

# Logiciels embarqués ambiants/iOS

## Chapitre 4 : User Inputs

Dr. Abdelkader Gouaïch<sup>1</sup>

<sup>1</sup>Department of Computer Science  
Université de Montpellier

2012

# Outline

- 1 Introduction
  - Objectifs du chapitre
- 2 Notification Center
- 3 Interface graphique avec OpenGL ES
  - Le rendu de l'interface graphique
  - Définir les zones
- 4 Interface graphique avec UIKit
  - Le pattern design MVC
  - Création d'interface avec Xcode
- 5 Travaux Pratiques

# Outline

- 1 Introduction
  - Objectifs du chapitre
- 2 Notification Center
- 3 Interface graphique avec OpenGL ES
  - Le rendu de l'interface graphique
  - Définir les zones
- 4 Interface graphique avec UIKit
  - Le pattern design MVC
  - Création d'interface avec Xcode
- 5 Travaux Pratiques

# Objectifs de ce chapitre

- Interface utilisateur
- Interface utilisateur directement avec OpenGL ES
- Interface utilisateur avec UIKit

# Résumé de la séance précédente

- Gestion des inputs de l'utilisateur
- Touches sur l'écran
- Accléromètre

# Notification Center

- Le service de notification permet de s'enregistrer comme listener avec un sélecteur et de répondre automatiquement à certains événements
- Il offre également la possibilité de poster des notifications/événements

# S'enregistrer dans le notification center

```
[[NSNotificationCenter defaultCenter] addObserver:self // observateur  
selector:@selector(maMethode:) // le selecteur de l objet qui sera appele  
name:@"NomDeEvenement" // Nom de la notification  
object:nil //objet source. Si nil il ne sera pas utilise  
]
```

# Poster des notifications

- (void)postNotification:(NSNotification \*)notification
- (void)postNotificationName:(NSString \*)notificationName  
object:(id)notificationSender



- Nous allons construire les interfaces graphiques à la main
- Il suffit simplement de bien définir les éléments graphiques à dessiner
- Il faut définir les zones d'interaction
- et comment réagir aux actions

# Outline

- 1 Introduction
  - Objectifs du chapitre
- 2 Notification Center
- 3 Interface graphique avec OpenGL ES
  - Le rendu de l'interface graphique
  - Définir les zones
- 4 Interface graphique avec UIKit
  - Le pattern design MVC
  - Création d'interface avec Xcode
- 5 Travaux Pratiques

- La construction d'une GUI en mode OpenGL ES n'est pas différente de la construction d'une scène normale de jeu
- Il nous faut donc créer une scène pour représenter un menu (par exemple la scène d'introduction du jeu)

- Nous allons donc simplement créer une scène (MenuScene) qui hérite de la classe AbstractScene
- Nous allons définir la méthode renderScene qui va dessiner les éléments du menu (images)

# Outline

- 1 Introduction
  - Objectifs du chapitre
- 2 Notification Center
- 3 Interface graphique avec OpenGL ES
  - Le rendu de l'interface graphique
  - Définir les zones
- 4 Interface graphique avec UIKit
  - Le pattern design MVC
  - Création d'interface avec Xcode
- 5 Travaux Pratiques

# Définir les zones d'interaction

- Le plus simple est de définir des zones (rectangles) qui vont contenir les images (boutons)
- Nous savons comment récupérer l'événement touch (voir le chapitre sur User Input)
- Il nous reste que vérifier si l'événement touch se trouve à l'intérieur de notre zone d'interaction

# Exemples pour simuler une image-button

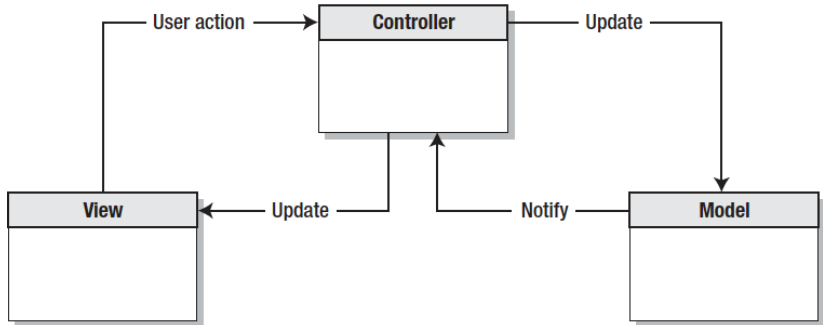
```
CGRect buttonBounds;  
buttonBounds = CGRectMake(0,0,10,20);  
// ...  
UITouch *touch = [[event touchesForView:aView] anyObject]  
if (CGRectContainsPoint(buttonBounds, touch))  
{ // traitement ici  
  
}
```

# Outline

- 1 Introduction
  - Objectifs du chapitre
- 2 Notification Center
- 3 Interface graphique avec OpenGL ES
  - Le rendu de l'interface graphique
  - Définir les zones
- 4 Interface graphique avec UIKit
  - Le pattern design MVC
  - Création d'interface avec Xcode
- 5 Travaux Pratiques



# MVC



# Outline

- 1 Introduction
  - Objectifs du chapitre
- 2 Notification Center
- 3 Interface graphique avec OpenGL ES
  - Le rendu de l'interface graphique
  - Définir les zones
- 4 Interface graphique avec UIKit
  - Le pattern design MVC
  - Création d'interface avec Xcode
- 5 Travaux Pratiques

# UIKit

- Le controleur hérite de UIViewController
- La vue hérite de UIView

# Interface Builder

- Interface Builder est un outil qui va nous permettre de créer facilement des interfaces UIKit
- L'outil est assez simple et intuitif
- Il produit des fichiers .xib
- L'interface (qui est la vue) sera composée de composants offerts par le framework UIKit

# IBOutlet et IBAction

- Vous pouvez lier les éléments de la vue avec les ivars (propriétés) d'un controleur
- Pour informer l'interface builder qu'une ivar peut être liée avec la vue il faut utiliser la macro IBOutlet
- Pour appeler une méthode directement depuis la vue (faire une action), il faut utiliser la macro IBAction dans la déclaration d'une méthode du controleur

# IBAction/IBOutlet

- Ces deux macro ne sont utilisées que pour donner des informations à l'IB
- Voici leurs vraies définitions
- `#define IBAction void`
- `#define IBOutlet`

# Hiérarchie de vues

- Pour afficher une vue, il suffit de la rajouter comme subview de la vue principale
- Exemple :

```
[sharedGameController.eaglView addSubview:self.view];
```

# Objectif 1 du TP

- Télécharger le fichier SLQTSOR.zip
- C'est un jeu complet qui utilise le moteur que nous avons présenté
- Regarder dans le détails les fichiers suivants
  - MenuScene
  - GameController
  - Toutes les classes dans le répertoire Classes/Views
  - Les fichiers .xib dans Resources



# Objectif du TP

- Suivre les étapes pour la construction d'une application simple

`https://developer.apple.com/library/ios/  
#DOCUMENTATION/iPhone/Conceptual/iPhone101/  
Articles/01\_CreatingProject.html`