



Piscine iOS Swift - Day 03

APM

Maxime LEMORT mlemort@student.42.fr
42 Staff pedago@42.fr

Résumé: Ce document contient le sujet du Day 03 de la piscine iOS Swift de [42](#)

Table des matières

I	Préambule	2
II	Consignes	4
III	Introduction	5
IV	Exercice 00 : Photos	6
V	Exercice 01 : Multithreads	7
VI	Exercice 02 : Alertes	8
VII	Exercice 03 : ScrollView	9
VIII	Exercice 04 : Zoom	10

Chapitre I

Préambule

Voici un extrait de la page wikipedia de Hubble :



Le télescope spatial Hubble (en anglais Hubble Space Telescope, en abrégé HST) est un télescope spatial développé par la NASA avec une participation de l'Agence spatiale européenne qui est opérationnel depuis 1990. Son miroir de grande taille (2,4 mètres de diamètre), qui lui permet de restituer des images avec une résolution angulaire inférieure à 0,1 seconde d'arc ainsi que sa capacité à observer à l'aide d'imageurs et de spectroscopes dans l'infrarouge proche et l'ultraviolet lui permettent de surclasser pour de nombreux types d'observation les instruments au sol les plus puissants handicapés par la présence

de l'atmosphère terrestre. Les données collectées par Hubble ont contribué à des découvertes de grande portée dans le domaine de l'astrophysique telles que la mesure du taux d'expansion de l'Univers, la confirmation de la présence de trous noirs supermassifs au centre des galaxies ou l'existence de la matière noire et de l'énergie noire.

Le développement du télescope Hubble, qui tient son nom de l'astronome Edwin Hubble, démarre au début des années 1970 mais des problèmes de financement, de mise au point technique et la destruction de la navette spatiale Challenger repoussent son lancement jusqu'en 1990. Une aberration optique particulièrement grave est découverte peu après qu'il a été placé sur son orbite terrestre basse à 600 km d'altitude. Dès le départ le télescope spatial avait été conçu pour permettre des opérations de maintenance par des missions des navettes spatiales. La première de ces missions en 1993 est mise à profit pour corriger l'anomalie de sa partie optique. Quatre autres missions, en 1997, 1999, 2002 et 2009, permettent de moderniser les cinq instruments scientifiques et remplacer certains équipements défaillants ou devenus obsolètes. La dernière mission de maintenance, réalisée en 2009, immédiatement avant le retrait définitif des navettes spatiales, doit permettre au télescope Hubble de fonctionner jusqu'à la fin de la décennie 2010, sauf imprévu. Pour les observations dans l'infrarouge il doit être remplacé vers 2018 par le télescope spatial James-Webb, aux capacités supérieures.

Chapitre II

Consignes

- Seule cette page servira de référence : ne vous fiez pas aux bruits de couloir.
- Lisez attentivement l'integralité du sujet avant de commencer.
- Le sujet peut changer jusqu'à une heure avant le rendu.
- Vos exercices seront corrigés par vos camarades de piscine.
- Le sujet fait foi, ne vous fiez pas toujours à la lettre aux demos qui peuvent contenir des ajouts supplémentaires non demandés.
- Vous devrez rendre une app par jour (sauf pour le Day 01) sur votre depot git, rendez le dossier du projet Xcode.
- Voici le manuel officiel de [Swift](#)
- Voici le manuel officiel de [Swift Standard Library](#)
- Il est interdit d'utiliser d'autres librairies, packages, pods... avant le Day 07
- Vous avez une question ? Demandez à votre voisin de droite. Sinon, essayez avec votre voisin de gauche.
- Pensez à discuter sur le forum Piscine de votre Intra !
- Lisez attentivement les exemples. Ils pourraient bien requérir des choses qui ne sont pas autrement précisées dans le sujet...
- Réfléchissez. Par pitié, par Odin ! Nom d'une pipe.



L'intra indique la date et l'heure de fermeture de vos dépôts.
Cette date et heure correspond également au début de la période de peer-evaluation pour le jour de piscine correspondant. Cette période de peer-evaluation dure exactement 24h. Une fois ces 24h passées, vos notes peer manquantes seront complétées par des 0.

Chapitre III

Introduction

Les **threads** ou *fil d'exécution* permettent d'effectuer les instructions d'un processus en suivant leurs propres pile d'appel. Au départ un processus se lance sur un seul thread, le **main thread**.

L'utilisation de plusieurs threads permet de paralléliser le traitement de plusieurs fonctions pour faire tourner du code en tâche de fond. Ce point est extrêmement important sur iOS pour éviter de bloquer l'interface utilisateur (UI) pendant que l'application fait des calculs ou attend la réponse d'un serveur.

Aujourd'hui vous allez voir plusieurs notions :

- Comment utiliser une **collection view**
- Comment faire du **multithread** sur iOS
- Comment faire des **alertes**
- Comment utiliser une **scroll view**

Tous cela dans une application qui téléchargera des images depuis le net.

