Journée Découverte du Métier d'Ingénieur

# Création d'une application iPhone

Association EpiMac


# Table des matières

# 1 Introduction

Le but de cette séance de travaux pratique est de vous permettre de réaliser une application iPhone : le jeu du Pierre/Feuille/Ciseaux. Nous utiliserons, pour créer notre application, le logiciel XCode. C'est un IDE. Il va entre autre vous permettre d'écrire le code de votre application puis de "compiler" votre projet. Nous utiliserons le langage Swift.

Langage de programmation : Un langage de programmation est un langage permettant destinée à formuler des algorithmes et produire des programmes informatiques qui les appliquent. Une fois écrit, le texte produit est analysé et traduit dans un langage compréhensible par la machine ("assembleur").

Compiler : Travail réalisé par un compilateur qui consiste à transformer un code source lisible par un humain en un programme lisible par une machine.

IDE : ("Integrated Development Environment" - Environnement de développement intégré), il vous permet de développer de manière plus intuitive qu'un simple éditeur de texte comme bloc note.

Swift : Swift, c'est un langage de programmation aussi fiable qu'intuitif mis au point par Apple pour développer des apps iOS, Mac, Apple TV et Apple Watch, créé en 2014.

# 2 Cours

## 2.1 Pour commencer

Pour pouvoir coder ce projet, nous ne vous lâchons pas directement dans le grand bain... Vous allez ici compléter un projet existant dans lequel nous avons déjà laissé des objets et des variables. Il ne vous restera plus qu'à coder 3 fonctions dont nous allons vous parler plus loin.

Pour commencer, ouvrez le logiciel "XCode" qui se trouve dans votre Dock (le lanceur d'application, situé dans la partie inférieure de votre écran)

Figure 1 – Logo de l'application XCode

Maintenant il ne vous reste plus qu'à ouvrir le projet déjà existant. Pour ce faire, cliquez sur "Open Another Project..." en bas à droite de la fenêtre.

Une boîte de dialogue va ensuite s'ouvrir pour vous proposer de choisir l'emplacement du dossier. Une fois trouvé, sélectionnez le dossier, sans entrer dedans, puis appuyez sur "Open" en bas à droite de la boîte de dialogue.


