

Storyboards et UIKit

Votre UI part de là...

Storyboards et UIKit

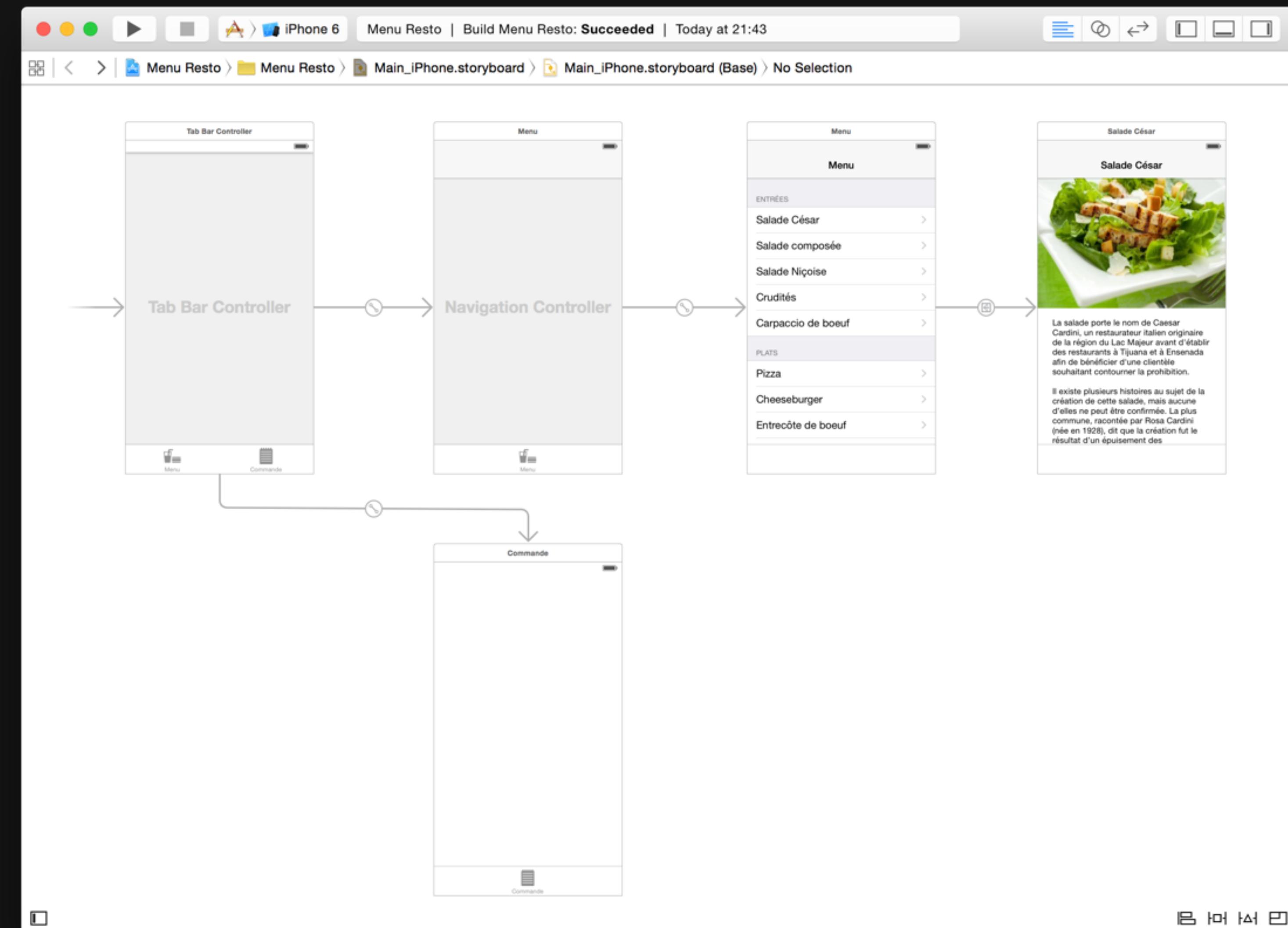
- Storyboards
- UIKit : organisation
- Adaptabilité de l'interface
- Objets courants
- Objets personnalisés

Storyboards

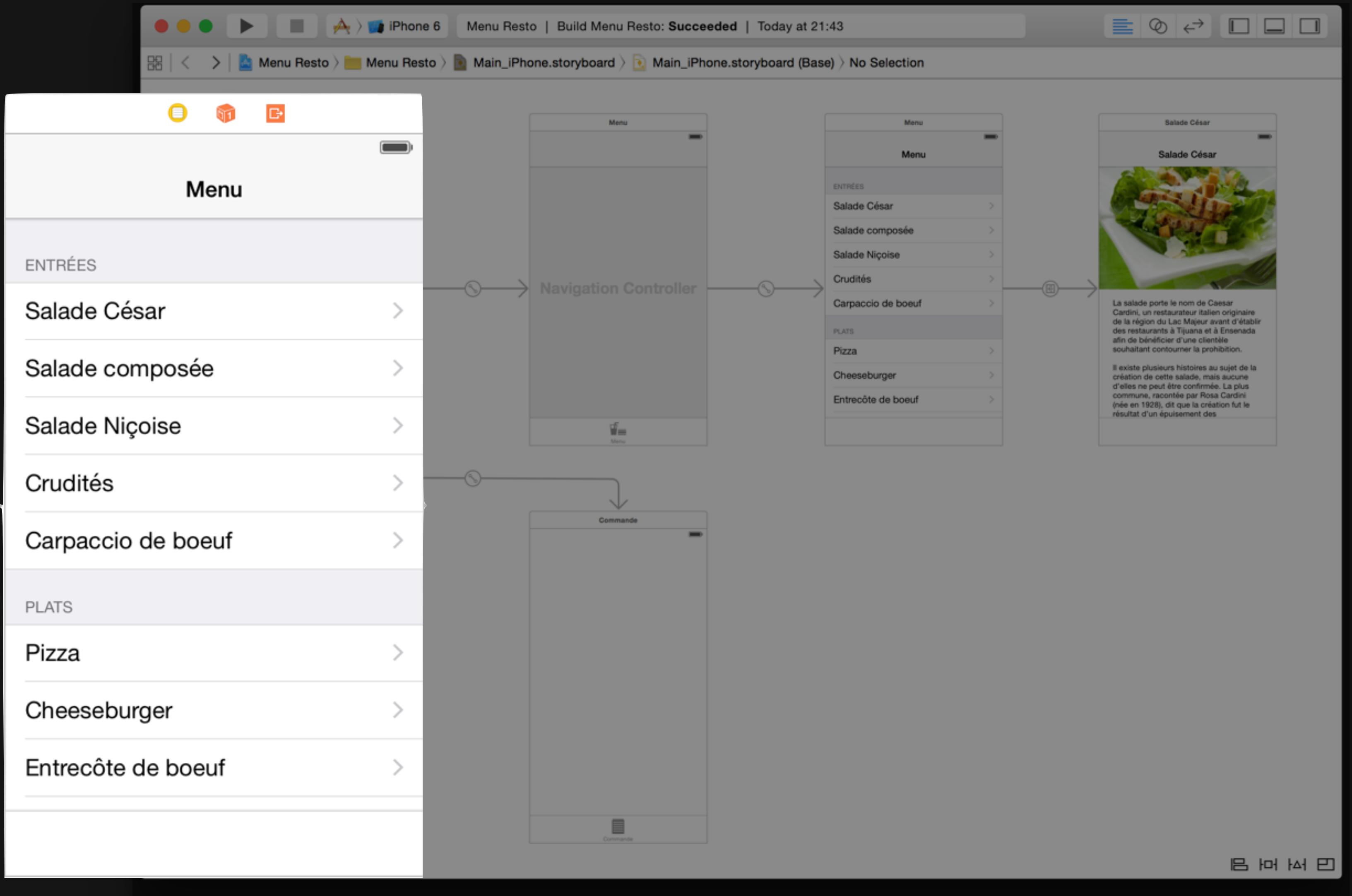
Storyboards

- Nouvelle façon de concevoir son UI
- Introduit avec iOS 5
- Fonctionne en reliant des scènes via des segues
- Chaque scène représente une instance de UIViewController

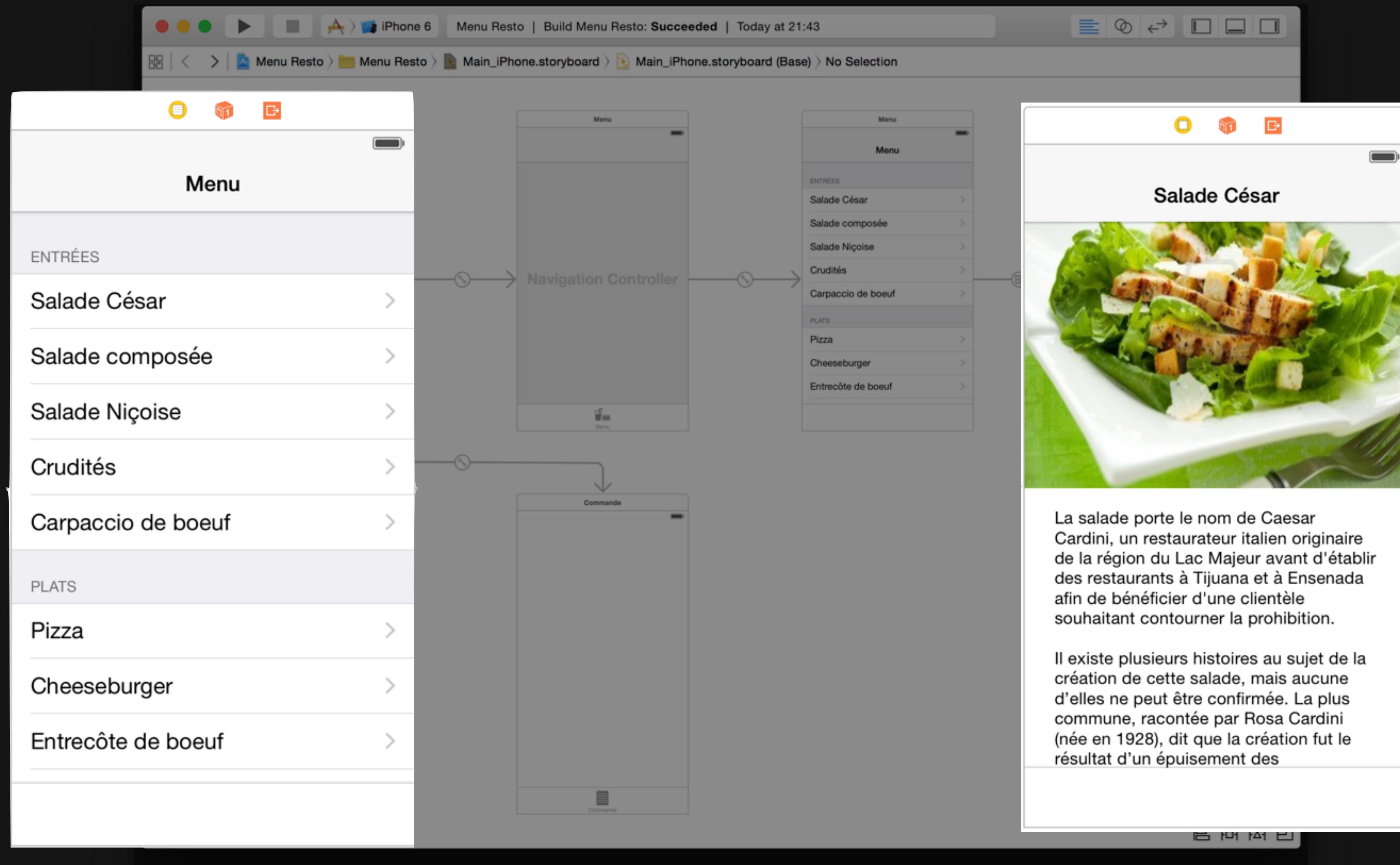
Storyboards



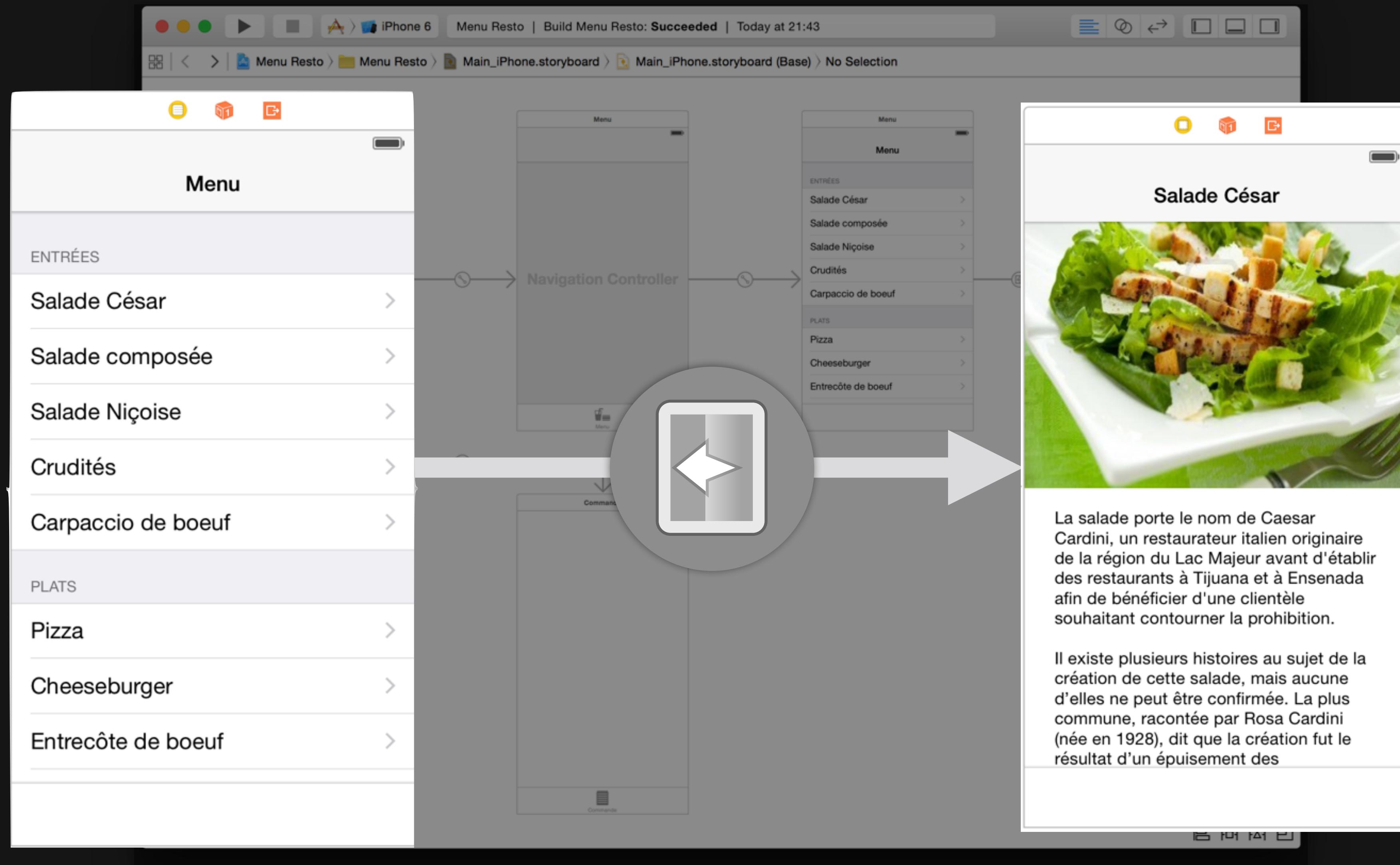
Storyboards



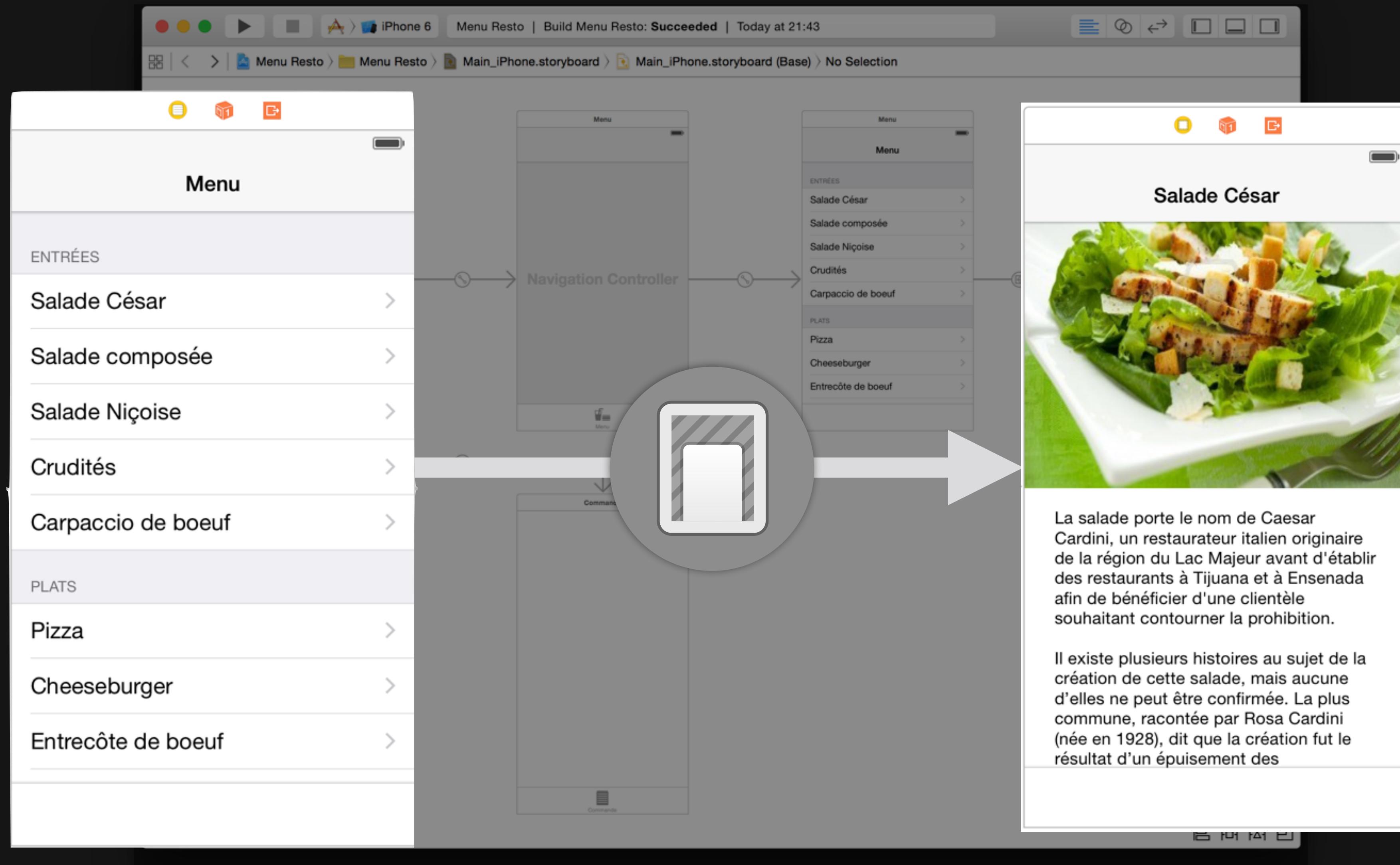
Storyboards



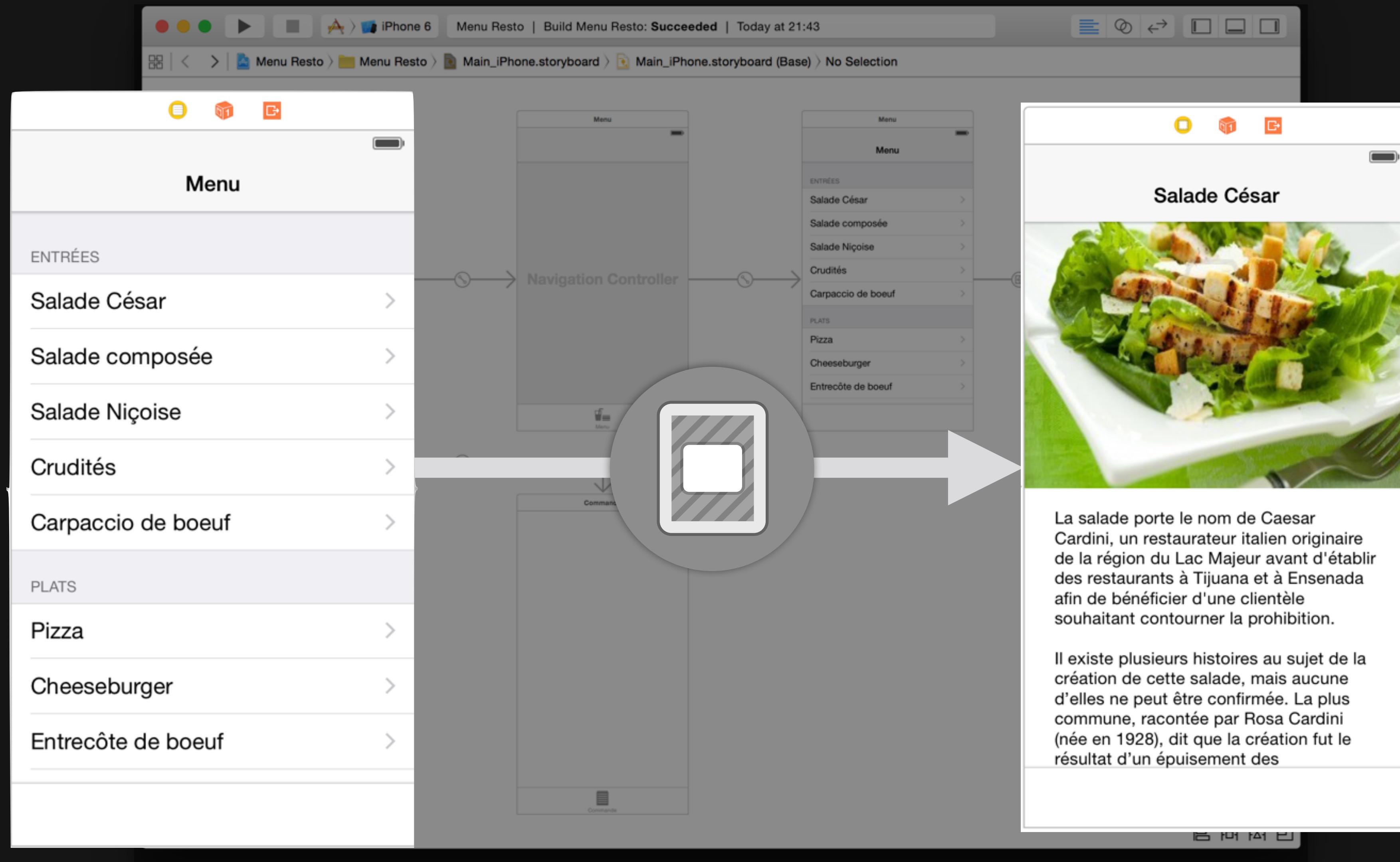
Storyboards



Storyboards



Storyboards



Storyboards



Carrier ⌘ 10:55 PM

Menu

ENTRÉES

- Salade César >
- Salade composée >
- Salade Niçoise >
- Crudités >
- Carpaccio de boeuf >

