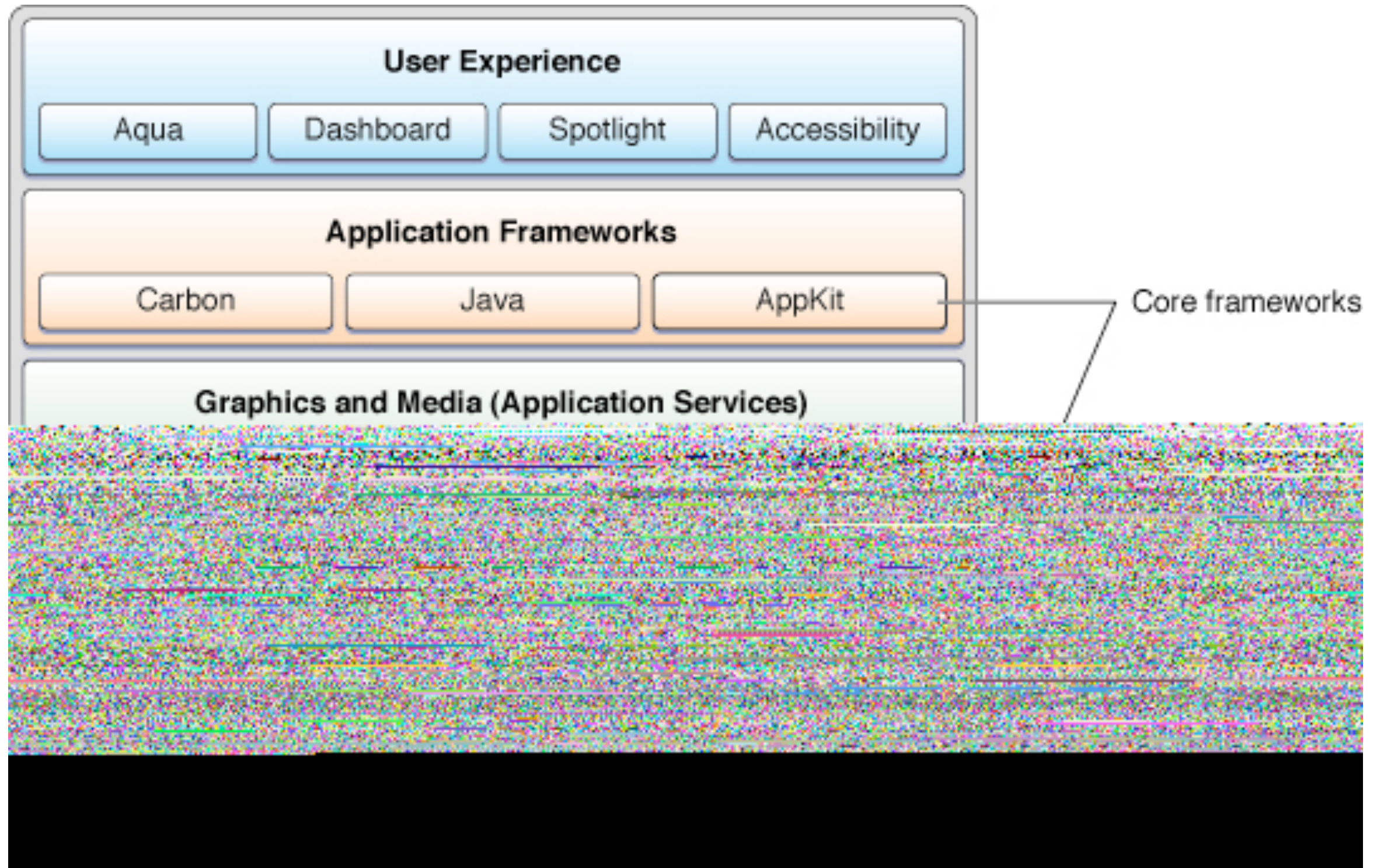


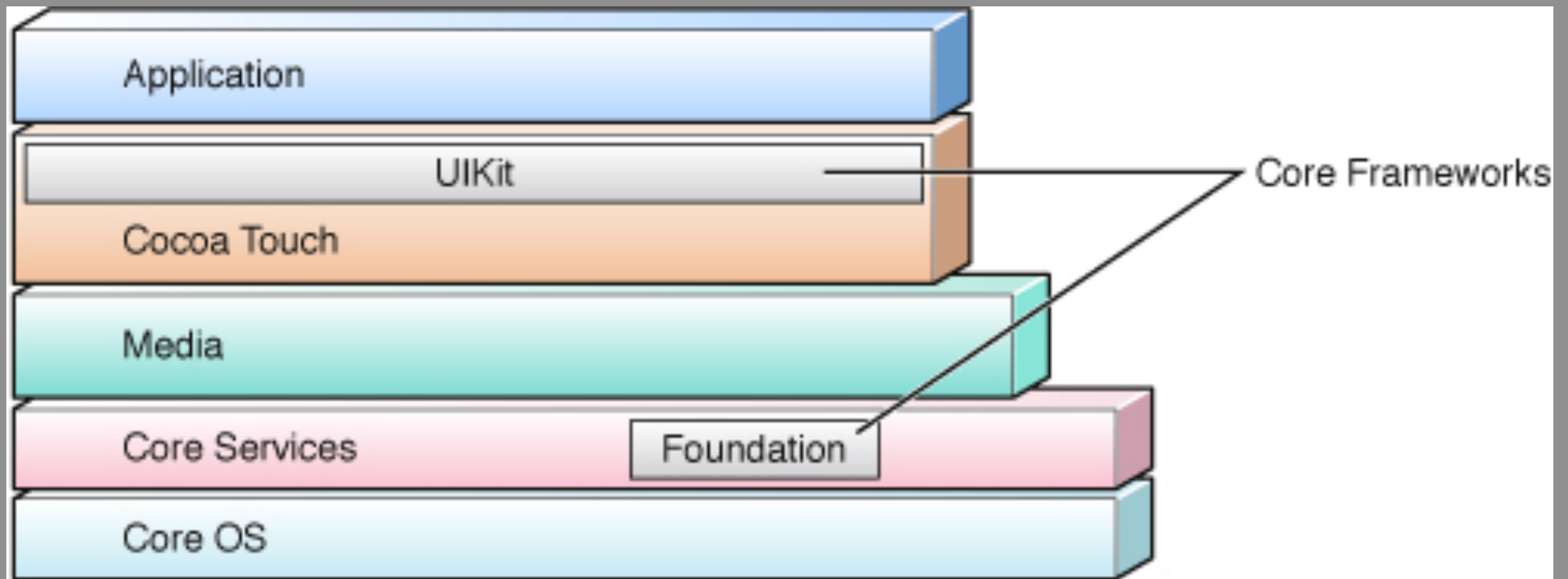
Cocoa

Jean-Baptiste.Yunes@univ-paris-diderot.fr

2012—2013

- deux Cocoa (pour le prix d'un) !
 - pour OSX
 - Foundation+AppKit
 - pour iOS
 - Foundation+UIKit
- Appkit : NS... (ex : NSButton)
- UIKit : UI... (ex : UIButton)





- pour simplifier donc :
- Cocoa c'est Foundation + l'interface graphique
- disons que ce sont les premières choses que l'on voit de Cocoa



