArm`, which has the following declaration:

```
TArm = Class(TObject)
  Property Hand : THand;
  Property Basis : TBasis;
  Property Width : TFloat;
  Property Elbow : T3DPoint;
  Property Wrist : T3DPoint;
end;
```

The `Wrist` and `Elbow` properties speak for themselves: they represent the position of the elbow and wrist. The `Width` is the detected diameter (cross-section) of the arm. The `Basis` property gives the orientation of the arm, it is an array of 3 vectors:

Basis[1] is the X-Basis, this is a vector perpendicular to the longitudinal axis of the bone.

Figure 1: Detailed fingers



Basis[2] is the Y-Basis, this is a vector perpendicular to the longitudinal axis of the bone.

Basis[3] is the Z-Basis, this is a vector aligned to the longitudinal axis of the bone. It is more positive towards the base of the arm (or finger).

Like the hand class, the finger class now has more detailed information about the bone structure. The leap motion now attempts to detect which finger it is, and provides information about the bones that make up the finger. This information is available based on the names of the bones, as depicted in figure 1 on page 3 (image taken from the Leap Motion website). The names have been converted into an enumerated: `TBoneType`:

bMetacarpal This bone is actually part of the palm of the hand. It is not present for the thumb.

bProximal This is the first bone of the finger.

bIntermediate This is the middle bone of the finger.

bDistal This is the tip of the finger.

All this results in a more extended `TFinger` class:

```
TFingerType = (ftThumb,ftIndex,ftMiddle,ftRing,ftPinky);

TFinger = Class(TPointable)
Public
  // Skeleton data
  Property Bones[Aindex : TBoneType] : TBone;
  Property FingerExtended : Boolean;
  Property CarpPosition : T3DPoint;
```

```
  Property TimeVisible : TFloat;
  Property FingerType : TFingerType;
end;
```

The properties are mostly self-explaining:

Bones access (by name) to the bones that make up the finger.

FingerExtended `True` if the finger is extended, false if it is bent.

carpPosition The position of the base of the metacarpal bone of the finger.

TimeVisible The time (in seconds) that the leap has been tracking this finger.

FingerType Which finger of the hand this instance represents.

4 conclusion

The leap developers have improved their Javascript API, making more detailed information about the hands and fingers available. The Object Pascal implementation makes use of the newly available information. However, none of the Object Pascal Leap demo programs makes use of these new properties.

The leap image API is not available for Object Pascal programmers: only the 'native' APIs can make use of this new feature. Making this information available to object pascal programmers would require interfacing directly with the Leap Motion API DLL. Since the DLL is written in C++ and exposes the C++ classes directly,interfacing with it from Object Pascal is problematic. This can be overcome in the same manner that the Qt API is exposed as a plain C API: with an intermediate C library that presents a flattened plain C API. However, this library is yet to be developed.