Annexe 2 – Interfaces graphiques Android

*** L'API ANDROID EST À : http://developer.android.com/reference/

1. On peut séparer les éléments composant la vue d'une application en 4 grands ensembles :

A) View :_____

B) ViewGroup :_____

C) Fragments : (depuis API 11) _____

_____

D) Activity : _____

_____

Pour assigner un GUI à une activité : `setContentView( R.layout.nom du fichier XML )`

→ permet de modifier l'apparence du GUI sans toucher au code (le modifier pour l'adapter à la configuration de l'écran au runtime)

2. Les layouts (LayoutManagers)

→ peuvent être imbriqués, de manière à dessiner des GUI plus complexes

→ les plus communs :

A) RelativeLayout :_____

B) LinearLayout: _____

C) FrameLayout : _____

D) GridLayout (depuis API 14) :

_____

*** Autant pour les layouts que pour les widgets, on utilise les constantes match_parent et wrap_content pour initialiser les largeurs et hauteurs plutôt que des valeurs fixes en pixels

→ permet que la disposition des layouts et des éléments soit indépendante de la résolution et de la dimension de l'écran.

Attributs communs (Layouts et views)	Valeurs possibles	Explications
android:layout_height android:layout_width	match_parent	
	wrap_content	
android:gravity	left, right, center_horizontal	

→ sinon, pour les tailles de polices ou certaines marges / paddings :

Unités de mesure	**signification**	**utilité**
pixels		
dp (ou dip)		
sp		

3. Relative Layout

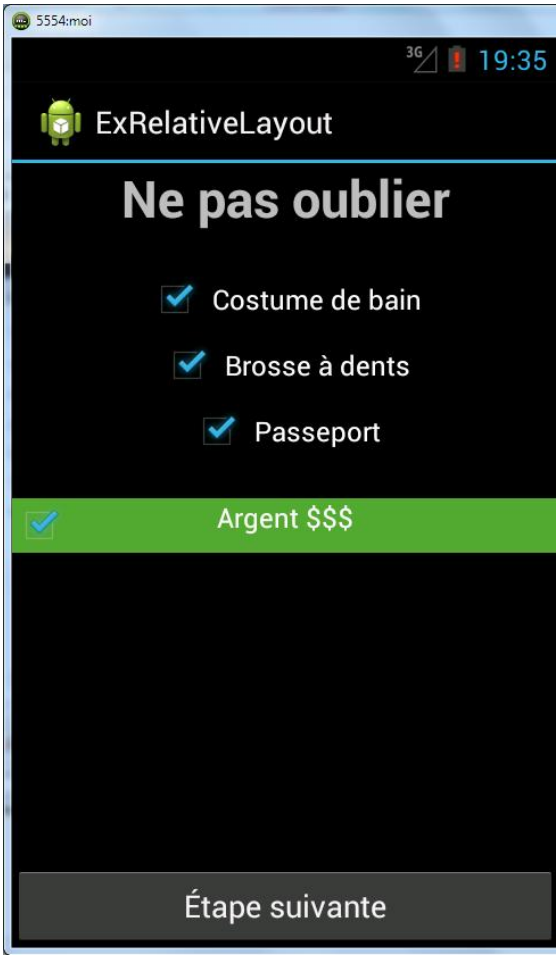
→ Les positions des éléments sont déterminés relativement à celles de leurs voisins ou de leurs parents

→ quelques attributs XML possibles : (dans l'API , View, ViewGroup, RelativeLayout ou RelativeLayout.LayoutParams)

Attribut XML	Équivalent Java	Détails
android:layout_alignParentLeft		
android:layout_centerHorizontal		
android:layout_marginRight	setMargins(int,int,int,int)	
android:layout_toLeftOf		

`android:background`	`setBackgroundCol or(Color c )`	

EX1 : Dans un nouveau projet, reproduisez l'interface suivante en utilisant un `RelativeLayout` (utilisez le fichier de positionnement .XML)



Cases à cocher ? _____

attribut pour que le bouton soit en bas ? _____

attributs pour la zone verte ? _____

Pourquoi le fond de mon application est noir plutôt que blanc ? _____

_____

4.LinearLayout

Résumé (LinearLayout)

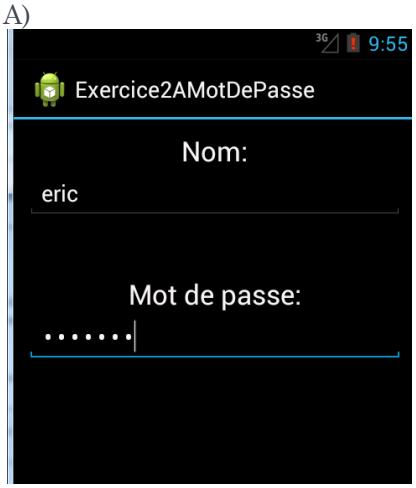
Propriétés XML	Equivalent Java	détails
`android : orientation`	`setOrientation`	`"horizontal"`: 1 seul widget par colonne `"vertical"` : 1 seul widget par ligne
`android :layout_width`	-	`"match_parent"` : prend toute la place restante, une fois les autres widgets placés avant `"wrap_content"` : Prend la place nécessaire (texte présent sur le bouton) Mesure en dip
`android :layout_height`	-	
`android :layout_weight`		Partager l'espace disponible (pourcentage ou valeur) IMPORTANT :
`android :gravity`	`setGravity ( Gravity.LEFT ) etc.`	Pour un élément → organise la disposition de tous les éléments contenus dans cet élément ("right", "horizontal_center"…)
`android :layout_gravity`		Positionne l'élément par rapport à son parent (si possible)

`android :padding`
`android :paddingTop`
`android:layout_marginBottom`
Etc.
	`setPadding`
	`setMargin`
	`etc`
		Donne un padding ou une marge à l'élément
		
		Padding permet de changer la forme des widgets car c'est la zone de dessin

Exercices

EX2. Reproduire les interfaces suivantes en utilisant principalement les valeurs « match_parent » et « wrap_content » et le layout LinearLayout

A)



attribut xml utilisé pour centrer les éléments du Layout :

_____

Moyen de faire un espace entre le champ texte nom et le TextView Mot de passe ?