Jean-Baptiste.Yunes@univ-paris-diderot.fr / 2012–2013 Master II

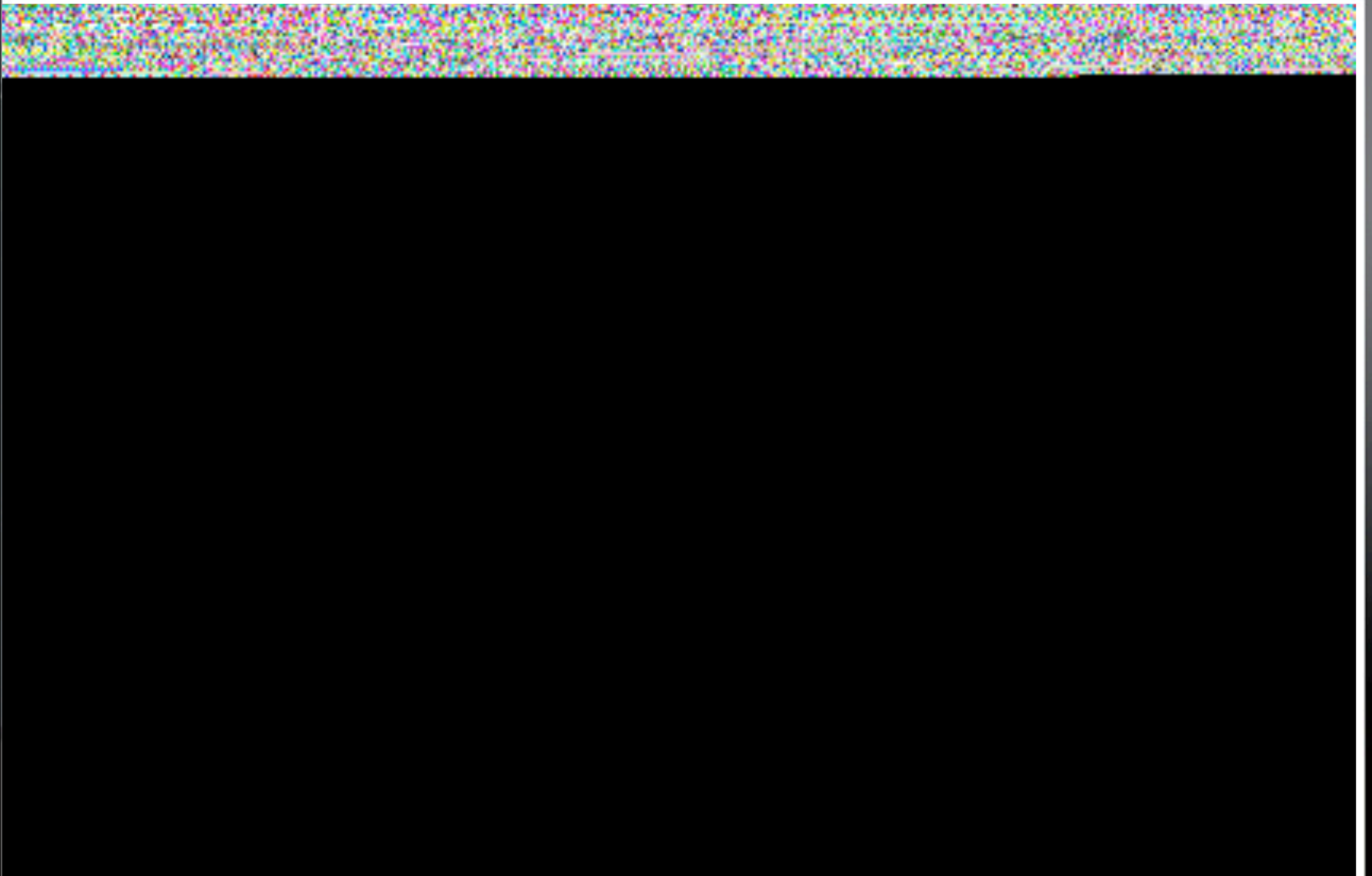
Jean-Baptiste.Yunes@univ-paris-diderot.fr / 2012–2013 Master II

Objective-C Application Kit Continued

Graphics

- NSAffineTransform
- NSAnimationContext
