UPMC
SORBONNE UNIVERSITÉS

Les vues, principes de base

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

Fabrice.Kordon@lip6.fr

En guise d'introduction…

Vue = élément de base dans iOS

- À commencer par les mécanismes d'interaction

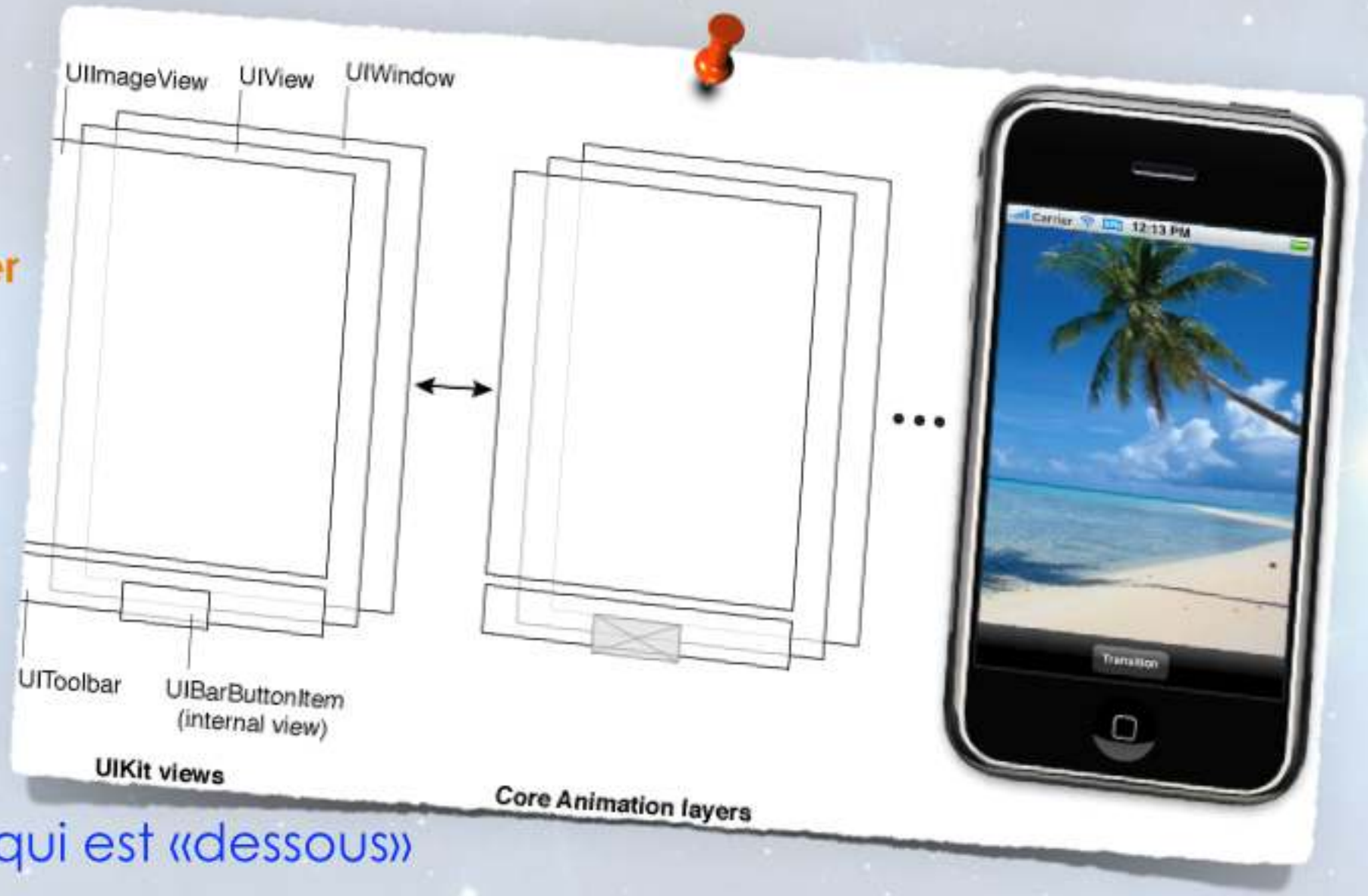
En guise d'introduction…

Vue = élément de base dans iOS

- À commencer par les mécanismes d'interaction

Qu'est-ce?