PLATS

- Pizza >
- Cheeseburger >
- Entrecôte de boeuf >

 Menu

 Commande

Carrier ⌘ 10:55 PM

< Menu Salade César



La salade porte le nom de Caesar Cardini, un restaurateur italien originaire de la région du Lac Majeur avant d'établir des restaurants à Tijuana et à Ensenada afin de bénéficier d'une clientèle souhaitant contourner la prohibition.

Il existe plusieurs histoires au sujet de la création de cette salade, mais aucune d'elles ne peut être confirmée. La plus commune, racontée par Rosa Cardini (née en 1928), dit que la création fut le résultat d'un épuisement des

 Menu

 Commande



prepareForSegue(_, sender:)

The diagram illustrates a storyboard sequence. It starts with a "Menu" screen on the left, which lists various food items under categories "ENTRÉES" and "PLATS". A green arrow points from the "Salade César" item in the "ENTRÉES" section to a second screen on the right. The second screen is titled "Salade César" and displays a photograph of the dish. Below the photo, there is descriptive text about the history of the Caesar salad.

Menu

Carrier 10:55 PM

ENTRÉES

- Salade César >
- Salade composée >
- Salade Niçoise >
- Crudités >
- Carpaccio de boeuf >

PLATS

- Pizza >
- Cheeseburger >
- Entrecôte de boeuf >

Menu Commande

Salade César

Carrier 10:55 PM

< Menu Salade César



La salade porte le nom de Caesar Cardini, un restaurateur italien originaire de la région du Lac Majeur avant d'établir des restaurants à Tijuana et à Ensenada afin de bénéficier d'une clientèle souhaitant contourner la prohibition.

Il existe plusieurs histoires au sujet de la création de cette salade, mais aucune d'elles ne peut être confirmée. La plus commune, racontée par Rosa Cardini (née en 1928), dit que la création fut le résultat d'un épisode des

Menu Commande

Storyboards



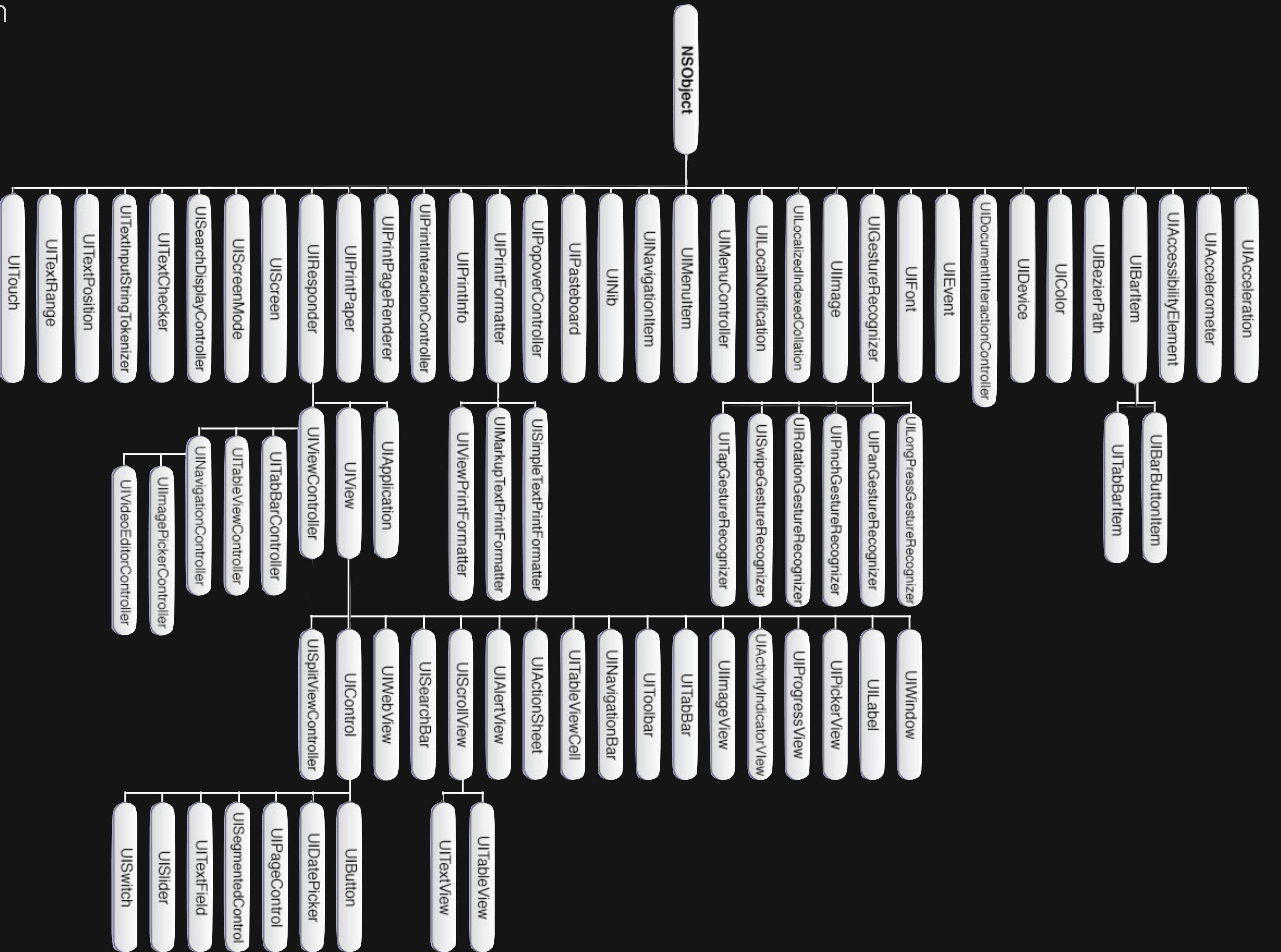
```
override func prepareForSegue(segue: UIStoryboardSegue, sender: AnyObject?) {  
    if segue.identifier == "displayDetails" {  
        var nextViewController = segue.destinationViewController  
        nextViewController.title = "The title of the next viewController"  
    }  
}
```

UIKit : organisation

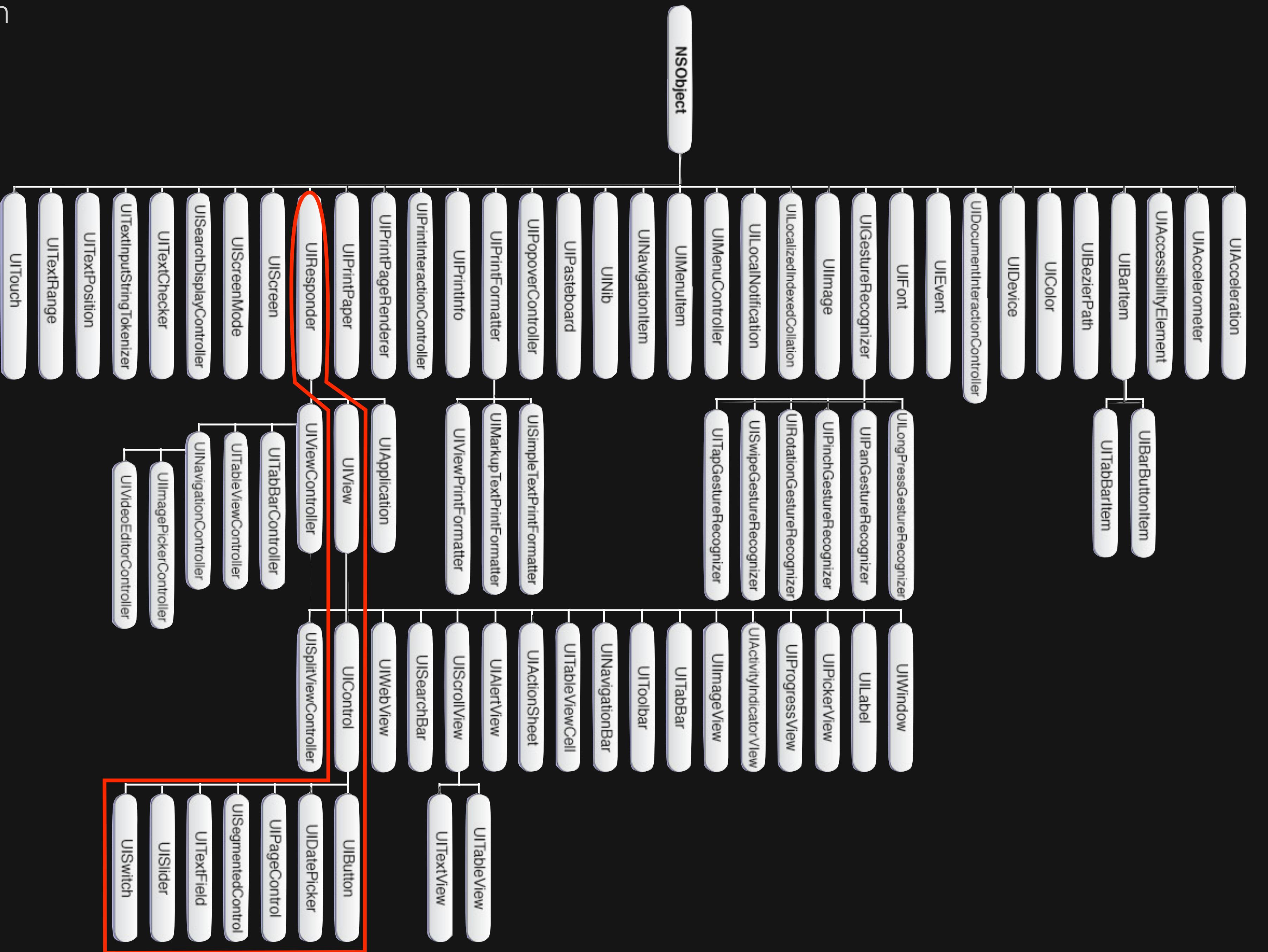
UIKit : organisation

- UIKit → Framework
- Contient toutes les classes pour vos UI
 - UIWindow, UIButton, UILabel...
- Travail simplifié pour le développeur

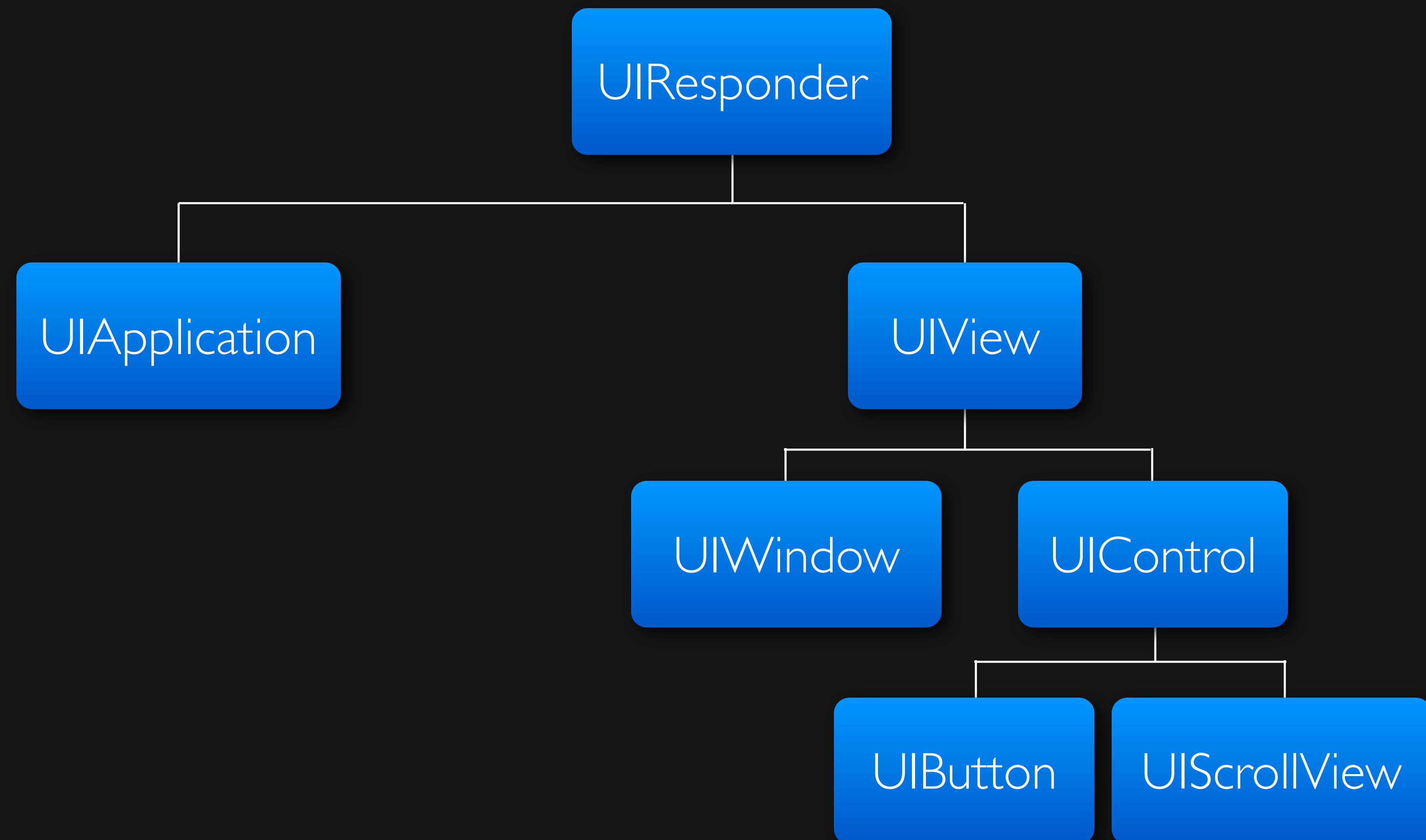
UIKit : organisation



UIKit : organisation

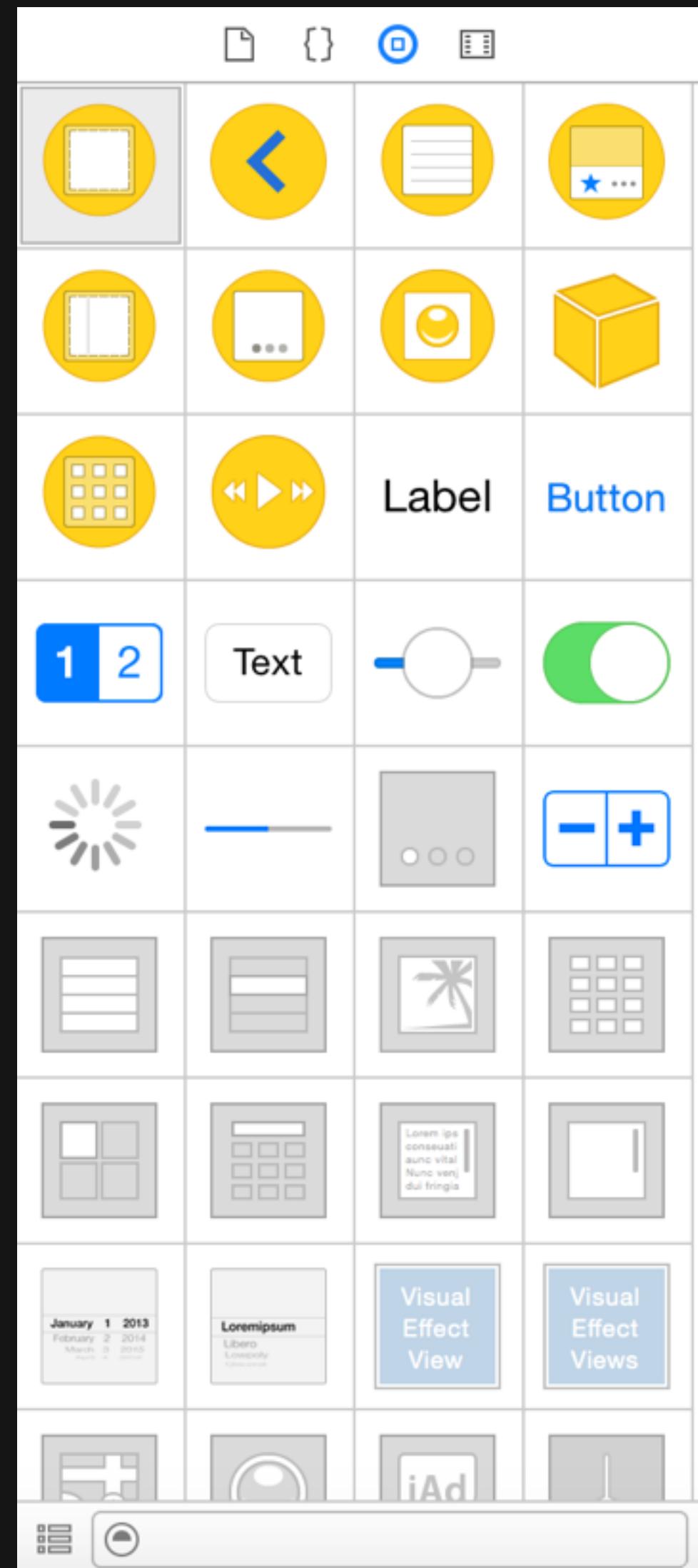


UIKit : organisation



UIKit : organisation

- Une grande variété d'objets disponibles
- Accessibles en drag'n drop, ou via le code





- UIResponder
- Gestion des touches
 - `func touchesBegan(touches: NSSet, withEvent event: UIEvent)`
 - `func touchesCancelled(touches: NSSet!, withEvent event: UIEvent!)`
 - `func touchesEnded(touches: NSSet, withEvent event: UIEvent)`
 - `func touchesMoved(touches: NSSet, withEvent event: UIEvent)`

UIWindow

- Point de départ de l'interface
- Une app iOS ne possède, en général qu'une fenêtre



