SCIENCES SORBONNE UNIVERSITÉ

UISplitViewController

F. Kordon - Sorbonne Université - CC2018

Fabrice.Kordon@lip6.fr

LIP6

As an introduction…

- A way to use the space in a device

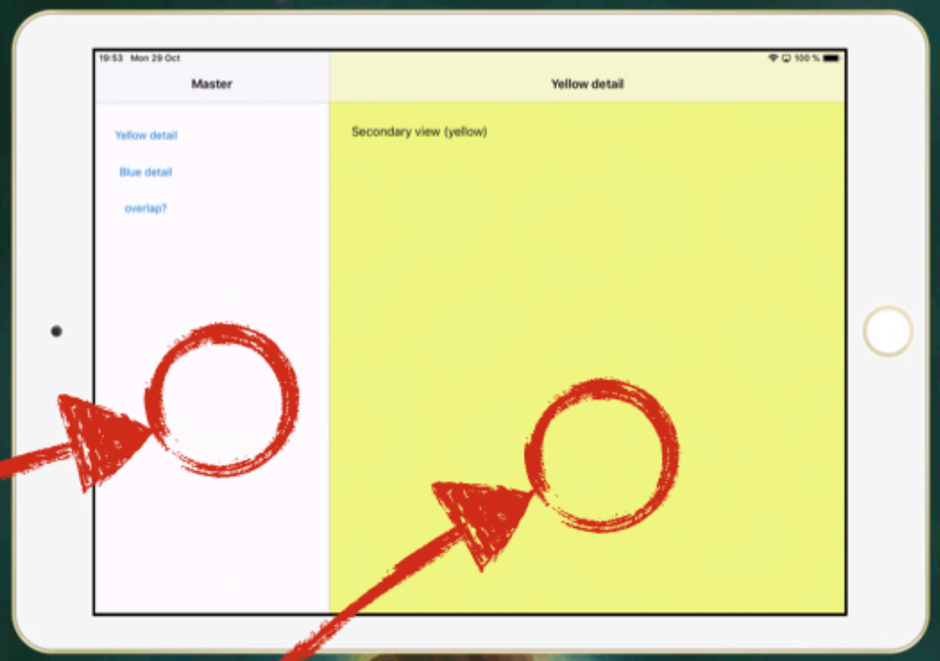
As an introduction…

A way to use the space in a device

Apple mail…

… on a large device

Typical architecture



AppDelegate

UISplitViewController

UINavigationController1

UINavigationController2

«master» ViewController

«detail» ViewController

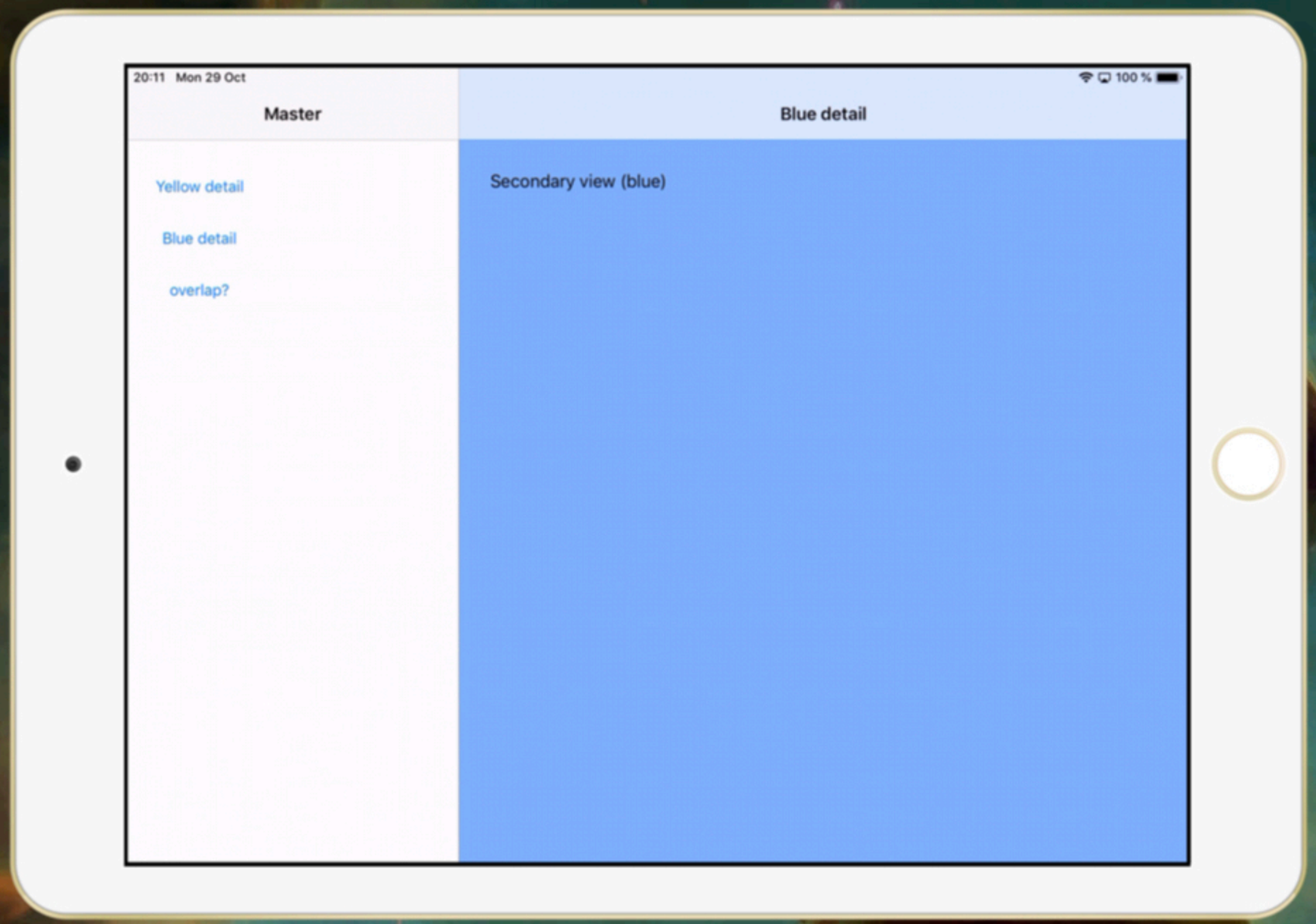
UIView1

UIView2

Orientation & side-by-side

Handled by the UISplitView

- At the level of the UISplitView only
- Master & detail views must handle this too



