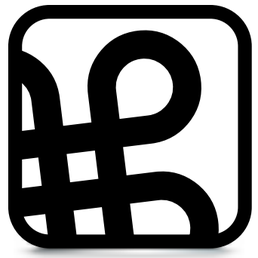


Macoun’11

Volle Kontrolle mit (und über) HID

Matthias Krauß
Multigrad UG

Suppenkoma

Ablauf

- HID – Was, warum, wie?
- HID: Konzepte
- HID in OSX
- HID-Kontrolle Schritt für Schritt
- Bonus: HID im echten Leben

HID – Was, warum, wie?

Was ist HID?

a) „Human Interface Device“

b) USB HID Device Class Definition

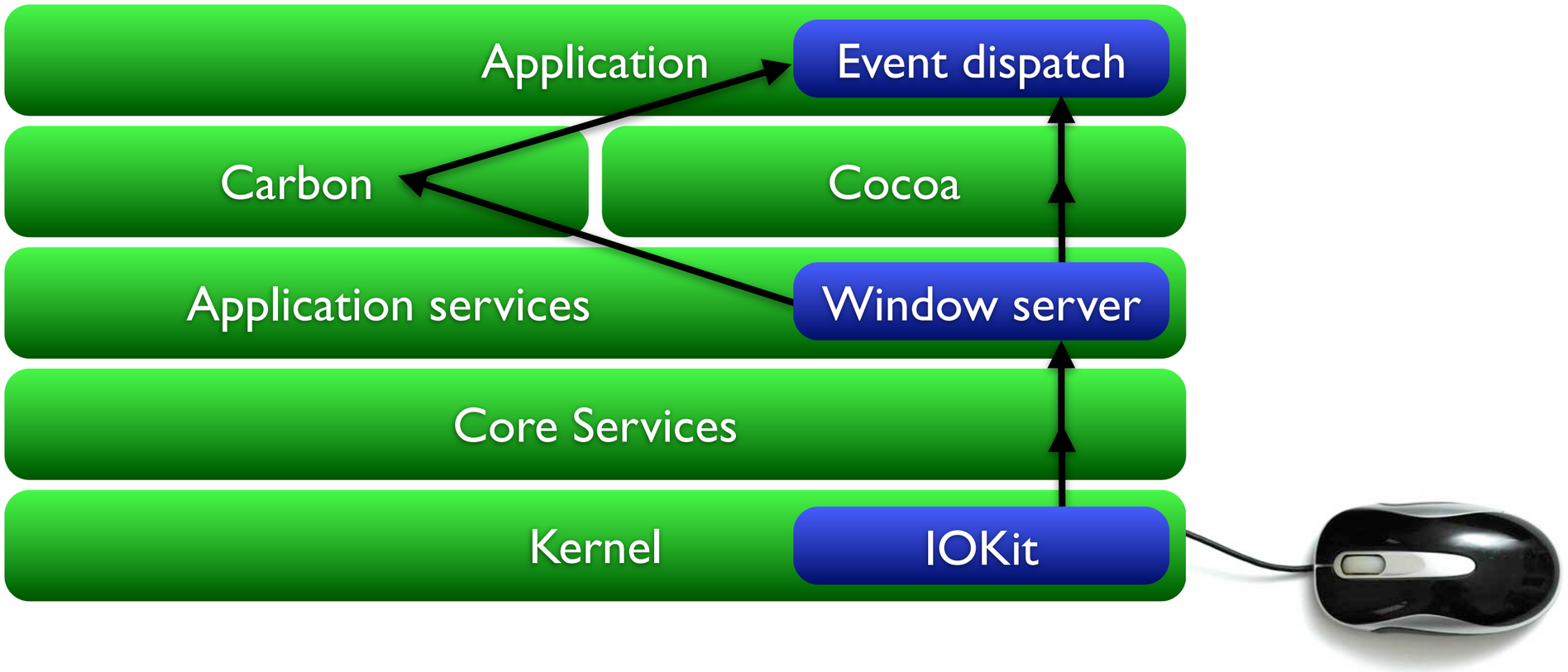
c) HID-Protokolle auf anderen Transport Layers *

