SCIENCES SORBONNE UNIVERSITÉ

«OKCorral»

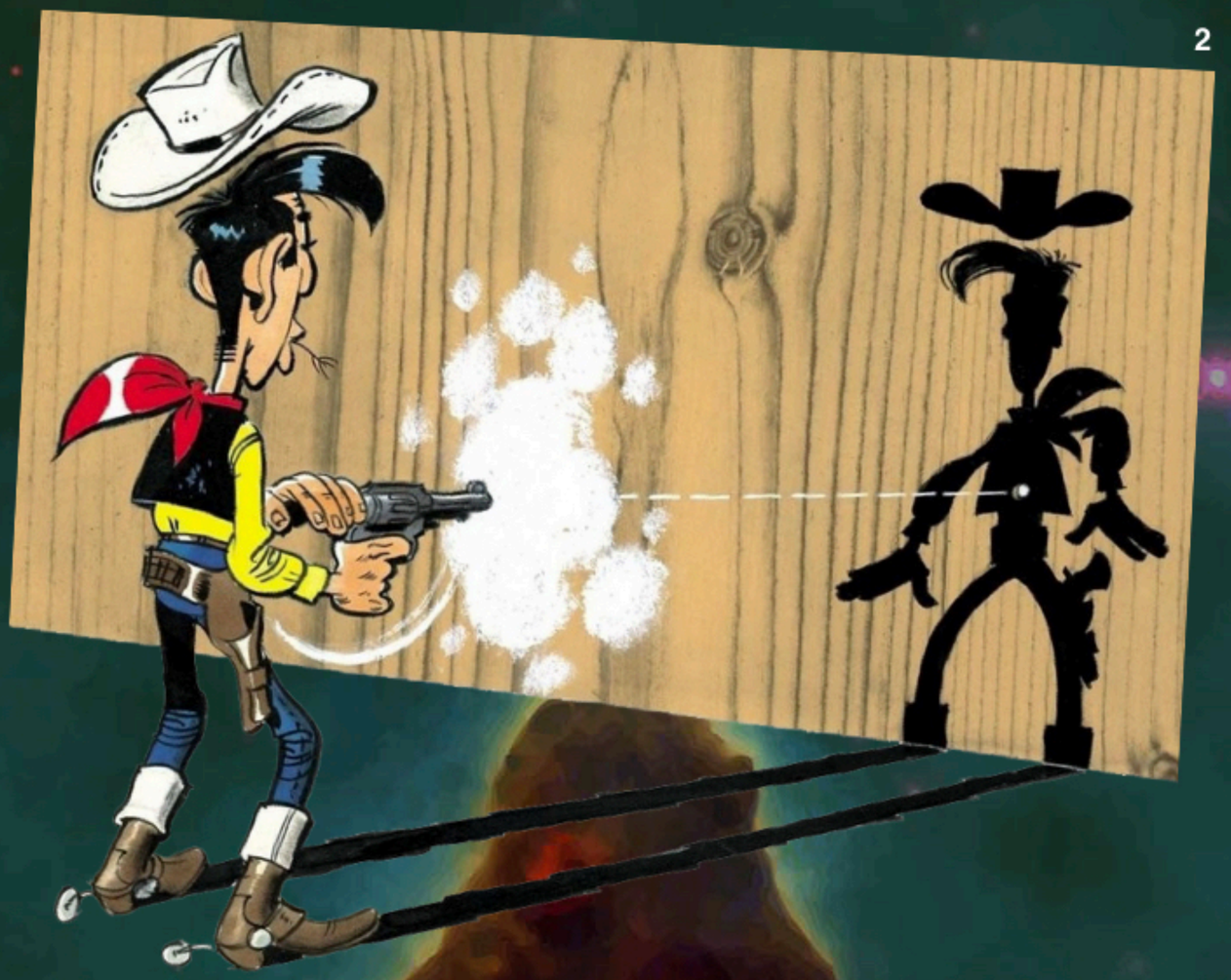
Fabrice.Kordon@lip6.fr

LIP6

Goal of the example

Play a duel

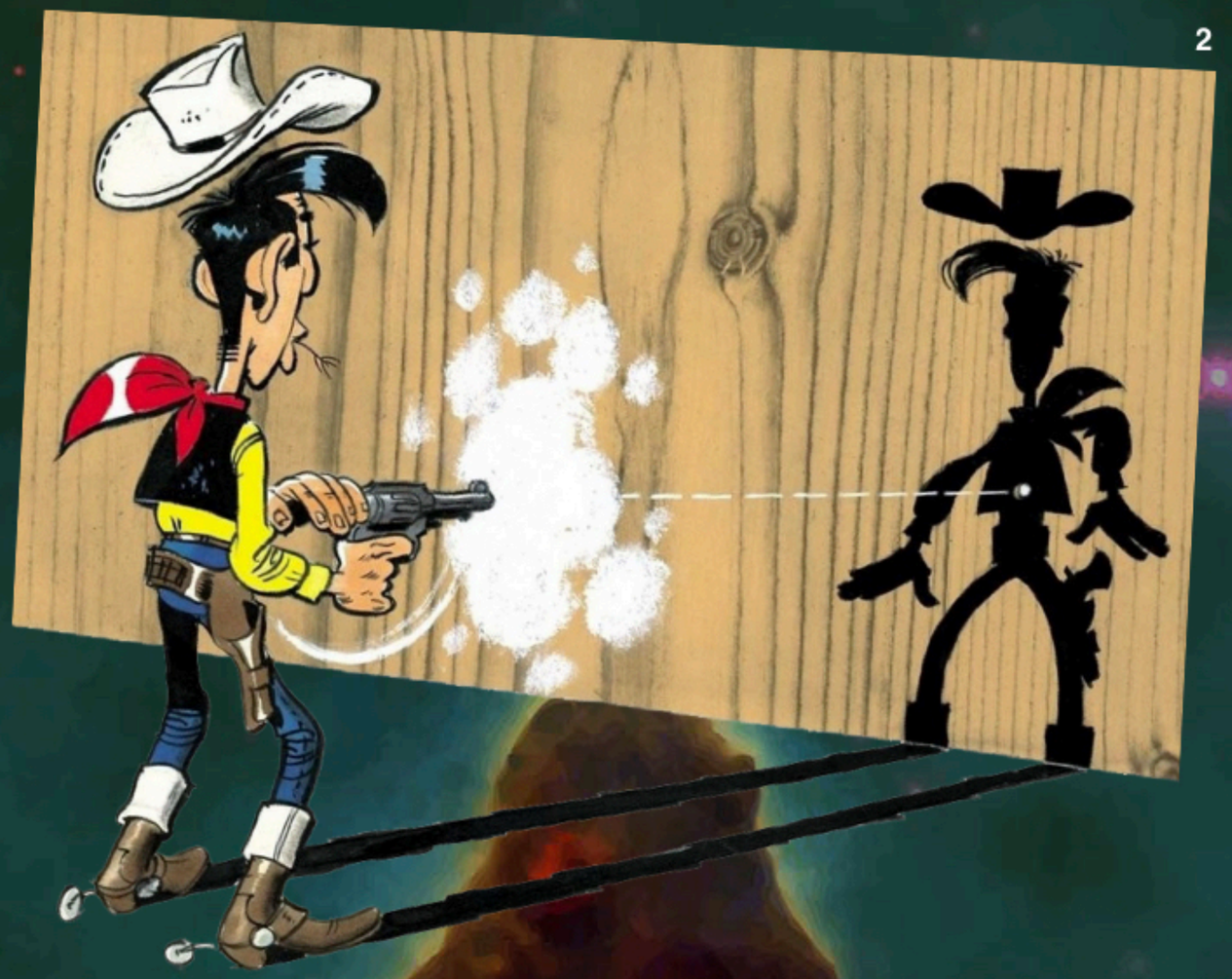
- Like « Gunfight at the O.K. Corral»
  - October 26, 1881
  - Tombstone, Arizona, USA
  - About 30 seconds
  - 3 dead people
- Numerous films…



To do

- On demand, search for a service
  - _okcorral
- Once detected
  - Propos a «fire» button
- Fire
  - _fire
  - The first one to receive this service is dead, the other unregister

Goal of the example



Play a duel

- Like « Gunfight at the O.K. Corral»
  - October 26, 1881
  - Tombstone, Arizona, USA
  - About 30 seconds
  - 3 dead people
- Numerous films…

To do

- On demand, search for a service
  - _okcorral
- Once detected
  - Propos a «fire» button
- Fire
  - _fire
  - The first one to receive this service is dead, the other unregister