- NSBezierPath
- NSCursor
- NSGradient
- NSGraphicsContext
- NSImage
- NSImageRep
- NSMovie

- NSBitmapImageRep
- NSCachedImageRep
- NSCIImageRep



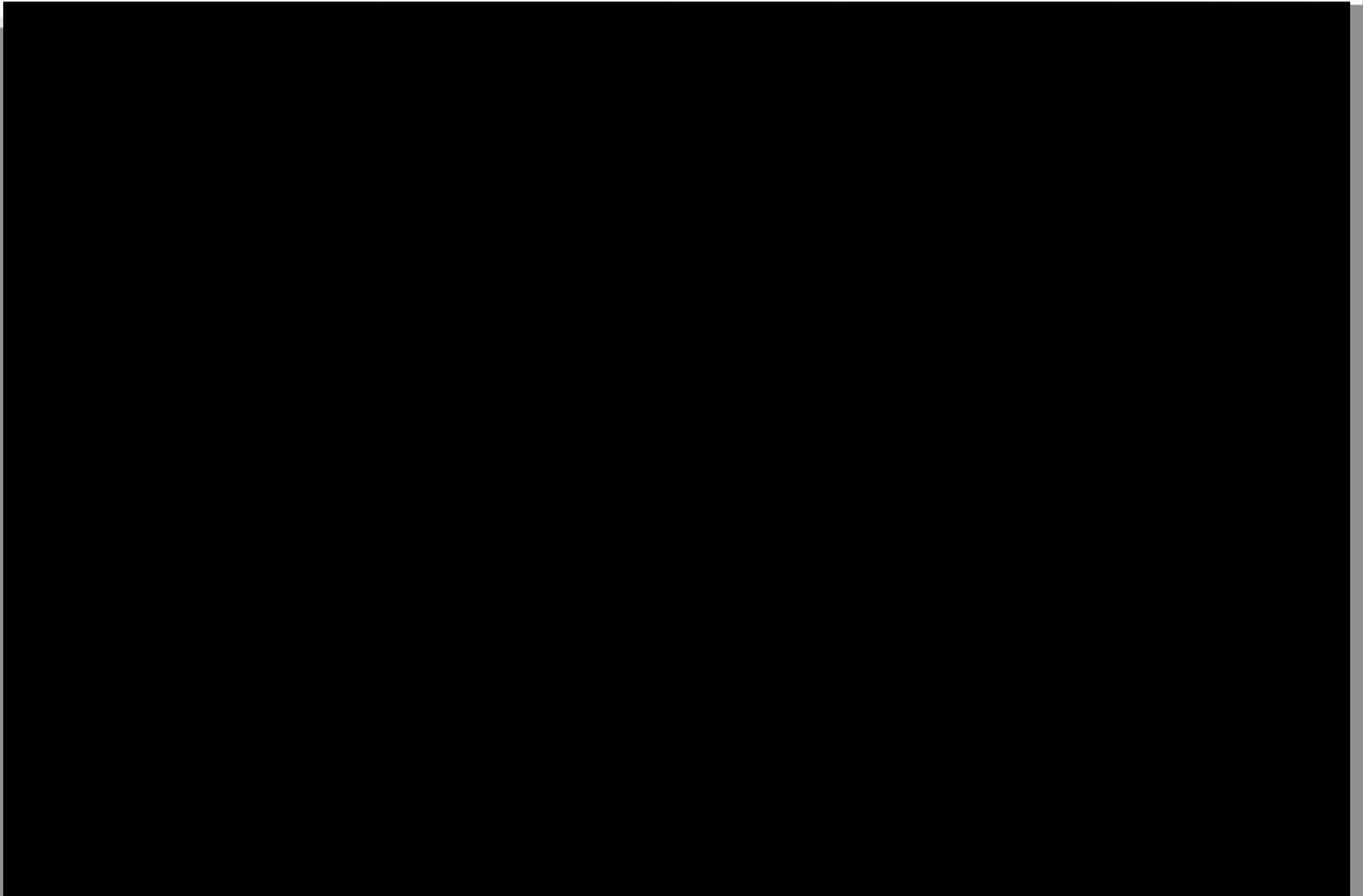
Jean-Baptiste.Yunes@univ-paris-diderot.fr / 2012–2013 Master II