d) OSX HID Subsystem *IOHIDLib* *

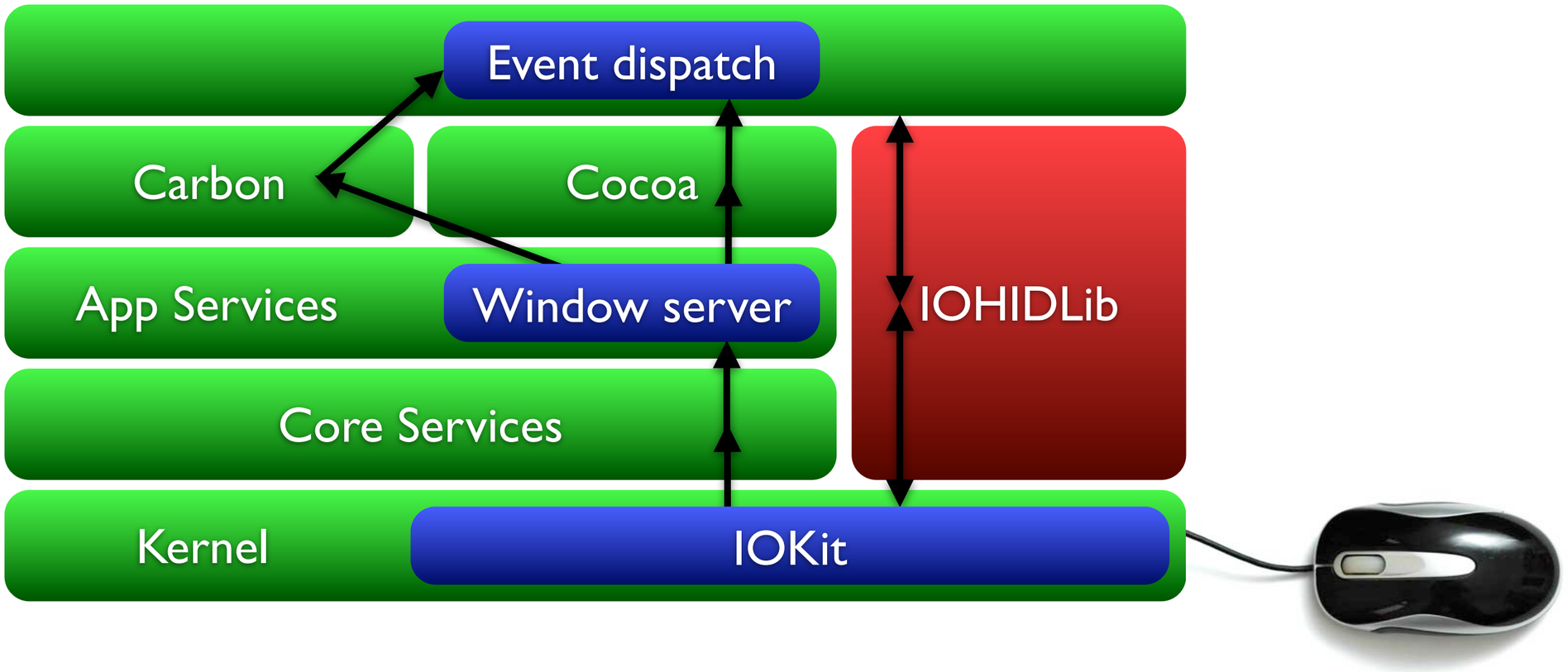
Warum HID?

- OSX versteckt HID gut
- Manchmal zu gut
- Unterstützung anderer Eingabegeräte
- Alle Features bekannter Geräte
- Filterung: Rohdaten, Latenz, Coalescing
- Rückkanal
- Tolles Protokoll für eigene Hardwareprojekte

HID - Wie?

