Wie man einen Multitouch DAW Controller für macOS mit SwiftUI baut

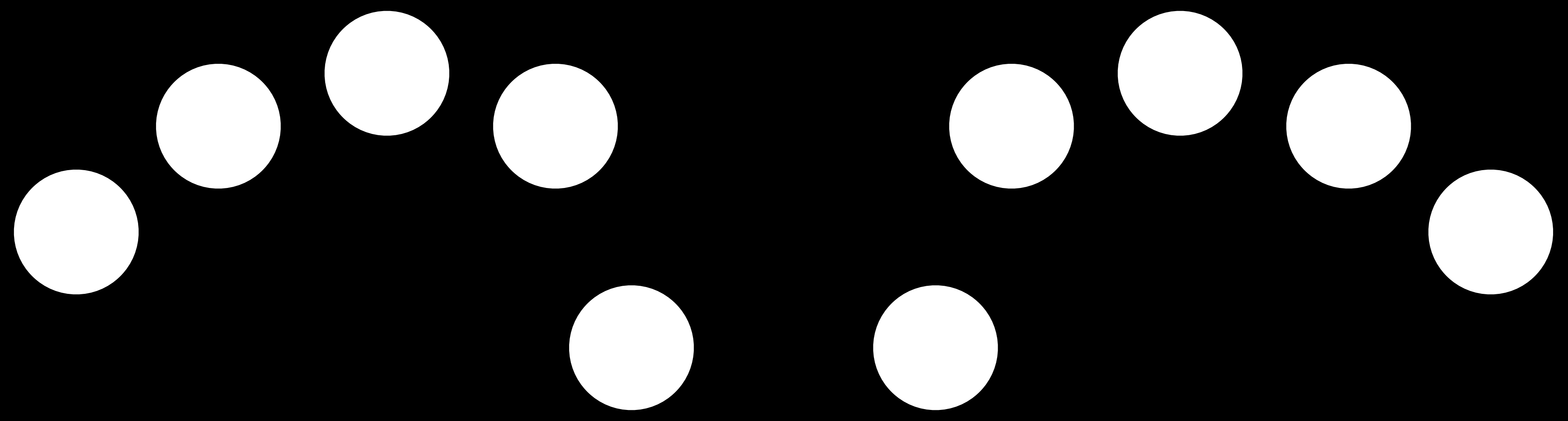
Torsten Lehmann, 17.10.2024, CocoaHeads LE

DAW

- **D**igital **A**udio **W**orkstation
- Beispiele:
  - Apple Logic Pro
  - Steinberg Cubase
  - Ableton Live
  - Bitwig Studio

Multitouch

- > 1 Finger können unabhängig und gleichzeitig mit der GUI interagieren



Problem 1: Controller

- existierende Controller sind meist auf 8 Spuren begrenzt (häufiges Umschalten notwendig)
- Displays häufig nicht vorhanden (Zuordnung Position/Farbe zu Instrument muss im Kopf passieren)