Figure 2 – Fenêtre de démarrage de XCode

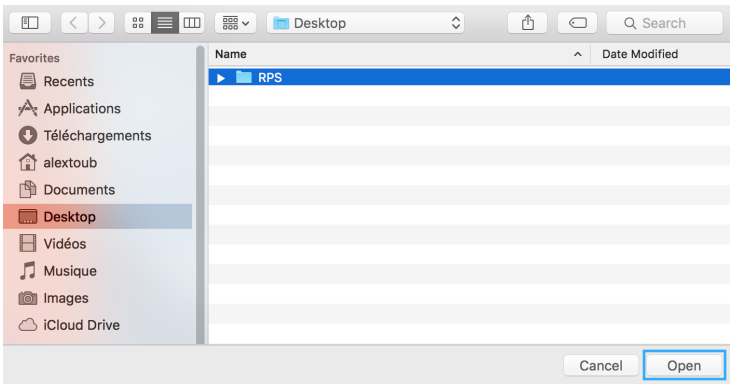


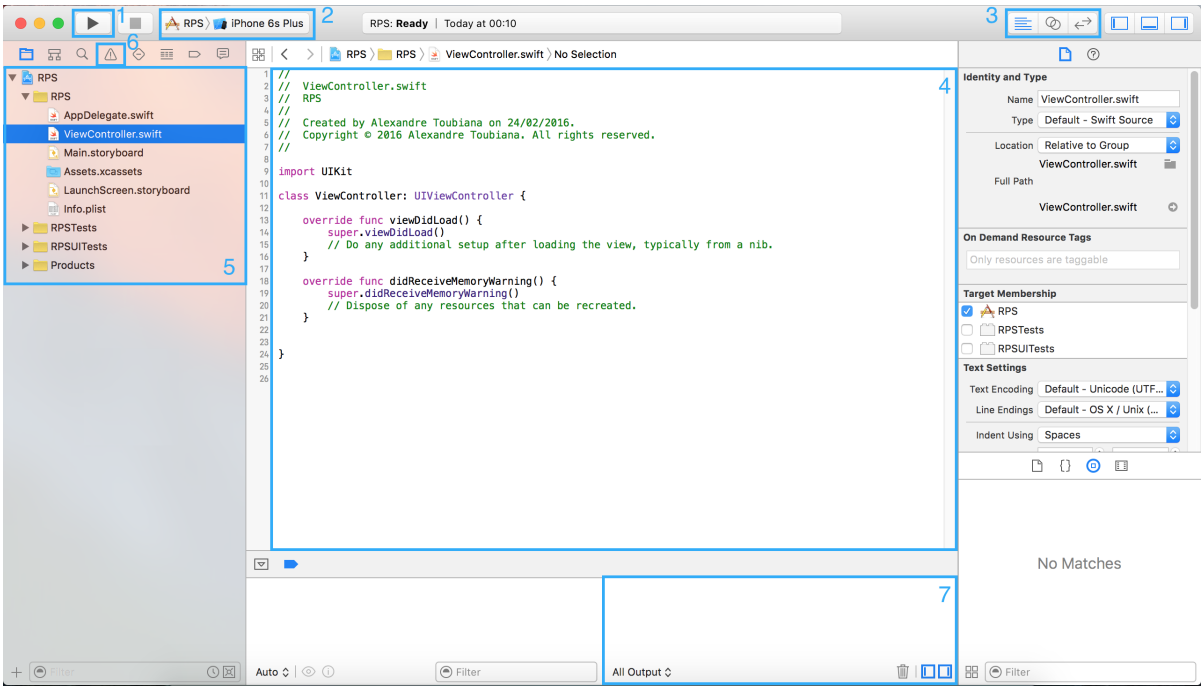

Figure 3 – Boîte de dialogue pour indiquer le dossier que l'on souhaite ouvrir

## 2.2 L'interface XCode

XCode est un logiciel très puissant et qui permet de faire énormément de chose dans le développement Apple, il est donc important de connaitre son fonctionnement. Pour ce TP, nous n'allons pas vous apprendre à vous servir entièrement du logiciel car beaucoup de choses vous sont inutiles pour l'instant.

Nous allons ici vous expliquer l'utilité de quelques parties de l'interface qui sont importantes aujourd'hui.

Tout d'abord, voici l'interface sur laquelle vous vous trouvez lorsque vous allez coder.




Nous avons numéroté les parties importantes et voici leur utilité :

1 - Bouton permettant de compiler le projet.

2 - Choix du simulateur dans lequel nous voulons exécuter l'application. Par défaut nous pouvons choisir la plupart des derniers terminaux iOS. L'application sera ensuite installée dans un simulateur de terminal.

3 - Choix de la manière dans laquelle nous souhaitons s'organiser pour coder, de base nous avons seulement le code et nous pouvons choisir d'afficher deux fichiers simultanément en cliquant sur le bouton du milieu.

4 - Affichage du fichier que nous traitons.

5 - Affichage de l'arborescence des fichiers qui composent le projet.

6 - Bouton permettant de voir toutes les erreurs qui font échouer la compilation de l'application.

7 - Affichage de la console (ce qui se passe en sortie de l'application). Il sera affiché ici tout ce que l'application écrit. Pour être plus clair, si dans le code vous écrivez par exemple la fonction `print("EpiMac")`, le mot "EpiMac" sera écrit dans la console et non dans l'application. Cela permet entre autres de vérifier si notre application fonctionne comme elle devrait.

## 2.3 Principes de bases de la programmation

### 2.3.1 Les types

Le but de l'informatique et de la programmation est le traitement des données et de l'information. Les données peuvent être des nombres, du texte ou encore des images, du son, etc...

Les types de données courant sont Int (les nombres) et String (chaine de caractères = texte)

Il existe un type appelé booléen qui peut avoir 2 valeurs : vrai ou faux, c'est en fait représenté sous la forme d'un entier qui peut prendre deux valeurs.