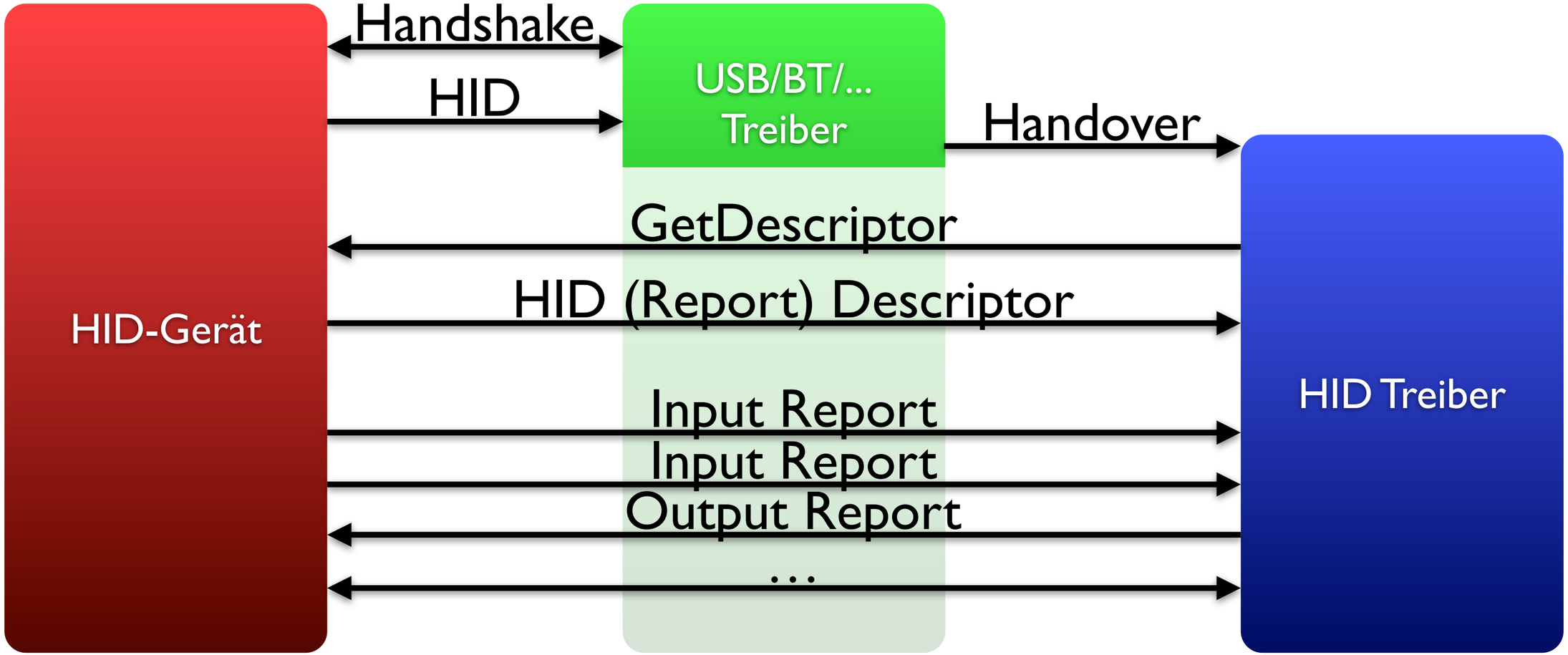

Application
Event dispatch
Carbon
Cocoa
Application services
Window server
Core Services
Kernel
IOKit

HID - Wie?


Event dispatch
Carbon
Cocoa
App Services
Window server
Core Services
Kernel
IOKit
IOHIDLib

