Introduction

Tout d'abord, il est à noter que l'application Android a été réalisée à l'aide du logiciel Android studio qui est le logiciel phare pour le développement sous Android. En effet, cet EDI (Environnement de Développement Intégré) créé par Google eux même est un must-have car il contient tous les éléments nécessaires à la réalisation d'une application Android qui sont :

- Éditeur de texte personnalisable avec l'auto complétion du code
- Compilateur pouvant aussi bien créer une version de débogage en reliant un téléphone à l'ordinateur, une version de débogage en utilisant un émulateur de téléphone tournant sous Android (peu recommandé car c'est extrêmement gourmand en ressources et soumis à des divers bugs et instabilités) ainsi qu'une version « release » exportable en .apk (extension pour installer l'application sur n'importe quel téléphone Android, après avoir préalablement autorisé l'installation des applications de source non vérifiées)
- Débuggeur pour pouvoir voir en temps réel les valeurs des variables, que ce soit en local avec un émulateur ou via un câble USB avec un vrai téléphone

Fichiers utiles

Le nombre de fichiers et de dossier peut paraître énorme au début mais il y a très peu de fichiers utiles au final, la plupart étant créés par Android Studio pour des raison m'étant inconnues. Je vais vous détailler ici les fichiers utiles et dont vous devrez peut-être modifier le contenu.

Spider drone.apk

Tout d'abord, vous trouverez, à la racine du dossier, un fichier nommé « Spider drone.apk » qui est utilisé pour installer l'application sur un nouvel appareil Android. Pour cette installation, il vous faut aller dans les paramètres -> sécurité (dans la catégorie « personnel » sur mon téléphone) -> autoriser l'installation d'applications issues de sources inconnues (dans la catégorie « gestion de l'appareil » sur mon téléphone).

Une fois cette autorisation effectuée, il vous faut installer l'apk, pour ce faire, vous pouvez déplacer ce fichier dans un dossier de la mémoire du téléphone, puis ouvrir ce fichier avec l'explorateur de fichier Android. Sinon, vous pouvez également utiliser des applications telles que airdroid pour installer à distance vos applications.

Une fois le fichier ouvert, que cela soit via l'explorateur ou airdroid, on vous demandera de confirmer l'autorisation de l'application d'accéder à certaines fonctionnalités du téléphone. Acceptez, patientez le temps que l'installation se finisse. Vous pourrez ensuite lancer l'application Android nommée « Spider Drone » avec le logo Android (pour la suite des explications, vous pouvez vous référer à la documentation utilisateur)