Problem 2: iOS DAWs

- viele DAWs nicht für iOS vorhanden
- iPads sind teilweise zu klein

Problem 3: macOS

- macOS ist kein Multitouch-OS
- keine Multitouch-DAW für macOS

Ansatz

- Touch-Devices als Multitouch-Controller
- TouchOSC
- OSC/PILOT

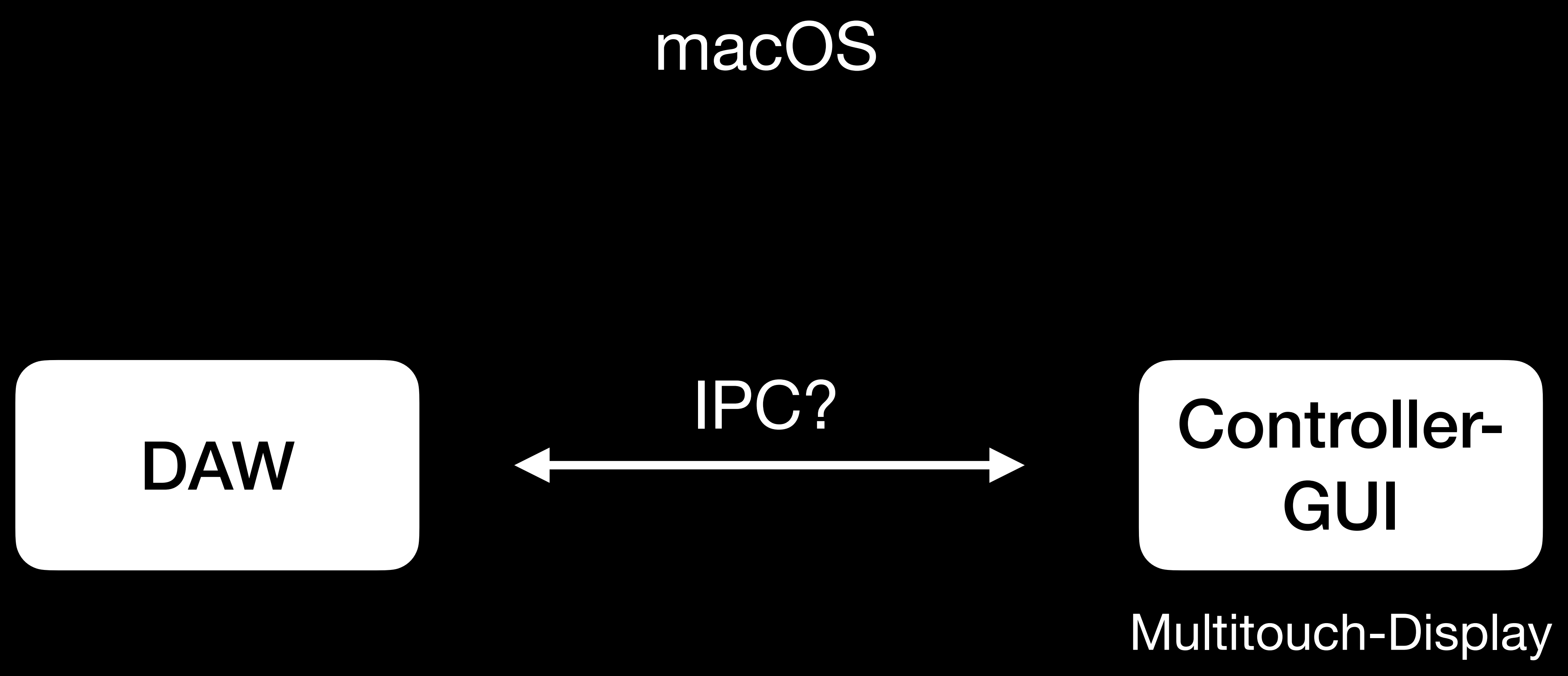
Ansatz

- TouchOSC
  - nur GUI-Editor
  - kein machine-readable Format
  - kein Import/Export-Format
- aufwändig zu erstellen/ändern

OSC/PILOT

- nicht probiert 🤷‍♂️
- endlich wieder ein SwiftUI-Projekt!

Bestandteile



OSC

- OSC: **O**pen **S**ound **C**ontrol
- UDP-basiert (TCP wohl auch möglich)
- Adresse (frei definierbar)
  - `/track/{1–8}/selected`
- Wert (frei definierbar)
  - `{0,1}`

DAW

- Bitwig Studio
- Extensions für Controller (Java oder JS by the way)
- Große OpenSource Library
  - https://github.com/git-moss/DrivenByMoss
  - enthält umfangreiche OSC-Extension

Controller-GUI

- Viele Komponenten (jeder Track wiederholt alle Bedienelemente)
- Fast alle Controls können sich updaten, da DAW außerhalb des Controllers geändert werden kann
- modernes GUI-Framework für macOS
- also: SwiftUI

