Application à un système vidéo

Romain Raveaux

Sommaire :

But du TP

1°) Visualiser les images issues de la camera.
2°) Passer de l'espace couleur YUV à l'espace couleur RGB.
3°) Monitorer l'exécution en calculant le nombre de Frame Per Second

Travail à faire

1°) Créer un projet Android de type HelloWorld (un projet simple de base)
2°) Récupérer la classe FrameProcessing via le lien suivant :
https://www.dropbox.com/s/28j2jcuue0mmv37/FramProcessing.java?dl=0

3°) La méthode de traitement de la Frame en cours :

```
protected Bitmap processFrame(byte[] data ) {

                    int frameSize = getFrameWidth() * getFrameHeight();
                    int[] rgba = new int[frameSize];

                    for (int i = 0; i < getFrameHeight(); i++)
                              for (int j = 0; j < getFrameWidth(); j++) {
                                        int y = (0xff & ((int) data[i * getFrameWidth() + j]));
                                        int u = (0xff & ((int) data[frameSize + (i >> 1) * getFrameWidth() + (j & ~1) + 0]));
                                        int v = (0xff & ((int) data[frameSize + (i >> 1) * getFrameWidth() + (j & ~1) + 1]));
                                        y = y < 16 ? 16 : y;

                                        int r = Math.round(1.164f * (y - 16) + 1.596f * (v - 128));
                                        int g = Math.round(1.164f * (y - 16) - 0.813f * (v - 128) - 0.391f * (u - 128));
                                        int b = Math.round(1.164f * (y - 16) + 2.018f * (u - 128));

                                        r = r < 0 ? 0 : (r > 255 ? 255 : r);
                                        g = g < 0 ? 0 : (g > 255 ? 255 : g);
                                        b = b < 0 ? 0 : (b > 255 ? 255 : b);


                                        rgba[i * getFrameWidth() + j] = 0xff000000 + (b << 16) + (g << 8) + r;
                              }


                    Bitmap bmp = Bitmap.createBitmap(getFrameWidth(), getFrameHeight(), Bitmap.Config.ARGB_8888);
                    bmp.setPixels(rgba, 0/* offset */, getFrameWidth() /* stride */, 0, 0, getFrameWidth(), getFrameHeight());
                    return bmp;
          }
```

5°) Expliquer ce que fait la fonction processFrame ?

6°) Récupérer la classe abstraite MyVideoView qui étend la classe `SurfaceView` et qui implémente les interfaces suivantes : `SurfaceHolder.Callback` et `Runnable`
`https://www.dropbox.com/s/owctss2jeup0tyj/MyVideoView.java?dl=0`

7°) Chercher et décrire à quoi sert la classe SurfaceView ?

8°) Chercher et décrire à quoi sert l'interface SurfaceHolder.Callback ?
9°) Chercher et décrire à quoi sert l'interface Runnable ?
10°) A quoi sert les méthode `getFrameWidth et getFrameHeight ?`

`11°) A quoi sert la méthode openCamera ?`

`12°) A quoi sert la méthode releaseCamera ?`

`13°) A quoi sert la méthode setupCamera ?`
`Lister les grandes étapes de cette fonction ?`

`14°) A quoi sert la méthode surfaceChanged ?`

`15°) A quoi sert la méthode surfaceCreated ?`

`16°) Que fait la méthode run ?`

17°) Dans votre activité principale, ajouter à votre layout un objet de type FrameProcessing après l’avoir instancié.

18°) Compiler et étudier les éventuels erreurs.

19°) Lancer l’application sur un émulateur.
21°) Créer la classe FramePerSecond qui permettra d’écrire à l’écran le nombre de frame par seconde de votre application tout en affichant la vidéo.
Vous utiliserez le Canvas pour écrire à l'ecran.

à Rendre :

1°) Le code source commenté
2°) Un rapport décrivant le fonctionnement et le but de ce TP. Qu’avez-vous compris ? Avez-vous rencontré des problèmes ? Ecrivez les éléments qui permettraient à un de vos collègues de faire ce TP facilement.
3°) Le tout doit être déposé sur moodle pour la semaine suivant le TP. (ou si moodle était récalcitrant par email à votre enseignant)