UIView

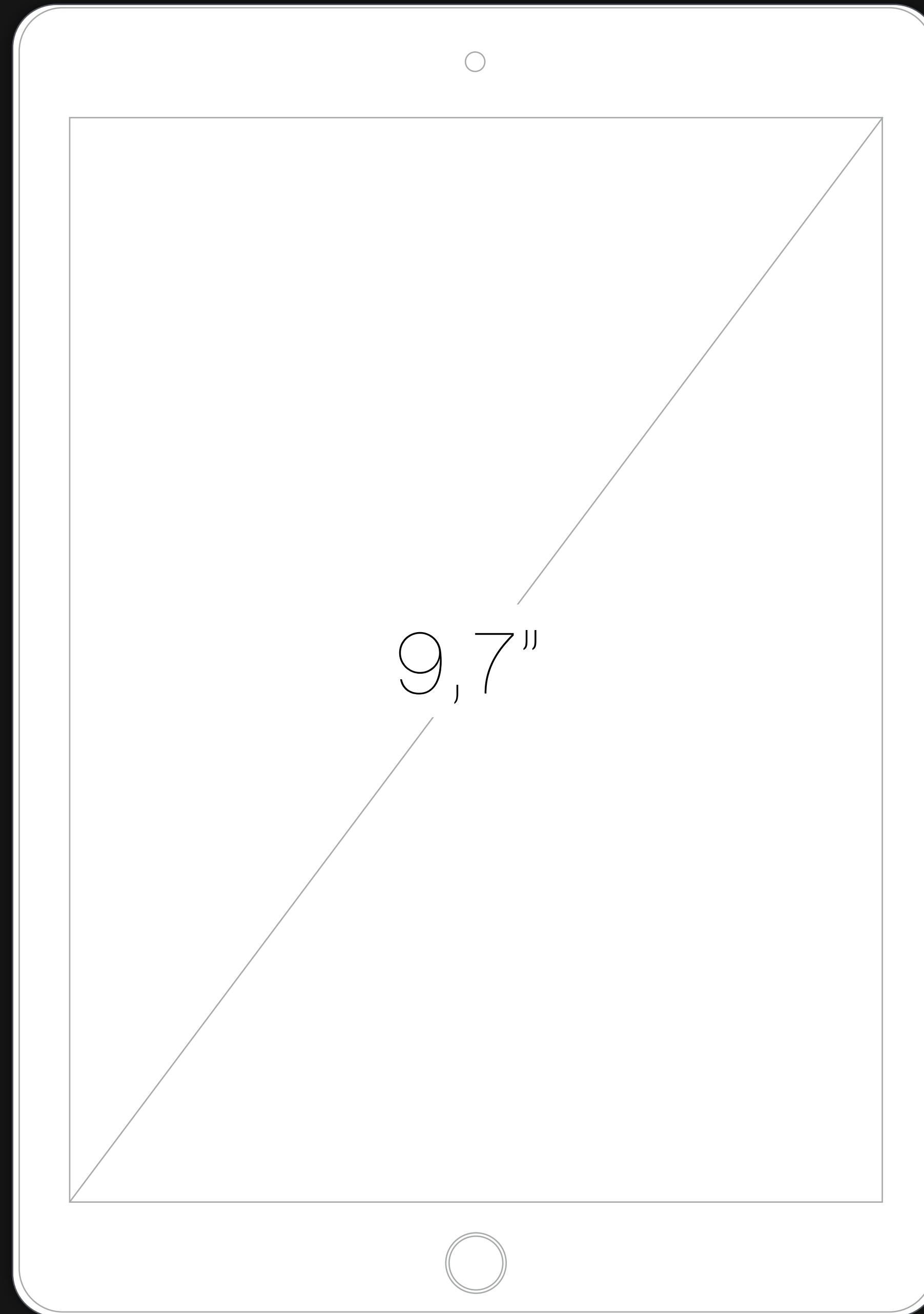
- Zone rectangulaire permettant d'afficher du contenu
- Par défaut : un rectangle rempli avec une couleur de fond
- Peut être sous-classé pour être personnalisé
- Une vue peut contenir des sous-vues
 - `var subviews: [UIView] { get }`
- Une vue peut posséder une vue parente
 - `var superview: UIView? { get }`

UIControl

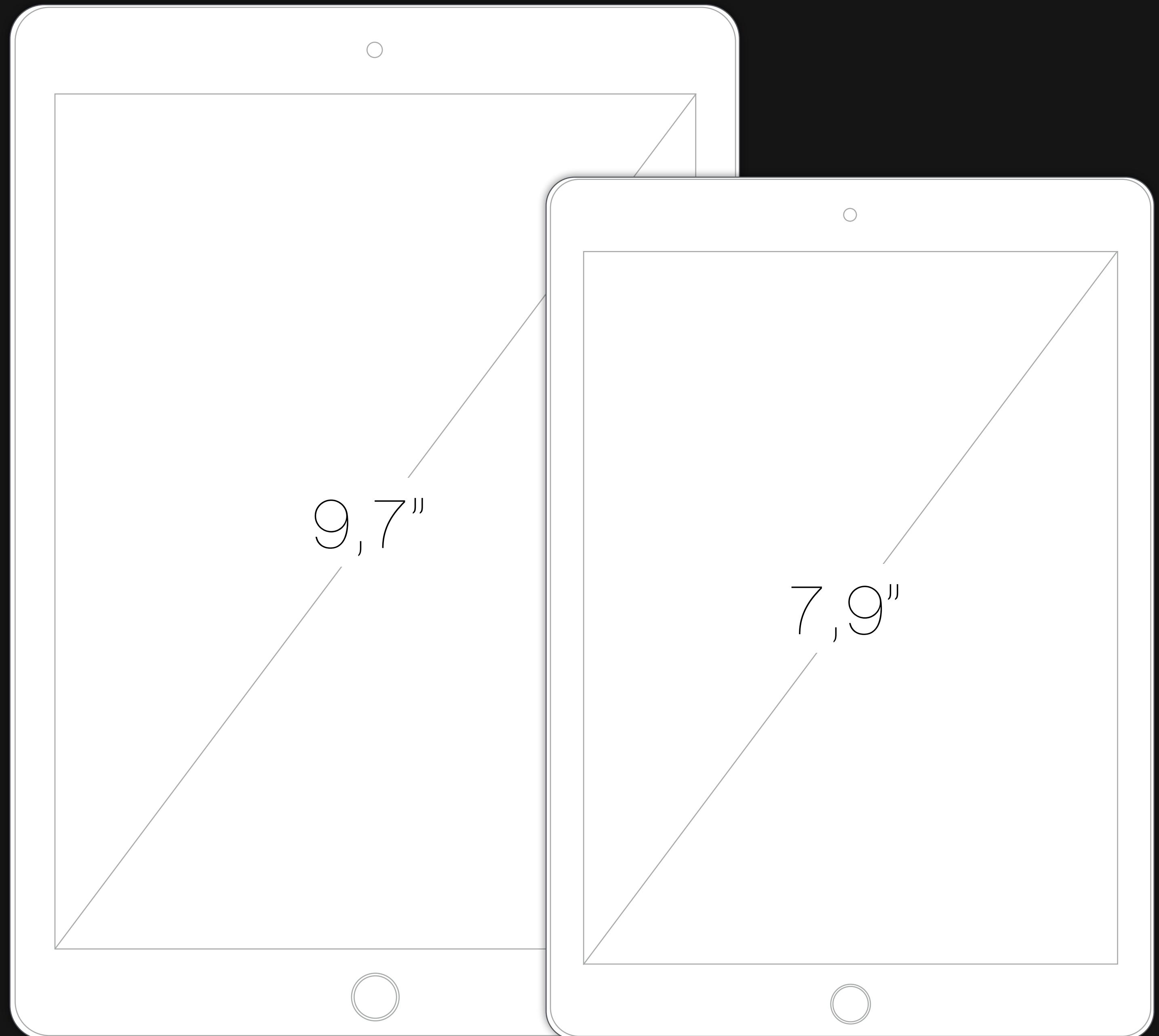
- Sous-classe de UIView
- Permet de réagir simplement aux taps en envoyant des actions (IBAction)

