UPMC
SORBONNE UNIVERSITÉS

Vues & contrôleurs de vues

F. Kordon - Université P. & M. Curie - Master SAR - année 2014/2015 - CC2014

Fabrice.Kordon@lip6.fr

En guise d'introduction…

Rappel, dans iOS tout est vue

- **Les «fonds», les éléments graphiques, les «widgets»**
  - UILabel, UIButton UISliders, UISegmentedControl, etc
  - Vous pouvez même inventer des objets «à la main»
- **Vous devez donc…**
  - Les créer
  - Associer explicitement des traitements à des événements
  - ATTENTION: certains événements ne sont pas supportés par tous les «widget»

Construire une application

Jusqu'à maintenant :

- Enrichissement d'un «viewControllen»
  - Création des structures de données
  - Création & agencement des vues
  - Création des IBAction
  - Affichage via StoryBoard

Problèmes :

- Pas de séparation entre les données et leur affichage
- Que faire lorsque plusieurs vues doivent se superposer
- Gestion efficace des différents appareils (iPad, iPhone, ...)
  - Isolation du code (comportement et liaison avec d'autres vues)

Construire une application

Jusqu'à maintenant :

- Enrichissement d'un «viewControllen»
  - Création des structures de données
  - [illegible] agencement des vues

MVC, Modèle, Vue, Contrôleur

Mise au point dans les années 1970 (Smalltalk — Xerox)
Lourd mais efficace

Omniprésent en Cocoa (Objective-C & Swift)

Objets pré-construits: les «ViewControllers»
La couche «data» sera vue plus tard (certains «widgets»)

Model View Controller

«gabarit de conception»

Le modèle

- Traitement des données + gestion de leur intégrité
- Offre des méthodes d'accès et de manipulation (setter/getter)

La vue

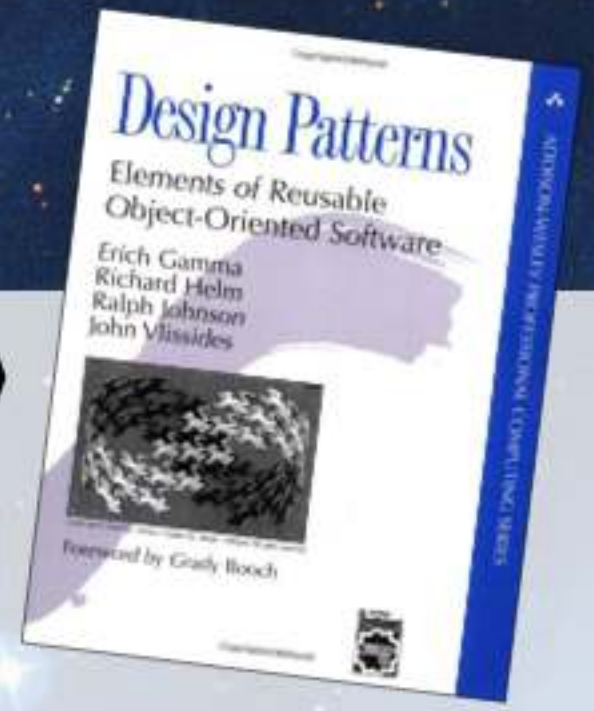
- Interaction avec l'utilisateur (entrées & sorties)

Le contrôleur

- Gestion des événements
  - **Mise à jour de la vue**
  - **Synchronisation entre le modèle et la vue**

Model View Controller

«gabarit de conception»



Le modèle

- Traitement des données + gestion de leur intégrité
- Offre des méthodes d'accès et de manipulation (setter/getter)