OSC & Swift

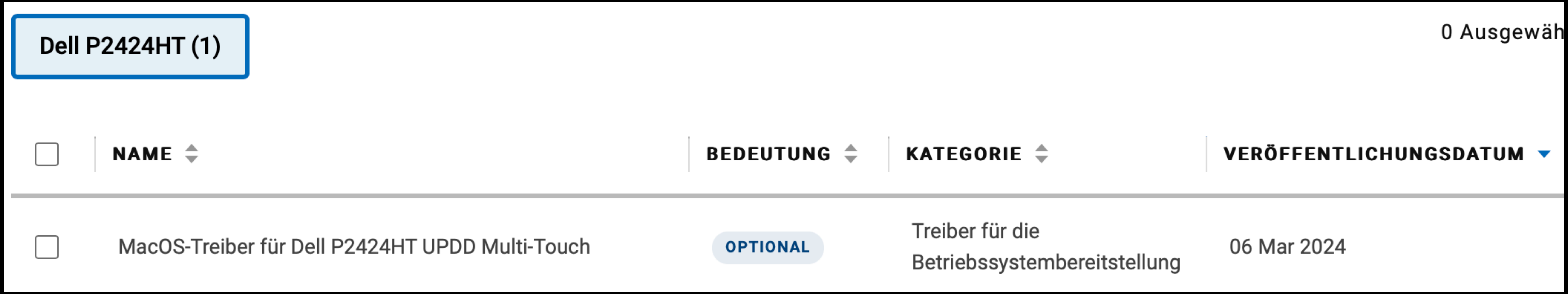
- https://github.com/orchetect/OSCKit
- Senden & Empfangen

SwiftUI & Multitouch?

- `func touchesMoved(with event: NSEvent)`
    - Trackpad-Events
- `DragGesture`
    - Cursor

Multitouch-Displays

- Wie kommt man an diese Events heran innerhalb von macOS?
  - Treiber selbst schreiben?
- Dell P2424HT