Jean-Baptiste.Yunes@univ-paris-diderot.fr / 2012–2013 Master II

UIAcceleration

UIAccelerometer

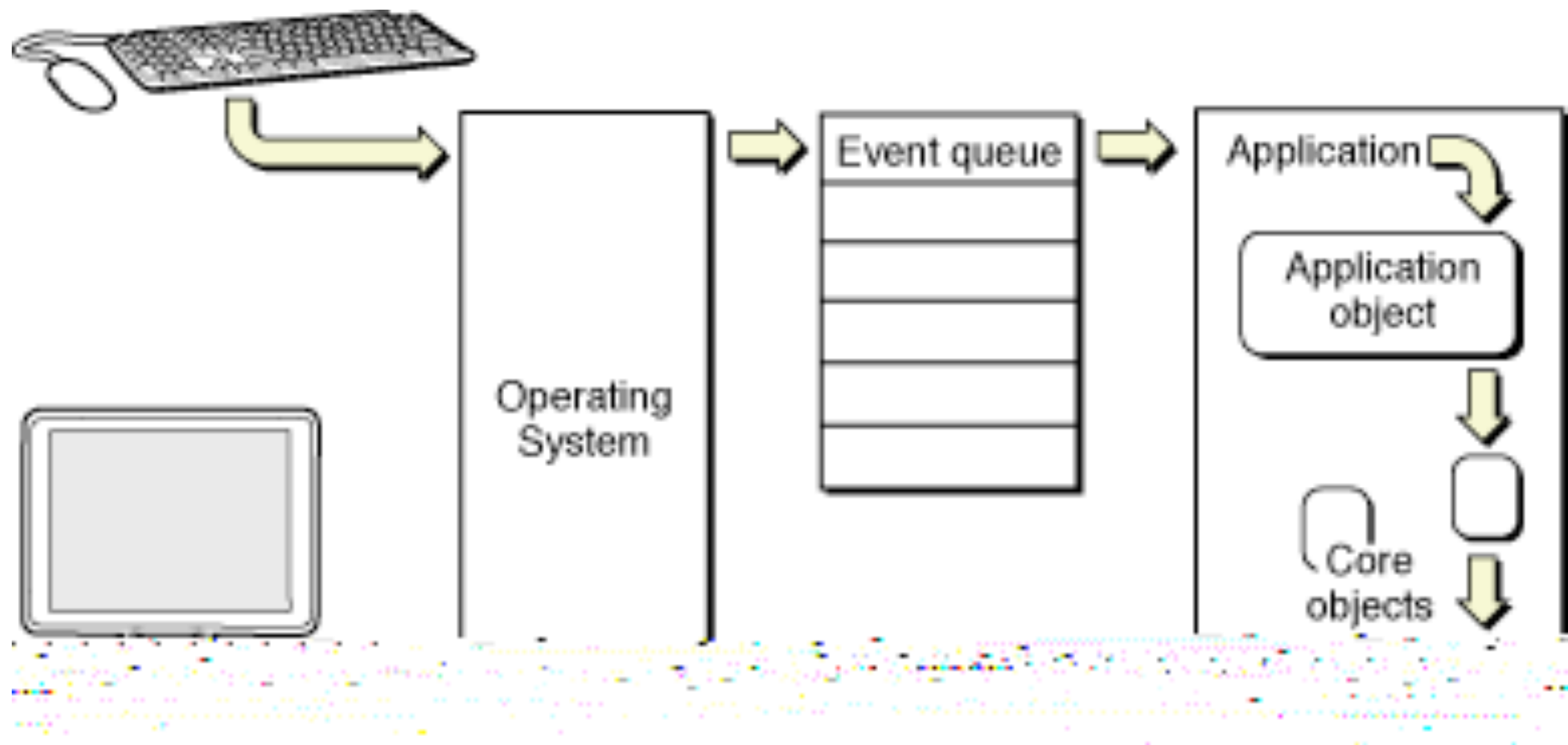
UIAccessibilityElement



Jean-Baptiste.Yunes@univ-paris-diderot.fr / 2012–2013 Master II

