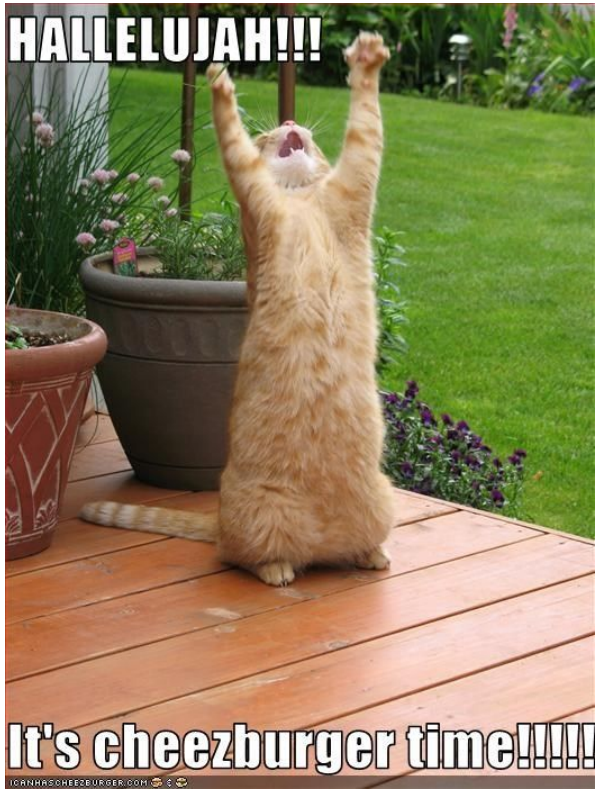


Cours 12

Plugins Android

- Unity donne un traitement particulier aux fichiers trouvés dans le répertoire **`Plugins/Android`** et ses sous-répertoires.
- La recherche dans les sous-répertoires est effectuée puisque, traditionnellement, les plugins Android étaient des projets ADT qui étaient installés dans des sous-répertoires de **`Plugins/Android`** et pré-compilés à l'aide de ADT. Unity recherchait alors ces répertoires pour les fichiers résultants.

- Avec Android Studio et l'invention du format AAR (**A**ndroid **AR**chive), il est maintenant préférable de garder les projets de vos plugins ailleurs et d'uniquement ajouter le fichier AAR résultant dans le répertoire **`Plugins/Android`**.
- Le format AAR est un ZIP contenant tous les fichiers nécessaires pour inclure un module Android dans un autre projet.

- Fichiers et répertoires reconnus par Unity:
  - **`AndroidManifest.xml`** : dans **`Plugins/Android`** et tous les sous-répertoires
  - **`*.jar`** : dans **`Plugins/Android`** et tous les sous-répertoires nommés **`bin`** ou **`libs`**
  - **`*.so`** : doivent être dans un sous-répertoire **`libs/armeabi-v7a`** ou **`libs/x86`**
  - **`assets/*`** : tout ce qui est présent dans un sous-répertoire **`assets`** se fait inclure directement dans votre APK
  - **`res/*`** : toutes les ressources présentes dans des sous-répertoires **`res`** de projets ADT

- **`*.aar`** : tous les fichiers AAR situés à la racine de **`Plugins/Android`**
- **`*.cs`** : dans **`Plugins/Android`** et tous les sous-répertoires

- Tout, à l'exception de fichiers C#, peut être installé sous forme de fichiers JAR ou AAR.
- D'ailleurs, un fichier AAR pourrait contenir tout ce qu'un fichier JAR contient, donc il y très peu de raisons d'utiliser autre chose que des fichiers AAR et C#.

- Lors de l’ajout de fichiers AAR et JAR, faites attention aux conflits de classes! Une classe ne peut être présente qu’une seule fois parmis tous les fichiers inclus par Unity.
- Par exemple, vous ne devriez pas inclure de bibliothèques JAR directement dans le plugin, puisque votre plugin ne pourrait alors pas être utilisé avec d’autres plugins avec la même bibliothèque.
- Les bibliothèques dont votre plugin nécessite devraient être installées séparément.

- Au résultat:
  - Tous les manifestes (incluant ceux dans AAR) seront combinés en un seul.
  - Tous les répertoires **`assets`** (incluant ceux dans AAR) seront combinés avec **`StreamingAssets`**.
  - Toutes les ressources seront incluses avec le nom de paquet du plugin contenant un répertoire **`res`** (incluant ceux dans AAR).
  - Toutes les classes dans des JAR et AAR seront combinées pour former le fichier **`classes.jar`** final du projet.
  - Toutes les bibliothèques SO pour **`x86`** ou **`armeabi-v7a`** seront ajoutées.