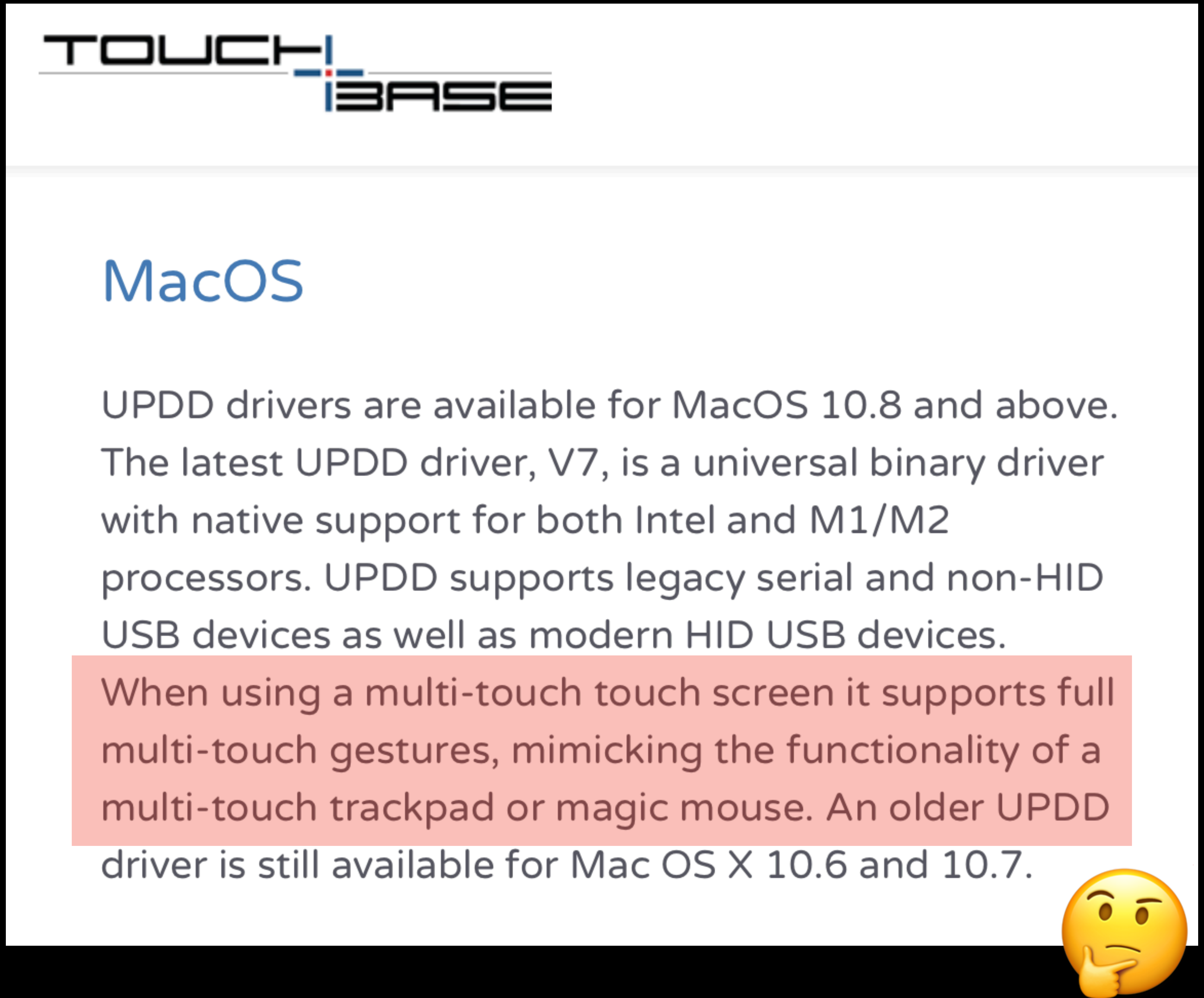


Dell P2424HT (1)

0 Ausgewäh

	NAME	BEDEUTUNG	KATEGORIE	VERÖFFENTLICHUNGSDATUM
☐	MacOS-Treiber für Dell P2424HT UPDD Multi-Touch	OPTIONAL	Treiber für die Betriebssystembereitstellung	06 Mar 2024

