SCIENCES SORBONNE UNIVERSITÉ

Views, zoom & scroll

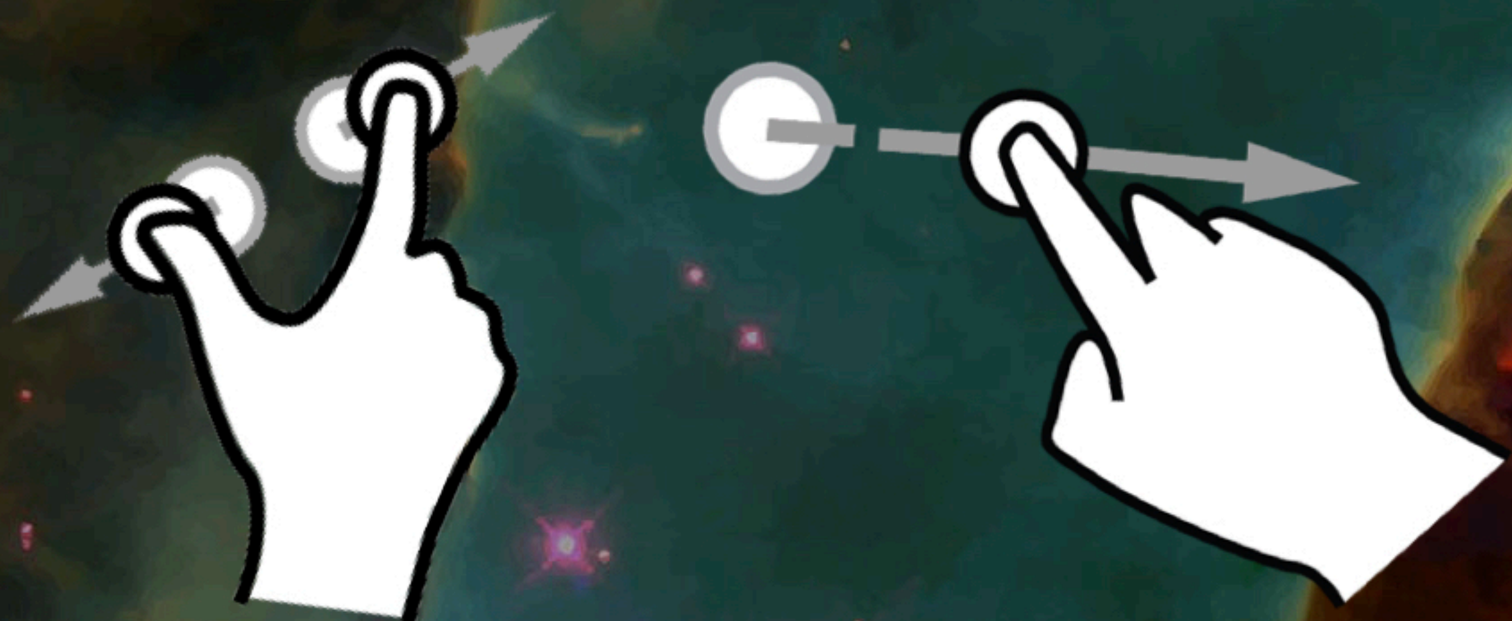
Fabrice.Kordon@lip6.fr

LIP6

As an introduction…

When a view is too large for the screen

- You display a part only
- You can scroll
- You can drag



This is built-in UIScrollView

- Handling scale and position
- Can be done «manually» (reinvent the wheel)

Delegation

Let's explain this notion

- First real contact

Key mechanism for iOS

- Used a lot
  - **Maps, sensors, etc**
- Here a «simple way» to operate

Already encountered *«in a different form»*

- Orientation handling
  - **This was an «implicit protocol»**

UIScrollView, principles

Encapsulation of the view

- The UIScrollView embeds the view to be scrolled/zoomed

You must answer to UIScrollViewDelegate

- This is a protocol
- It allows to define the minimal/maximal scales
- You define a «delegate»
  - **Liaison with the framework**
  - **Implementation of predefined methods**
  - **Numerous «call back» to foresee**
- In our case, you fill some methods
  - **viewForZoomingInScrollView:**
  - **scrollViewDidEndZooming: scrollView withView: atScale**
  - **etc.**