La boucle d'évènements

Jean-Baptiste.Yunes@univ-paris-diderot.fr / 2012–2013 Master II



Usage des classes Cocoa

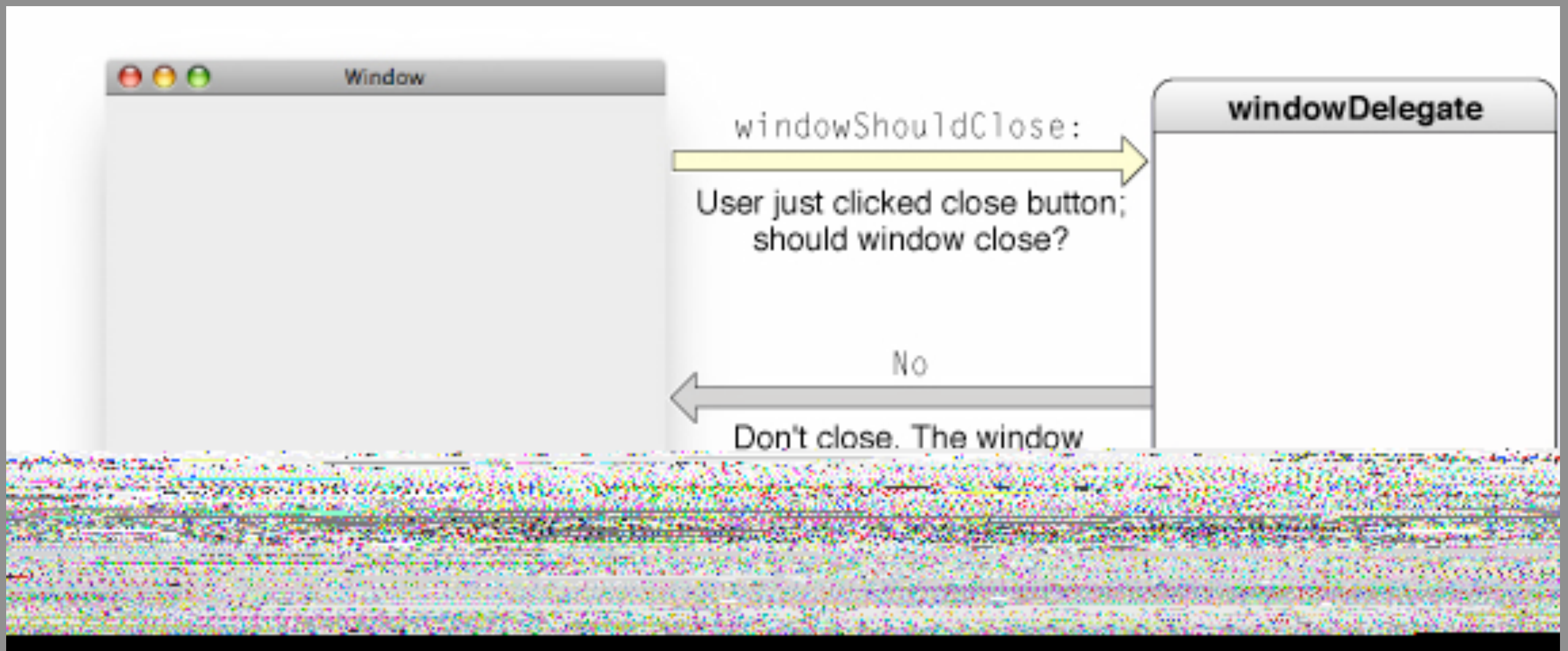
Jean-Baptiste.Yunes@univ-paris-diderot.fr / 2012–2013 Master II