Chapitre IV

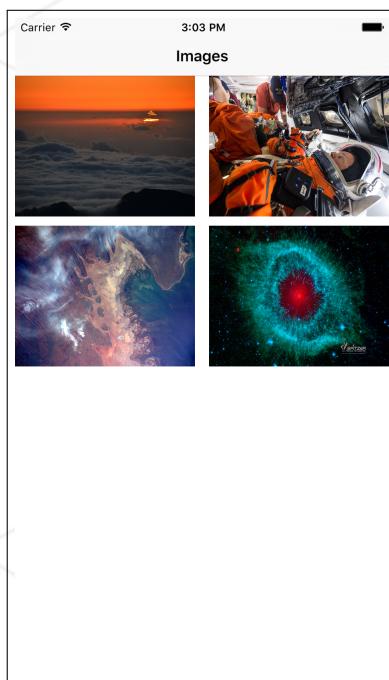
Exercice 00 : Photos

	Exercice : 00
	Photos
	Fichiers à rendre : .xcodeproj et tous les fichiers nécessaires
	Fonctions Autorisées : Swift Standard Library, UIKit
	Remarques : n/a

La **collection view** est un outil qui permet d'afficher des données différemment d'une **table view** mais leur utilisation est quasiment identique.

Créez une **collection view** qui affiche au moins 4 photos du web de votre choix. Les 4 photos doivent être affichées en entier dans la **collection view**.

Prenez des photos lourdes pour que les téléchargements prennent du temps. Vous pouvez chercher des photos sur le site de la [nasa](#) par exemple.



Chapitre V

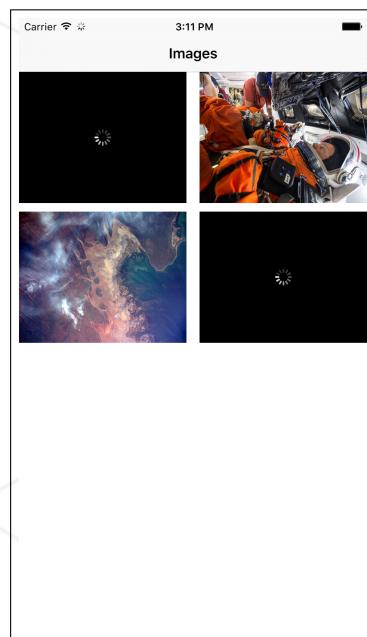
Exercice 01 : Multithreads

	Exercice : 01
	Multithreads
	Fichiers à rendre : .xcodeproj et tous les fichiers nécessaires
	Fonctions Autorisées : Swift Standard Library, UIKit
	Remarques : n/a

Vous avez remarqué que le temps que les images se téléchargent, l'UI est bloquée et iOS ne répond pas. Les appels sur le **main thread** nuisent à l'expérience utilisateur. Pour pallier à ce problème vous allez rendre ces appels asynchrones.

Ajoutez aussi un **activity monitor** sur chaque vue de la **collection view** qui doit tourner lorsque l'image se télécharge et disparaître lorsque l'image s'affiche.

Vous devez aussi faire tourner le **network activity indicator** lorsque l'application utilise le réseau et l'arrêter lorsqu'elle ne l'utilise plus.

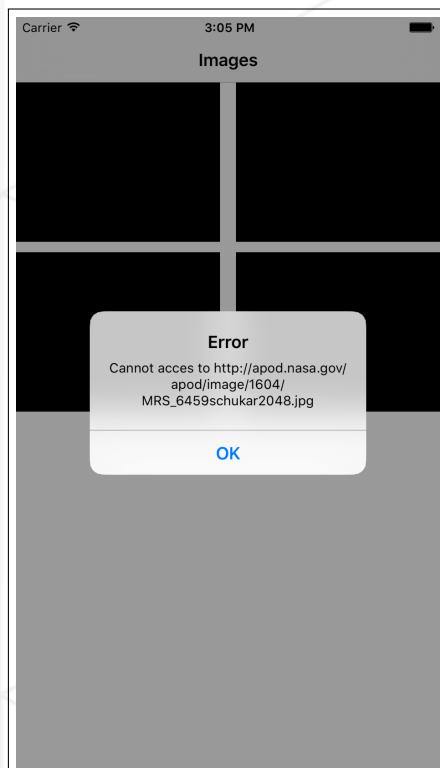


Chapitre VI

Exercice 02 : Alertes

	Exercice : 02
	Alertes
	Fichiers à rendre : .xcodeproj et tous les fichiers nécessaires
	Fonctions Autorisées : Swift Standard Library, UIKit
	Remarques : n/a

S'il y a un problème pendant le téléchargement de la photo, vous devez faire apparaître une **alerte** simple qui explique le problème avec un bouton "Ok" pour faire disparaître l'**alerte**.



Chapitre VII

Exercice 03 : ScrollView

	Exercice : 03
	ScrollView
	Fichiers à rendre : .xcodeproj et tous les fichiers nécessaires
	Fonctions Autorisées : Swift Standard Library, UIKit
	Remarques : n/a

Ajoutez une **navigation bar** avec un titre pour chaque vue.

Créer une nouvelle vue contenant une **scroll view**. Lorsqu'on click sur une cellule de la **collection view**, vous devrez afficher la **scroll view** avec l'image en grand. On doit pouvoir déplacer l'image.



Chapitre VIII

Exercice 04 : Zoom

	Exercice : 04
	Zoom
	Fichiers à rendre : .xcodeproj et tous les fichiers nécessaires
	Fonctions Autorisées : Swift Standard Library, UIKit
	Remarques : n/a

Déplacer l'image c'est bien, pouvoir zoomer c'est mieux. Faites en sorte que l'on puisse zoomer et dézoomer sur l'image.

Il faut aussi que l'image tienne parfaitement en largeur avec le dézoom maximal, et ce quelque soit le device et son orientation !

