TP 2 de programmation Android

Résumé

L'objectif de ce second TP est de transformer l'application créee au TP1 en lecteur video utilisant la classe MediaPlayer.

Une adresse utile : `http://developer.android.com/guide/components/index.html`

En local : `/mnt/n7fs/ens/tp_cregut/android-sdk-linux_86/docs/offline.html`

1 Préparation d'Éclipse

Afin d'éviter les soucis rencontrés au premier TP (débordement de quota), assurez-vous d'abord de disposer de quelques centaines de Mo libres sur votre compte.

Utiliser la version Éclipse qui est installée dans :

`/mnt/n7fs/ens/tp_android/eclipse/eclipse`

Si la perspective proposée par défaut n'est pas la perspective Java, ouvrez perspective Java (Window > open perspective > other > Java).

Lancez l'AVD (Window > Android Virtual Device Manager).

2 Préparation du projet

2.1 Ajout de video sur la carte SD

Ajoutez une vidéo sur la carte SD virtuelle. Pour cela, ouvrez la perspective DDMS pour accéder au file manager (ou ouvrez la vue si vous l'avez ajoutée), et faites "push file on device" (petit bouton en haut à droite). Choisissez la vidéo :

`/mnt/n7fs/ens/tp_android/TP_01/documentariesandyou.mp4`

On pourra se renseigner sur les formats de videos lisibles à la page : `http://developer.android.com/guide/appendix/media-formats.html`

2.2 Ouverture de vidéo par intention

Finir rapidement le TP 1 jusqu'en 4.2 (ouverture de vidéo par intention). Notre application appelle maintenant le lecteur media par défaut pour lire et contrôler la vidéo.

3 Intégration de la vidéo

Dans la version précédente, une nouvelle activité est lancée pour afficher la vidéo. Il serait intéressant que la vidéo soit lancée directement dans notre application. Plusieurs classes peuvent être utilisées, notamment VideoView (simple à mettre en place, mais limité) ou MediaPlayer (plus de fonctionnalités).

3.1 Layout

Nous allons commencer par modifier le layout, afin que l'application ressemble à un lecteur vidéo classique.

Créer :

- Un bouton "Play from start"
- Un bouton "Pause / Resume"
- Une "SurfaceView" dans laquelle nous afficherons la vidéo
- Une "Seekbar" pour afficher la position courante dans la vidéo (et pour s'y déplacer ensuite)
- Le bouton Go peut être conservé, tout comme le composant "EditText" que nous utiliserons pour le choix de la vidéo.

Note : votre classe principale doit maintenant implémenter SurfaceHoler.Callback :

```
public class BrowserActivity extends Activity implements SurfaceHolder.Callback {
```

3.2 Préparation du Media Player

La lecture de vidéo par intention est gourmande en mémoire, et ne peut être modifiée selon ses besoins ou ses goûts. La classe MediaPlayer est plus étendue. Transformer l'application pour que la vidéo soit gérée par cette classe. Pour cela il faut d'abord initialiser le SurfaceHolder :

surfaceView = (SurfaceView) findViewById(R.id.surfaceview) ;
surfaceHolder = surfaceView.getHolder() ;
surfaceHolder.addCallback(this) ;
surfaceHolder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS) ;

et associer ce Holder au MediaPlayer :

```
public void surfaceCreated(SurfaceHolder holder) {
 mediaPlayer.setDisplay(surfaceHolder) ;
}
```

3.3 Lecture de la vidéo

Etudiez la page de la Doc correspondant à la classe MediaPlayer : `http://developer.android.com/reference/android/media/MediaPlayer.html`

En ajoutant un listener sur le bouton "play", faites en sorte que la vidéo soit lancée dans le SurfaceHolder défini plus haut. Il faut dans l'ordre :

- choisir le fichier vidéo à lire (**setDataSource**)
- préparer le MediaPlayer à la lecture (**prepare**)
- lancer la video (**start**)

Les fonctions **mediaPlayer.*** ** permettront de contrôler la vidéo, ainsi que d'obtenir de nombreuses informations sur la vidéo (taille, durée...) que vous pourrez éventuellement afficher sous la vidéo. En utilisans les différents Listener de la classe et le LogCat, vous étudierez les différentes étapes du chargement de la vidéo.