Orientation & side-by-side

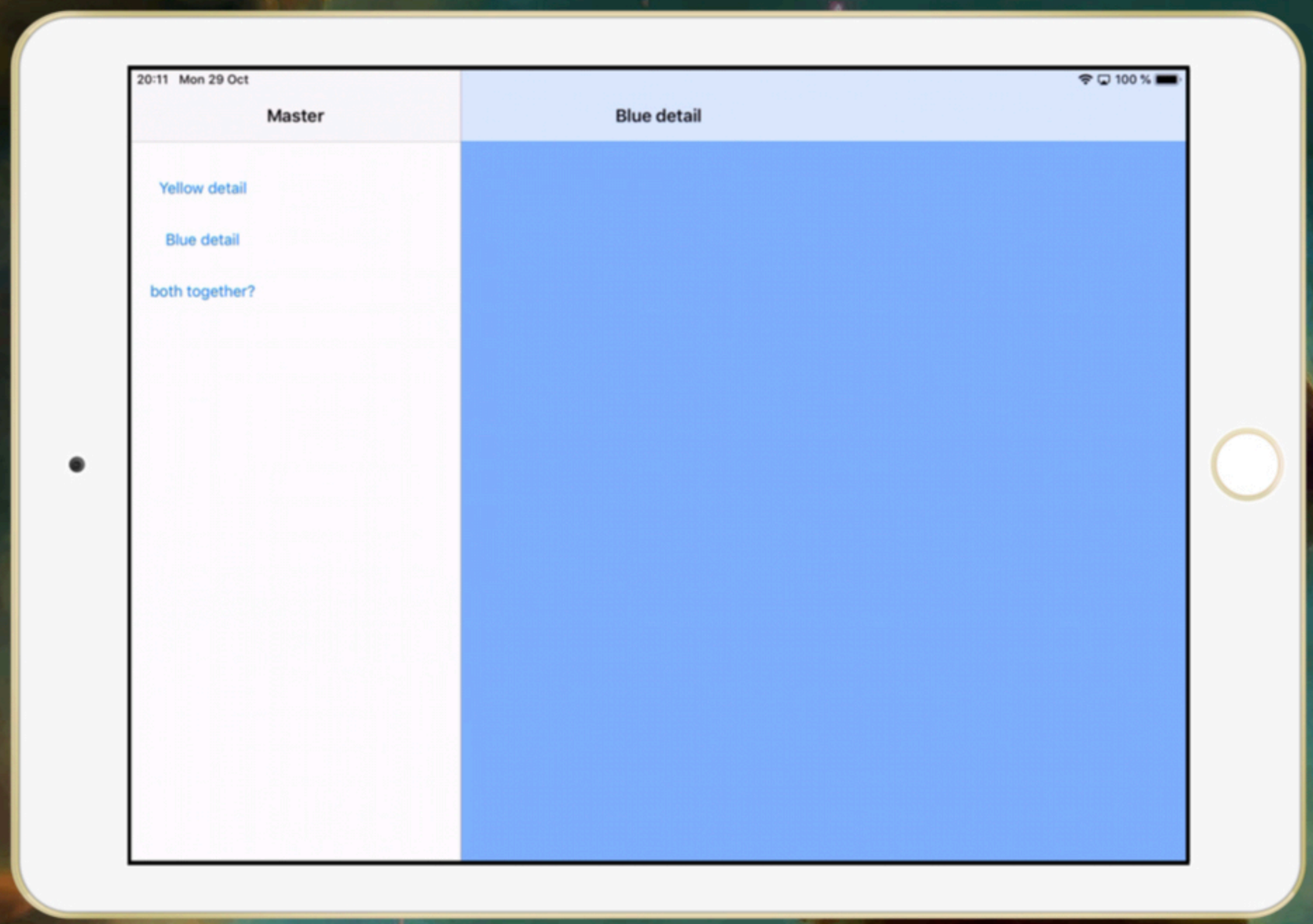
Handled by the UISplitView

- At the level of the UISplitView only
- Master & detail views must handle this too