HID: Konzepte

Ablauf

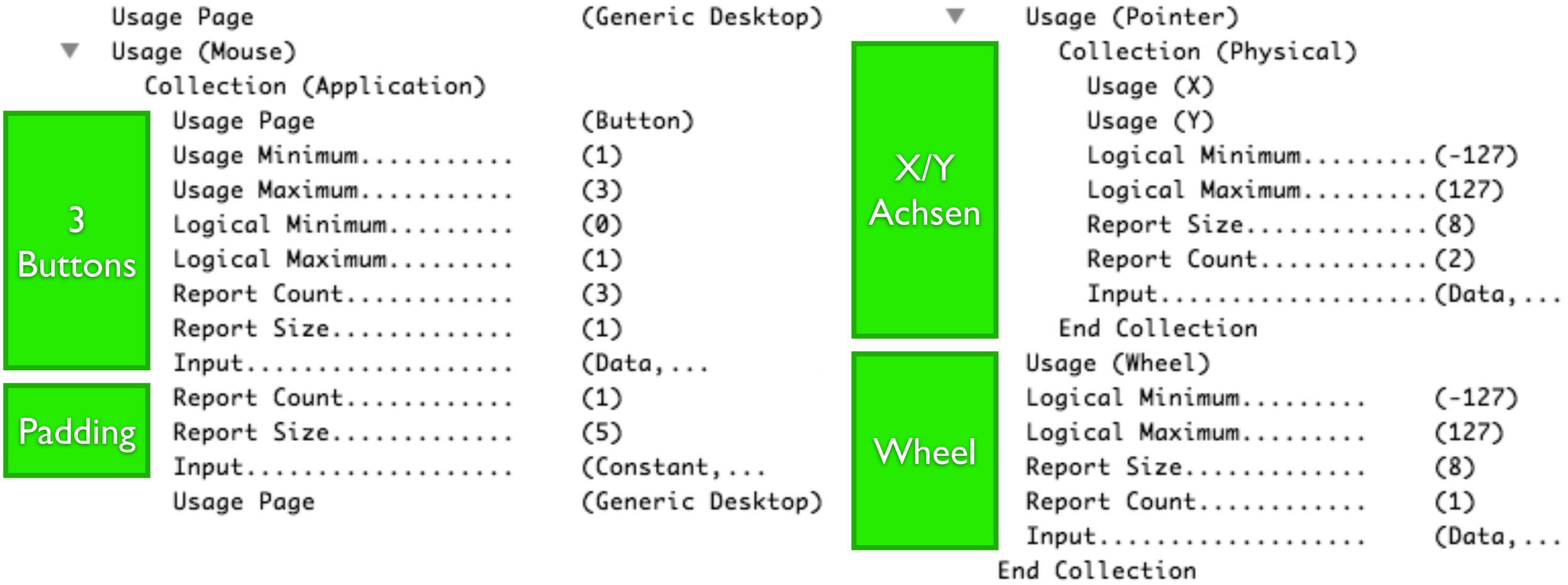

HID-Gerät
Handshake
HID
USB/BT/...
Treiber
Handover
HID Treiber
GetDescriptor
HID (Report) Descriptor
Input Report
Input Report
Output Report
...

HID Report Descriptor

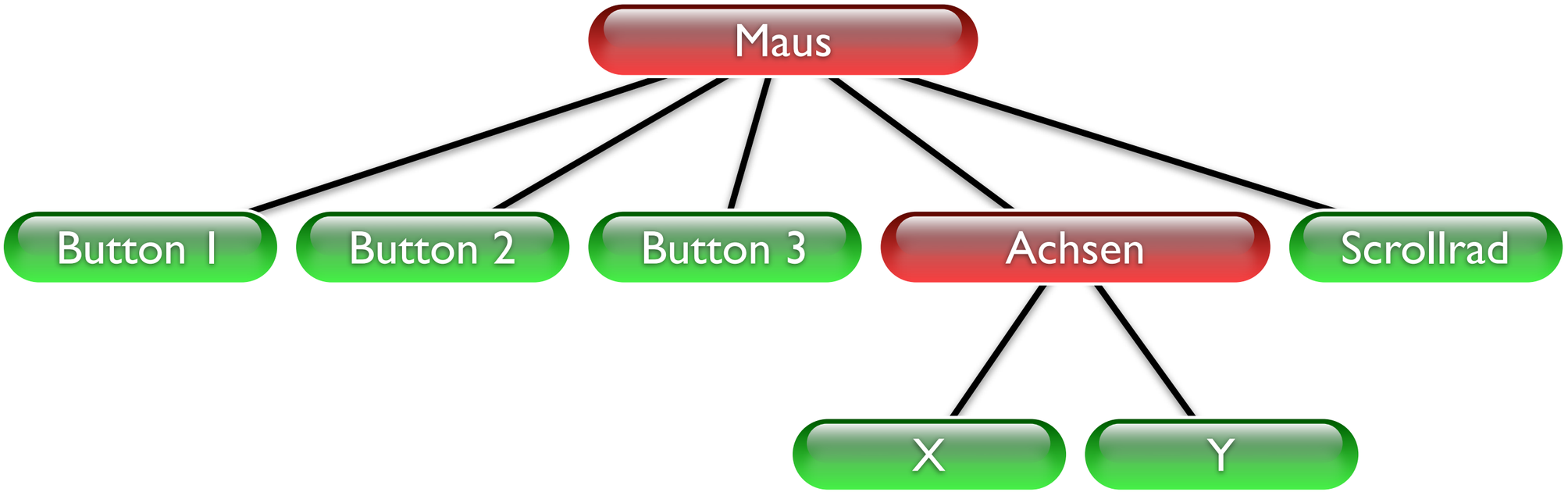
- Binärformat
- Beschreibt:
  - a) Gerätestruktur als Baum von Elementen
  - b) Datenformat von ein- und ausgehenden Reports
- Jedes Element in Syntax und Semantik

