Cours n°7 — Médias, graphismes et localisation

Médias et graphismes

Une application peut nécessiter l'affichage ou l'acquisition de médias.

iOS propose une fonctionnalité pour organiser les ressources d'une application plus proprement et qui s'adaptent à l'appareil sur lequel se trouve l'application.

De plus, de nombreux frameworks permettent de gérer des éléments graphiques à un niveau plus ou moins poussé selon les besoins, qu'il s'agisse de graphismes 3D ou de traitement de fichiers PDF par exemple.

Ressources

Pour fournir des ressources statiques dans une application, il est évidemment possible d'inclure des fichiers standards comme vu dans le cours n°5 sur les données locales avec l'exemple d'un fichier représentant une base SQLite 3.

Cependant, pour s'adapter correctement à l'environnement d'exécution, par défaut, une application possède un « Assets catalog » permettant de stocker les ressources de façon à les rendre adaptées à l'appareil sur lequel elles doivent être utilisées.

Le catalogue peut contenir des sets d'images, des icônes pour l'application, des set de textures ou des sets de couleurs. L'avantage est que chaque set peut contenir un élément adapté pour un appareil ou un environnement en particulier. Il est possible de configurer les éléments par appareil, thème (clair ou sombre), langue et densité de pixels dans le cas des images. Le catalogue par défaut se trouve dans le dossier `Assets.xcassets`.

Comme indiqué en diapositive 5, pour ajouter des éléments il suffit de cliquer sur le plus (+) situé en bas à gauche du panneau central de Xcode quand le catalogue est ouvert.

Une fois l'élément créé, il est possible d'indiquer les variations possibles du set en ouvrant l'inspecteur d'attributs dans le panneau droit de Xcode. (c.f. diapositive 6)

Il est ensuite possible, comme d'habitude, d'utiliser les éléments du catalogue via les storyboards ou via du code. Côté code, il faudra utiliser la classe appropriée pour charger l'élément voulu, en s'assurant de lui avoir donné un identifiant au préalable (exemple en diapositive 7 pour charger une image).

Simple point de « culture » : en ce qui concerne l'icône de l'application, plusieurs conseils sont donnés pour la conception de l'icône de votre application en diapositive 8 ; cette partie de la documentation donne de bons conseils généraux sur la conception visuelle d'applications iOS.

À l'instar des sets d'images, il existe également des sets de couleurs qui peuvent être configurées selon l'environnement et utilisés ou dans les storyboards ou côté code une fois qu'ils possèdent un identifiant. (c.f. diapositives 9 et 10)

Graphismes

Différents frameworks existent dans le SDK iOS pour la gestion de graphismes, allant de simples graphismes en 2D (Core Graphics), au rendu d'éléments 3D (SceneKit) en passant par la réalité augmentée (ARKit).

Core Graphics est évidemment un des plus simples à utiliser et permet, entre autres, la création d'images par le biais de tracés, à la manière de l'élément canvas dans les technologies web, mais également la création de fichiers PDF ou la gestion d'ombres ou dégradés sur les objets.

La diapositive 13 présente un code permettant de réaliser un losange. Comme avec le Canvas en HTML, on place dans un premier temps le « crayon » à l'origine puis on réalise des tracés en le déplaçant. Il faut simplement veiller à fermer le tracé avec la méthode `close` et à définir la couleur de remplissage (*fill*) et de bordure (*stroke*), chacune étant optionnelle.

Il est possible d'imaginer utiliser une vue personnalisée en redéfinissant sa méthode `draw` pour créer un élément visuel réutilisable. (c.f. diapositive 14, le commentaire avec les trois points désigne le code de la diapositive précédente)

Médias

A l'instar des graphismes, la gestion des médias est assurée par plusieurs frameworks présents dans le SDK. Les deux frameworks principalement utilisés sont AVFoundation et PhotoKit permettant respectivement de gérer les éléments audio-visuelles de l'application et de gérer la bibliothèque photos et vidéos de l'utilisateur.