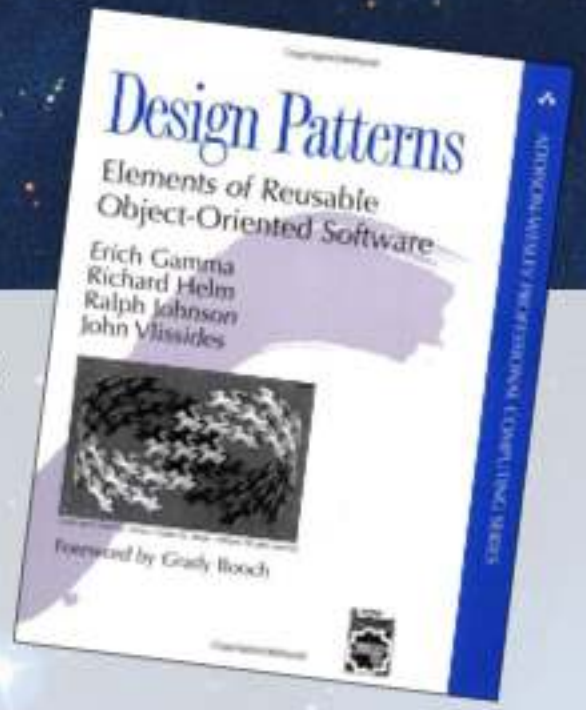
La vue

- Interaction avec l'utilisateur (entrées & sorties)

Le contrôleur

- Gestion des événements
  - **Mise à jour de la vue**
  - **Synchronisation entre le modèle et la vue**

Model View Controller



t de conception»

Avantage
Architecture claire!!!

Le

- Traitement des données + gestion de leur intégrité
- Offre des méthodes d'accès et de manipulation (setter/getter)

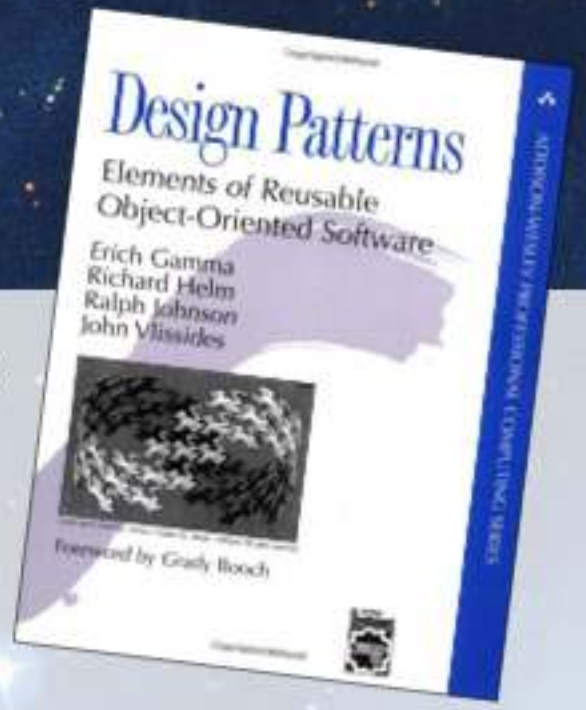
La vue

- Interaction avec l'utilisateur (entrées & sorties)

Le contrôleur

- Gestion des événements
  - **Mise à jour de la vue**
  - **Synchronisation entre le modèle et la vue**

Model View Controller



t de conception»

Avantage
Architecture claire!!!

Le

- Traitement des données + gestion de leur intégrité
- Offre des méthodes d'accès et de manipulation (setter/getter)