Spider_Drone.java

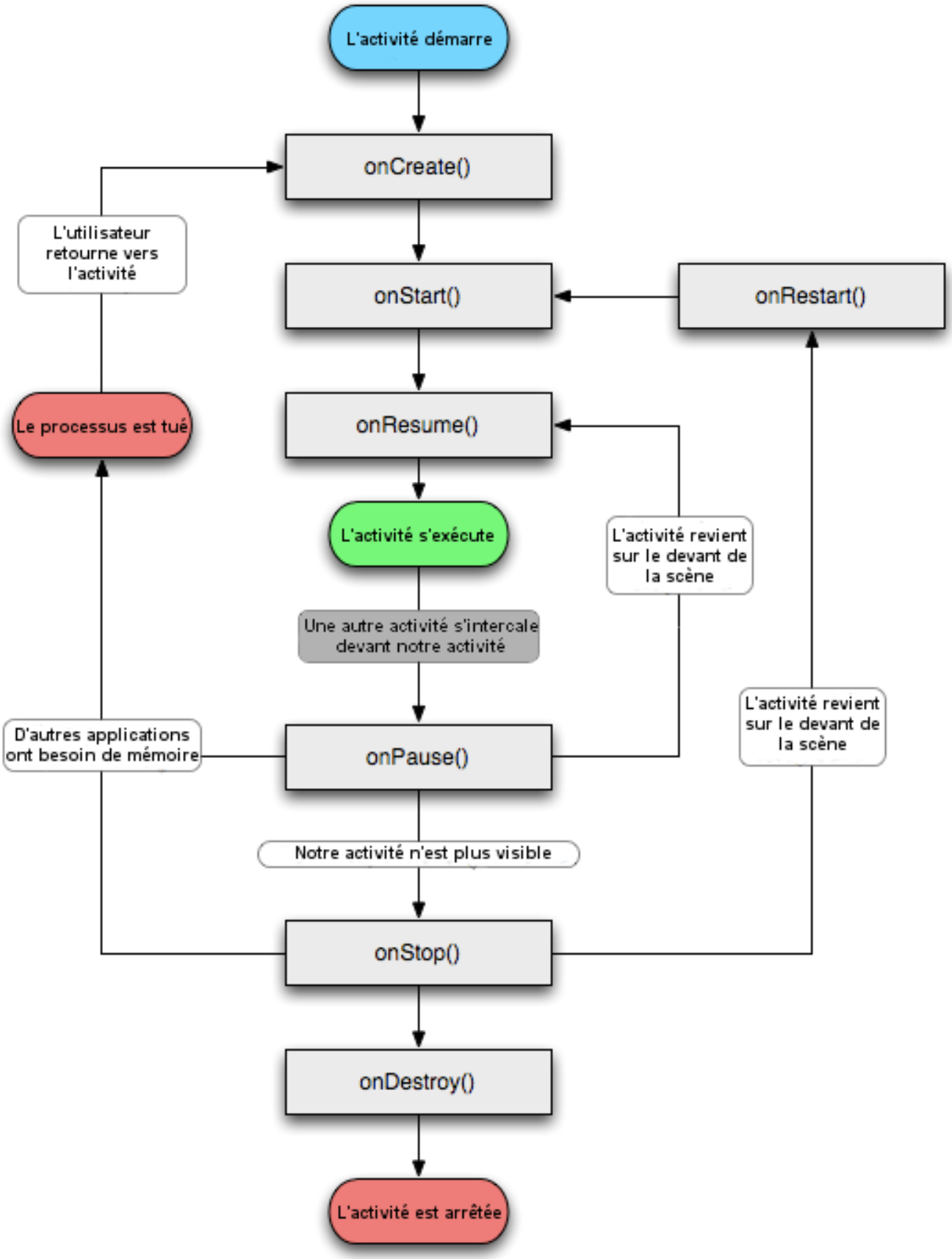
Vous trouverez ce fichier ici :

« SpiderDrone/app/src/main/java/fr/upmc/polytech/spiderdrone/Spider_Drone.java »

C'est le fichier contenant le code java servant à la réalisation de l'application. Au début, je fois tout une liste d'include nécessaires à l'application. Ensuite, tout le code se trouve dans une unique classe nommée « Spider_Drone » (mais cette classe contient différentes classes que je détaillerai plus tard). Ensuite, vous pouvez lire que j'initialise le port de communication pour les paquets UDP avec le port 2000 (qui est un port appartenant à une plage libre donc qui a peu de chance d'être pris. Cependant, vous serez peut-être amenés à le modifier). Ensuite, je mets l'adresse IP de la machine qui devra recevoir le paquet UDP (ce n'est donc pas votre ou vos téléphones mais la raspberry pi servant à commander le drone). Pour obtenir cette adresse IP, il suffit de taper la commande « ifconfig » sur la raspberry pi. Il est à noter que malgré le fait que cette adresse IP soit fixée dans le code, elle est modifiable directement depuis l'exécution de l'application.

Passons désormais aux. Le premier bloc est uniquement la déclaration d'un type de listener personnel. Il servira uniquement, quand il est appelé à identifier sur quel bouton l'utilisateur a touché. Selon le bouton, il changera le champ de texte contenu sur l'application (qui sert à l'utilisateur pour voir quelle est la commande qu'il émet vers le robot). Ce champ de texte sera réutilisé par la suite. De plus, si l'utilisateur relâche le bouton, on refait passer le champ de texte à « aucun ».

Pour la suite, il faut d'abord comprendre les bases du système d'exploitation Android. En effet, cet OS très spécial n'utilise pas de fonction « main » classique comme dans n'importe quel autre programme java mais des fonctions aux noms imposées appelées à différents moments. En effet, le schéma suivant (merci à openclassroom, le site remplacant le Site Du Zéro pour ce schéma) montre comment « l'activité » (c'est comme cela qu'un programme est appelé sous Android) marche.