- 4 façons d'utiliser des classes Cocoa
 - nature : on utilise des objets prédéfinis que l'on paramètre (ex : NSButton)
 - sans le savoir : des objets sont créés en arrière-plan
 - générique (ex : UIView)
 - par délégation ou notification

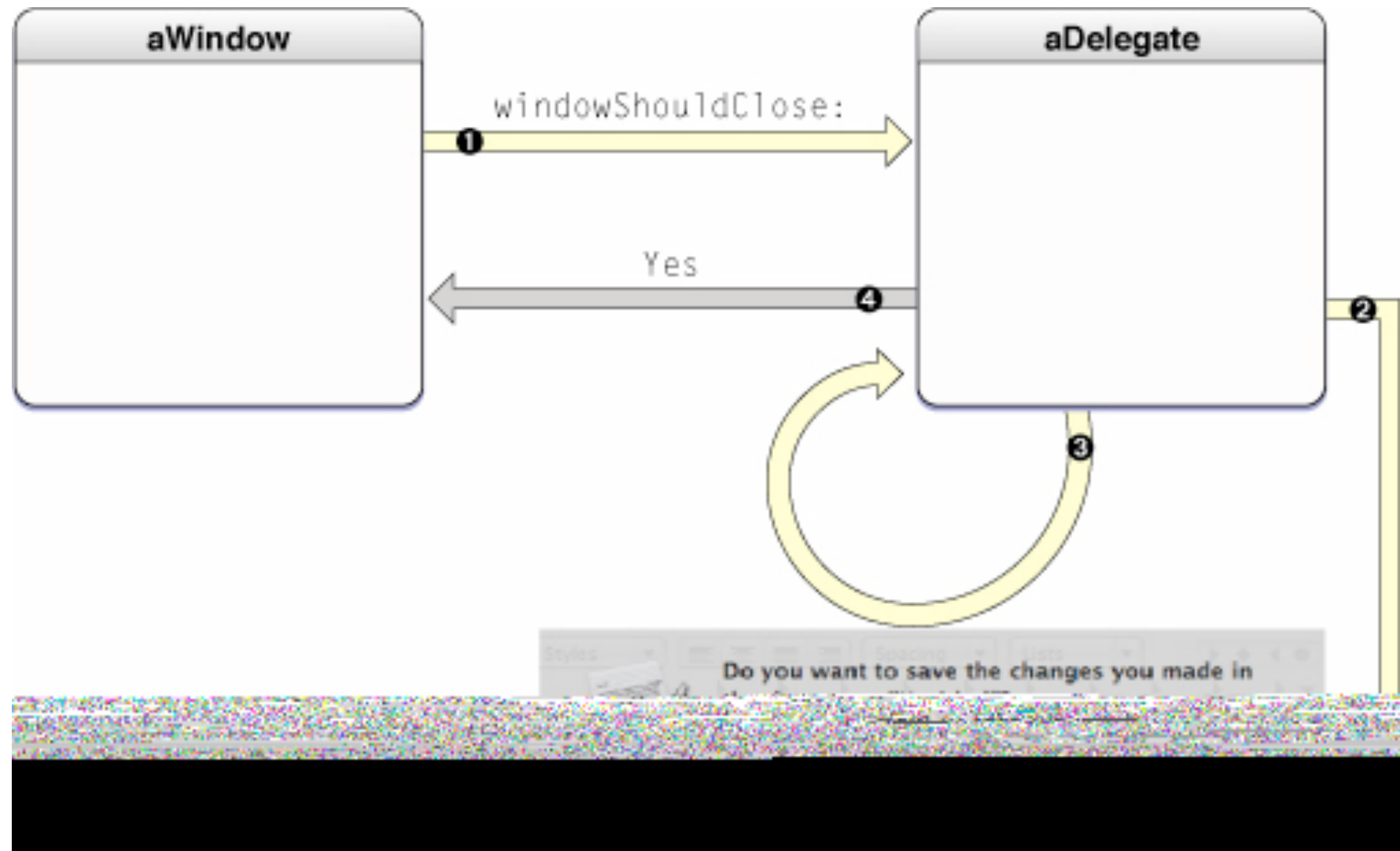
- Les outlets
 - variable d'instance qui référence un autre objet
 - manipulable par l'éditeur d'interface (configurable et archivable)

```
@interface MaClasse : NSObject  
{  
    IBOutlet NSObject *out;  
}
```