Adaptabilité de l'interface

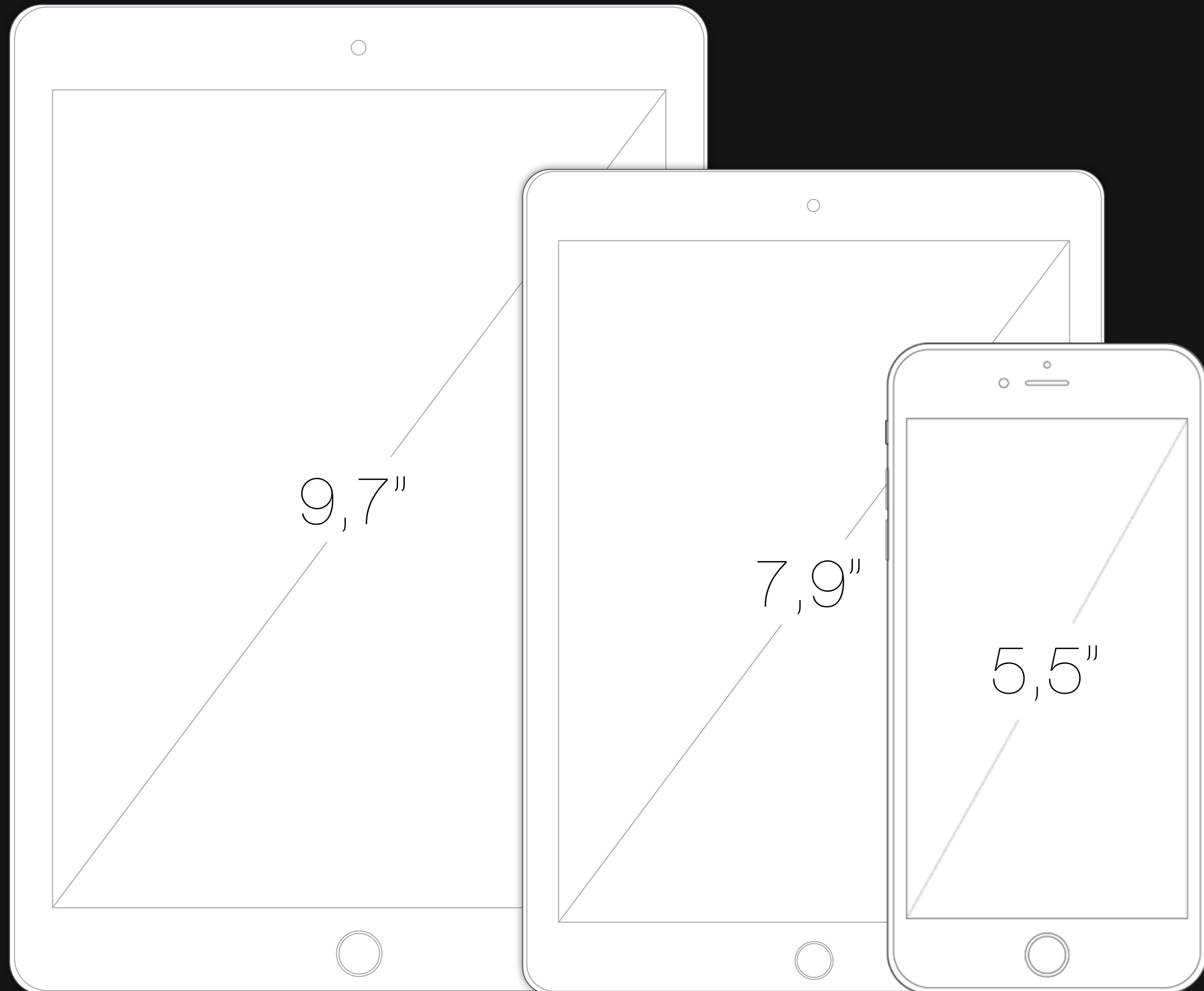
Adaptabilité de l'interface



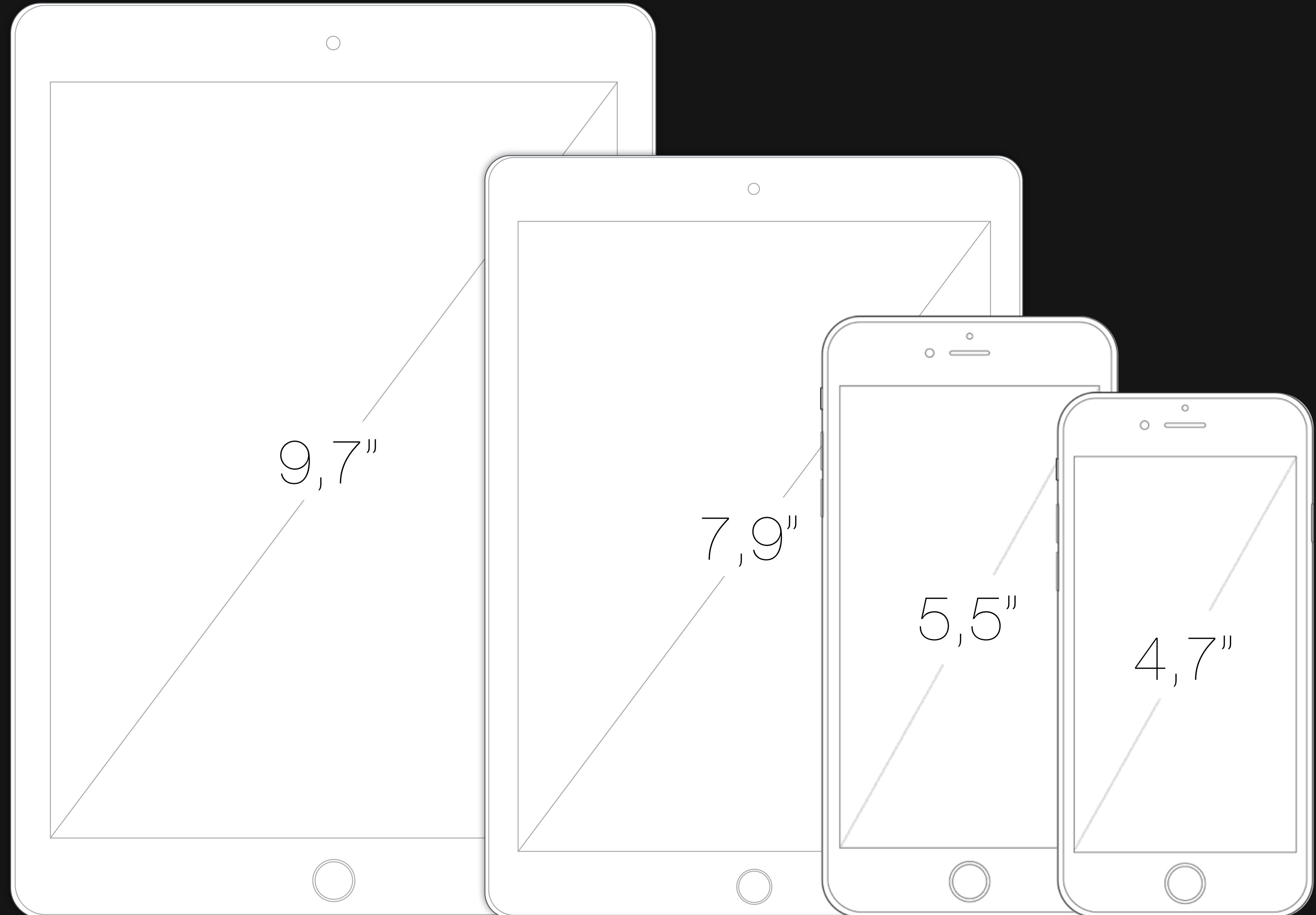
Adaptabilité de l'interface



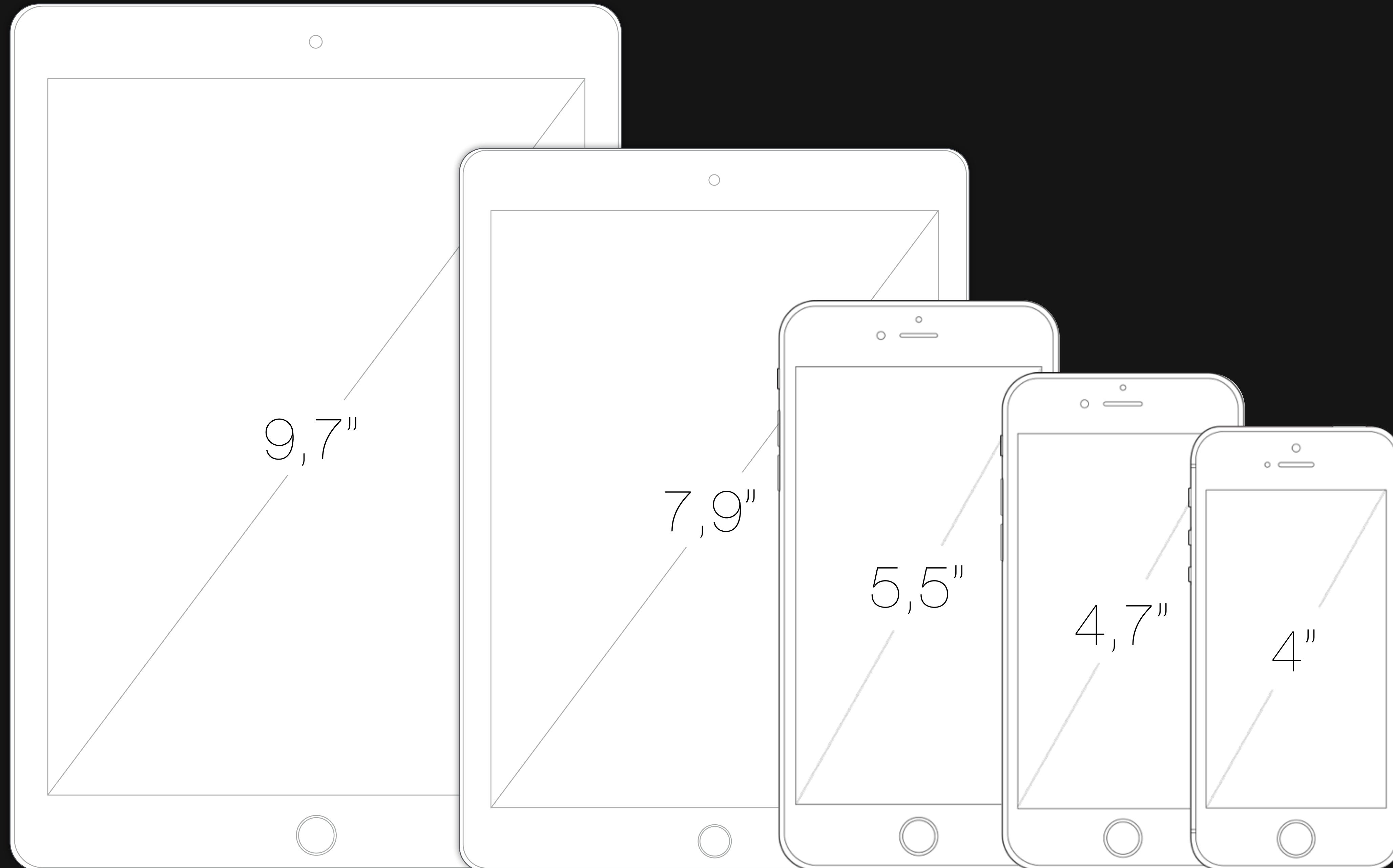
Adaptabilité de l'interface



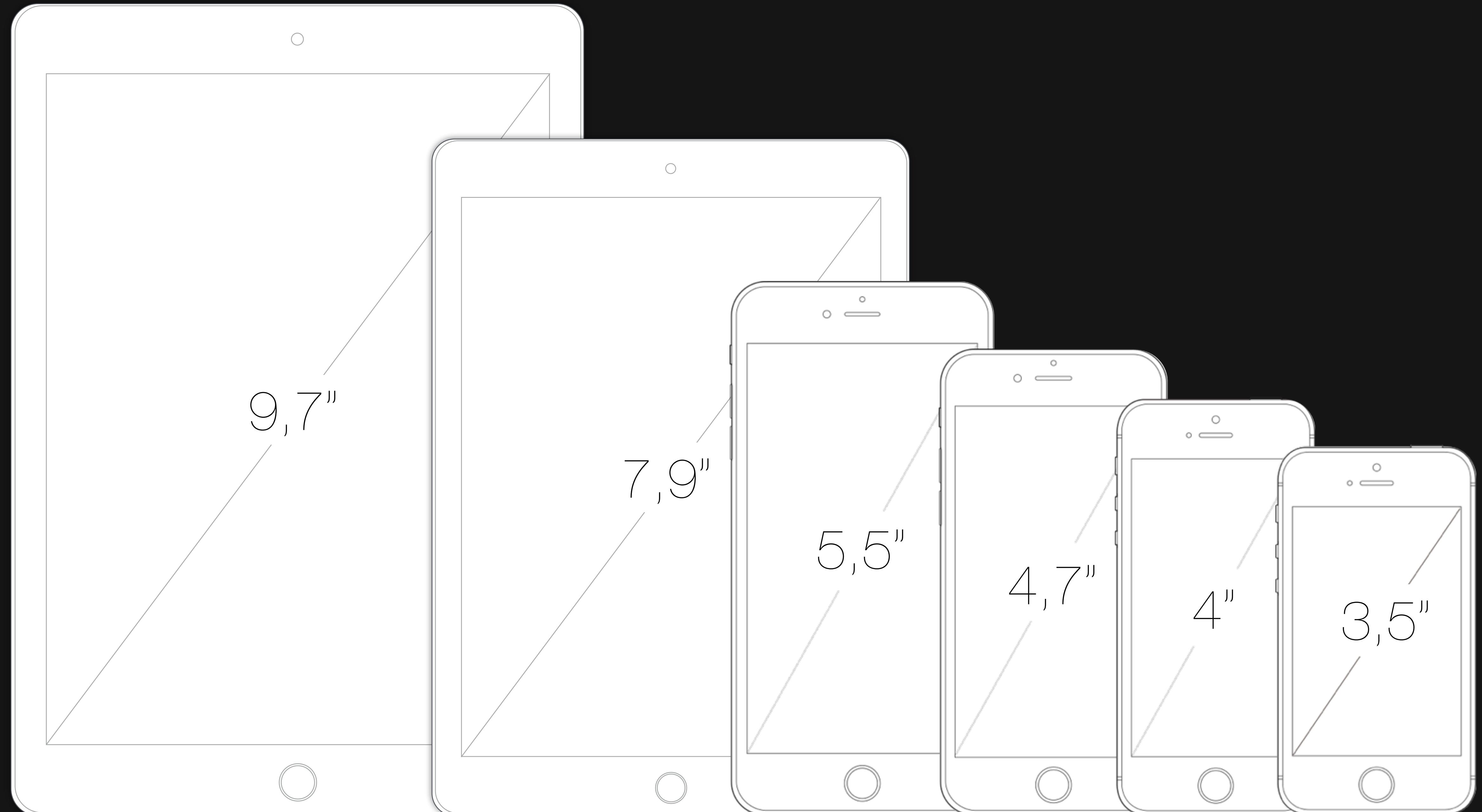
Adaptabilité de l'interface



Adaptabilité de l'interface



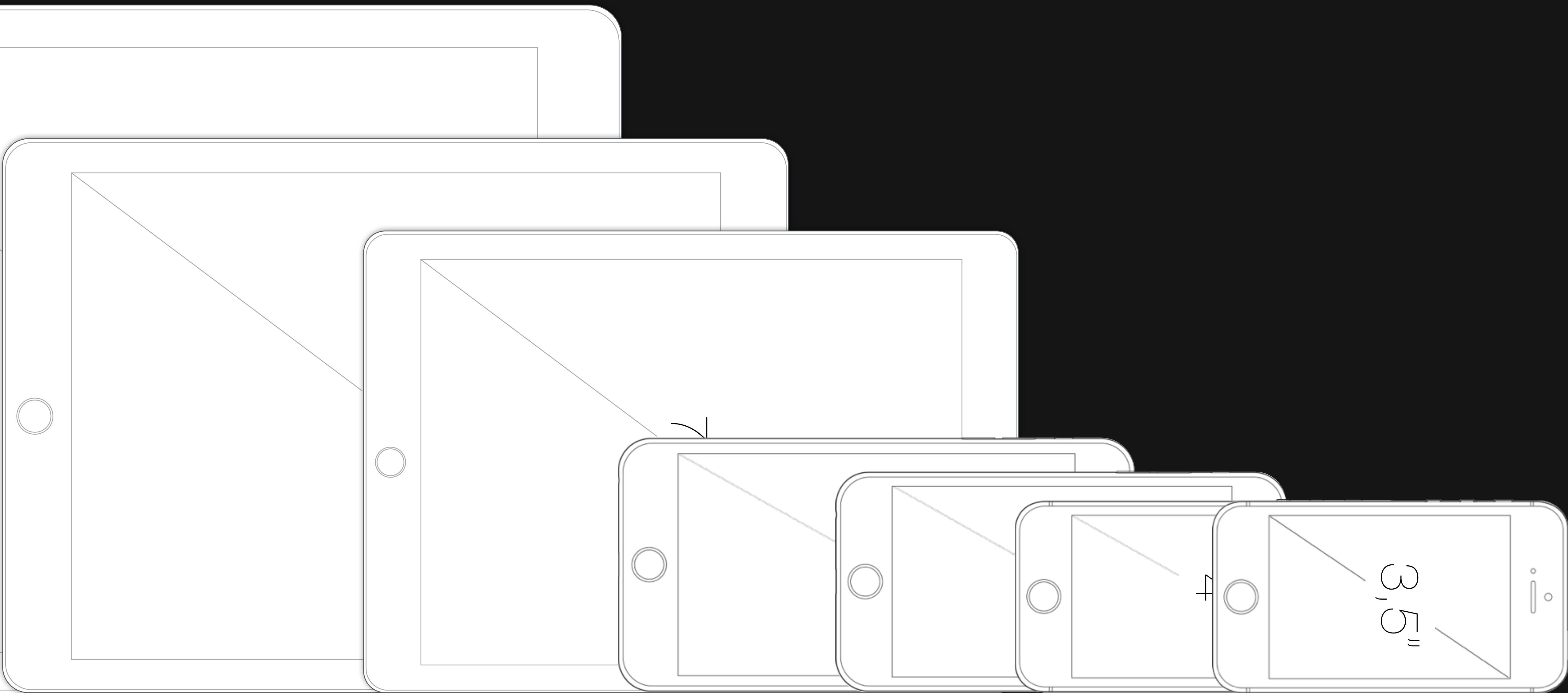
Adaptabilité de l'interface



Adaptabilité de l'interface



Adaptabilité de l'interface



Adaptabilité de l'interface

- iOS fonctionne sur une grande variété d'appareils
- Chaque appareil peut être utilisé en portrait ou paysage
- Cela donnerait énormément d'écrans à concevoir
- Auto Layout, les classes de tailles et les contrôleurs adaptatifs nous aident à réduire ce travail

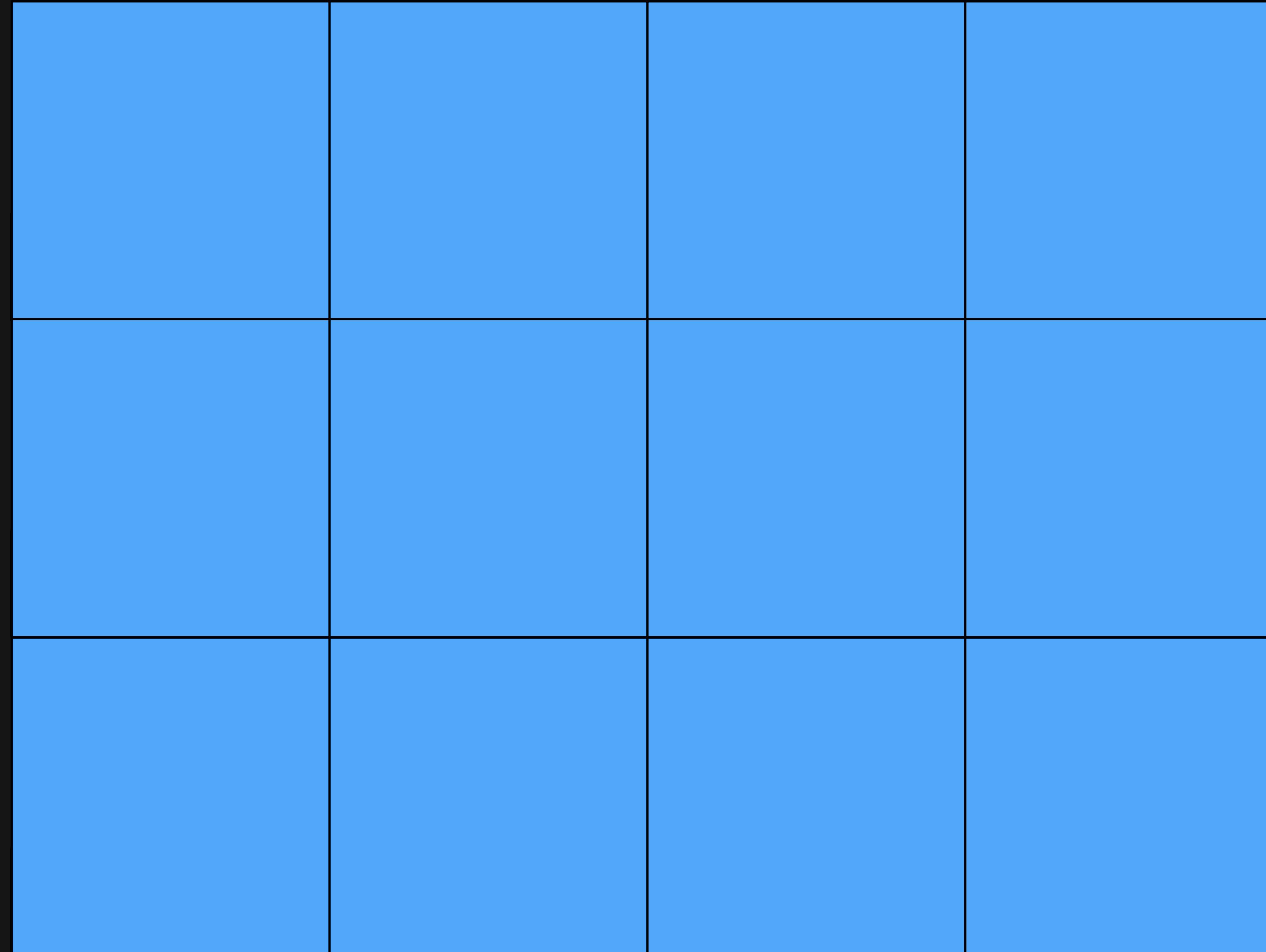
Coordonnées

- CGFloat : type défini à partir de float ou de double utilisé dans Core Graphics
- CGPoint : structure représentant un point
 - (CGFloat x, CGFloat y)
- CGSize : structure représentant une taille
 - (CGFloat width, CGFloat height)
- CGRect : structure représentant un rectangle
 - (CGPoint origin, CGSize size)

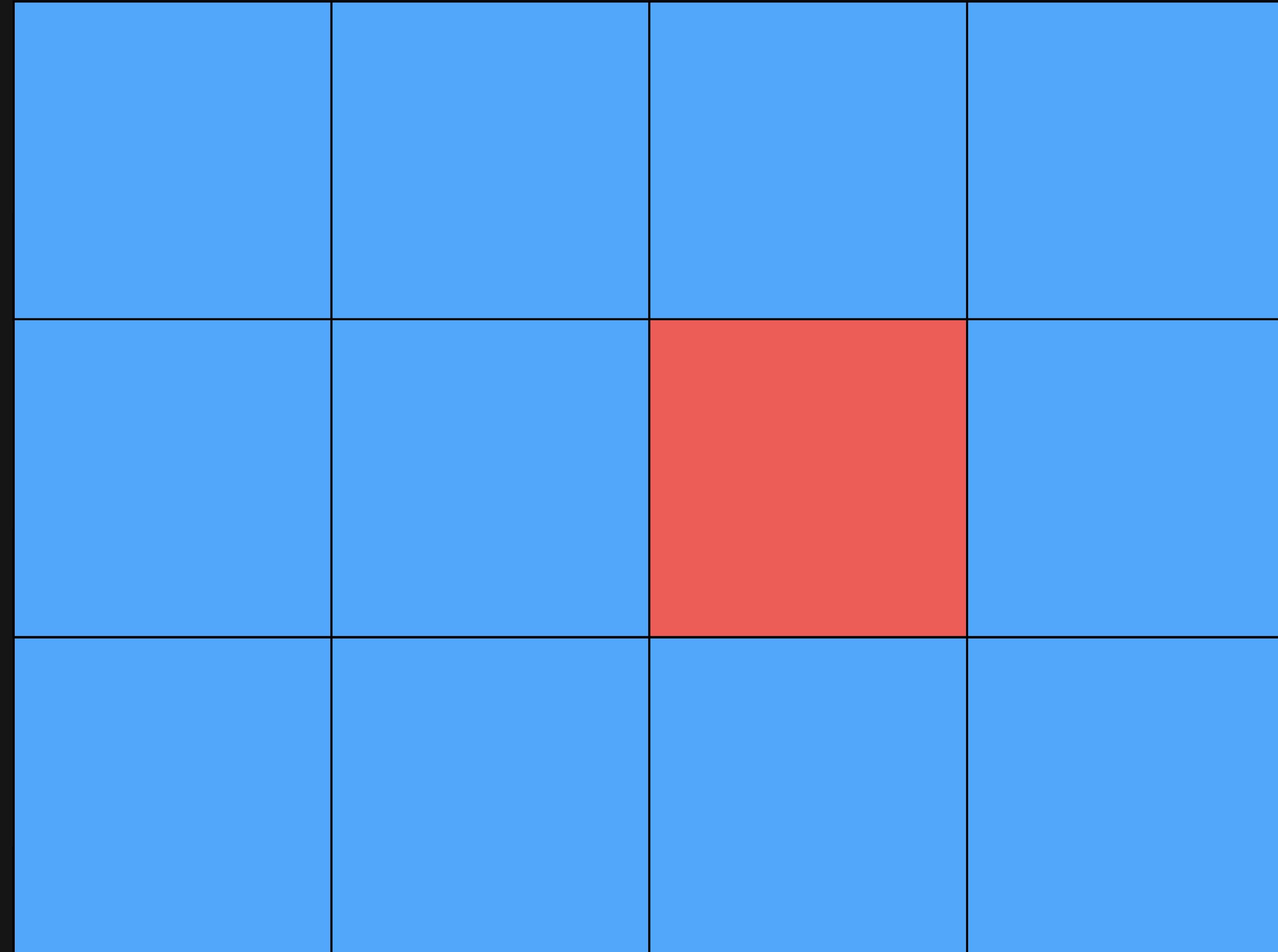
Coordonnées

- Origine du système de coordonnées en haut à gauche
- Coordonnées en *points*. Les *points* sont liés aux *pixels* en fonction de la densité de l'écran.