### 2.3.2 Les variables

Pour simplifier ce traitement de la donnée, on utilise ce que l'on appelle les variables. Une variable permet de stocker une donnée et de lui donner un nom. Voici comment créer une variable :

```
var jeSuisUneVariable = "Je suis un texte"
```

Ici on associe le nom "jeSuisUneVariable" à la donnée (du texte en l'occurence) "Je suis un texte". Il y a des rêgles à respecter lorsque l'on nomme une variable :

```
var ca ne marche pas = "je ne marche pas" // Pas d'espaces
var 1337foo = "toujours pas..."           // On ne commence pas par un chiffre
```

Une fois la variable créée. on peut lui associer de nouvelles données :

```
var jeSuisUneVariable = "Salut"
jeSuisUneVariable = "Hi"       // Ici la variable jeSuisUneVariable contient "Hi"
```

### 2.3.3 Les structures conditionelles

Maintenant, on aimerait interargir avec les données. Par exemple, on aimerait faire certaines choses si une condition est rempli. Imaginons que l'on ait une variable appelée inconnu. On ne connait pas sa valeur mais si elle est supérieur à 10, on veut faire une certaine action. Pour sela on utilise if (si en français). Exemple :

```
var resultat = "42"
if inconnu > 10 {
  resultat = "Supérieur à 10"
}
else {
  resultat = "Inférieur ou égal à 10"
}
```

Mais que fait on si on à beaucoup de cas à gérer, on enchaine les if ? En réalité il existe une structure conditionnelle plus approprié dans le cas ou on doit faire un traitement spécifique en fonction de la valeur d'une variable (entier), cela s'appelle le switch :

```
switch x
{
  case 0:
    resultat = "la variable x est égal à 0"
  case 1:
    resultat = "la variable x est égal à 1"
  case 2:
    resultat = "la variable x est égal à 2"
  default:
    resultat = "la variable x est supérieur à 2"
}
```

### 2.3.4 Les fonctions

Une fonction comporte une séquence d'instructions réalisant un calcul ou une tâche. Elle peut, ou non, comporter des arguments sur lesquels elle va appliquer des traitements.

# 3 Développement

Il est maintenant temps de mettre en pratique ce que vous avez appris et commencer la réalisation de votre application ! Pour cela nous vous avons préparé une interface toute faite, à laquelle, il faudra coder quelques fonctions bien précises, puis laisser libre court à votre imagination quant au design de l'application.

Aller donc dans le fichier ViewController.swift, c'est ici que s'écriront le code qui va "contrôler" les différentes parties de l'application.

## 3.1 Initialiser les variables : SETUP

Cette fonction a pour but d'initialiser les variables avant le commencement du jeu. En effet il faut donner une valeur initiale si on ne veut pas que le programme fasse n'importe quoi au lancement (les scores à 0 par exemple). Pensez bien à toutes les variables qu'on doit initialiser.

```
// Fonction qui restaure les variables par default
func setup()
{
    /* ------------ */
    /* FIXME Setup */
    /* ------------ */
}
```

Indice : il y a 8 initialisation a faire. 4 liés au code pur et 4 liés à l'interface utilisateur (l'UI).

## 3.2 Définir le choix des joueurs : MAKECHOICE

Ici vous devez définir les différents choix des joueurs. Cette fonction se fait en 3 étapes : le choix du premier joueur, le choix du second et enfin on lance le calcule du resultat en fonction des deux choix.

```
// Fonction associée aux trois boutons "Pierre/Feuille/Ciseaux"
// Cette fonction permet de décider l'action du joueur.
@IBAction func makeChoice(sender: AnyObject)
{
    /* ---------------- */
    /* FIXME MakeChoice */
    /* ---------------- */
}
```

### 3.2.1 Choix du joueur "humain"

Il y a 3 choix différents (pierre, feuille ou ciseaux), la variable playerchoice peut donc prendre 3 valeurs différents, en fonction du bouton sur lequel l'utilisateur appuie. Servez-vous donc de la structure conditionnelle la plus appropriée.