Orientation & side-by-side

Handled by the UISplitView

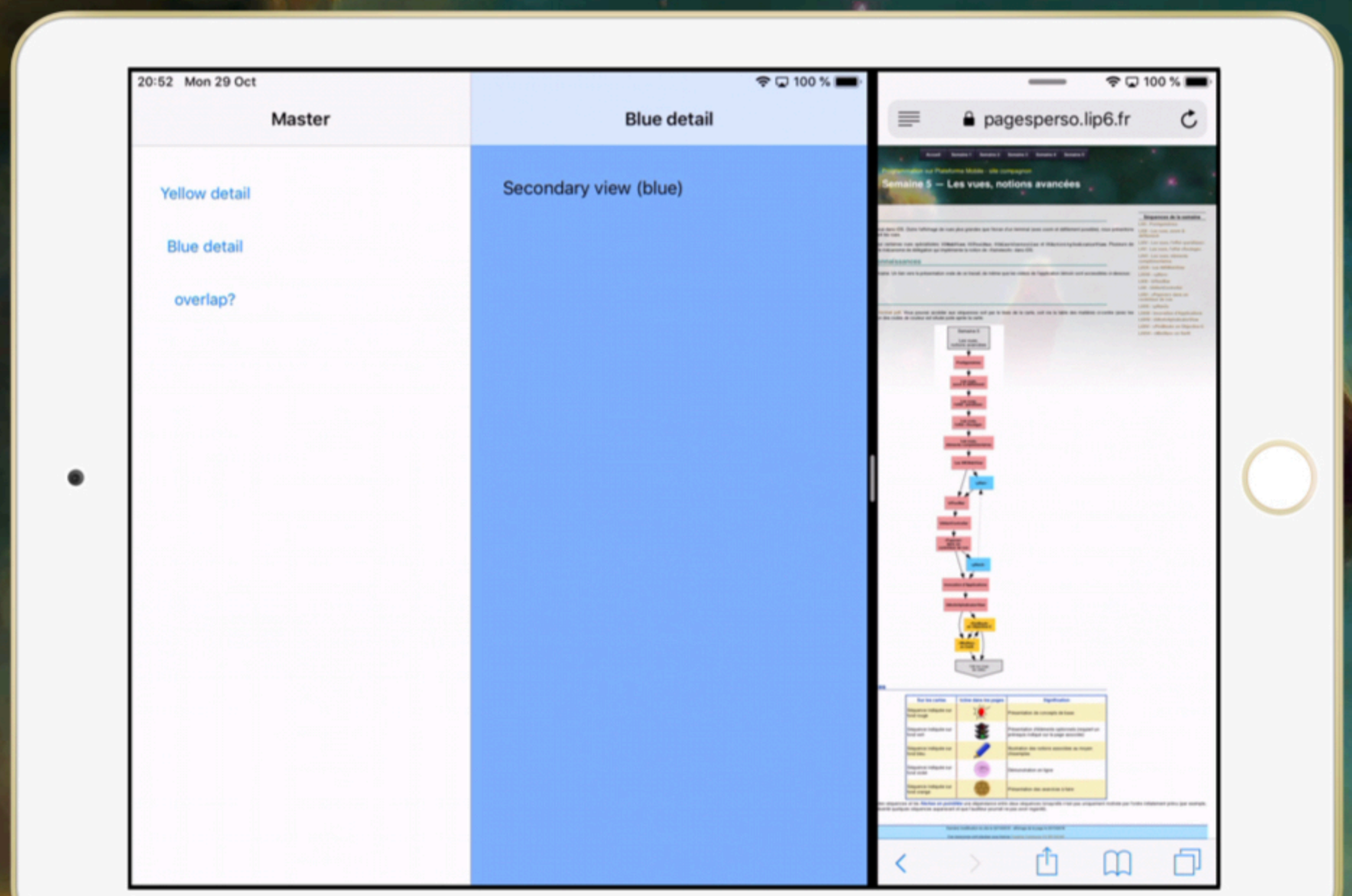
- At the level of the UISplitView only
- Master & detail views must handle this too