Demo

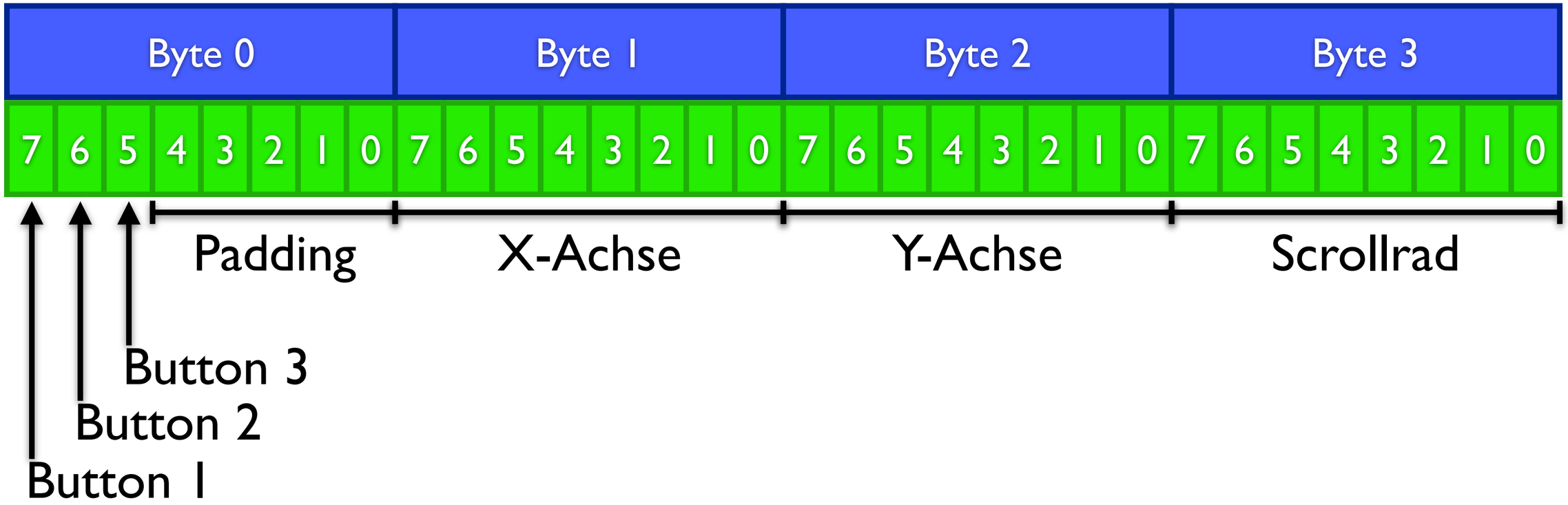
Anatomie einer Maus



Maus: Elementbaum

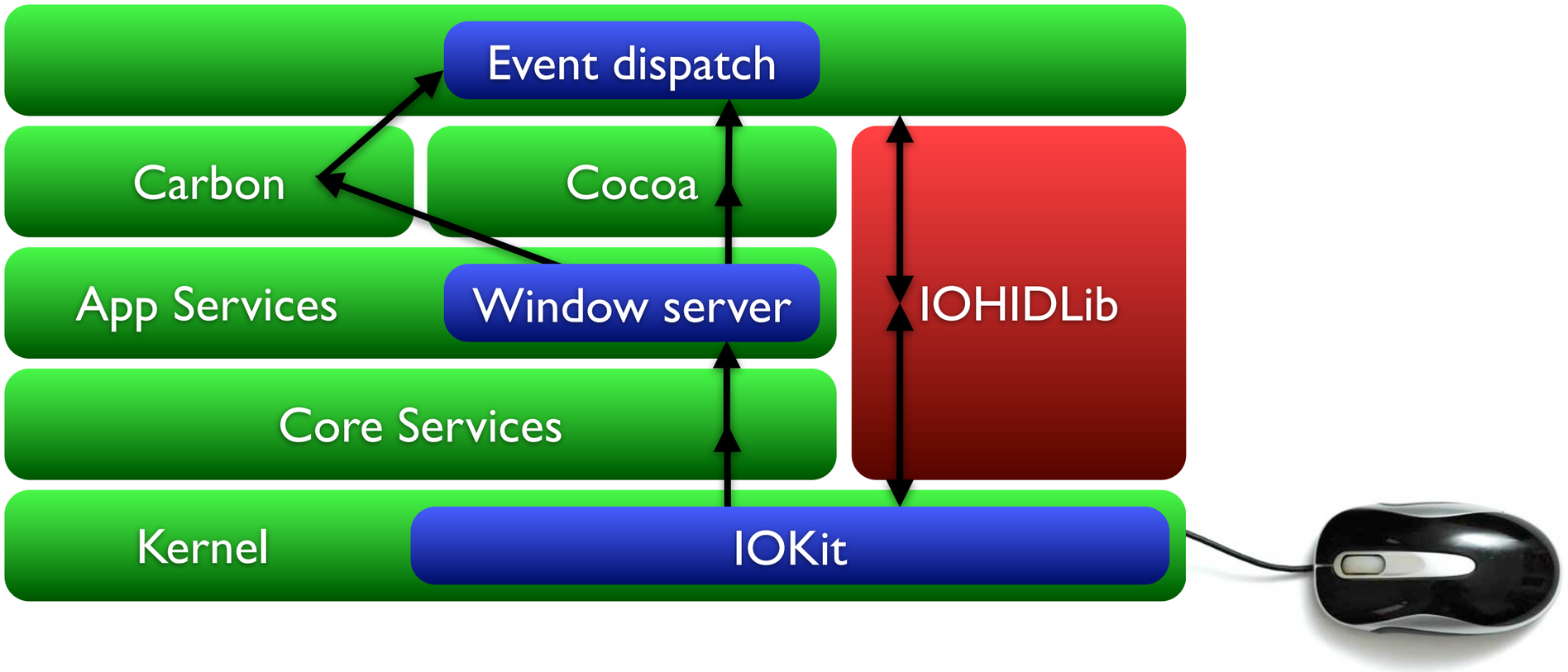