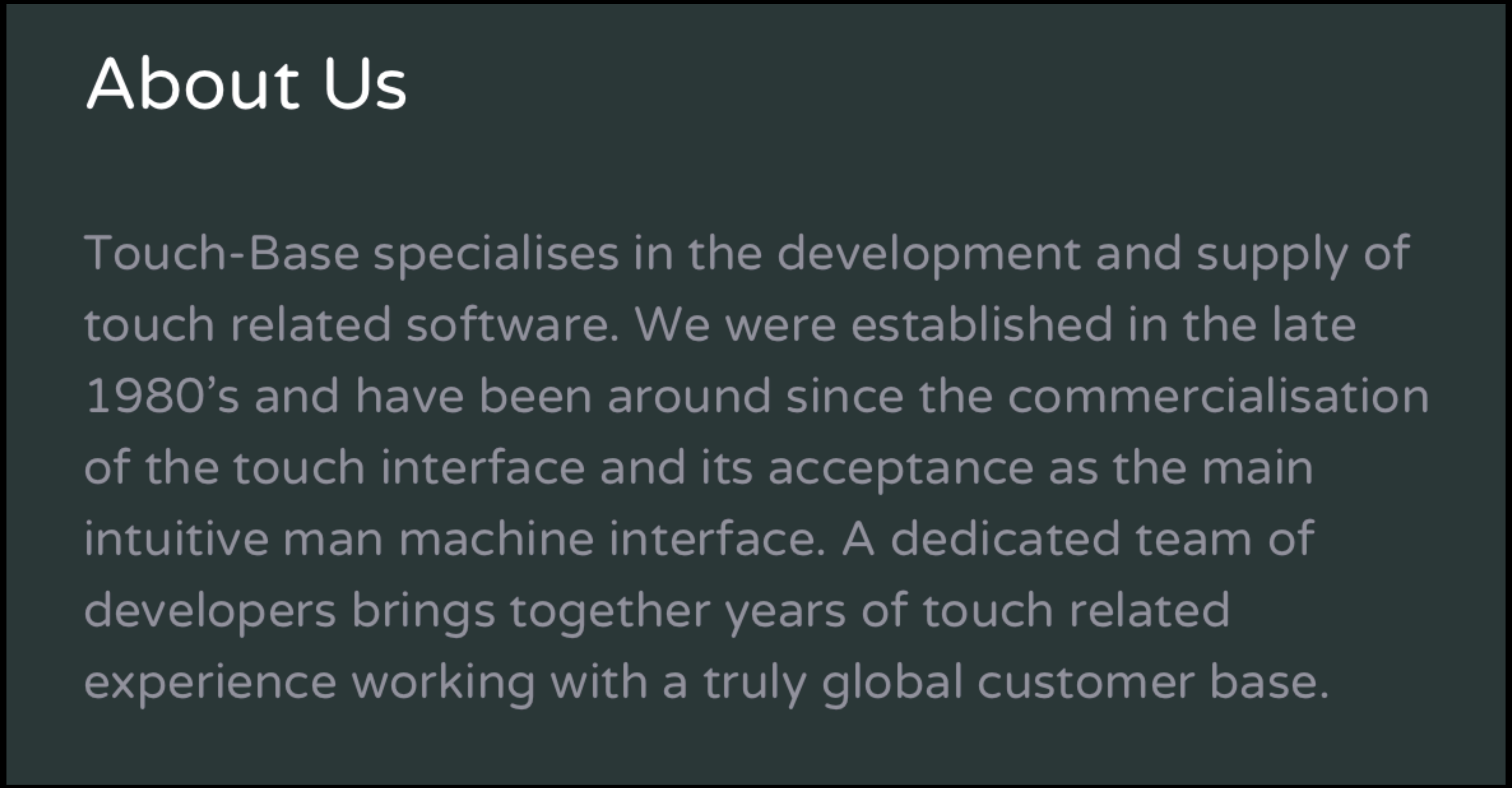
UPDD



TOUCH-BASE

MacOS

UPDD drivers are available for MacOS 10.8 and above. The latest UPDD driver, V7, is a universal binary driver with native support for both Intel and M1/M2 processors. UPDD supports legacy serial and non-HID USB devices as well as modern HID USB devices. When using a multi-touch touch screen it supports full multi-touch gestures, mimicking the functionality of a multi-touch trackpad or magic mouse. An older UPDD driver is still available for Mac OS X 10.6 and 10.7.