L'activité démarre
onCreate()
L'utilisateur retourne vers l'activité
onStart()
onRestart()
Le processus est tué
onResume()
L'activité s'exécute
L'activité revient sur le devant de la scène
Une autre activité s'intercale devant notre activité
D'autres applications ont besoin de mémoire
onPause()
L'activité revient sur le devant de la scène
Notre activité n'est plus visible
onStop()
onDestroy()
L'activité est arrêtée

La première méthode est donc « onCreate », son rôle est d'instancier les boutons et différents champs, définis par leur ID (dans un fichier que nous verrons dans la prochaine partie). On leur ajoute ensuite le listener crée précédemment (pour factoriser le code et ne pas créer un listener spécifique à chaque bouton, ce qui aurait été une optimisation très peu efficace).

Ensuite, nous passons aux méthode « onResume » et « onPause » dont le fonctionnement est complémentaire. « onResume » sert à créer un thread (processus parallèle au processus principal) dont le rôle sera détaillé après. On appelle ensuite la fonction start de ce thread (ce qui fait que le programme principal est bloqué jusqu'à l'exécution complète de ce thread). « onPause », au contraire interrompt le processus et gère ensuite les possibles cas d'erreurs (grâce à un lever d'exception).

Dans le thread nommé « ThreadUDP », je crée juste un thread donc le but va être de créer un autre thread avec une certaine fréquence (grâce au sleep et à toutes les gestions d'exception associées). Tant que le thread n'est pas interrompu, il crée donc un thread nommé « ThreadEnvoiPaquetUDP » et lance ce thread dans un processus secondaire (grâce à la méthode run, qui n'est pas bloquante contrairement au start vu précédemment). La périodicité peut être réglée manuellement grâce au sleep (actuellement, un paquet UDP est envoyé toutes les 1 ms)

Enfin, dans le thread nommé ThreadEnvoiPaquetUDP, nous récupérons la commande utilisateur, contenue dans le champ de texte dont j'ai parlé précédemment. Ce thread risque d'être compliqué à comprendre tant que vous n'aurez pas eu vos cours de réseau mais en gros, ce qu'il faut retenir, c'est qu'il envoie à la rapsberry pi une information sous la forme d'un string. Nous avons utilisé des paquets UDP et non des paquets TCP car nous travaillons en temps réel, le renvoi de paquet n'ayant pas atteint leur cible est donc inutile (et même nuisible). Il faudra donc faire avec les caprices possibles du réseau et espérer qu'un grand nombre de paquets arriveront. Là encore, de nombreuses exceptions sont levés.

activity_spider_drone.xml

Vous trouverez ce fichier ici : « SpiderDrone/app/src/main/res/layout/activity_spider_drone.xml ». Ce fichier est un fichier de ressource, type de fichier là encore spécifique au système d'exploitation Android. C'est dans ce fichier que je définis la liste des éléments utilisé, leur positionnement ainsi que leur contenu et leur nom. Ainsi, je vais détailler un seul des cas, le reste étant assez facile à comprendre. Le premier bouton aura l'ID stabilisation. Cet ID est important car c'est grâce à cela que je peux le trouver dans le fichier java. Ensuite, je précise que se bouton prendre toute la taille nécessaire pour s'afficher entièrement. Ensuite, je précise quel est le contenu de ce bouton grâce au champ text. Enfin, je le centre au milieu de la fenêtre (ensuite, je placerais tous les autres composants par rapport aux autres, afin de faciliter l'utilisation de cette application sur un grand nombre de téléphone, ce qu'un positionnement absolu n'aurait pas permis).

AndroidManifest.xml

Vous trouverez ce fichier ici : « SpiderDrone/app/src/main/AndroidManifest.xml ». Ce fichier est extrêmement simple, malgré sa syntaxe très spéciale (c'est un document xml, comme le précédent). Ce fichier a pour seul but de lister les autorisations que l'application demandera lors de son installation (des autorisations nécessaires à l'utilisation du réseau dans notre cas).