La vue

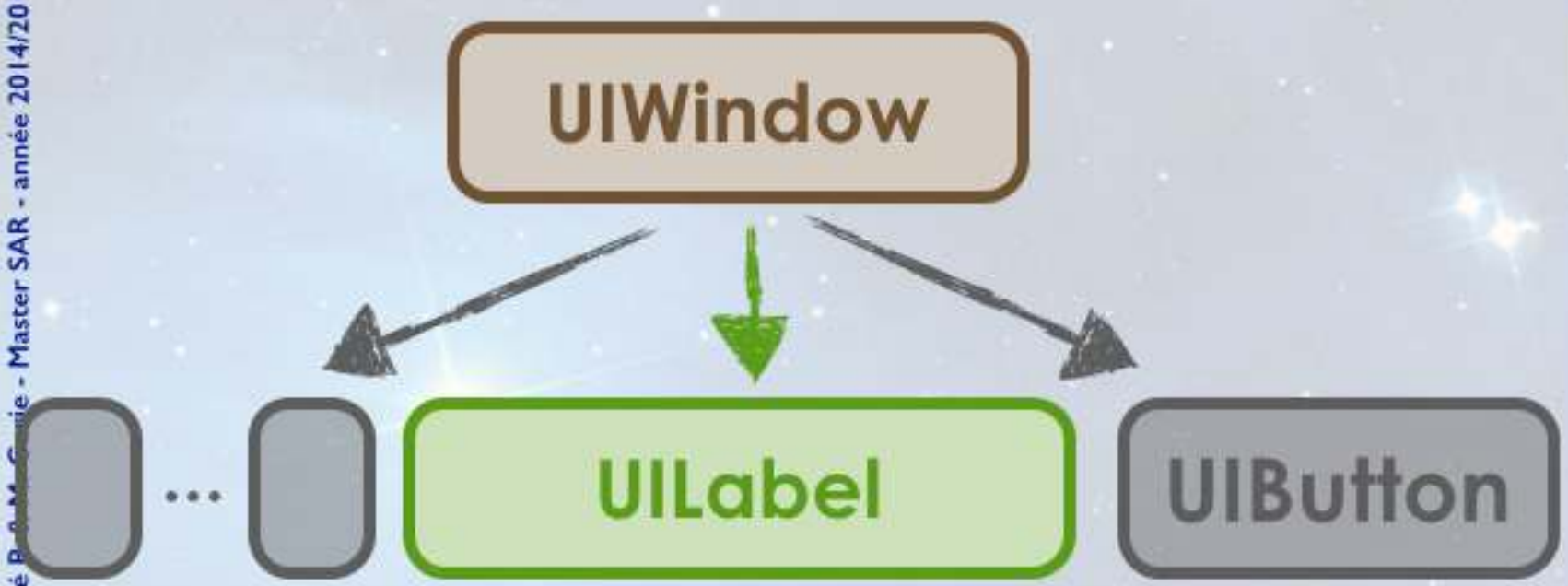
- Interaction ave

Le contrôleu

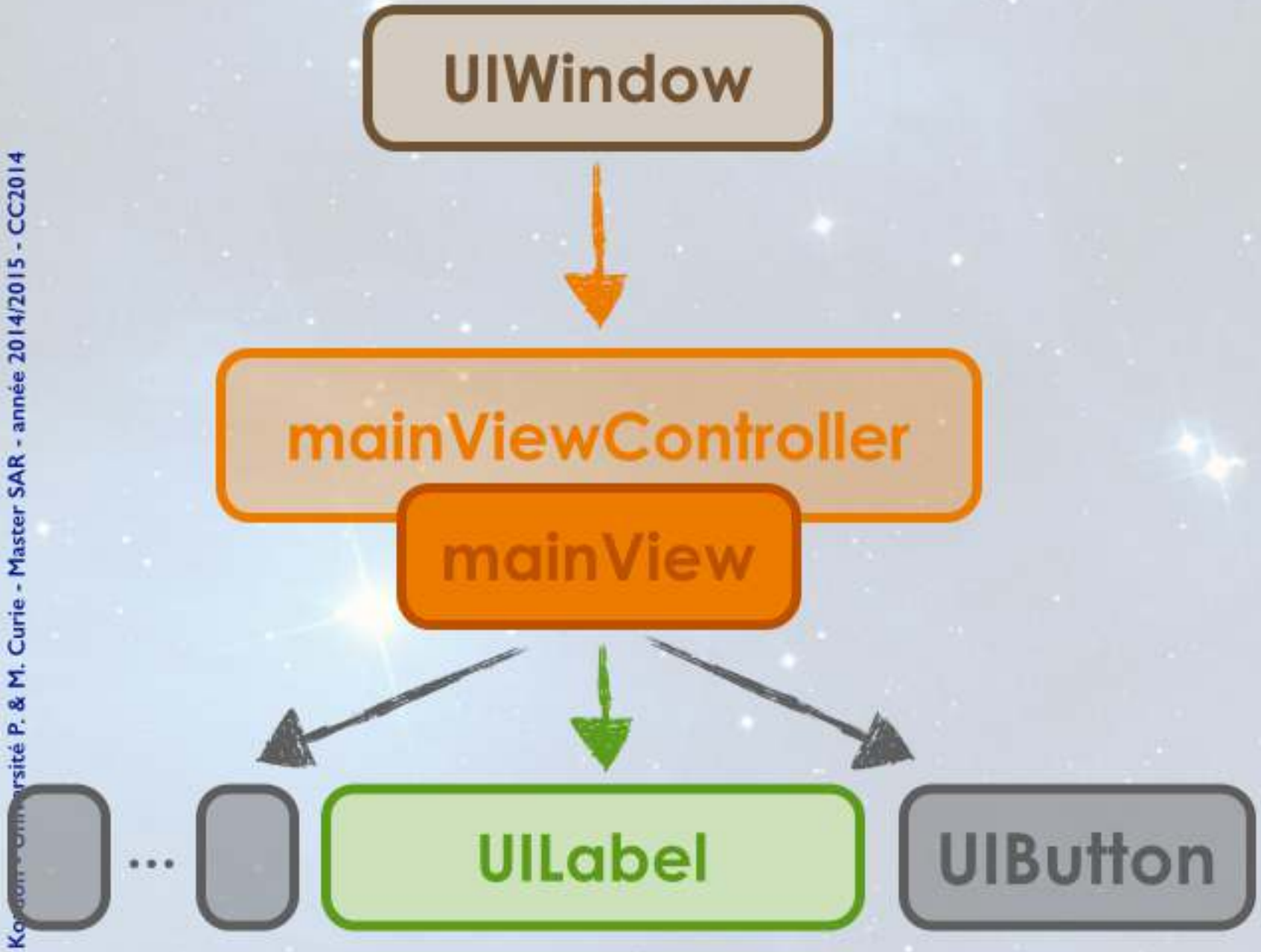
- Gestion des é
  - **Mise à jour de l**
  - **Synchronisation entre le modèle et la vue**