Maus: Input Report 1



HID in OSX

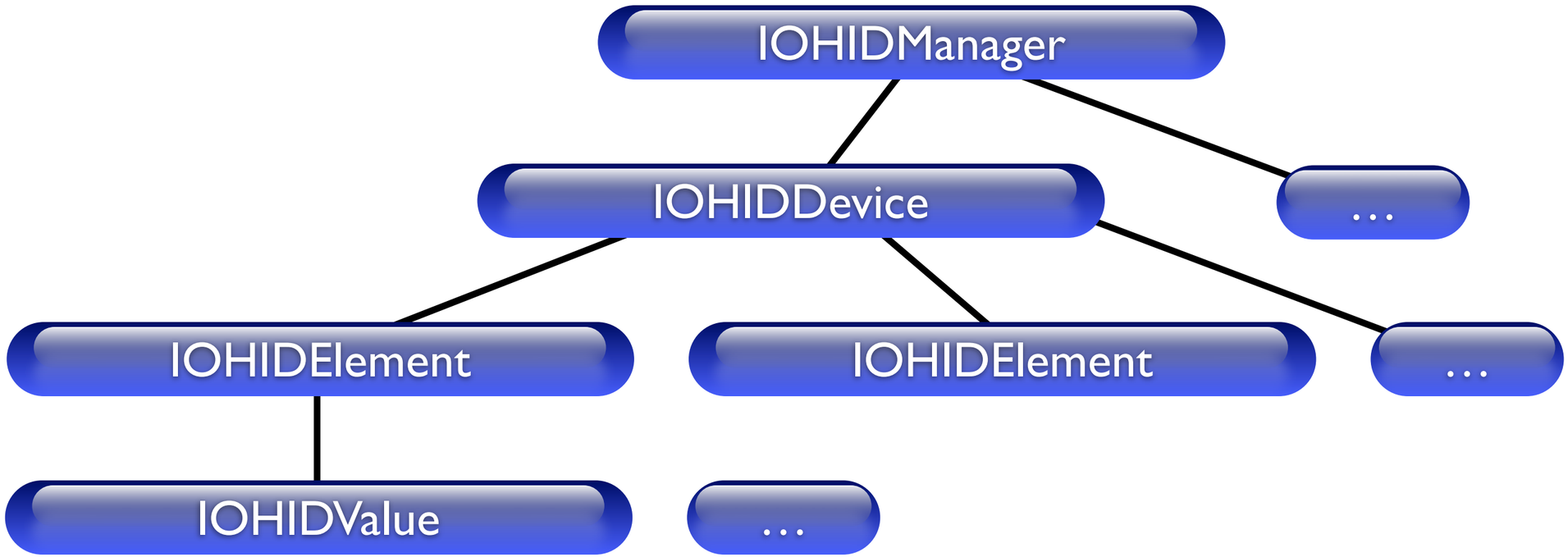
HID - Wie?