### 3.2.2 Choix du jouer "ordinateur"

Ici nous allons simplement faire un appel à la fonction computerChoice, qui renvoie un entier entre 0 et 2.

### 3.2.3 Calcul du résultat

Une fois que vous avez les deux entiers nous pouvons appeler la fonction finality que vous devez implémenter par la suite.

### 3.3 Définirs les règles et le résultat des parties : FINALITY

Voici la dernière fonction : le calcul du résultat en fonction des choix des joueurs.

```
// Fonction qui définit le résultat en fonction
// du choix de l'ordinateur et du joueur.
func finality(PC: Int, CC: Int)
{
    /* ------------- */
    /* FIXME finality */
    /* ------------- */
}
```

Tout d'abord, la première chose à faire est d'afficher l'image correspondant au choix de l'adversaire. Pour cela on va utiliser la même structure conditionnelle que précèdemment et afficher l'image de la manière suivante :

```
computerChoiceImage.image = UIImage(named: "Rock")
```

Ensuite on va donc comparer les choix des deux joueurs afin d'afficher le résultat et d'augmenter le score des joueurs le cas écheant.

Et voila le partie code est terminée ! Vous pouvez maintenant compiler et vérifier que tout marche grâce au simulateur iPhone. Cependant comme vous pouvez le constater c'est assez moche ...
Il va donc falloir à présent personnaliser l'interface !

## 4 Travailler le Design de l'application

Avant toute chose il faut définir le design de l'application lors de son lancement, pourquoi pas mettre quelques couleurs ?

Rendez dans le StoryBoard approprié : LaunchScreen.storyboard

Une fois ceci fait, allez pouvoir personnaliser le design de l'application, pour cela ouvrez le fichier Main.storyboard

Voici une liste, non exaustif, des choses à faire :
— changement du titre
— choix des couleurs
— changement des images
— affichage gagné/perdu en vert/rouge

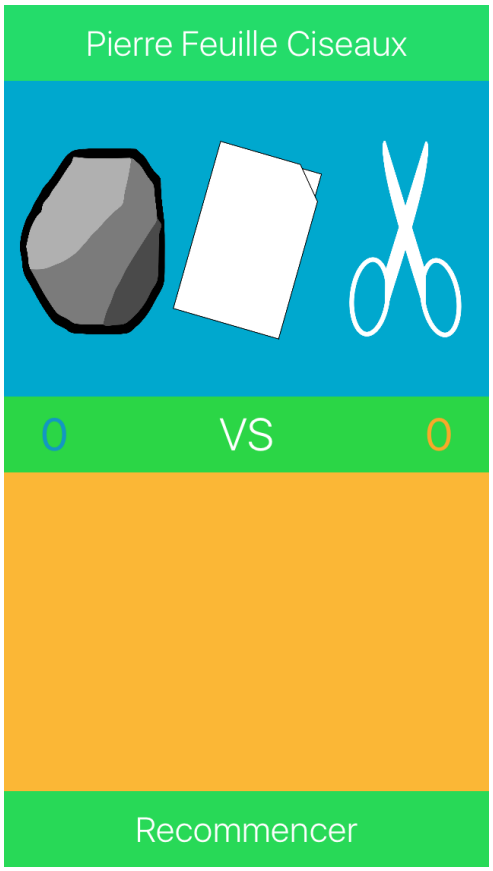



Figure 4 – Exemple de Rendu possible de l'application

## 5 Annexes

Epimac est l'association d'Epita qui regroupe tout l'univers Apple. Elle organise des conférences d'apprentissage au développement mobile mais aussi des ateliers pour tous les niveaux. Elle propose également des offres promotionnelles sur les produits Apple. Pour nous contacter, rendez vous sur notre site http ://epimac.org ou envoyez un mail à contact@epimac.org.

Pour récupérer le code source de l'application vous pouvez aller sur GitHub afin de "clone" le projet ou le télécharger (https ://github.com/EpiMac).

Si vous désirez aller plus loin et apprendre les bases du développement iOS, il y a de très bon tutorial sur internet.

Vous pouvez également nous contacter ou passer nous voir au local nous serons heureux de vous aider :)