Orientation & side-by-side

Handled by the UISplitView

- At the level of the UISplitView only
- Master & detail views must handle this too



Handling the detail view?

- Use of delegation
  - UISplitViewControllerDelegate
- Who is taking care of it?

The master view?

The most obvious solution

Another idea?

A specialisation of a UISplitViewController

Principles

- **Creation of a UISplitViwController**
- **Elaboration of the associated view controllers**
  - Embedded in a UINavigationController
- **Association of the two involved controllers**
  - Property viewControllers
    - **Array of two UIViewController**
    - **first = master**
    - **second = detail**
  - Associate a delegate to the UISplitViewDelegate
    - **Usually the master view controller**

Replacing views

Approach 1 (😭)

- Replace the main view in the involved controller
- Poorly considered (but it works)
  - **No presentation animation from iOS**

Approach 2 (😍)

- Use dedicated primitives
  - **Inherited from UIViewController**

```
func show(_ vc: UIViewController, sender: Any?)

func showDetailViewController(_ vc: UIViewController, sender: Any?)
```

- The way to proceed
  - **Display is then optimised**

UISplitViewControllerDelegate

Display mode

- automatic (set a preferred style)
- primaryHidden
- allVisible
- primaryOverlay
  - **Primary may appear on swipe**

Changes of display

- When the delegate needs to get a display mode

```
func targetDisplayModeForAction(in svc: UISplitViewController)
                -> UISplitViewController.DisplayMode
```

- When the delegate must told the display mode changes

```
func splitViewController(_ svc: UISplitViewController,
            willChangeTo displayMode: UISplitViewController.DisplayMode)
```

- Etc.

UISplitViewControllerDelegate

Display mode

- automatic (set a preferred style)
- primaryHidden
- allVisible
- primaryOverlay
  - **Primary may appear on swipe**