Event dispatch
Carbon
Cocoa
App Services
Window server
Core Services
Kernel
IOKit
IOHIDLib

IOHIDLib

- Einfachere API ab 10.5
- Core Foundation-Konventionen
- Toll-Free Bridging ist ein Freund!
- Framework: `/System/Library/Frameworks/IOKit.framework`
- `#import <IOKit/hid/IOHIDLib.h>` *

IOHIDLib


IOHIDManager
IOHIDDevice
…
IOHIDElement
IOHIDElement
…
IOHIDValue
…

IOHIDManager

- Typischerweise eine Instanz pro Anwendung
- Zugang zu verfügbaren Geräten
- Hot plugging
- Werte und Reports geräteübergreifend

IOHIDDevice

- Typischerweise eine Instanz pro Gerät
- Beschreibung
- Werte und Reports gerätespezifisch lesen
- Werte und Reports gerätespezifisch schreiben

HID-Kontrolle Schritt für Schritt

1. Hello HID!

```
IOHIDManagerRef hid = IOHIDManagerCreate(kCFAllocatorDefault, 0);
IOHIDManagerOpen(hid, 0);

NSDictionary* matchDict = [NSDictionary dictionary];
IOHIDManagerSetDeviceMatching(hid, (CFDictionaryRef)matchDict);

CFSetRef devices = IOHIDManagerCopyDevices(hid);
// Etwas tolles mit den Ergebnissen anstellen
CFRelease(devices);

IOHIDManagerClose(hid, 0);
CFRelease(hid);
```

Demo

2. Hot plugging

```
IOHIDManagerRegisterDeviceMatchingCallback
                      (hid, pluggedCallback, self);
IOHIDManagerScheduleWithRunLoop
                      (hid, CFRunLoopGetMain(), kCFRunLoopDefaultMode);
```

```
void pluggedCallback (void* context, IOReturn result,
                      void* sender, IOHIDDeviceRef device) {
   id myself = (id)context;
   [myself devicePlugged:device];
}
```

```
- (void) devicePlugged:(IOHIDDeviceRef)device {
  // Unfug mit dem eingesteckten Geraet treiben
}
```

Demo

3a. Elemente finden

```
NSDictionary* matchDict =
                        [NSDictionary dictionaryWithObjectsAndKeys:…];
CFArrayRef elements = IOHIDDeviceCopyMatchingElements
                        (device, (CFDictionaryRef)matchDict, 0);
// Etwas lustiges mit den Elementen machen
CFRelease(elements);
```

Key	Value
kIOHIDElementUsagePageKey	kHIDPage_GenericDesktop
kIOHIDElementUsageKey	kHIDUsage_GD_X
kIOHIDElementTypeKey	kIOHIDElementTypeInput_Misc

3b. Wertänderungen abonnieren

```
IOHIDDeviceSetInputValueMatchingMultiple(device, matchArray);
IOHIDDeviceRegisterInputValueCallback(device, valueCallback, self);
// Optional: RunLoop setzen (sonst wie IOHIDManager)
```