Cours 12

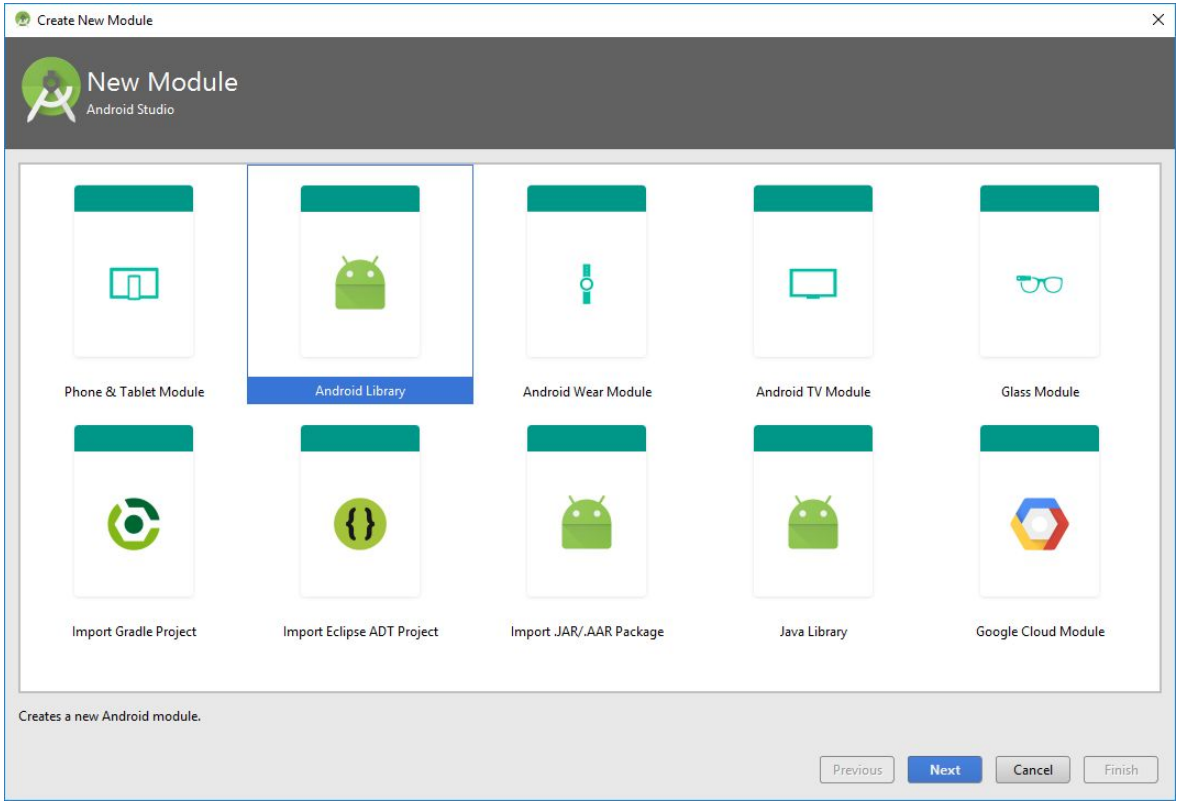
Création de Plugins

- La majorité de la création de plugin pour Android peut se faire à l’intérieur de Android Studio.
- Sauf si votre objectif est de publier un plugin réutilisable sur le Asset Store de Unity, je vous recommande de combiner tout votre code natif dans un seul plugin par projet. Ceci permet d’y inclure la majorité des dépendances et d'optimiser le résultat final.

- Premièrement, assurez-vous que votre projet Unity est situé dans un répertoire n'ayant aucun espace nul part dans son chemin. Certains outils, surtout lorsque le NDK est impliqué, ne supporte pas bien les espaces dans des noms de répertoires.
- Ensuite, vous allez créer un nouveau projet Android Studio à la racine de votre projet Unity (à côté du répertoire **`Assets`**) pour qu'il soit ignoré par Unity mais facilement ajouté aux sources de votre projet.

- Lors de la création de votre projet, Android Studio vous offre les options nécessaires à la création d'une application complète. Android Studio n'offre pas la possibilité de créer un projet AAR uniquement à partir de son écran de départ.
- Il est recommandé que ces informations soient les mêmes que votre projet Unity si vous voulez garder le module **`app`** que Android Studio créera (par exemple, pour facilement vous connecter avec Android Studio). Par contre, vous pouvez sauter la création d'activités.

- Une fois votre projet créé, vous pouvez maintenant créer des modules, et l'une des options permet la création de AAR.
- Faites **File > New > New Module** …



- L’option Android Library permet de créer un fichier AAR à l’intérieur de votre projet.

- ***ATTENTION!*** Le nom de paquet ne devrait pas être le même que votre projet Unity sinon vous aurez des conflits avec les ressources et la classe **`R`**! Chaque plugin devrait ici avoir son propre paquet.