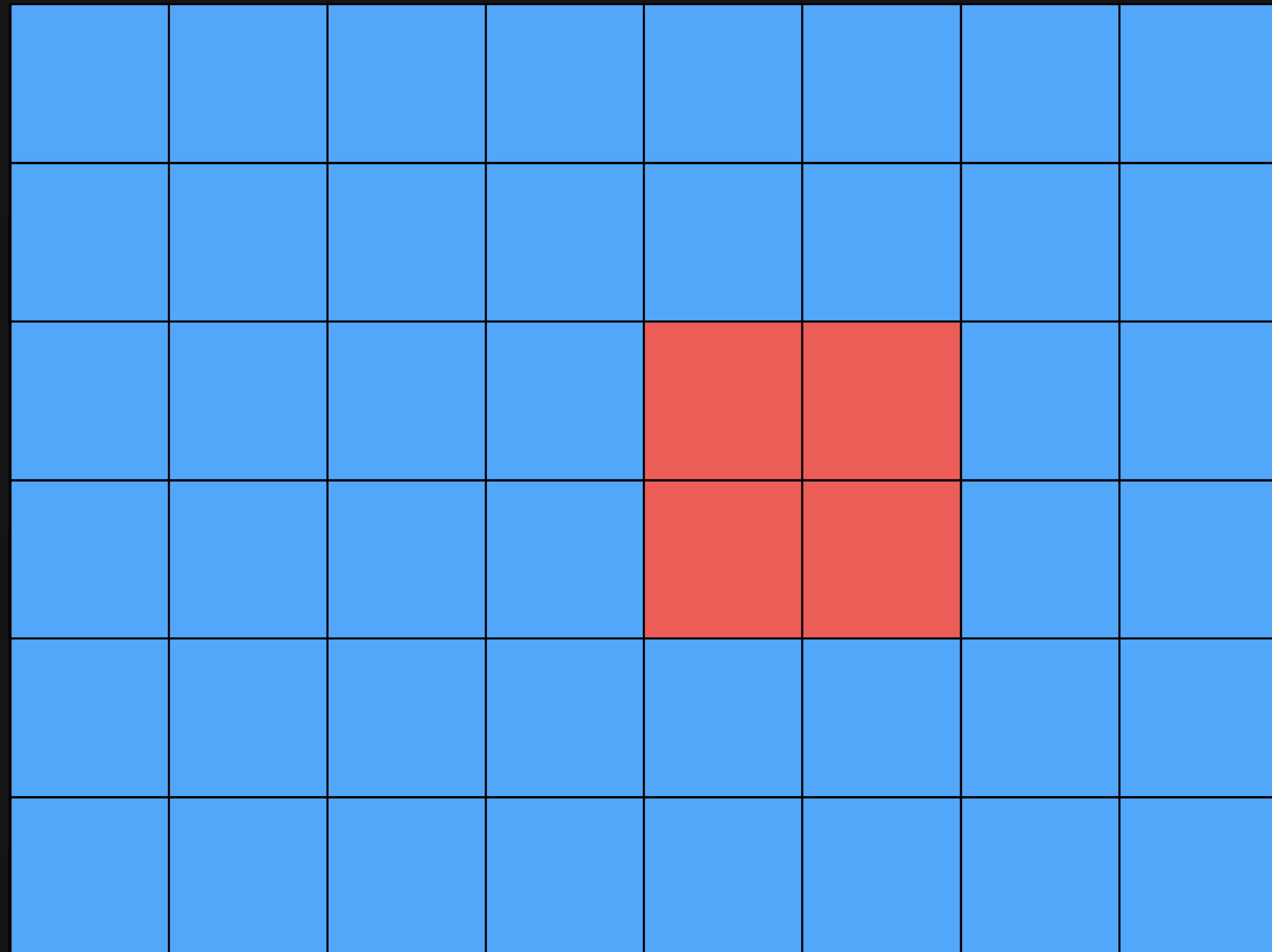
Adaptabilité de l'interface



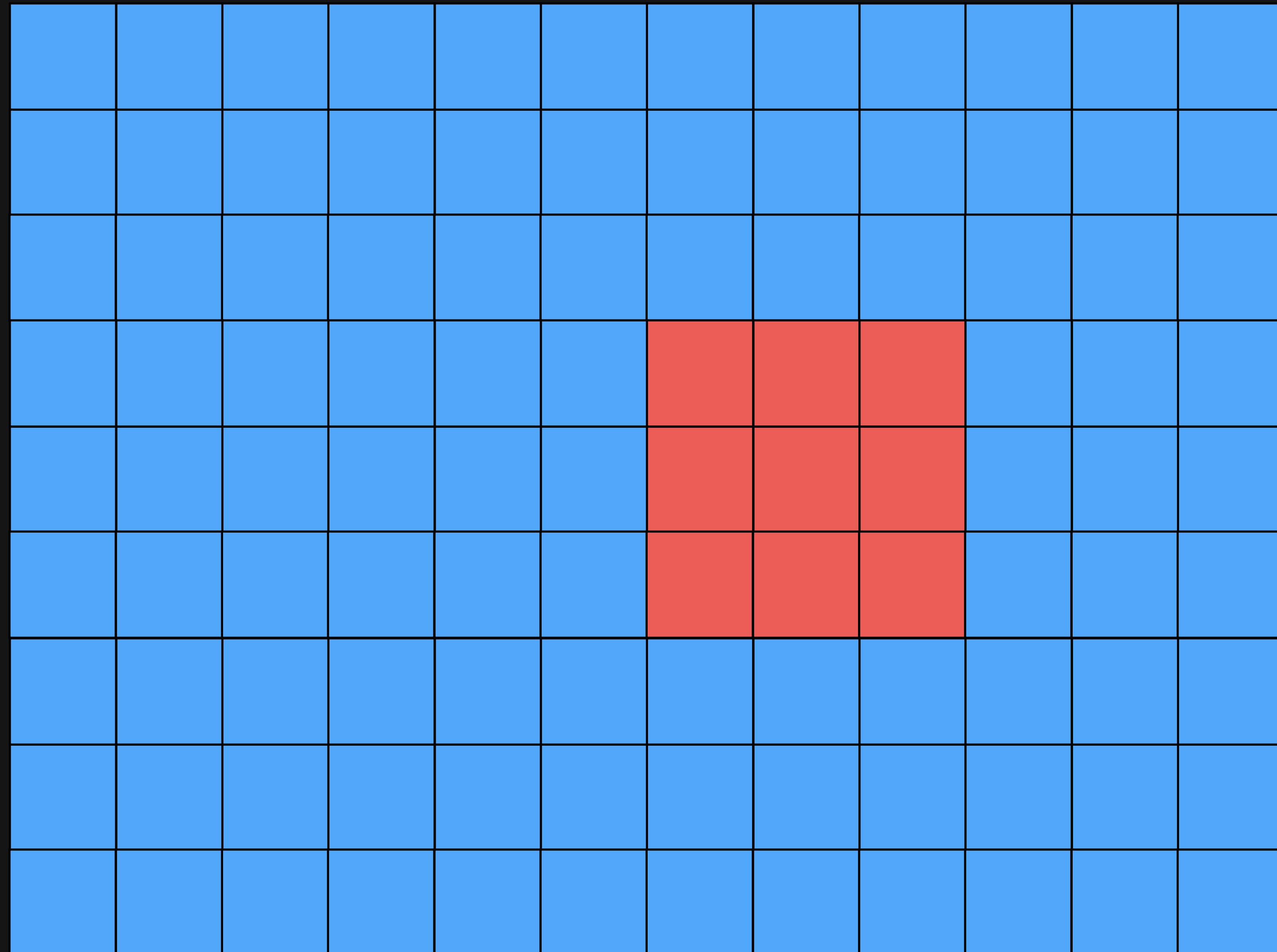
Adaptabilité de l'interface



Adaptabilité de l'interface



Adaptabilité de l'interface

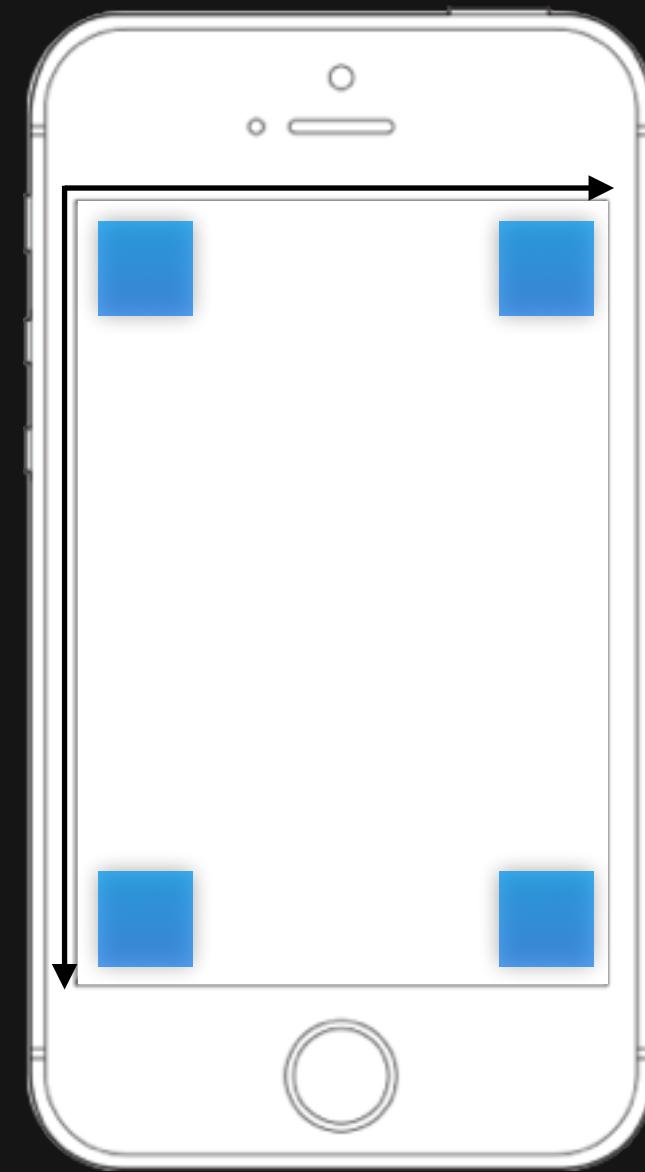


Auto Layout

- Par défaut, nos éléments sont positionnés de manière fixe dans le système de coordonnées

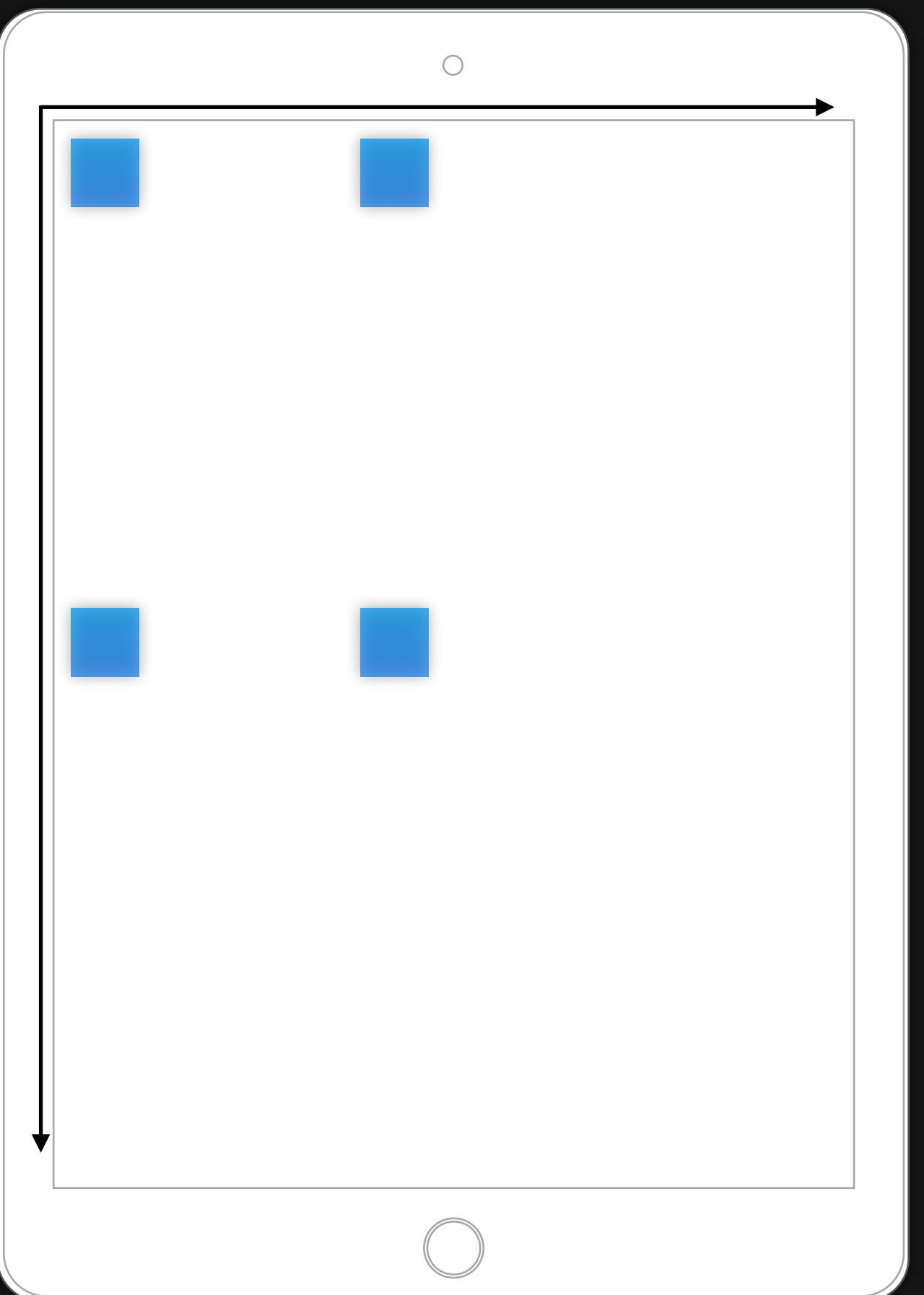
Auto Layout

- Par défaut, nos éléments sont positionnés de manière fixe dans le système de coordonnées



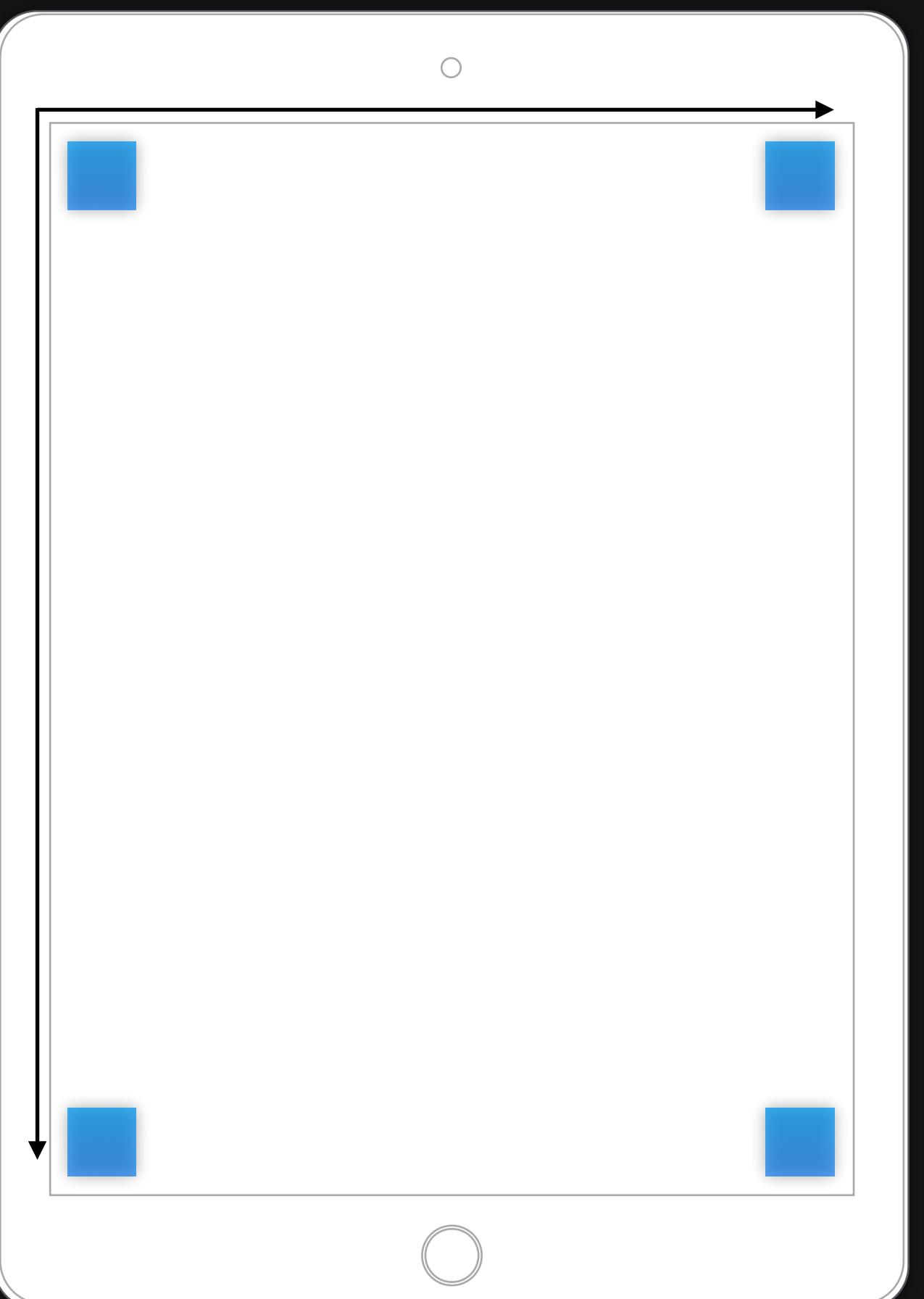
Auto Layout

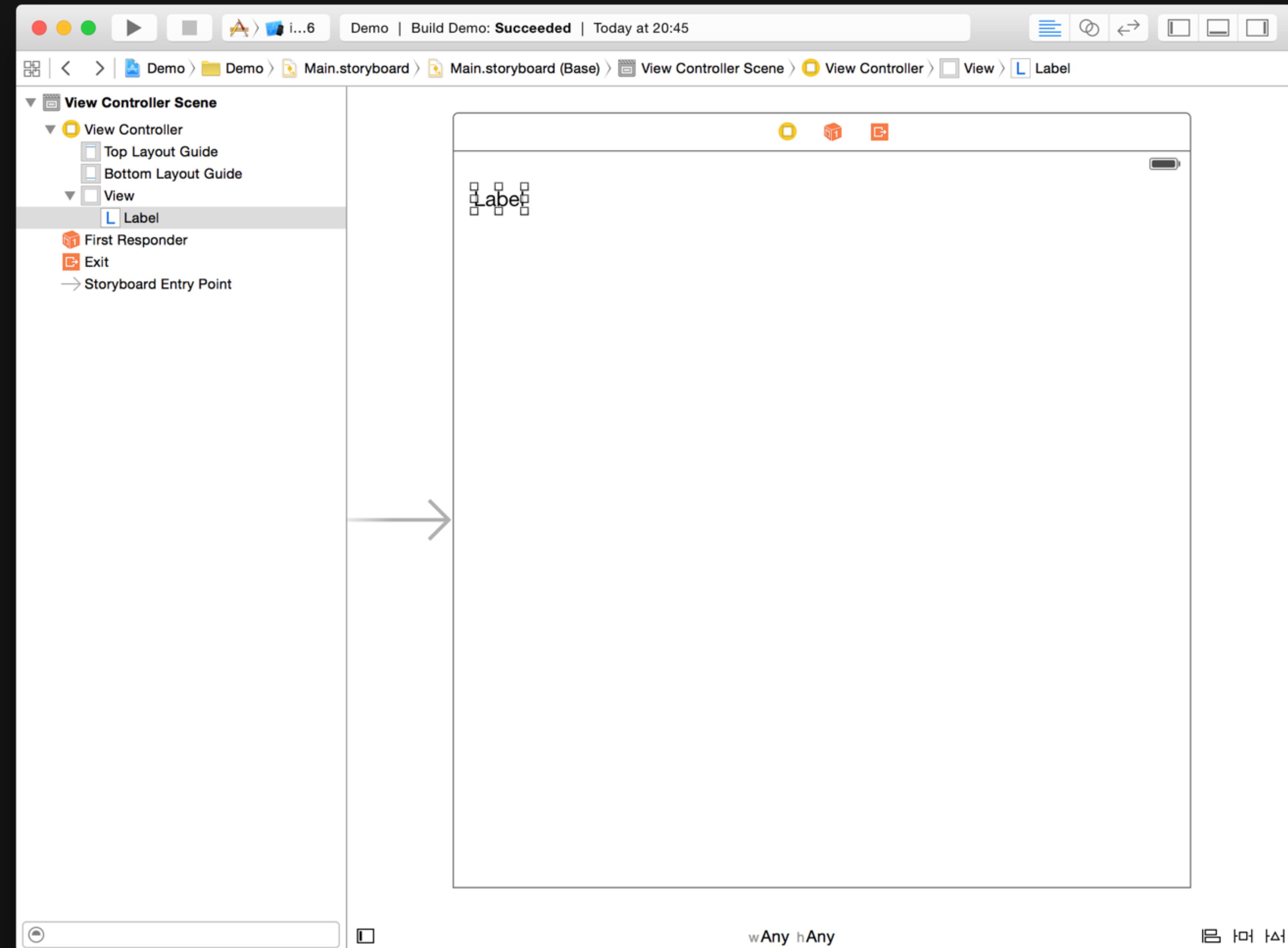
- Par défaut, nos éléments sont positionnés de manière fixe dans le système de coordonnées