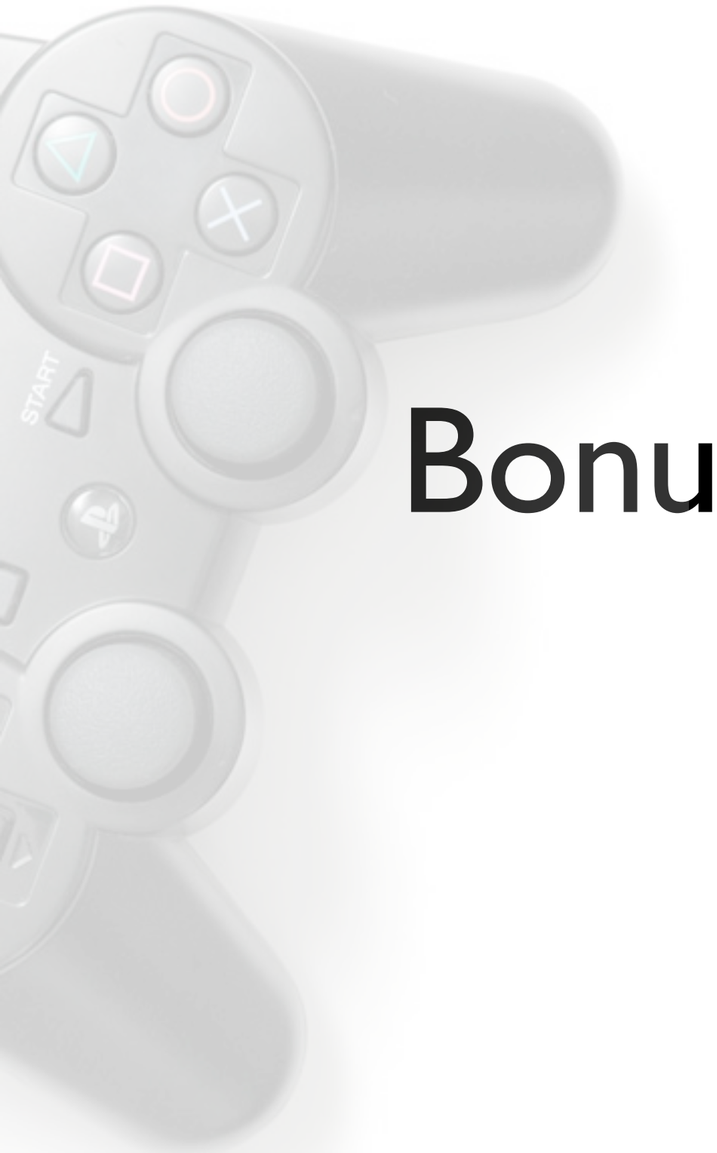
```
void valueCallback(void* context, IOReturn result,
                   void* sender, IOHIDValueRef value) {
   id myself = (id)context;
   [myself valueCallback:value];
}
```

```
- (void) valueCallback:(IOHIDValueRef)value {
  // Wert auslesen
}
```

Demo

4. Den Rest anbinden

Demo



Bonus: HID im echten Leben

(alles subjektiv)

Alles ganz anders

- Zwischenschicht:
  - Physikalische / anwendungslogische Elemente
  - Player matching
  - Preferences, Kalibrierung
  - Multimodale Kontrolle
- ForceFeedback.framework, TUIO etc.

Dos und Don’ts

- Keine Angst!
- Wenn möglich asynchron
- Matching ausgiebig nutzen
- IOHIDDevice: Mindestens einen Callback setzen
- IOHIDValue prüfen
- Selbst besser machen als andere

HID-Konformität

- Spezifikation vs. „geht unter XP“
- nicht verstanden vs. nicht hinbekommen vs. ignoriert
- Gut: Tastaturen, Mäuse, Trackballs, Joysticks, Lenkräder etc.
- Mittelmäßig: Tablets, Gamepads, Wiimote etc.
- Schmerzhaft: Alles andere
- Ansatz: Standardkonform + Platzhirsche

Wrapper

- Grundsätzlich ok
- Hauptvorteil: Gerätespezifische Tweaks (theoretisch)
- Oft buggy, schlecht gepflegt
- Alte API
- Arbeitserleichterung vs. Universalität

Dokumentation

- HID Class Device intro
  http://developer.apple.com/library/mac/#documentation/DeviceDrivers/Conceptual/HID/intro/intro.html
- USB HID Specification
  http://www.usb.org/developers/devclass_docs/HID1_11.pdf
  http://www.usb.org/developers/devclass_docs/Hut1_12v2.pdf
  http://www.usb.org/developers/hidpage/
- Dokumentation in IOHIDManager.h, IOHIDDevice.h, …
- HID Explorer

Der richtige Spaß

- IOKit open source
  http://www.opensource.apple.com/source/IOHIDFamily/
  http://www.opensource.apple.com/source/IOUSBFamily/IOUSBFamily-440.4.1/IOUSBHIDDriver/
- USB Prober: USB Logger
- Linux-Treiber
- USB Snoopy
- Logic Analyzer

Fragen?

krauss@multigrad.de

Vielen Dank

5. HID Reports

```
NSMutableData* reportBuffer;
IOHIDDeviceRegisterInputReportCallback (deviceRef,
                                        [reportBuffer mutableBytes],
                                        [reportBuffer length],
                                        myReportCallback,
                                        self);
```

```
void myReportCallback (void* context, IOReturn result,
                       void* sender,  IOHIDReportType type,
                       uint32_t reportID, uint8_t* report,
                       CFIndex reportLength) {
    // Gemeine Sachen machen
}
```

Demo