- Un rectangle
  - Pour y dessiner
  - Pour insérer d'autres vues
  - Sensible aux événements
- Organisation hiérarchique
  - 1 superview
  - 0+ subviews
- Recouvre ce qui est «dessous»
  - Opacité…



En guise d'introduction…

Vue = élément de base dans iOS

- À commencer par les mécanismes d'interaction

Qu'est-ce?

- Un rectangle
  - Pour y dessiner
  - Pour insérer d'autres vues
  - Sensible aux événements
- Organisation hiérarchique
  - 1 superview
  - 0+ subviews
- Recouvre ce qui est
  - Opacité…

La vue racine est contenue dans une fenêtre (UIWindow)

UIView — éléments principaux

Personnalisation

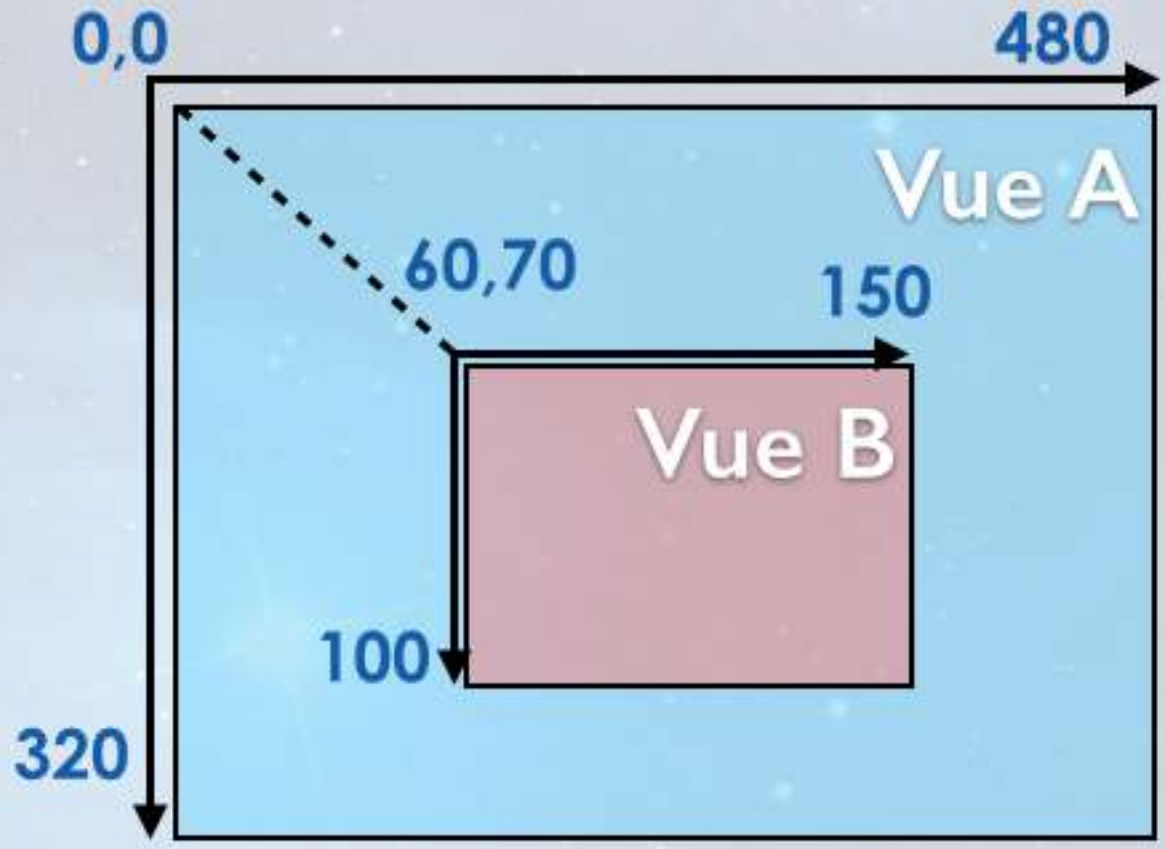
- frame — *position + coordonnées dans la vue parente*
- bounds — *origine + taille*
- backgroundcolor — *couleur du fond (transparent par défaut)*
- hidden — *booléen indiquant si la vue est visible*
- etc.