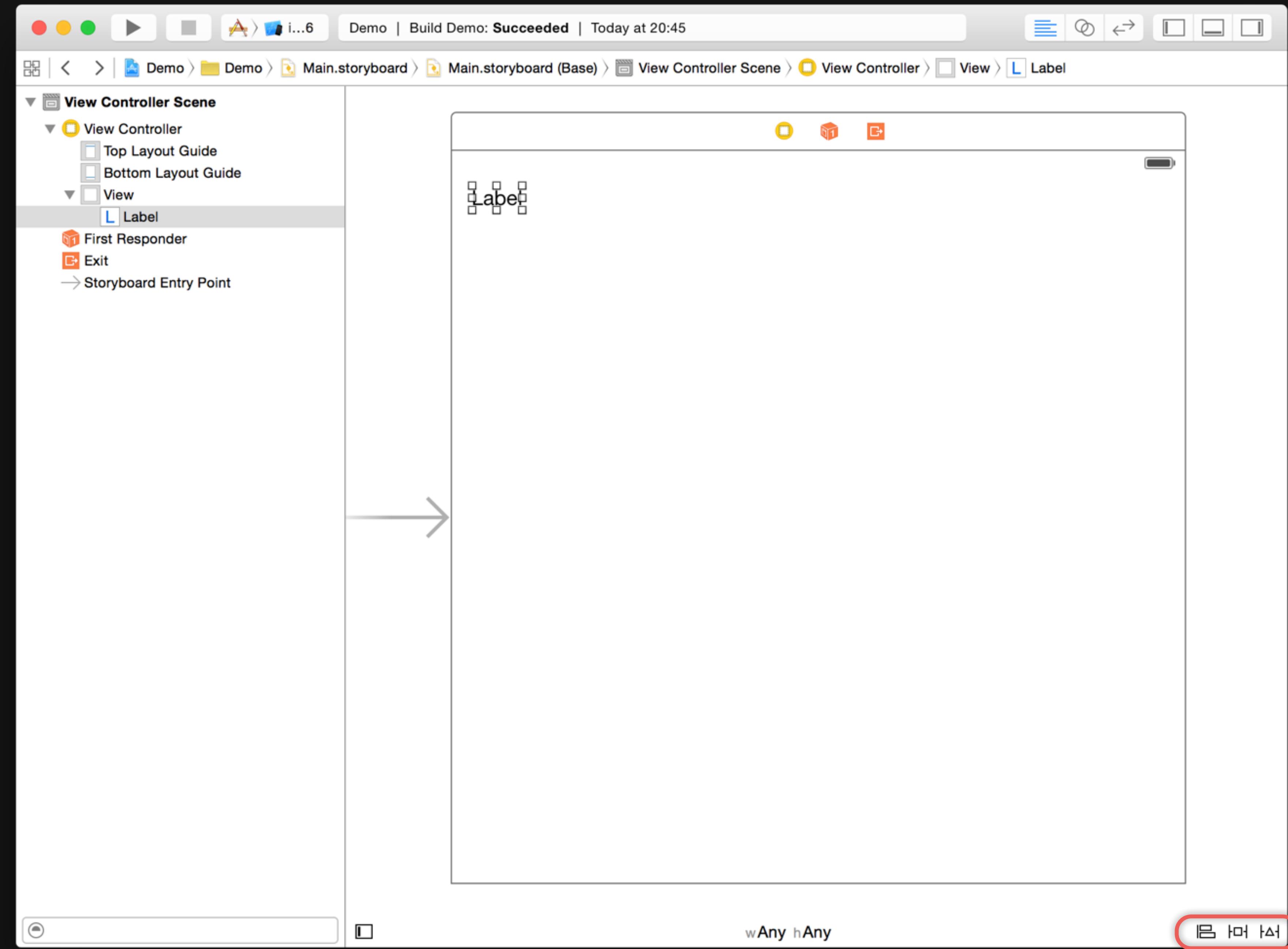


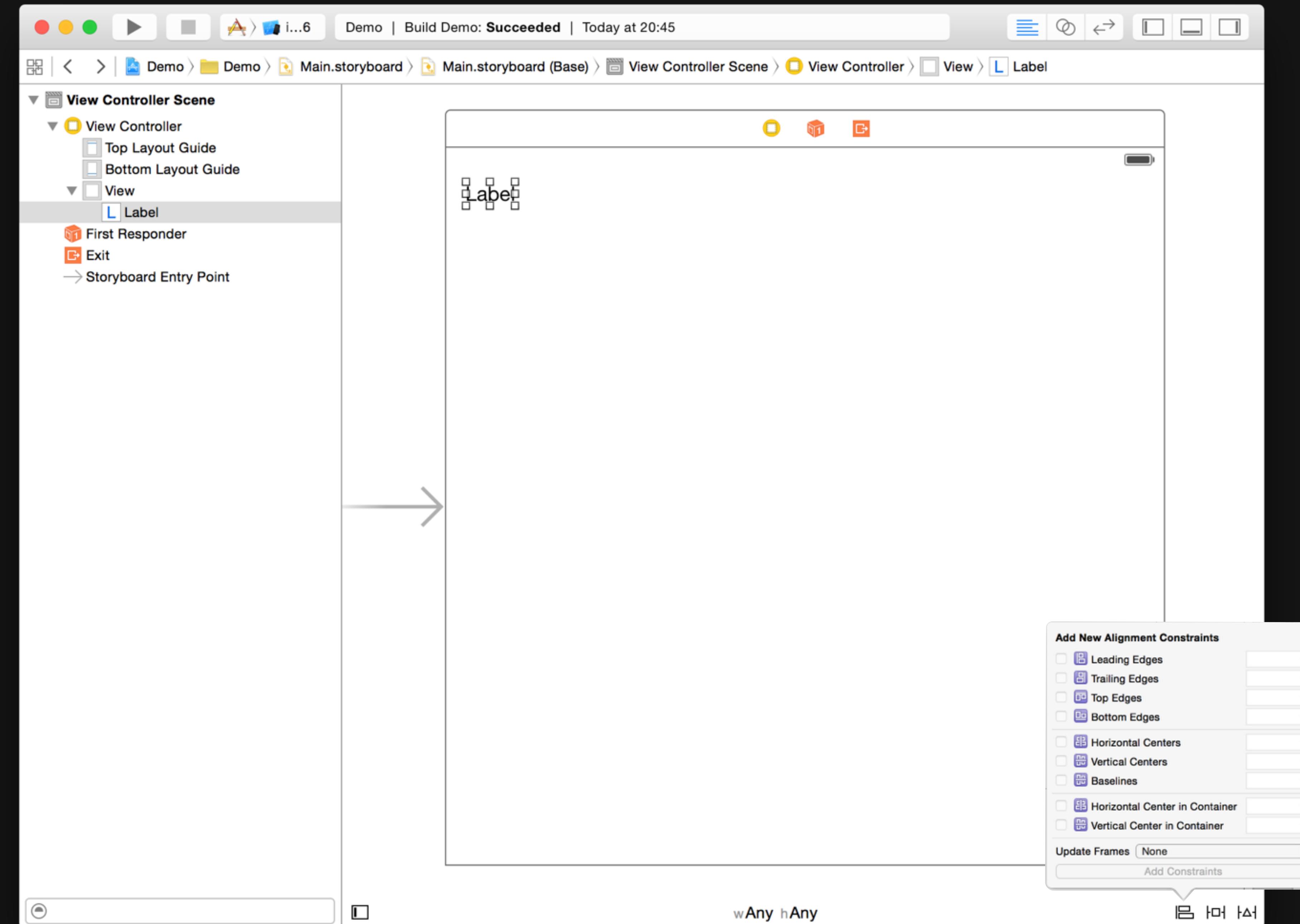
Auto Layout

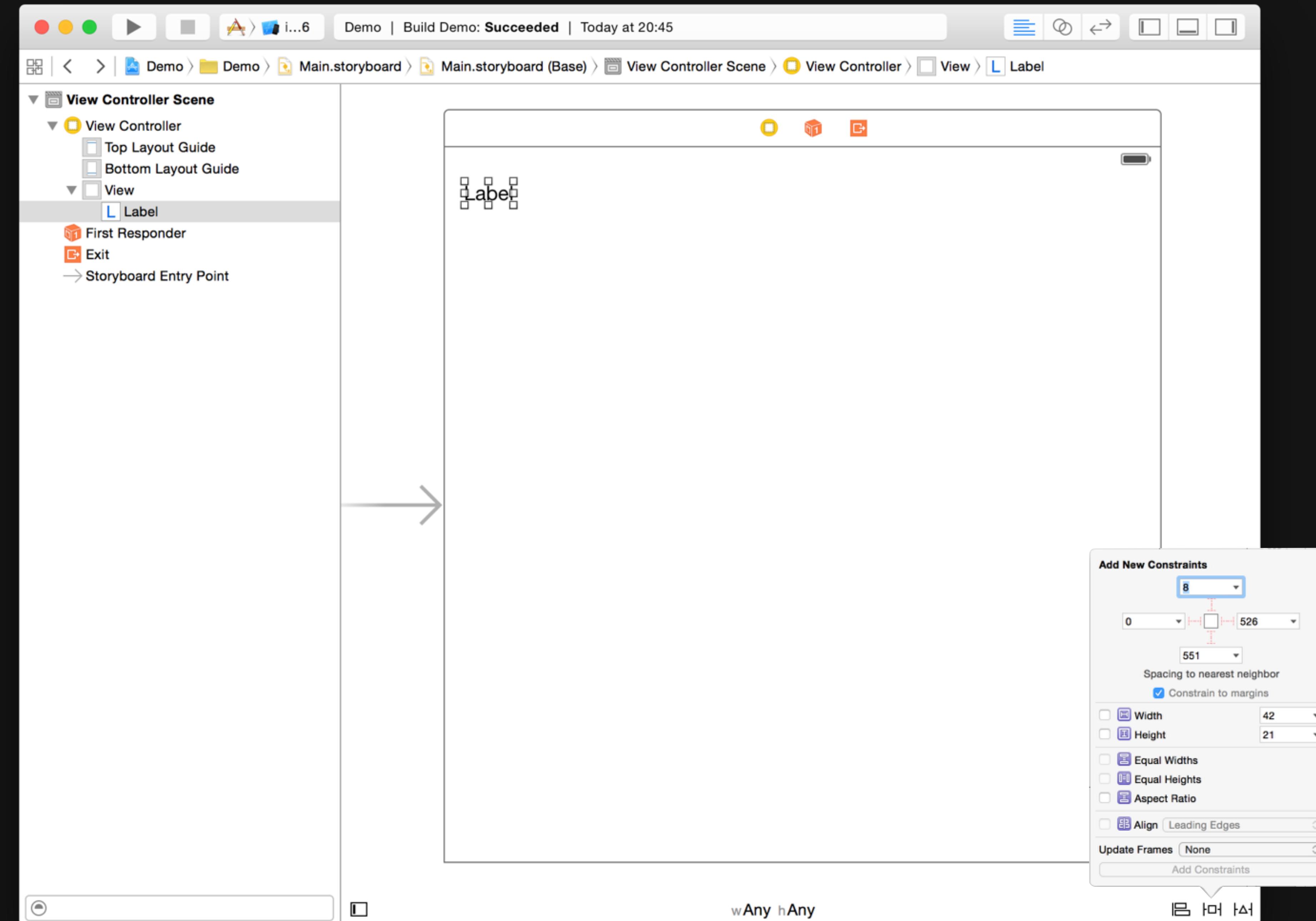
- Par défaut, nos éléments sont positionnés de manière fixe dans le système de coordonnées
- Auto Layout nous permet d'exprimer les positions grâce à des contraintes qui seront évaluées dynamiquement
- Auto Layout définit des relations mathématiques entre nos objets graphiques.
- On doit définir suffisamment de contraintes pour placer les objets sans ambiguïtés

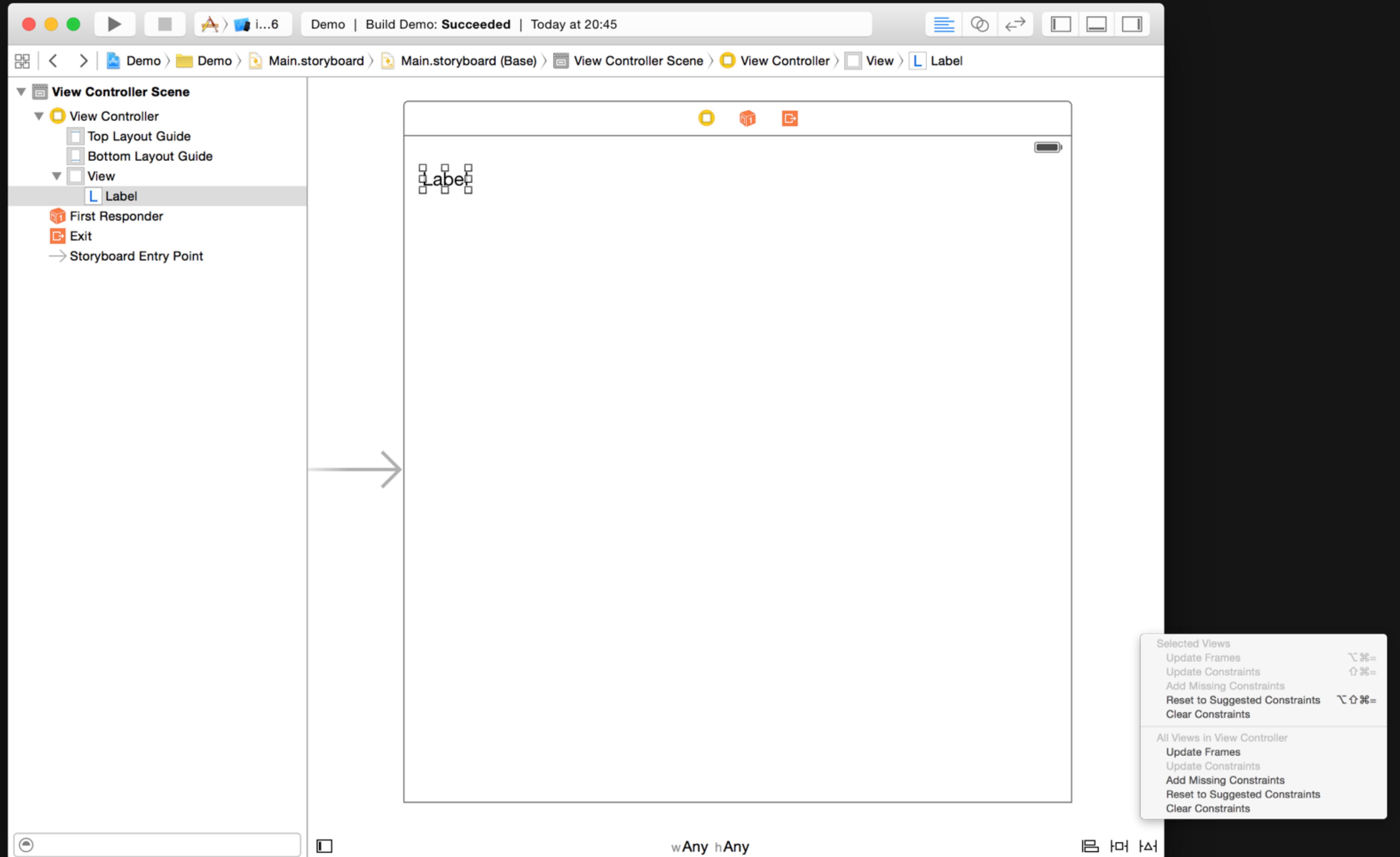












Auto Layout

- Possibilité de définir les contraintes dans le code
- Contraintes définies sous forme d'une équation linéaire
 - élément1.attribut (=, <=, >=) multiplicateur x élément.attribut + constante
- NSLayoutConstraint



Auto Layout

```
convenience init(item view1: AnyObject,  
attribute attr1: NSLayoutAttribute,  
relatedBy relation: NSLayoutRelation,  
toItem view2: AnyObject?,  
attribute attr2: NSLayoutAttribute,  
multiplier multiplier: CGFloat,  
constant c: CGFloat)
```

Size Classes

- Classification de nos interfaces en fonction de la taille.
- Chaque dimension peut avoir être soit :
 - Compact
 - Regular

Size Classes

- Depuis le storyboard on peut définir des contraintes différentes pour chaque combinaison de classes
- On peut aussi choisir de retirer ou d'ajouter des composantes graphiques selon les combinaisons de classes

Size Classes

Width : Regular

Height : Regular



iPad (Portrait / Paysage)

Size Classes

Width : Regular

Height : Compact



iPhone 5,5" paysage

Size Classes

Width : Compact

Height : Compact



iPhone 3,5" -> 4,7" (Paysage)