- Le nom de votre module sera le nom du fichier de plugin, choisissez un nom simple et court!

- À partir de ce moment, vous aurez un projet d’application incluant un sous-module de bibliothèque Android. Vous pourriez effacer le module app si vous le désirez, mais il pourrait être utile pour débogage ultérieur.
- Pour que le module de votre plugin devienne un véritable plugin pour Unity, vous devrez effectuer quelques modifications au projet.

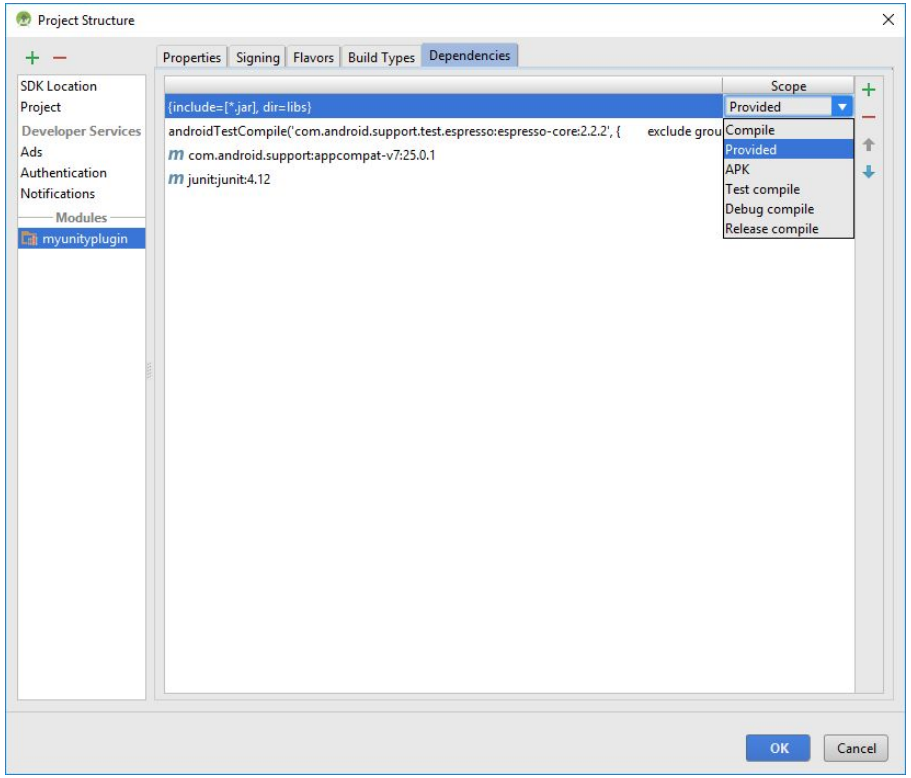
- Vous devez première y ajouter la bibliothèque de Unity. Dans l'installation de Unity, allez chercher le fichier suivant:

```
Editor/Data/PlaybackEngines/AndroidPlayer/Variations/mono/Release/Classes/classes.jar
```

- Naviguez vers le répertoire de votre projet, et placez ce fichier dans le répertoire **`libs`** de votre module de plugin (faites attention de ne pas le mettre dans le répertoire **`app`**!).

- Android Studio ajoute automatiquement tous les fichiers JAR trouvés dans le répertoire **`libs`**. Par contre, puisque Unity ne supporte pas avoir des classes dupliquées, vous devez changer ce comportement pour que les classes de Unity soient uniquement utilisées lors de la compilation mais exclues du fichier AAR.

- Retournez dans Android Studio, et allez dans le menu **File > Project Structure**.
- Allez dans votre module, et changez le mode d'inclusion du répertoire **`libs`** de *Compile* à *Provided*.



- Selon la documentation de Unity, vous devriez également aller dans l'onglet ***Options*** et y mettre ***Source*** et ***Target Compatibility*** à ***Java 1.6***.
- Votre projet devrait maintenant vous permettre de développer un plugin Unity sans problèmes.
- Toutes les classes que vous y ajoutez, ressources et entrées de manifeste, seront mises dans un fichier AAR que vous pouvez copier dans le répertoire **`Plugins/Android`** de Unity.

- Pour compiler votre plugin, vous pouvez accéder à Gradle à droite de la fenêtre de Android Studio, et y choisir la cible **`assembleDebug`** (ou **`assembleRelease`**) de votre plugin. Ceci créera un fichier AAR dans le répertoire **`build/outputs/aar`** du module de votre plugin. Vous n'avez qu'à copier ce fichier vers Unity. Faites attention! Vous ne pouvez pas y mettre à la fois la version **`debug`** et **`release`**, vous devez choisir une ou l'autre!