Associez maintenant un listener au bouton "pause/resume" afin de mettre en pause la vidéo ou de la relancer, suivant son état (on pourra utiliser un simple booléen pour suivre cet état).

3.4 Cycle de vie

Afin d'éviter les conflits, il faut libérer le MediaPlayer lorsque l'utilisateur quitte l'application, ou sauvegarder la position courante jusqu'à ce qu'il revienne dans l'application.

En utilisant les fonctions **public void onPause()** et **public void onResume()**, gérez les différents scénarios possibles :

- Lorsque l'utilisateur quitte l'application (**onPause()**, il faut enregistrer la position courante (**getCurrentPosition()**) et libérer le MediaPlayer (**reset()** et **release()**.
- Lorsqu'il y revient, il faut réinitialiser le MediaPlayer (**setDataSource**, **prepare**), et lancer la vidéo à la dernière position enregistrée : (**seekTo(position)** et **start()**).

3.5 Gestion de l'avancement

Plutôt que de démarrer à chaque fois la vidéo au début, il est intéressant de pouvoir s'y déplacer au moyen d'une barre de défilement, qui permettra aussi de visualiser l'avancement de la vidéo. Nous allons pour cela utiliser la SeekBar, ainsi que des threads.

Pour se déplacer dans la vidéo, ajoutez un "OnSeekBarChangeListener" à la SeekBar, dans laquelle vous définirez une fonction **public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser)** qui déplacera la vidéo à la position "progress" (fonction **mediaPlayer.seekTo(progress)**).

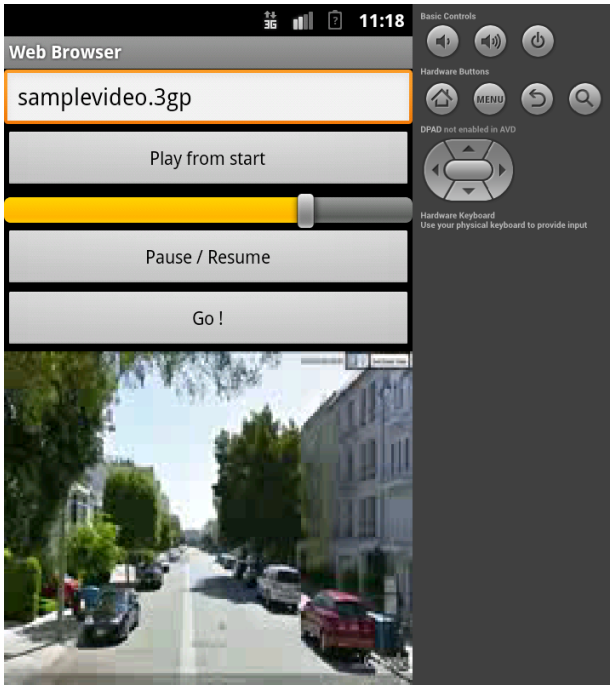
Essayez de vous déplacer dans la vidéo grâce à cette SeekBar. Remarquez que la position n'est pas mise à jour en "continu" lors de la lecture de la vidéo. Il faut pour cela utiliser un "thread".

Créez une classe "gestionAvancement" implémentant "Runnable" : **class GestionAvancement implements Runnable{ }**

Cette classe doit contenir les méthodes **run()**, **stop()** et **start()**. Dans la méthode **run()**, une boucle "while" vérifiant que la vidéo n'est pas terminée récupèrera la position courante (**mediaPlayer.getCurrentPosition()**), mettra à jour la SeekBar (**seekbar.setProgress(currentPosition)**) et attendra 50ms (**Thread.sleep(50)**).

Enfin, le thread sera lancé par le Listener associé au bouton play : **progressUpdate = new Thread(gestionAvancement) ; progressUpdate.start()**. Modifiez le cycle de vie pour prendre en compte le démarrage et l'arrêt du thread.

La barre de défilement est maintenant mise à jour pendant la lecture, et le lecteur vidéo contient les fonctionnalités basiques attendues. Votre application devrait maintenant ressembler à ca :



3.6 Personnalisation de l'application

Vous pouvez maintenant personnaliser l'application, pour par exemple ne pas afficher la seekBar tant que la vidéo n'est pas démarrée (**setVisibility(View.INVISIBLE)**), changer l'apparence et la taille des boutons (via le layout)...