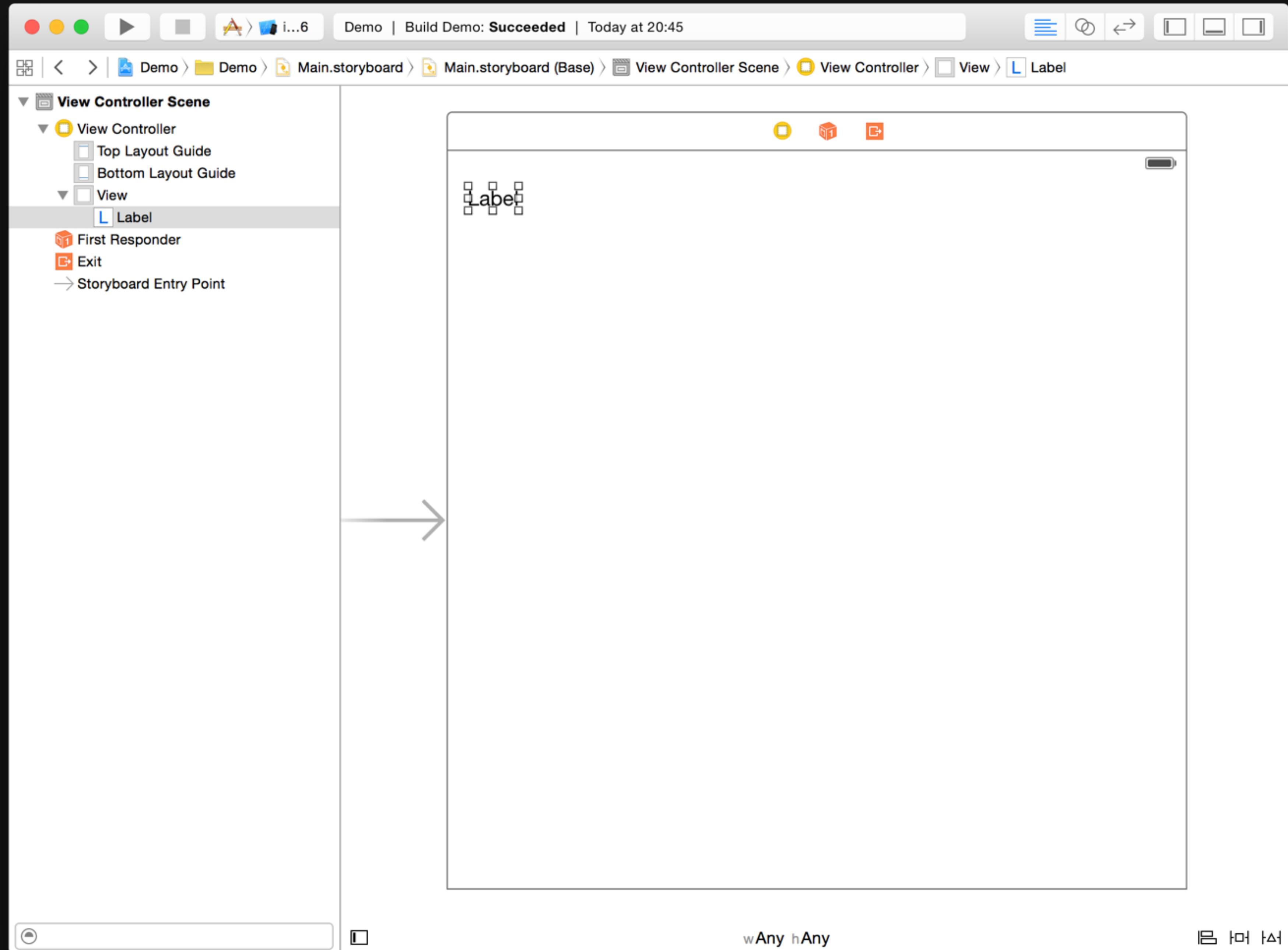
Size Classes

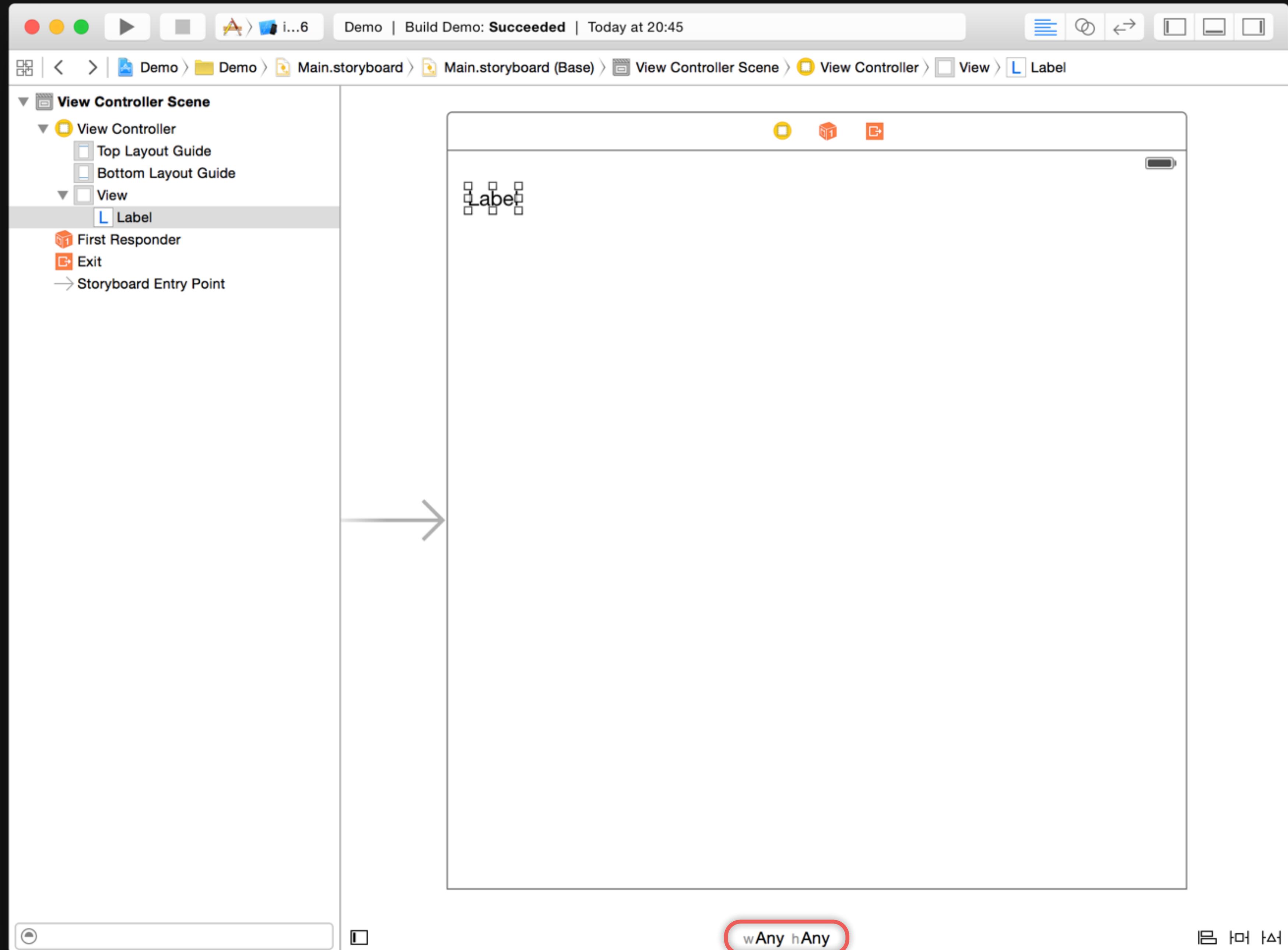
Width : Compact

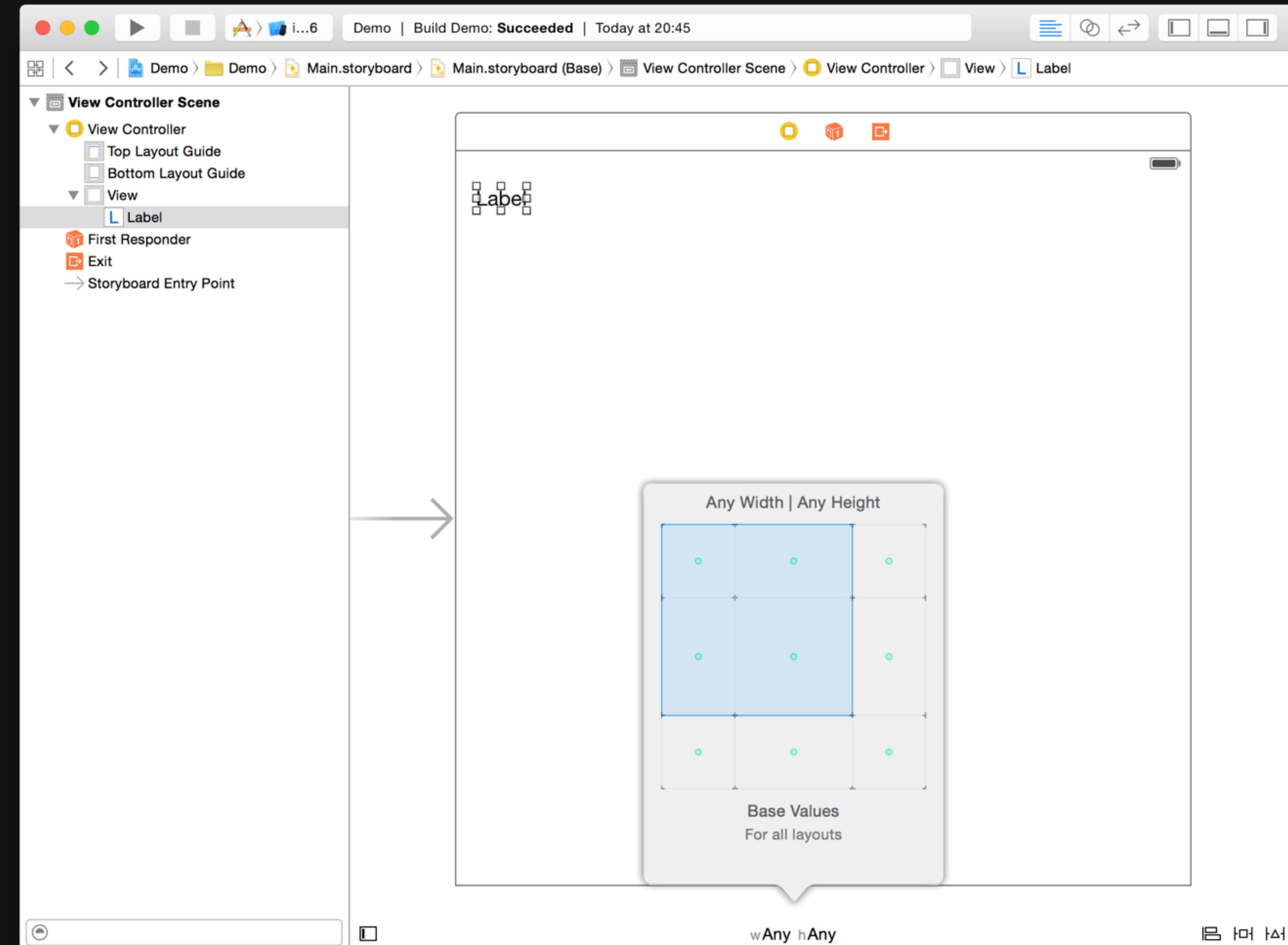
Height : Regular

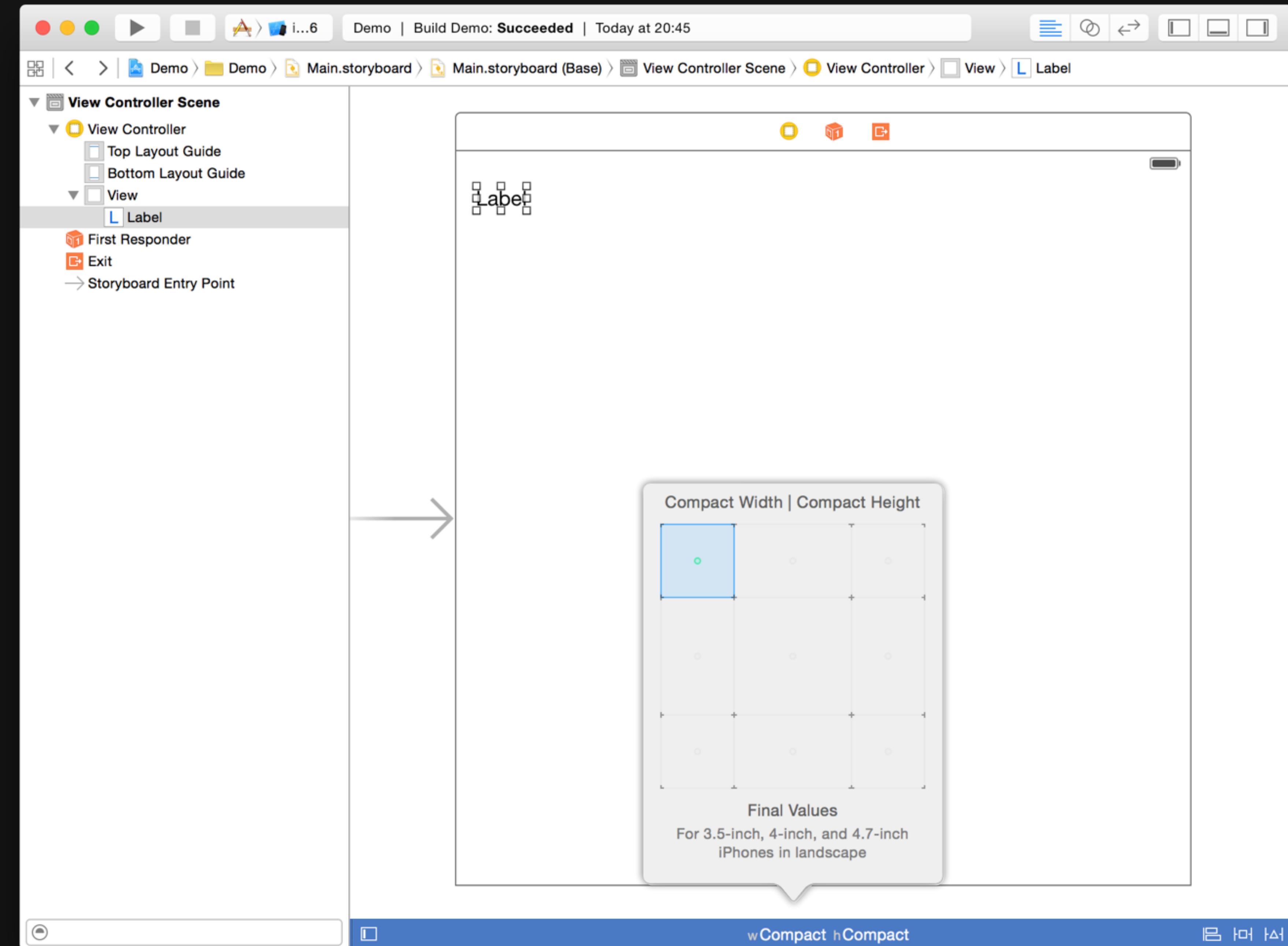


iPhone (Portrait)











Trait collection

- Les UIView, UIViewController, UIPresentationController se conforment à UITraitEnvironment
- Ils exposent une propriété traitCollection

```
var userInterfaceIdiom: UIUserInterfaceIdiom { get }  
var displayScale: CGFloat { get }  
var verticalSizeClass: UIUserInterfaceSizeClass { get }  
var horizontalSizeClass: UIUserInterfaceSizeClass { get }
```

- Peuvent surcharger une méthode pour réagir au changement de classe

```
func traitCollectionDidChange(_ previousTraitCollection: UITraitCollection?)
```

Objets courants



Boutons

- UIButton

- `class func buttonWithType(_ buttonType: UIButtonType) -> AnyObject`
- `var titleLabel: UILabel? { get }`
- `var imageView: UIImageView? { get }`

Button





Boutons

- UIButton → UIButtonTypeCustom
 - `func setImage(_ image: UIImage?, forState state: UIControlState)`
- UIControlState
 - UIControlStateNormal
 - UIControlStateHighlighted
 - UIControlStateDisabled
 - UIControlStateSelected

Table View

- UITableView

- Affiche une liste d'éléments
- Organisation en section
- Deux styles d'affichage
- Contenu statique ou dynamique (via un dataSource)

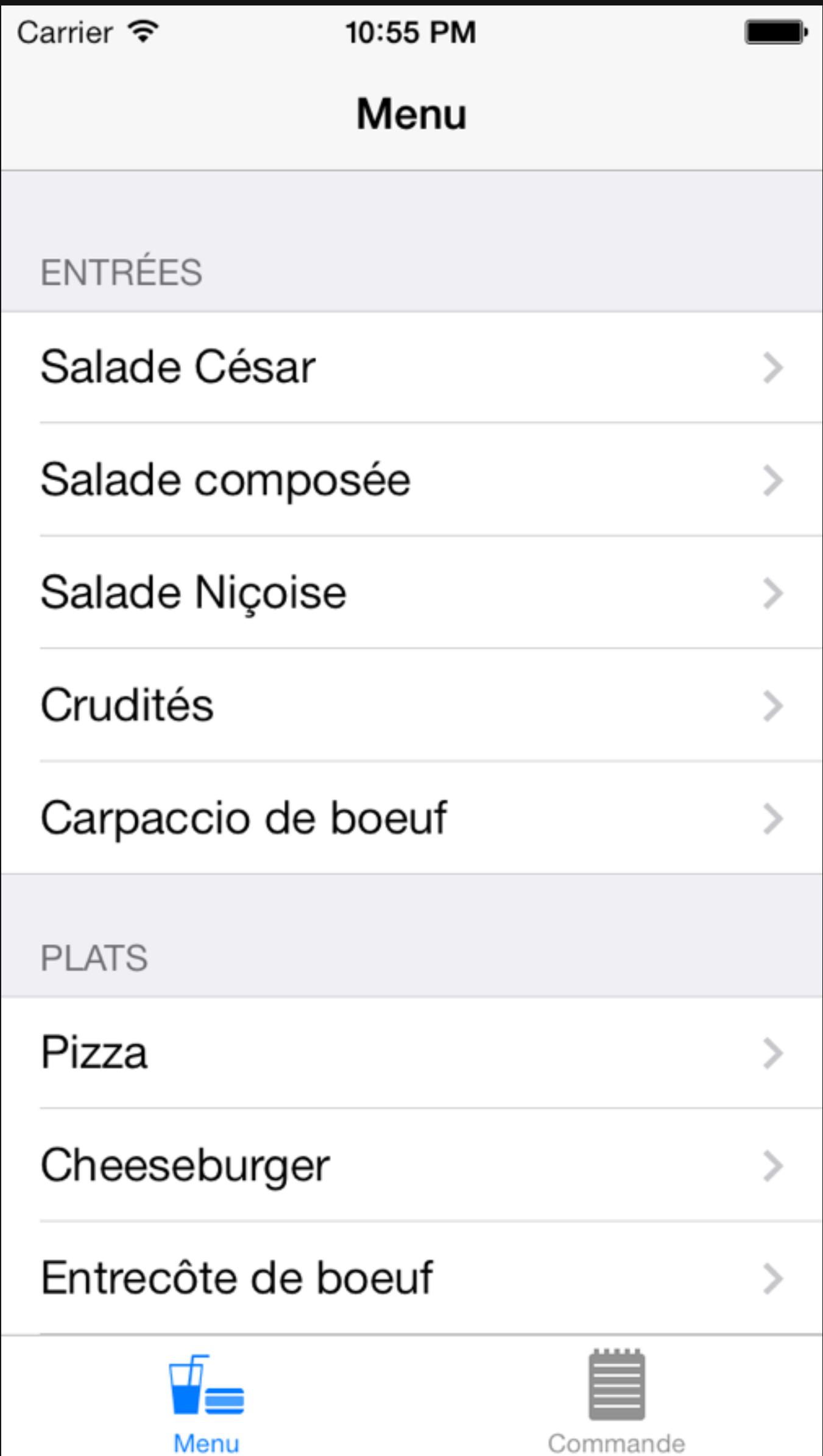


Table View

- UITableView

- Affiche une liste d'éléments
- Organisation en section
- Deux styles d'affichage
- Contenu statique ou dynamique (via un dataSource)

Section 0

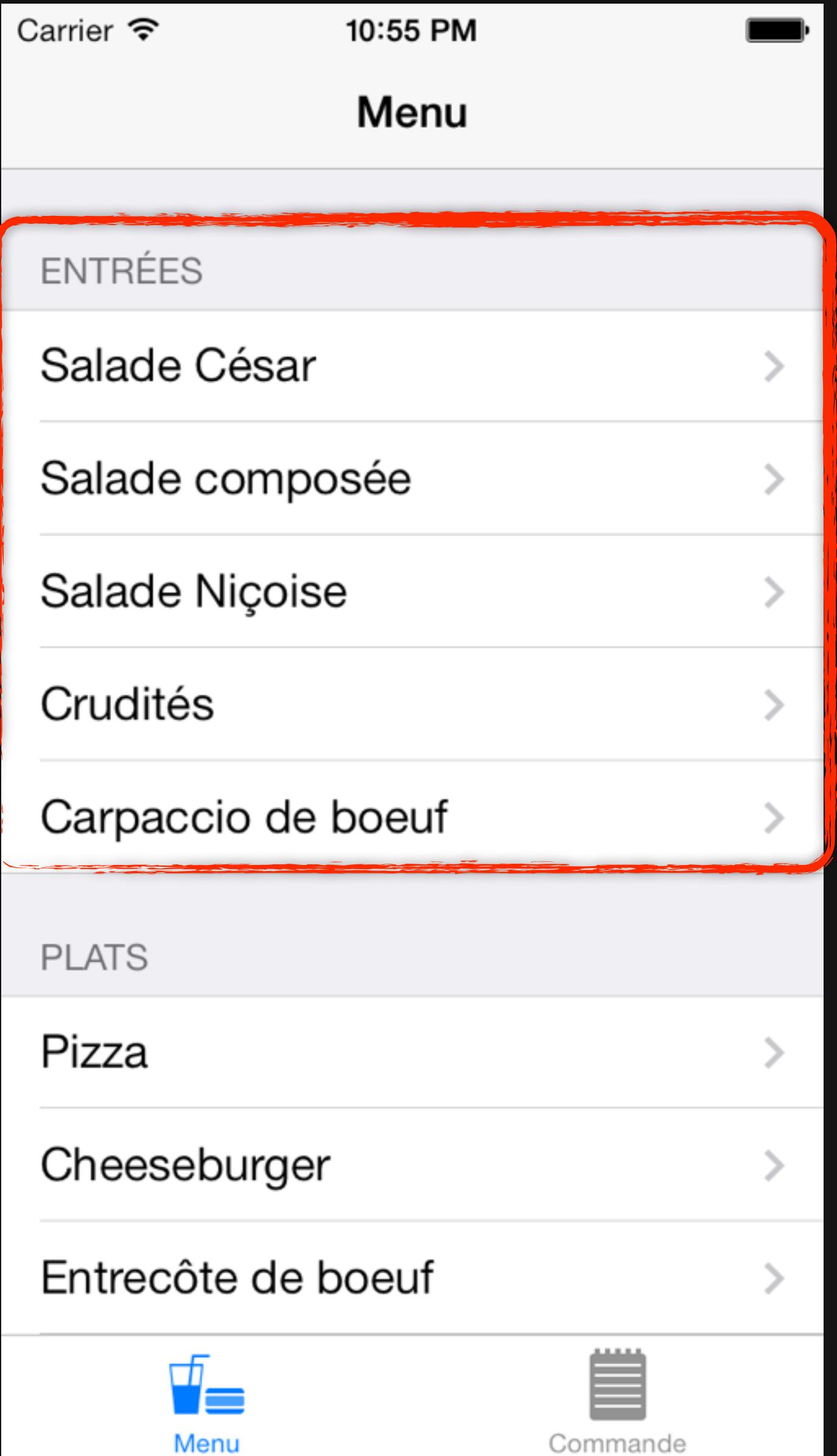


Table View

- UITableView

- Affiche une liste d'éléments
- Organisation en section
- Deux styles d'affichage
- Contenu statique ou dynamique (via un dataSource)

Section Header

Section 0

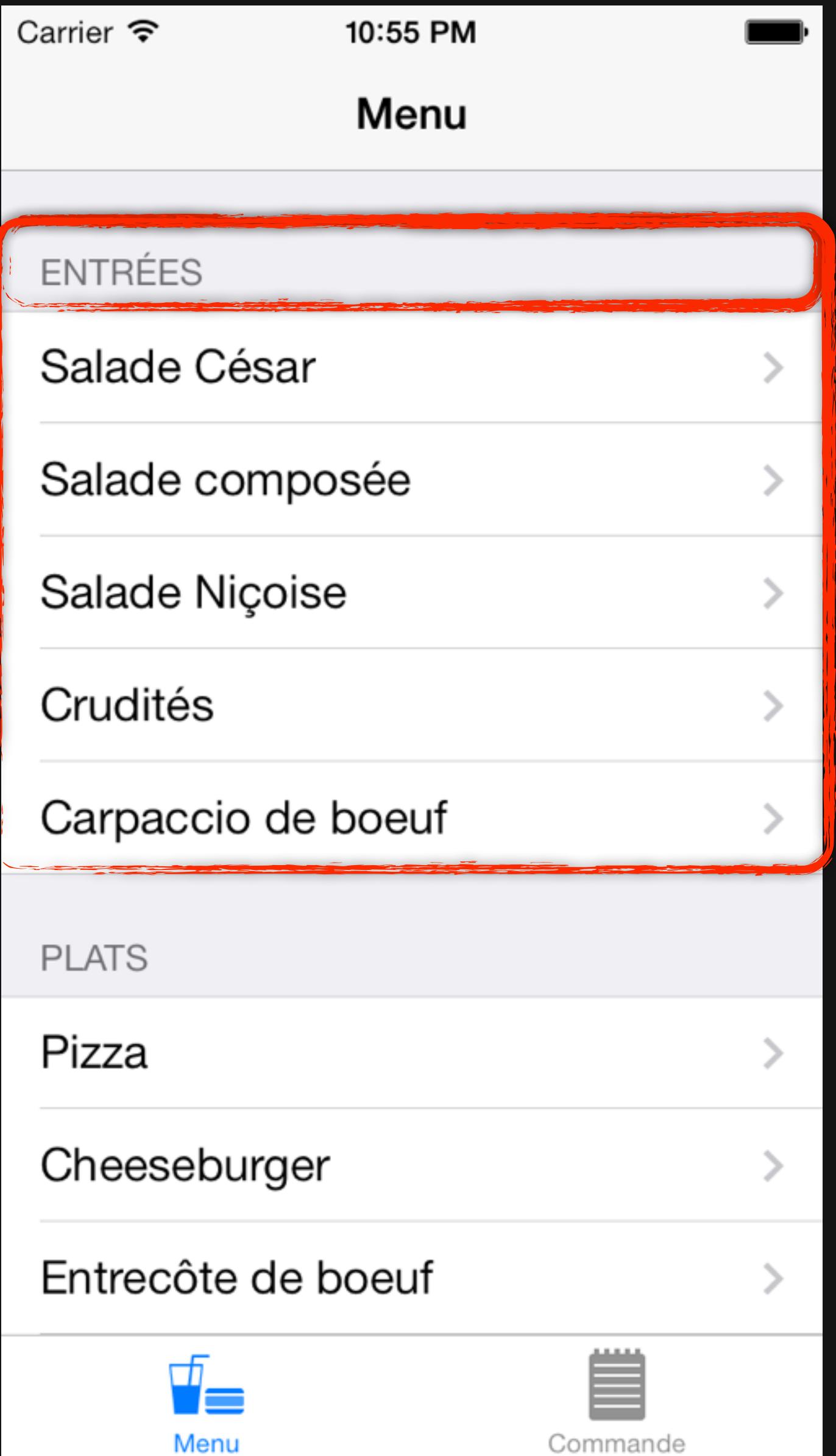


Table View

- UITableView

- Affiche une liste d'éléments
- Organisation en section
- Deux styles d'affichage
- Contenu statique ou dynamique (via un dataSource)

Section Header

Section 0

10:55 PM

Menu

ENTRÉES

Salade César

Salade composée

Salade Niçoise

Crudités

Carpaccio de boeuf

PLATS

Cell section 1 line 0

Pizza

Cheeseburger

Entrecôte de boeuf



Menu



Commande

Navigation Controller

- **UINavigationController**
- Fonctionne avec une pile de ViewControllers
- Gère le titre et le bouton précédent en fonction des propriétés title des ViewControllers
- Gère les animations



La salade porte le nom de Caesar Cardini, un restaurateur italien originaire de la région du Lac Majeur avant d'établir des restaurants à Tijuana et à Ensenada afin de bénéficier d'une clientèle souhaitant contourner la prohibition.

Il existe plusieurs histoires au sujet de la création de cette salade, mais aucune d'elles ne peut être confirmée. La plus commune, racontée par Rosa Cardini (née en 1928), dit que la création fut le résultat d'un épisode de

Navigation Controller

- **UINavigationController**
- Fonctionne avec une pile de ViewControllers
- Gère le titre et le bouton précédent en fonction des propriétés title des ViewControllers
- Gère les animations



La salade porte le nom de Caesar Cardini, un restaurateur italien originaire de la région du Lac Majeur avant d'établir des restaurants à Tijuana et à Ensenada afin de bénéficier d'une clientèle souhaitant contourner la prohibition.