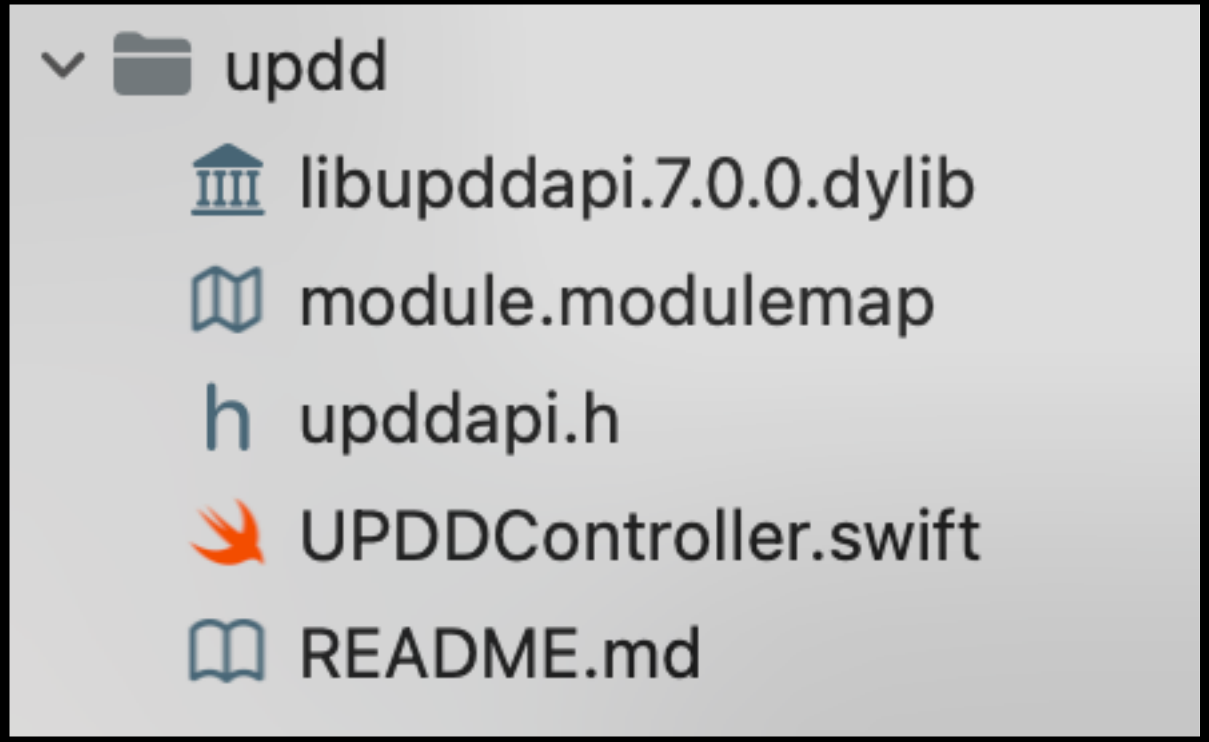


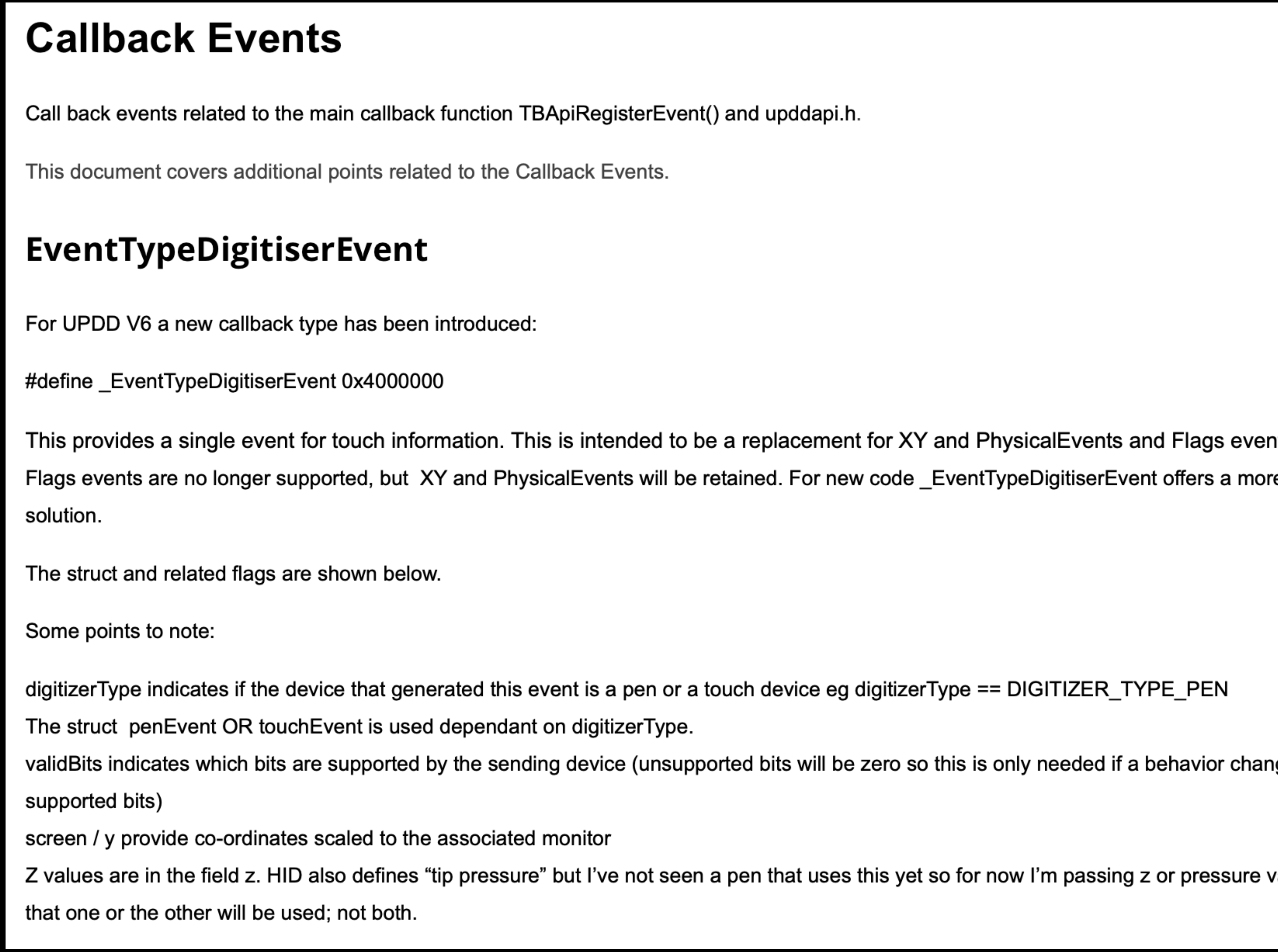
About Us

Touch-Base specialises in the development and supply of touch related software. We were established in the late 1980's and have been around since the commercialisation of the touch interface and its acceptance as the main intuitive man machine interface. A dedicated team of developers brings together years of touch related experience working with a truly global customer base.

UPDD – back then

- https://support.touch-base.com/Documentation/50135/Utilising-the-API
- C API


updd
libupddapi.7.0.0.dylib
module.modulemap
upddapi.h
UPDDController.swift
README.md


Callback Events

