# Annexe 8 – Animations

Principes derrière les animations « utiles » dans une interface graphique :

* Informative : permet de montrer les liens entre les informations 🡪 glisser « swipe » pour voir plus d’informations
* Attractive : une animation peut permettre d’attirer l’attention sur ce qui est important 🡪 animation sur un bouton pour indiquer qu’on doit le cliquer.
* Originale : donne du style à une application si revient à plusieurs reprises

1. ObjectAnimator

* Permet d’animer certaines propriétés d’un objet graphique
* Des méthodes de la classe permettent de définir la ou les propriétés qui seront soumises à l’animation
* On peut établir ces animations en XML ou en Java
* La méthode invalidate est automatiquement appelée afin que l’apparence visuelle soit actualisée

Exemples :

Objcetanimator anim = ObjectAnimator.offloat( ce qu’on veut anime (un bouton),ou on va la chercher View.X, depart500 px, arriver 1000px)

Anim.start();

Objcetanimator anim2 = ObjectAnimator.offloat(image, View.x, view.Y, path )

ofARGB(textView, "textColor",Color.RED,co)

Objcetanimator anim2 = ObjectAnimator.offloat(image, View.x, view.Y, path )

Resources.getSystem().getDisplayMehcs().

Propriétés pouvant être utilisées :

[Property Animation Overview  |  Android Developers](https://developer.android.com/develop/ui/views/animations/prop-animation#views)

entres autres…

* View. X : en fonction de l’origine
* View.TRANSLATION\_X : en fonction de la position au départ ( nonobstant d’autres translations qui auraient eu lieu )

\*\*\* Les unités sont en pixels. Si besoin est vous pouvez faire les équivalences en dp à l’aide des formules suivantes :

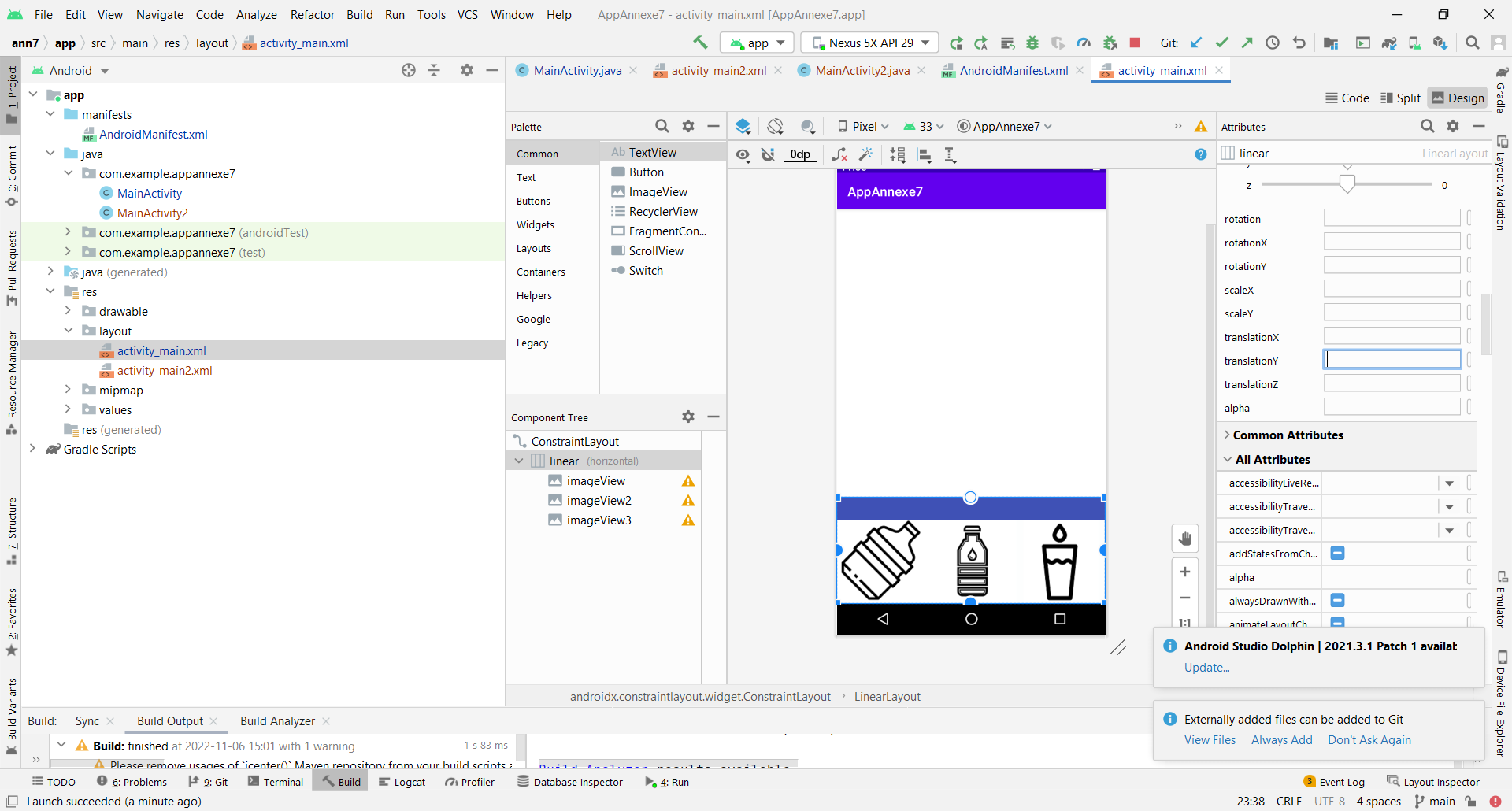
**public static int** pxToDp(**int** px) {  
 **return** (**int**) (px / Resources.*getSystem*().getDisplayMetrics().**density**);