Il existe plusieurs histoires au sujet de la création de cette salade, mais aucune d'elles ne peut être confirmée. La plus commune, racontée par Rosa Cardini (née en 1928), dit que la création fut le résultat d'un épisode de

Navigation Controller

- **UINavigationController**
- Fonctionne avec une pile de ViewControllers
- Gère le titre et le bouton précédent en fonction des propriétés title des ViewControllers
- Gère les animations

Back button



La salade porte le nom de Caesar Cardini, un restaurateur italien originaire de la région du Lac Majeur avant d'établir des restaurants à Tijuana et à Ensenada afin de bénéficier d'une clientèle souhaitant contourner la prohibition.

Il existe plusieurs histoires au sujet de la création de cette salade, mais aucune d'elles ne peut être confirmée. La plus commune, racontée par Rosa Cardini (née en 1928), dit que la création fut le résultat d'un épisode de

Tab Bar

- UITabBarController
- Fonctionne avec un tableau de ViewControllers
- Affiche l'image monochrome de la propriété tabBarItem de chaque ViewController
- Un TabBar doit toujours rester à l'écran



Tab Bar

- UITabBarController
- Fonctionne avec un tableau de ViewControllers
- Affiche l'image monochrome de la propriété tabBarItem de chaque ViewController
- Un TabBar doit toujours rester à l'écran



Tab Bar

- UITabBarController
- Fonctionne avec un tableau de ViewControllers
- Affiche l'image monochrome de la propriété tabBarItem de chaque ViewController
- Un TabBar doit toujours rester à l'écran

Tab Bar item

Tab Bar



Tab Bar

- UITabBarController
- Fonctionne avec un tableau de ViewControllers
- Affiche l'image monochrome de la propriété tabBarItem de chaque ViewController
- Un TabBar doit toujours rester à l'écran



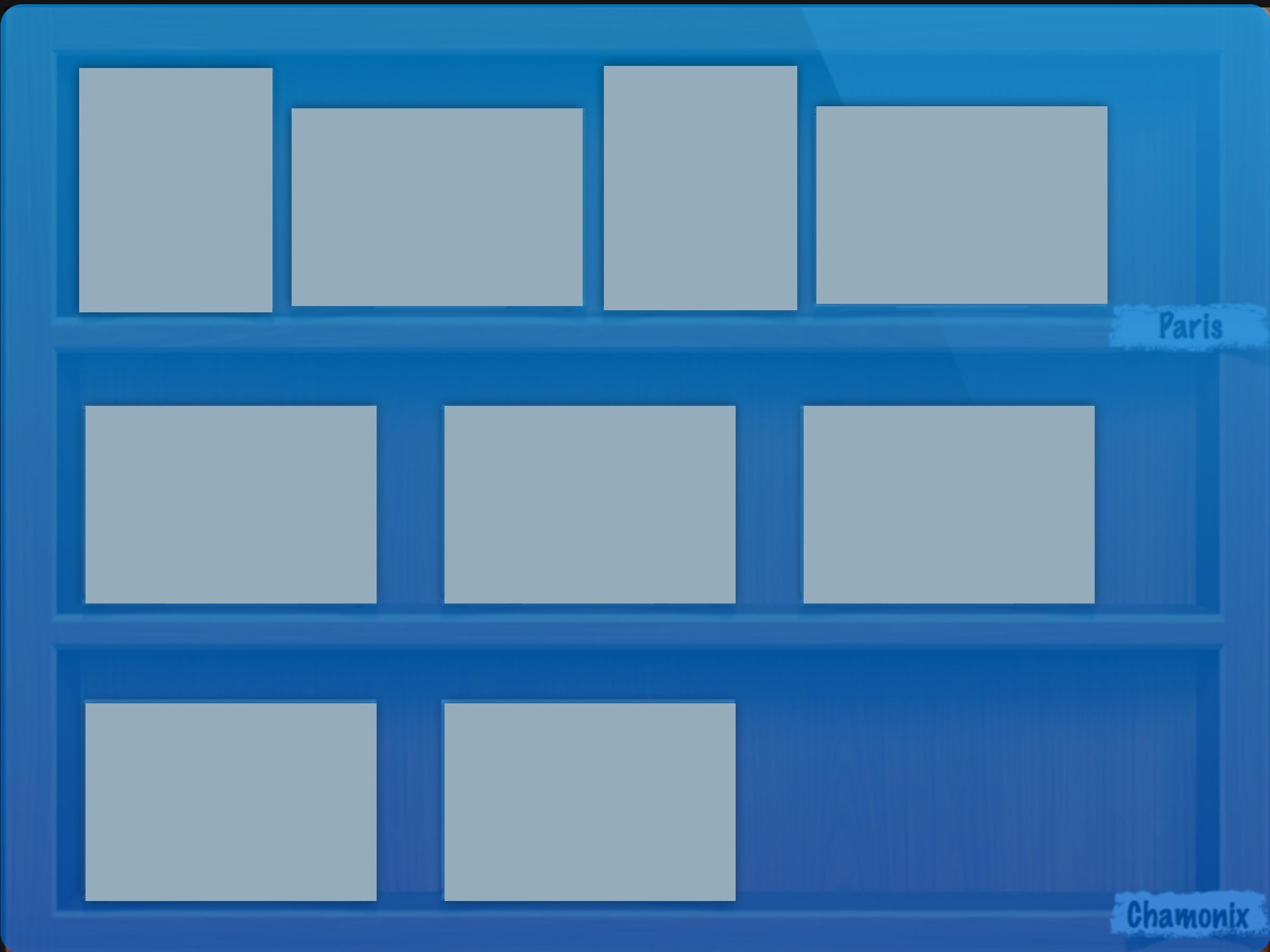
Collection View

- UICollectionView
- UITableView aux stéroïdes
- Affiche une collection d'éléments
- Complétement personnalisable



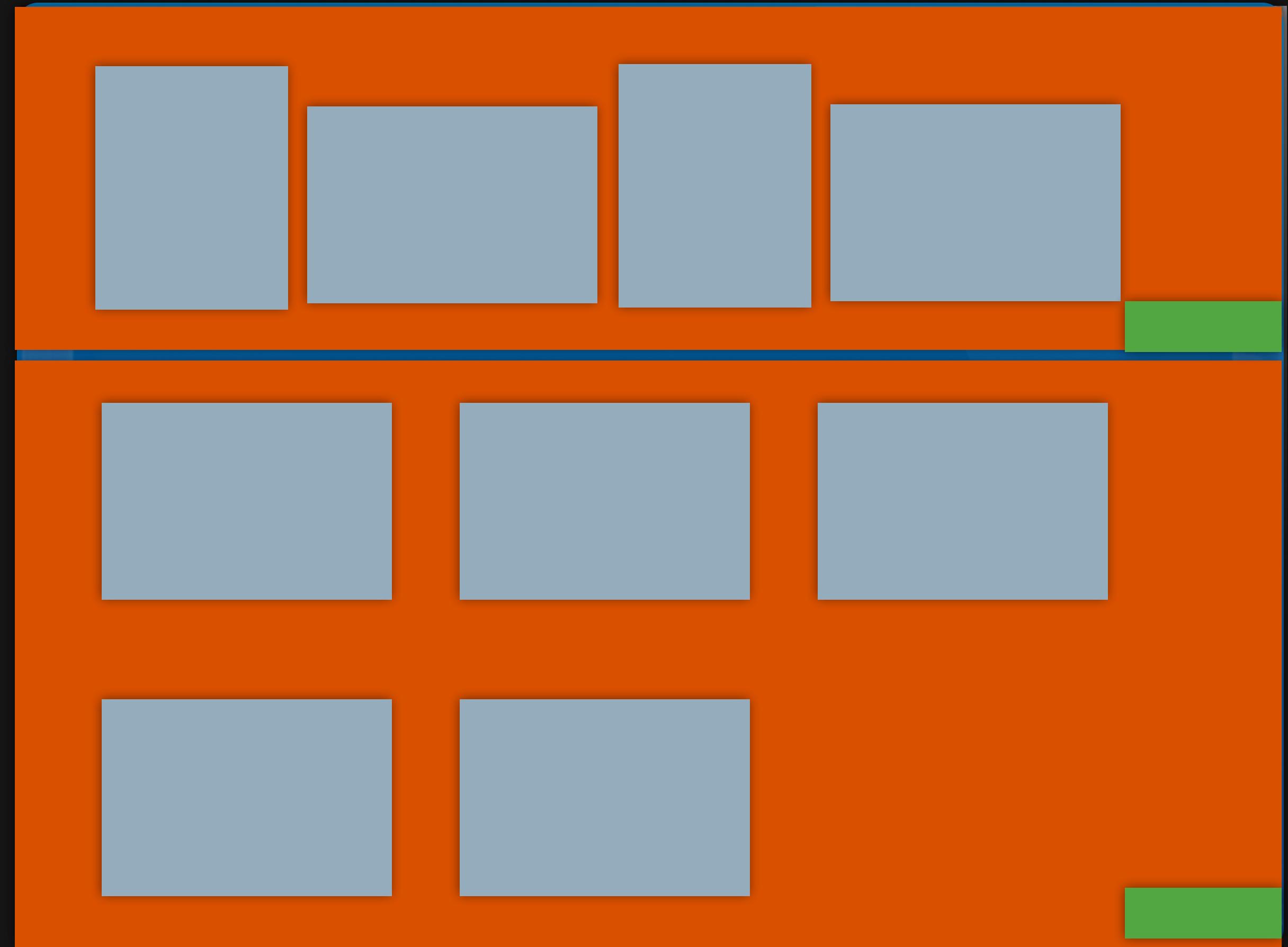
Collection View

- UICollectionView
 - UITableView aux stéroïdes
 - Affiche une collection d'éléments
 - Complétement personnalisable



Collection View

- UICollectionView
 - UITableView aux stéroïdes
 - Affiche une collection d'éléments
 - Complétement personnalisable



Collection View

- UICollectionView
- UITableView aux stéroïdes
- Affiche une collection d'éléments
- Complétement personnalisable



Objets personnalisés

Objets personnalisés

- Créer son propre contrôleur graphique
- Vérifier avant que l'existant ne peut être personnalisé
- Coller au fonctionnement des objets UIKit.

Objets personnalisés

- ❖ Personnaliser l'image d'un bouton



Objets personnalisés

- ❖ Personnaliser l'image d'un bouton
 - ❖ Utiliser l'existant



▼ Button

Type	Rounded Rect
State Config	Default
Title	Plain
Button	
Font	System Bold 15.0
Text Color	Default
Shadow Color	Default
Image	Default Image
Background	Default Background Image

Objets personnalisés

- ❖ Personnaliser l'image d'un bouton
 - ❖ Utiliser l'existant



▼ Button

Type Rounded Rect

State Config
✓ Default
Highlighted
Selected
Disabled

Title Placeholder

Font System Bold 15.0

Text Color Default

Shadow Color Default

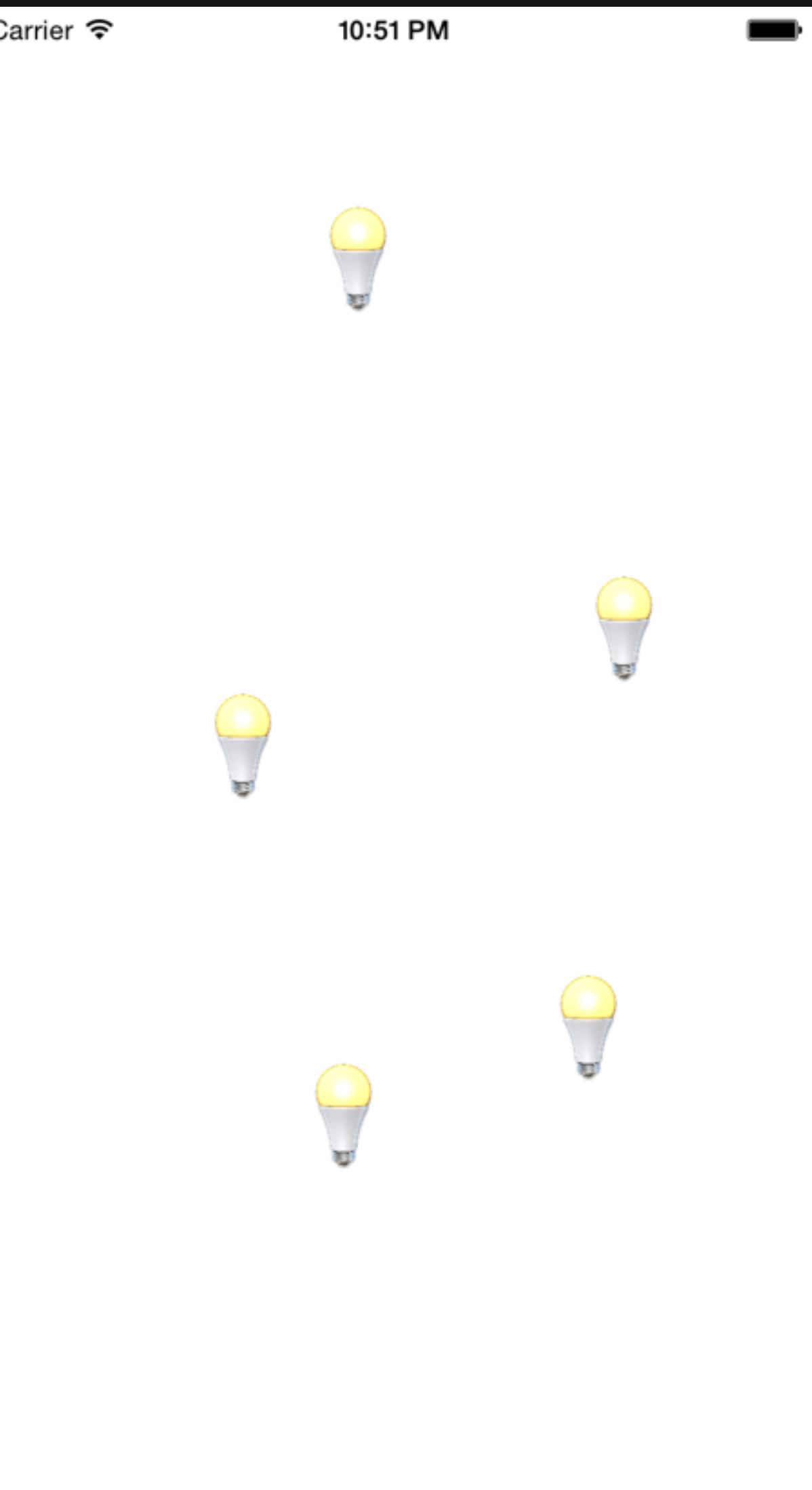
Image Default Image

Background Default Background Image

A screenshot of a user interface element for customizing a button. The main title is "Button". Under "Type", "Rounded Rect" is selected. In the "State Config" section, "Default" is checked, while "Highlighted", "Selected", and "Disabled" are options. Below that, there are fields for "Title" (placeholder text), "Font" (set to "System Bold 15.0"), "Text Color" (blue square), "Shadow Color" (black square), "Image" (dropdown set to "Default Image"), and "Background" (dropdown set to "Default Background Image"). A dropdown menu for "State Config" is open, showing the four states: Default (checked), Highlighted, Selected, and Disabled.

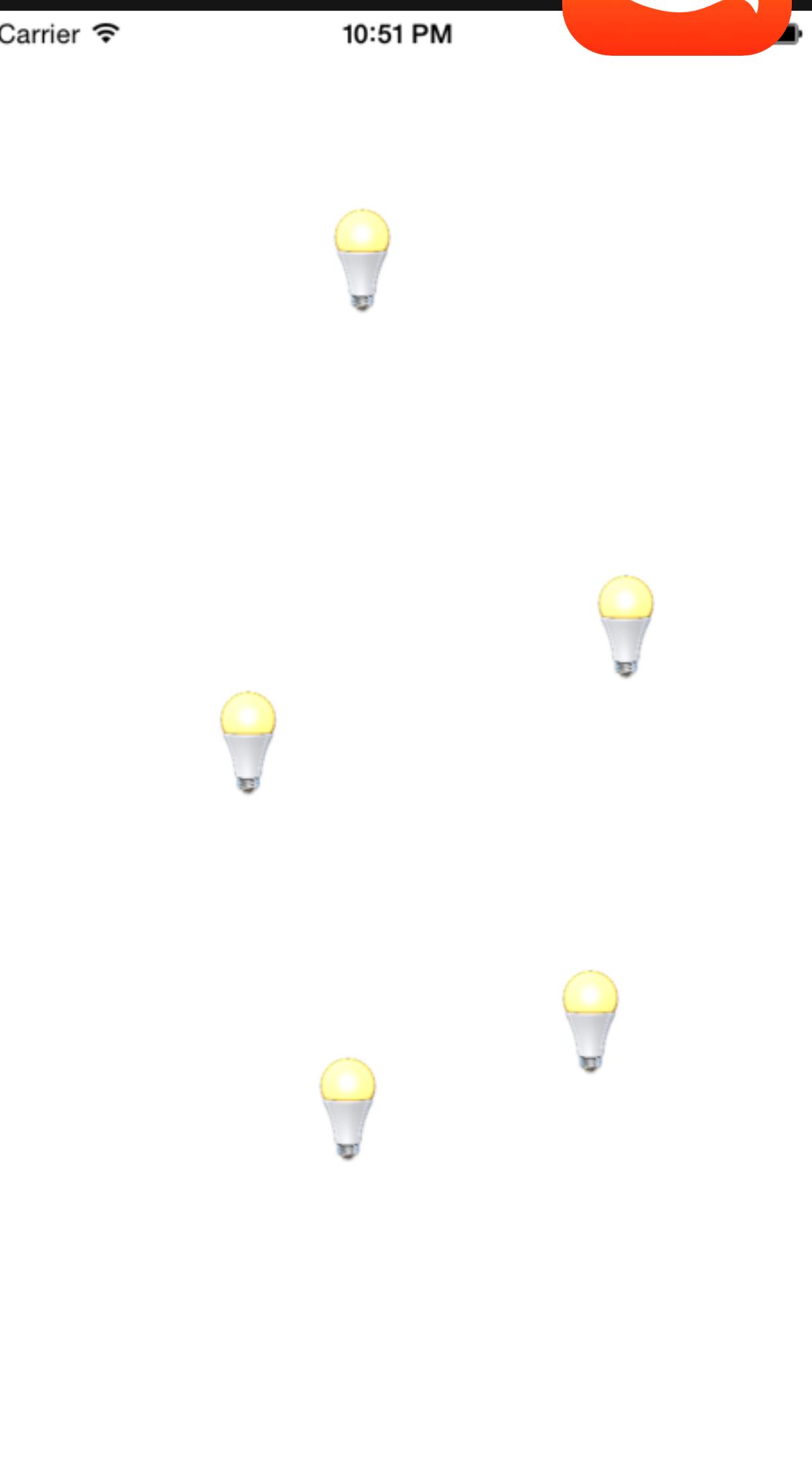
Objets personnalisés

- ❖ Pour créer une vue personnalisée : sous-classer UIView



Objets personnalisés

```
class CustomView: UIView {  
  
    var lastImage: UIImageView = UIImageView()  
  
    override func touchesBegan(touches: NSSet, withEvent event: UIEvent) {  
  
        let targetPoint: CGPoint = touches.anyObject()!.locationInView(self)  
  
        lastImage = UIImageView(frame: CGRectMake(targetPoint.x, targetPoint.y, 50, 50))  
        lastImage.image = UIImage(named: "light.png")  
        self.addSubview(lastImage)  
  
    }  
  
    override func touchesMoved(touches: NSSet, withEvent event: UIEvent) {  
  
        let targetPoint: CGPoint = touches.anyObject()!.locationInView(self)  
  
        lastImage.frame = CGRectMake(targetPoint.x, targetPoint.y, 50, 50)  
  
    }  
  
    override func touchesCancelled(touches: NSSet!, withEvent event: UIEvent!) {  
  
        lastImage.removeFromSuperview()  
  
    }  
}
```



- Utiliser les UIGestureRecognizer
 - UITapGestureRecognizer
 - UIPinchGestureRecognizer
 - UIRotationGestureRecognizer
 - UISwipeGestureRecognizer
 - UIPanGestureRecognizer
 - UILongPressGestureRecognizer

Pour aller plus loin . . .



- Designing for iOS
- UI Elements
- iOS Human Interface Guidelines (iBook)