Demo

ViewController


```swift
import UIKit

class ViewController: UIViewController {

    // To se the status bar to light colors
    override var preferredStatusBarStyle: UIStatusBarStyle {
        return .lightContent
    }

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view, typically from a nib.
        self.view = MyScrollView(frame:UIScreen.main.bounds)
        self.view.backgroundColor = UIColor.white
    }
}
```

MyScrollView


```swift
import UIKit

class MyScrollView: UIView, UIScrollViewDelegate {

    private let myImageView = UIImageView(image: UIImage(named: "Pillars_of_Creation"))
    private let scrollV = UIScrollView(frame: UIScreen.main.bounds)

    override init (frame: CGRect) {
        scrollV.maximumZoomScale = 1.0
        scrollV.minimumZoomScale = 0.05
        scrollV.backgroundColor = UIColor.white
        scrollV.addSubview(myImageView)
        scrollV.setZoomScale(0.2, animated: true)
        super.init(frame: frame); // We need self now
        self.addSubview(scrollV)
        scrollV.delegate = self // self = target of the protocol methods
    }

    required init?(coder aDecoder: NSCoder) {
        fatalError("init(coder:) has not been implemented")
    }
```

MyScrollView


```
    func viewForZooming(in scrollView: UIScrollView) -> UIView? {
        return myImageView // There could be a selection among several images
    }

    func scrollViewDidEndZooming(_ scrollView: UIScrollView,
                                 with view: UIView?,
                                 atScale scale: CGFloat) {
        scrollView.zoomScale = scale
        // For one dimension only, it could be
        //view?.frame.size.width = view?.frame.size.width ?? 0.0 * scale
    }
}
```

MyScrollView


```
    func viewForZooming(in scrollView: UIScrollView) -> UIView? {
        return myImageView // There could be a selection among several images
    }

    func scrollViewDidEndZooming(_ scrollView: UIScrollView,
                                 with view: UIView?,
                                 atScale scale: CGFloat) {
        scrollView.zoomScale = scale
        // For one dimension only, it could be
        //view?.frame.size.width = view?.frame.size.width ?? 0.0 * scale
    }
}
```

Trick

You may have several resolution of the same image and switch to the appropriate one according to the current scale

A bit of objective-C?


```
//
//  MyScrollView.h
//  zoomObjc
//
//  Created by Fabrice Kordon on 30/09/2018.
//  Copyright © 2018 Sorbonne Université. All rights reserved.
//

#import <UIKit/UIKit.h>

@interface MyScrollView : UIView <UIScrollViewDelegate>

@end
```

A bit of objective-C?


```
#import "MyScrollView.h"

@implementation MyScrollView

UIImageView *MyImageView;
UIScrollView *scrollV;

- (id) initWithFrame:(CGRect)frame {
    self = [super initWithFrame:frame];
    if (self) {
        UIImage *monImage = [UIImage imageNamed:@"orion"];
        MyImageView = [[UIImageView alloc] initWithImage:monImage];
        scrollV = [[UIScrollView alloc] initWithFrame:frame];
        [scrollV setBackgroundColor:[UIColor whiteColor]];
        [scrollV setMaximumZoomScale:1.0];
        [scrollV setMinimumZoomScale:0.05];
        [scrollV setDelegate:self]; // self is the delegate
        [scrollV addSubview:MyImageView];
        [self addSubview:scrollV];
        // Pas de release sur monImage?
        [MyImageView release];
        [scrollV release];
        [scrollV setZoomScale:0.2 animated:YES];
    }
    return self;
}
```

A bit of objective-C?


```objc
// Protocole UIScrollViewDelegate
- (UIView*)viewForZoomingInScrollView:(UIScrollView*) scrollView {
    return MyImageView;
}

- (void)scrollViewDidEndZooming:(UIScrollView*)scrollView
                       withView:(UIView *)view
                        atScale:(CGFloat)scale {
    [scrollView setZoomScale:scale];
}

@end
```

As a conclusion…

Nice and easy to get…

- You can do many things with it

The protocol is much richer

- Catch scrolling before and after
- Catch zooming before and after
- Etc.