Relations avec les autres vues

- superview — *la vue de niveau supérieur*
- subviews — *tableau des vues contenues dans la vue*
- window — *fenêtre contenant la vue (ou nil)*
- etc.

Des coordonnées lié à la vue englobante

Par l'exemple…



- **Vue A frame**
  origine: 0,0
  taille: 480,320
- **Vue A bounds**
  origine: 0,0
  taille: 480,320
- **Vue B frame**
  origine:60,70
  taille: 150,100
- **Vue B bounds**
  origine: 0,0
  taille150,100

- Frame : configuration «à l'extérieur»
- Bounds : configuration «à l'intérieur»

Des coordonnées lié à la vue englobante

Par l'exemple…

Rappel sur les tailles d'écrans

3,5" — 320 x 480 pts
4" — 320 x 568 pts
4,7" — **375 x 667** pts
5,5" — 414 x 736 pts
iPad* — 1024 x 768 pts

- **Vue A frame**
  origine: 0,0
  taille: 480,320

bounds
: 0,0
480,320

frame
e:60,70
150,100

bounds
e: 0,0
taille150,100

- Frame : configuration «à l'extérieur»
- Bounds : configuration «à l'intérieur»

Manipulation des vues

Objective-C

```
- (void)addSubview:(UIView*) view;
- (void)removeFromSuperview;
- (void)insertSubview:(UIView*)view atIndex:(int)index;
- (void)exchangeSubviewAtIndex:(int)index
                withSubviewAtIndex:(int)otherIndex;
```

Swift

```
func addSubview(view: UIView)
func insertSubview(view: UIView,
     aboveSubview siblingSubview: UIView)
func exchangeSubviewAtIndex(index1: Int,
     withSubviewAtIndex index2: Int)
func removeFromSuperview()
```

Manipulation des vues

Objective-C

```
- (void)addSubview:(UIView*) view;
- (void)removeFromSuperview
- (void)insertSubview:(UIV
- (void)exchangeSubviewAtI
                withSubviewAtI
```

Une vue est propriétaire de ses sous-vues

En objective-C sans ARC, il faut en tenir compte...

... avec des release!!!

Swift

```
func addSubview(view: UIView)

func insertSubview(view: UIView,
     aboveSubview siblingSubview: UIView)

func exchangeSubviewAtIndex(index1: Int,
     withSubviewAtIndex index2: Int)

func removeFromSuperview()
```

Construire sa propre classe de type vue

- Créer une classe héritant de UIView
- Construire le contenu de la classe
- Éventuellement l'associer à un contrôleur de vue

Autres actions importantes

- Réafficher une partie de la vue

```
- (void) drawRect:(CGRect)rect;
```

```
func drawRect(rect: CGRect)
```

  - Utile pour les vues graphiques

- Forcer le rafraîchissement du contenu

```
- (void) setNeedsDisplay;
```

```
func setNeedsDisplay()
```

En guise de conclusion…

Maintenant il faut «dessiner» dans les vues

Avec UIKit

- Vous êtes condamnés aux rectangles
  - Mais on peut ruser avec la transparence (cf icônes)
- nombreux «widgets» à votre disposition

Avec CoreGraphics

- Bibliothèque complète de dessin
- Contexte (pour dessiner)
  - Initialisé par défaut (variable de type CGContextRef à déclarer)
- Formes (notion de «chemin»)
- Couleur
- Polices
- etc.

En guise de conclusion…

Maintenant il faut «dessiner» dans les vues

Avec UIKit

- Vous êtes condamnés aux rectangles
  - Mais on peut ruser [illegible] la transparence (cf icônes)
- nombreux «widge[illegible]

Avec CoreGra[illegible]

- Bibliothèque cor[illegible]
- Contexte (pour [illegible]
  - Initialisé par défaut (va[illegible]
- Formes (notion de «chemin»)
- Couleur
- Polices
- etc.

Autres mécanismes possibles

OpenGL *(ES)*

SceneKit