Call back events related to the main callback function TBApiRegisterEvent() and upddapi.h.

This document covers additional points related to the Callback Events.

EventTypeDigitiserEvent

For UPDD V6 a new callback type has been introduced:

#define _EventTypeDigitiserEvent 0x4000000

This provides a single event for touch information. This is intended to be a replacement for XY and PhysicalEvents and Flags even
Flags events are no longer supported, but XY and PhysicalEvents will be retained. For new code _EventTypeDigitiserEvent offers a more
solution.

The struct and related flags are shown below.

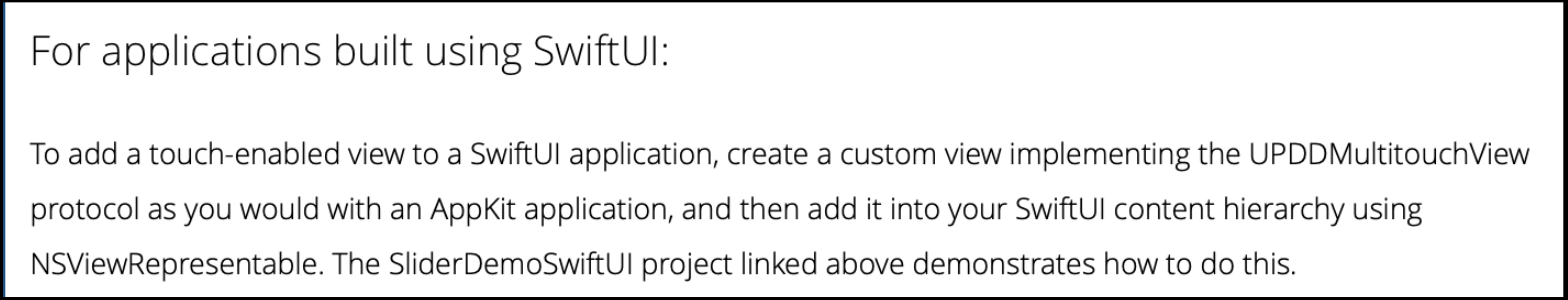
Some points to note:

digitizerType indicates if the device that generated this event is a pen or a touch device eg digitizerType == DIGITIZER_TYPE_PEN
The struct penEvent OR touchEvent is used dependant on digitizerType.
validBits indicates which bits are supported by the sending device (unsupported bits will be zero so this is only needed if a behavior chan
supported bits)
screen / y provide co-ordinates scaled to the associated monitor
Z values are in the field z. HID also defines “tip pressure” but I’ve not seen a pen that uses this yet so for now I’m passing z or pressure v
that one or the other will be used; not both.

DEMO

UPDD – now

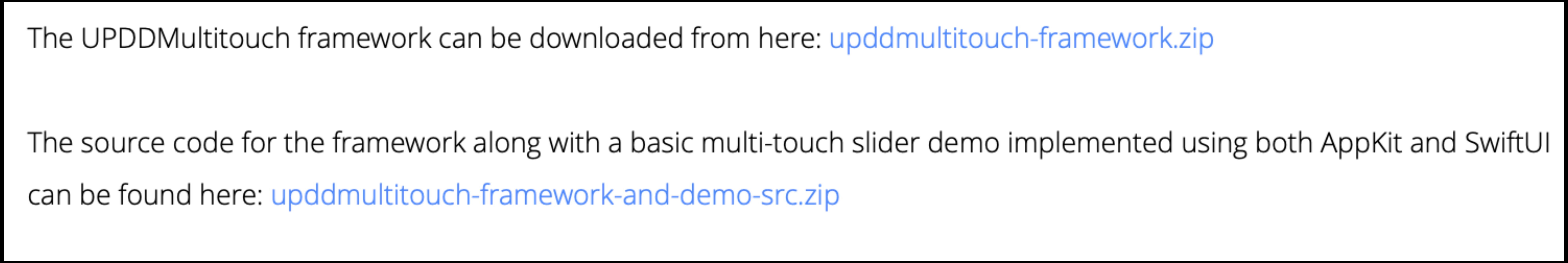
- https://support.touch-base.com/Documentation/50847/macOS-Native-Client-Applications



For applications built using SwiftUI:

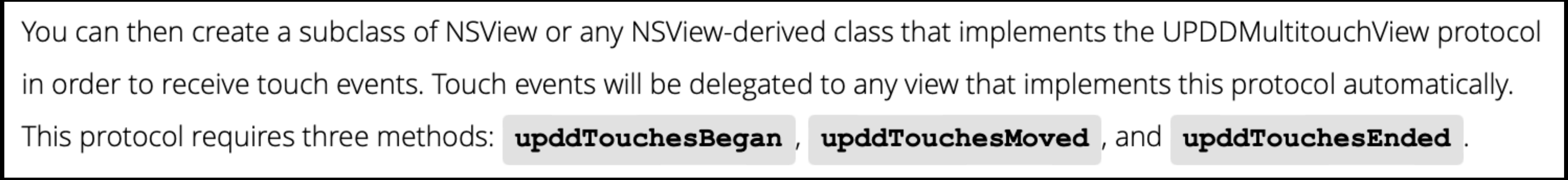
To add a touch-enabled view to a SwiftUI application, create a custom view implementing the UPDDMultitouchView protocol as you would with an AppKit application, and then add it into your SwiftUI content hierarchy using NSViewRepresentable. The SliderDemoSwiftUI project linked above demonstrates how to do this.

UPDD – now



The UPDDMultitouch framework can be downloaded from here: upddmultitouch-framework.zip

The source code for the framework along with a basic multi-touch slider demo implemented using both AppKit and SwiftUI can be found here: upddmultitouch-framework-and-demo-src.zip



You can then create a subclass of NSView or any NSView-derived class that implements the UPDDMultitouchView protocol in order to receive touch events. Touch events will be delegated to any view that implements this protocol automatically. This protocol requires three methods: `upddTouchesBegan`, `upddTouchesMoved`, and `upddTouchesEnded`.