Eliminate yourself

Neither as enemy nor as a target

Demo

A bit about conventions

Two services

- When you «search» for the enemy
  - _okcorral._tcp.
  - Detected = I saw you
- When you «fire» at the enemy
  - _fire._tcp.
  - Detected = I got a bullet in my chest

Our players are «Gun Ace» (never miss)

How to proceed

- You listen for both services
- Service _fire._tcp. is published only when «Fire» is tapped
  - The «fire» button is shown only when you «see» the adversary

ViewController

Sake of simplicity…

Code located in a ViewController

ViewController


```
import UIKit

class ViewController: UIViewController,
                      NetServiceDelegate,
                      NetServiceBrowserDelegate {

    private let myBounds = UIScreen.main.bounds
    private let search = UIButton(type: .system)
    private let fire = UIButton(type: .system)
    private let state = UITextView()
    private let debug = UITextView()
    private let bg = UIImageView(image: UIImage(named: "tombstone"))

    private var whoIsVisible = ""

    private var seeService : NetService?
    private var fireService : NetService?
    private var seeBrowser : NetServiceBrowser?
    private var fireBrowser : NetServiceBrowser?

    private var lineNb = 1 // number lines in debug mode
```

ViewController


```
override func viewDidLoad() {
    super.viewDidLoad()
    self.view = UIView()
    self.view.addSubview(bg)

    search.setTitle("Searching ennemy", for: .normal)
    search.tintColor = UIColor.white
    search.titleLabel?.font = UIFont.boldSystemFont(ofSize: 24)
    search.addTarget(self, action: #selector(doSearch(s:)), for: .touchDown)
    self.view.addSubview(search)

    fire.setTitle("fire!", for: .normal)
    fire.tintColor = UIColor.red
    fire.titleLabel?.font = UIFont.boldSystemFont(ofSize: 60)
    fire.addTarget(self, action: #selector(doFire(s:)), for: .touchDown)
    fire.isHidden = true
    self.view.addSubview(fire)

    state.font = UIFont.boldSystemFont(ofSize: 24)
    state.backgroundColor = UIColor.clear
    state.textAlignment = .center
    state.isEditable = false
    state.isSelectable = false
    self.view.addSubview(state)

    debug.backgroundColor = UIColor(red: 1, green: 1, blue: 1, alpha: 0.5)
    debug.isHidden = true
    debug.isEditable = false
    debug.isSelectable = false
    self.view.addSubview(debug)
    self.drawInSize()
}
```

ViewController


```
func drawInSize() {
    bg.center = CGPoint(x: myBounds.size.width / 2,
                        y: myBounds.size.height / 2)
    search.frame = CGRect(x: myBounds.size.width / 2 - 120,
                          y: 30, width: 240, height: 60)
    fire.frame = CGRect(x: myBounds.size.width / 2 - 80,
                        y: myBounds.size.height - 255,
                        width: 160, height: 80)
    state.frame = CGRect(x: 20, y: myBounds.size.height - 150,
                         width: myBounds.size.width - 40,
                         height: 100)
    debug.frame = CGRect(x: 20, y: 80,
                         width: myBounds.size.width - 40,
                         height: myBounds.size.height - 320)
}
```

ViewController


```
@objc func doSearch (s: UIButton) {
    self.traceInfo(s: " ==> search engaged...")
    search.isHidden = true
    seeService = NetService(domain: "local",
                            type: "_okcorral._tcp.",
                            name: UIDevice.current.name,
                            port: 9090)
    seeService?.delegate = self
    seeService?.includesPeerToPeer = true // Activate Bluetooth too
    seeService?.publish()
    // the service for searching
    seeBrowser = NetServiceBrowser()
    seeBrowser?.delegate = self
    seeBrowser?.searchForServices(ofType: "_okcorral._tcp.",
                                  inDomain: "local")
    // the service for setting up firing
    fireBrowser = NetServiceBrowser()
    fireBrowser?.delegate = self
    fireBrowser?.searchForServices(ofType: "_fire._tcp.",
                                   inDomain: "local")
}
```

ViewController


```
@objc func doFire (s: UIButton) {
    self.traceInfo(s: "==> I fired...")
    fireService = NetService(domain: "local",
                             type: "_fire._tcp.",
                             name: UIDevice.current.name,
                             port: 9090)
    fireService?.includesPeerToPeer = true // Activate Bluetooth
    fireService?.delegate = self
    fireService?.publish()
}
```

```
func traceInfo (s : String) {
    if debug.isHidden {
        debug.isHidden = false
    }
    debug.text = debug.text + "\(lineNb) - " + s + "\n"
    lineNb += 1
    debug.setNeedsDisplay()
}
```