Changes of displ

- When the delegate r splay mode

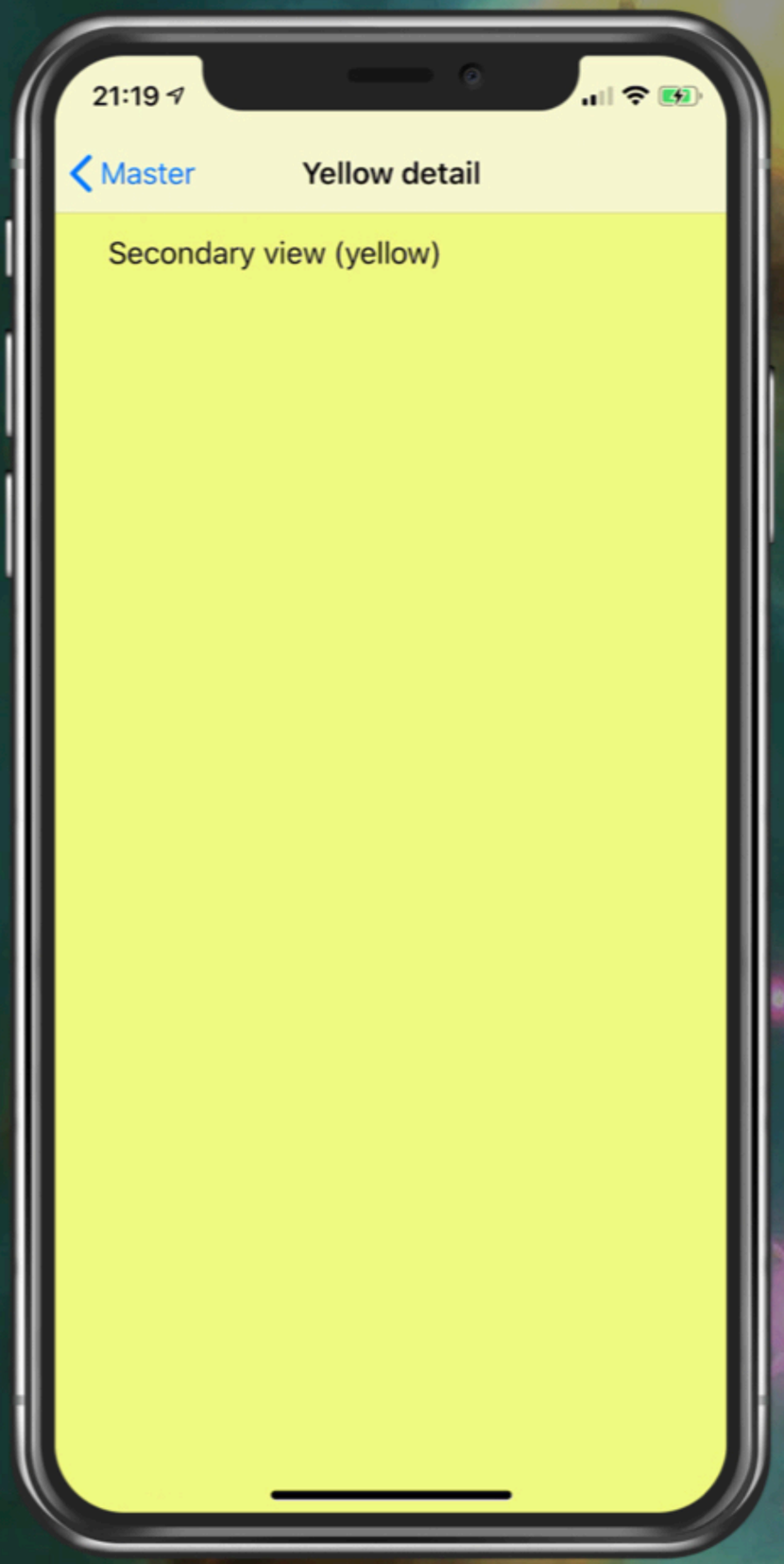
```
func targetDisplayModeForAct  svc: UISplitViewController)
        -> UISplitViewController.DisplayMode
```

- When the delegate must told the display mode changes

```
func splitViewController(_ svc: UISplitViewController,
        willChangeTo displayMode: UISplitViewController.DisplayMode)
```

- Etc.

What about small devices?



Before iOS8

since iOS8

- Handled in a small device
  - **Opening on the detailed view**
  - **Pop mechanism**

Usual substitution in a small device

- UITabBarController (2 items)
- UINavigationController
- Substitution handled explicitly
  - **In the AppDelegate**

As a conclusion…

Adapted to large devices

- Less adapted for small devices
  - **But some correspondance is supported**
- Association possible with a UITableViewController
  - **As a master view**
- Be aware of «strange devices»
  - **Phablets (6+, 6s+, 7+, 8+, X max)**
  - **Overlay possible (landscape mode)**

You must use it!