- Une classe qui implémente ce pattern possède une propriété nommée delegate

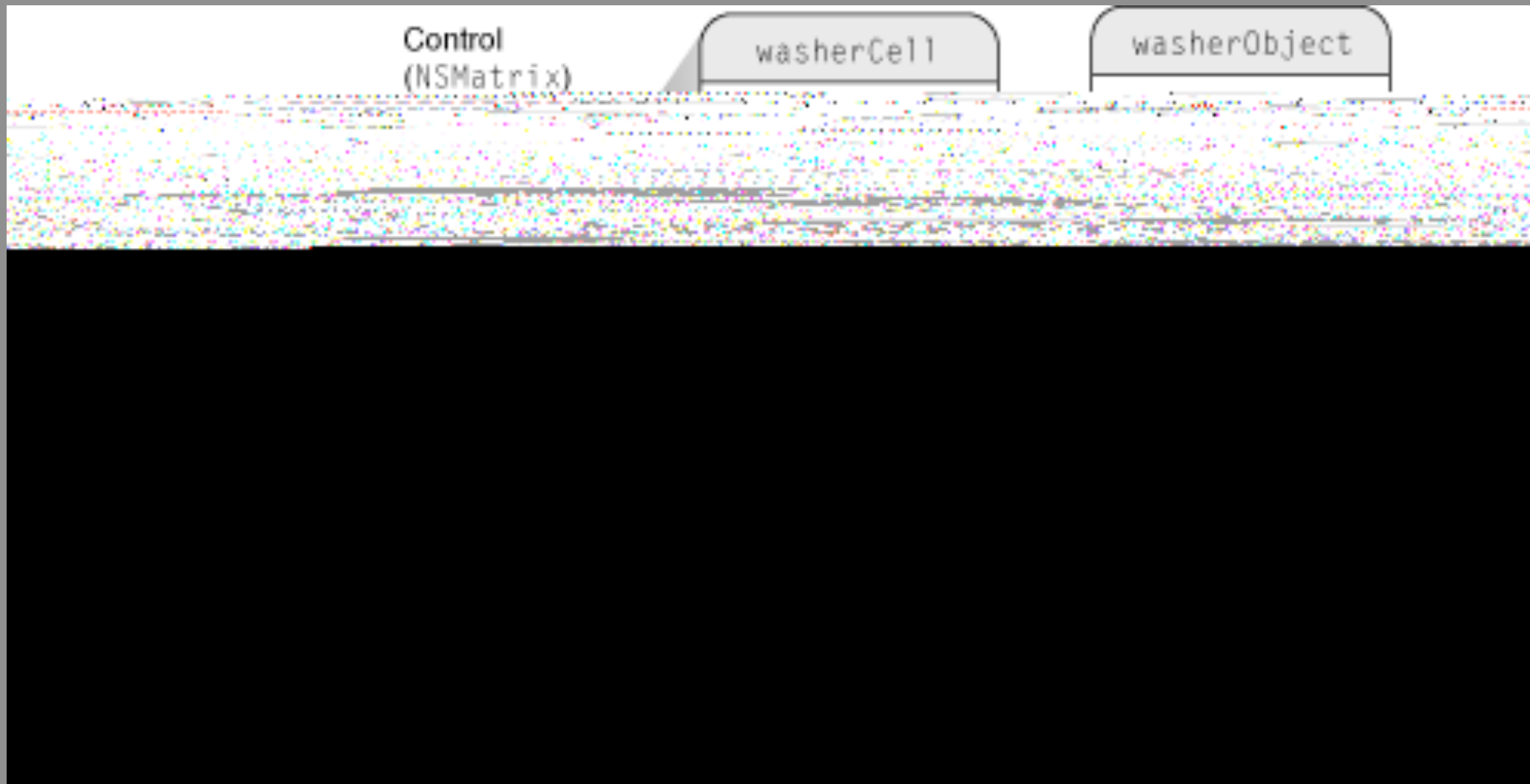


- Un scénario de délégation plus riche

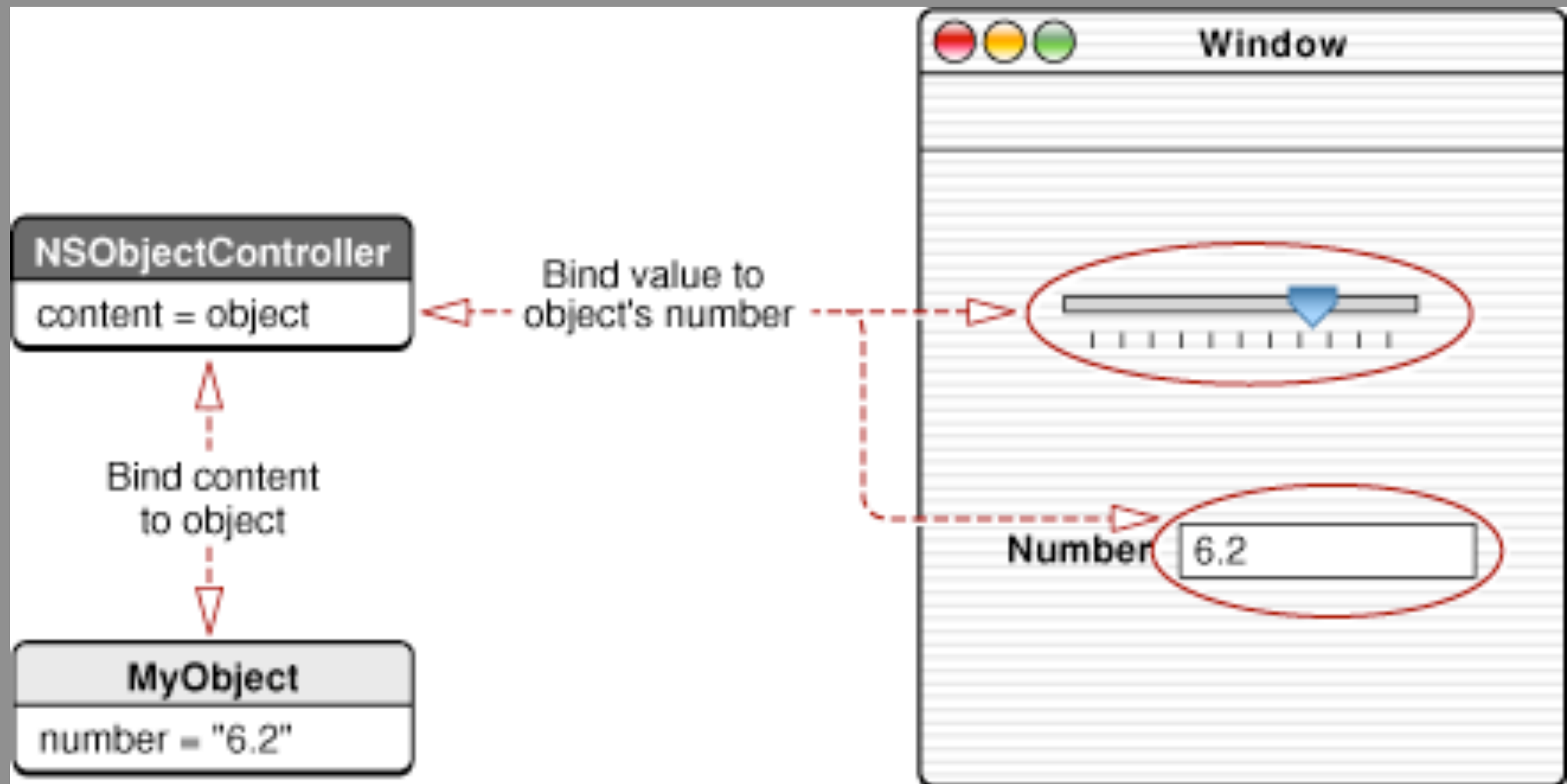


- Il existe une délégation pour les sources de données (data source) et non les interfaces
- Le grand classique est le UITableView

- Le pattern target-action
 - La cible (target) est le receveur d'un message correspondant à une action; il s'agit généralement d'une instance d'une classe personnalisée
 - L'action est le message que le contrôle envoie à la cible
- Une classe éligible pour le pattern doit posséder une méthode signalée comme IBAction
 - (IBAction)faisMoiMal:(id)source
- IBAction n'est pas un type, c'est un tag pour l'éditeur d'interface



- Les bindings
 - dans le cadre du MVC
 - view : affiche les données
 - model : représente le concept
 - controller : médiateur entre le vue et le modèle
 - reposent sur des objets conforme aux patterns KVO et KVC (suffisant pour les modèles)



- les points d'entrée et sortie dans les objets sont :
 - `+initialize` initialisation de la classe
 - `-init` initialisation d'un objet
 - `-initWithCoder:` initialisation par désérialisation
 - `-awakeFromNib:` réveil d'un objet lors de sa désérialisation depuis un fichier d'interface
 - `-encodeWithCoder:` sérialisation si nécessaire
 - `-dealloc` contrôle de la destruction de l'objet