ViewController


```swift
//  NSNetServiceBrowserDelegate Protocol

func netServiceBrowser(_ browser: NetServiceBrowser,
                       didFind service: NetService,
                       moreComing: Bool) {
    self.traceInfo(s: "service found : \(service.type)/\(service.name)")
    if service.type == "_okcorral._tcp." {
        // We do not fire on ourself
        if service.name != UIDevice.current.name {
            state.backgroundColor = UIColor(red: 1, green: 0.8, blue: 0, alpha: 0.5)
            state.text = "I see \(service.name)"
            whoIsVisible = service.name // find on who we fire
            fire.isHidden = false
        }
    } else if service.type == "_fire._tcp." {
        // Eliminate our own bullets ;-)
        if service.name != UIDevice.current.name {
            state.text = "AAAAAAAAAAAAARG!\n\(service.name) killed me..."
            fire.isHidden = true
            seeService?.stop()
            fireService?.stop()
        }
    }
}
```

ViewController


```
func netServiceBrowser(_ browser: NetServiceBrowser,
                       didRemove service: NetService,
                       moreComing: Bool) {
    self.traceInfo(s: "netServiceBrowser (didRemove = \(service.name))")
    if service.name == whoIsVisible {
        state.text = "BRAVO!\n you killed\n\(service.name)"
        fire.isHidden = false
    }
}

func netServiceBrowser(_ browser: NetServiceBrowser,
                       didFindDomain domainString: String,
                       moreComing: Bool) {
    self.traceInfo(s: "netServiceBrowser (didFindDomain = \(domainString))")
}
func netServiceBrowser(_ browser: NetServiceBrowser,
                       didRemoveDomain domainString: String,
                       moreComing: Bool) {
    self.traceInfo(s: "netServiceBrowser (didRemoveDomain = \(domainString))")
}

func netServiceBrowser(_ browser: NetServiceBrowser,
                       didNotSearch errorDict: [String : NSNumber]) {
    self.traceInfo(s: "netServiceBrowser (didNotSearch = \(errorDict))")
}
func netServiceBrowserWillSearch(_ browser: NetServiceBrowser) {
    self.traceInfo(s: "netServiceBrowserWillSearch ()")
}
func netServiceBrowserDidStopSearch(_ browser: NetServiceBrowser) {
    self.traceInfo(s: "netServiceBrowserDidStopSearch ()")
}
```

ViewController


```
    // NSNetServiceDelegate protocol

    func netServiceWillPublish(_ sender: NetService) {
        self.traceInfo(s: "netServiceWillPublish (\(sender.type)/\(sender.name))")
    }

    func netService(_ sender: NetService,
                    didNotPublish errorDict: [String : NSNumber]) {
        self.traceInfo(s: "netService didNotPublish (\(sender.type)/\(sender.name), \
(errorDict))")
    }

    func netServiceDidPublish(_ sender: NetService) {
        self.traceInfo(s: "netServiceDidPublish (\(sender.type)/\(sender.name))")
    }

    func netServiceWillResolve(_ sender: NetService){
        self.traceInfo(s: "netServiceWillResolve (\(sender.type)/\(sender.name))")
    }

    func netService(_ sender: NetService,
                    didNotResolve errorDict: [String : NSNumber]) {
        self.traceInfo(s: "netService didNotResolve (\(sender.type)/\(sender.name), \
(errorDict))")
    }
```

ViewController


```
    func netServiceDidResolveAddress(_ sender: NetService) {
        self.traceInfo(s: "netServiceDidResolveAddress (\(sender.type)/\(sender.name))")
    }

    func netService(_ sender: NetService,
                    didUpdateTXTRecord data: Data) {
        self.traceInfo(s: "netService (\(sender.type)/\(sender.name), didUpdateTXTRecord=
(data.base64EncodedString())")
    }

    func netServiceDidStop(_ sender: NetService) {
        self.traceInfo(s: "netServiceDidStop (\(sender.type)/\(sender.name))")
    }
}
```

As a conclusion…

Nice is'n't it?

- Maybe it's easy too

Now you may play with it…

- This brings added value to your Apps
- Numerous ones already use this
  - Local data exchange (business card)
  - Also used in Airplay (for detection & configuration)