Image picker controller

En ce qui concerne la prise de photos ou vidéos, un contrôleur par défaut est disponible. Il est nullement personnalisable mais permet d'ajouter très rapidement à une application la capacité de prendre des photos et vidéos en ayant une interface standard.

On retrouve un exemple d'utilisation du contrôleur en diapositive 18. Dans la méthode `viewDidLoad`, on vérifie dans un premier temps si un appareil photo est

disponible ; s'il ne l'est pas, le bouton (préalablement créé par nos soins dans le storyboard) permettant de lancer l'appareil photo est caché. Sinon, on associe l'appel de la méthode permettant d'afficher le contrôleur (`prendrePhoto` ici) au clique sur ce bouton.

La méthode `prendrePhoto` instancie le contrôleur et utilise la méthode `present` pour l'afficher.

Cette méthode définit également l'attribut `delegate` du contrôleur. La compréhension de cette partie n'est pas d'une grande importance ; il s'agit juste de préciser quel objet va recevoir la photo une fois celle-ci prise (ici se sera `self`, donc notre contrôleur).

Cet objet en question doit suivre deux protocoles particuliers ; on retrouve l'implémentation de ces protocoles pour notre contrôleur à la diapositive 19. Ici, la gestion de la photo consiste simplement à cacher le contrôleur gérant l'appareil photo (avec `dismiss`) et de fournir la photo qui vient d'être prise à une vue (`imageView` ici, elle aussi ajoutée par nos soins dans le storyboard).

AVFoundation

Le contrôleur fournit par défaut est très pratique mais pour pouvoir proposer une interface personnalisée ou réaliser des traitements sur l'image rendue, il faut faire appel à AVFoundation. Son utilisation ne se limite pas à la prise de photos ou vidéos mais il ne sera question que de ça ici.

Dans le cadre de la prise de photos ou vidéos, l'élément centrale est la session. C'est la session à laquelle il faut connecter les entrées (un micro, une caméra) et les sorties (un fichier ou un objet traitant des méta-données). (c.f. diapositive 21 à 23)

Après avoir designé comme souhaité une interface côté storyboard, il est relativement aisé d'implémenter la prise de photos ou vidéos côté code. Si l'on part du principe que notre scène de storyboard contient un bouton et une vue basique qui va se charger d'accueillir le rendu de l'appareil photo, on retrouve des diapositives 24 à 26 le code pour réaliser une capture de vidéo. La propriété `started` permet de savoir si l'on doit démarrer ou stopper l'enregistrement sachant qu'ici le même bouton permet de réaliser les deux actions.

Le framework AVFoundation va permettre également de jouer de l'audio ou de la vidéo, ce qui peut être pratique une fois que l'on a réalisé une prise par exemple. Cependant, comme dans l'exemple précédent, la réalisation de l'interface est à la charge du développeur. (c.f. diapositives 27 à 29)

Localisation

Lorsqu'il s'agit de fournir une application dans plusieurs langues, plusieurs processus peuvent intervenir. Comme définies en diapositive 32, l'internationalisation et la localisation sont les deux principaux.

Il est effectivement possible de traduire dans différentes langues les éléments d'une application. Tout d'abord, il est possible de traduire les storyboards comme exposé dans les diapositives 32 à 34. Les ressources du catalogue peuvent être localisées ce qui peut s'avérer utile pour des images contenant du texte par exemple. (c.f. diapositive 36)

Finalement, comme il est possible de créer des vues dynamiquement, certaines chaînes de caractères peuvent nécessiter d'être affichées ou générées dynamiquement aussi. Pour cela, il faut créer un fichier `Localizable.strings`, le décliner selon les langues comme avec les storyboards en diapositive 36 et créer des clés (qui seront les mêmes quelque soit la langue) et leur associer la traduction souhaitée. (c.f. diapositives 38 à 40)

Pour faciliter les tests durant le développement, il est possible de définir la langue par défaut de l'application (qui est normalement celle de l'utilisateur) ; cela peut s'avérer pratique pour ne pas avoir à changer la langue sur chacun des simulateurs utilisés. (c.f. diapositives 41 et 42)