}  
  
**public static int** dpToPx(**int** dp) {  
 **return** (**int**) (dp \* Resources.*getSystem*().getDisplayMetrics().**density**);  
}

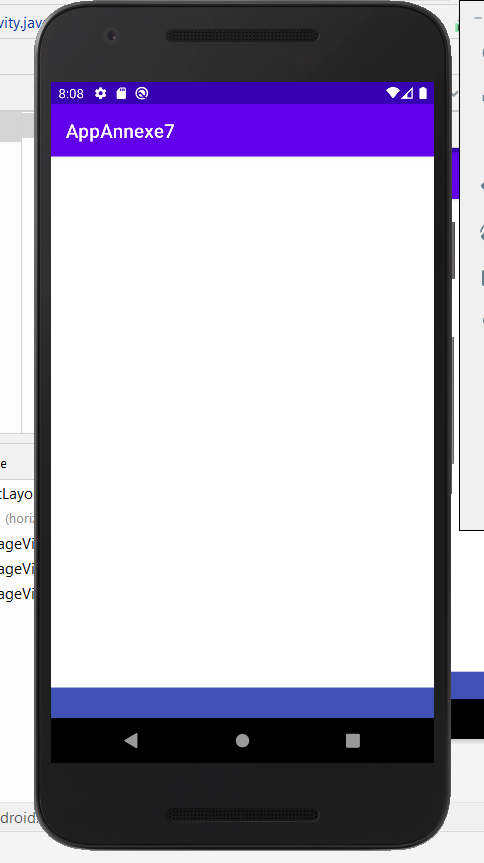
Resources.*getSystem*().getDisplayMetrics().**heightPixels**

Exercice 1

Dessinez un conteneur qui apparaitra lorsqu’on le cliquera :



1. Créez un nouveau projet, dans le fichier .xml de positionnement, utilisez un LinearLayout que vous accrocherez au ConstraintLayout d’origine et placez-y ces trois images connues.
2. Donnez un fond blanc aux ImageViews et un fond bleu au LinearLayout. Donnez un padding de manière à ce qu’on voie une bande bleue en haut des images.
3. Dans l’onglet design, utilisez la propriété translationY sur le LinearLayout de manière à cacher tout son contenu sauf la bande bleue.



D) Utilisez un ObjectAnimator avec une propriété adéquate de manière à ce que le LinearLayout apparaisse progressivement jusqu’à s’afficher au complet quand on clique sur la bande bleue :



E) Trouvez une méthode dans la classe ObjectAnimator afin de rabaisser le menu une fois ouvert

F) Codez votre gestion d’événements en utilisant une expression lambda. Faites en sorte que le LinearLayout apparaisse lorsqu’il est caché et qu’il disparaisse lorsqu’il est déjà visible. ( en alternance )