Concrètement!!!
Contrôleur = UIViewController
Vue = UIView
Modèle = une/des classes gérant les données

Structuration d'une vue



Structuration d'une vue



Le cycle de vie d'un ViewController

- **Init ou initWithFrame (ou InitWithNibName)**
- **ViewDidLoad (initialisation & création)**
  - Permet de construire le contenu (aussi dans loadView)
- **D'autres méthodes possibles**
  - isViewLoaded, viewWillAppear, viewDidAppear, viewWillDisapear,...
    - **RTFM**
- **didReceiveMemoryWarning**
  - Alerte en cas de besoin de mémoire
    - **Disposer de ce qui peut l'être**
- **Gestion de la rotation**
  - Présenté plus loin

Créer une vue dans l'application…

- **Créer une classe qui hérite de UIView**
  - Peut supporter le protocole UIScrollViewDelegate
- **Compléter loadView ou ViewDidLoad dans le contrôleur de vue**
  - Allocation/initialisation de la vue
  - Création de la «filiation» (à la vue racine associée)

```
- (void)viewDidLoad {
    [super viewDidLoad];
    // Do any additional setup after loading the view
    UIScreen *ecran = [UIScreen mainScreen];
    CGRect rect = [ecran bounds];
    MaVue *v = [[MaVue alloc] initWithFrame:rect];
    [self setView:v];
    [v release];
}
```

Créer une vue dans l'application…

- **Créer une classe qui hérite de UIView**
  - Peut supporter le protocole UIScrollViewDelegate
- **Compléter loadView ou ViewDidLoad dans le contrôleur de vue**
  - Allocation/initialisation de la vue
  - Création de la «filiation» (à la vue racine associée)

```
override func viewDidLoad() {
    super.viewDidLoad()
    let ecran = UIScreen.mainScreen()
    let rect = ecran.bounds
    let v = MaVue(frame: rect)
    self.view = v
}
```

En guise de conclusion…

Finalement, c'est plus simple que Storyboard…

- Il faut juste gérer les coordonnées

Conseil avisé…

- Prenez le temps de «dessiner»
- Pensez en position relative…

Scoop !

- Il existe un système de gestion de contraintes, **AutoLayouts**