_____

_____

Propriété pour que les caractères du mot de passe soient cachés :

_____

Ajoutez un bouton sur cette interface. Faites en sorte que ce bouton permette de valider la connexion si le nom est "Foo" est le mot de passe est "Bar". Afficher connexion réussie ou fermer l'application si les données ne correspondent pas

Pour afficher connexion réussie : utiliser un Toast
http://developer.android.com/guide/topics/ui/notifiers/toasts.html

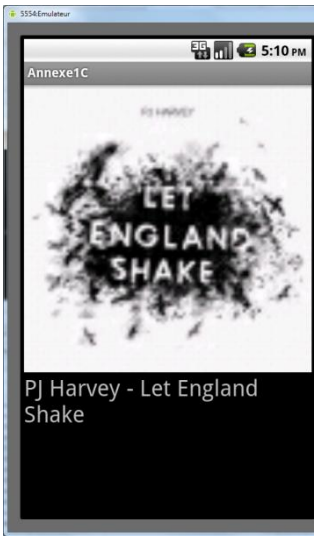
Pour fermer l'application : finish();

B)
Stratégie à utiliser pour placer plusieurs widgets sur la même ligne :

_____

_____

_____



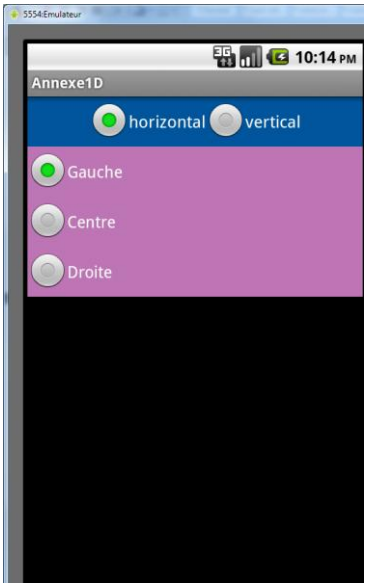
C) Faites en sorte, en utilisant un poids, que la surface consacrée à l'image soit 2 fois plus grande que celle consacrée au texte

Quelle propriété fait en sorte que l'image est de même taille que son conteneur ImageView ? _____

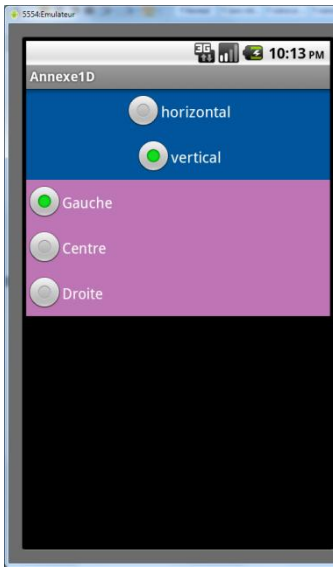
D) Utilisez 2 RadioGroups, le premier avec ses RadioButtons centrés horizontalement, l'autre avec ses RadioButtons alignés à gauche.

Faites la gestion des événements de manière à ce que le premier groupe change son orientation de vertical à horizontal

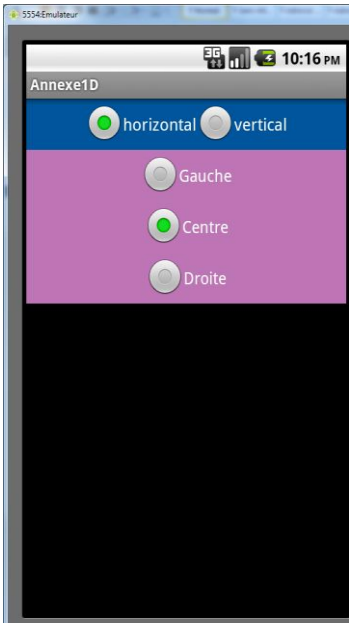
Faites la gestion des événements du deuxième pour que les RadioButtons changent de gauche, au centre, à droite



au départ



…suite au clic « vertical »



…suite au clic « centre »

5.GridLayout

Plus facile à utiliser dans le fichier xml qu'avec l'éditeur graphique.

site web expliquant bien les principes :
http://blog.stylingandroid.com/archives/669

*** On peut créer d'autres layouts pour la même Activité en ajoutant des fichiers `main.xml` dans des dossiers différents que le dossier `layout`.

Par exemple, si on veut un affichage différent si la personne utilise l'application en position « Paysage », on peut faire le design en créant un `main.xml` dans un dossier `layout-land` toujours dans le répertoire `res`
Le même principe existe pour les icöns représentant l'application, c'est pour ça que 3 dossiers existent.

`res/layout-xlarge/main.xml` pour les écrans à large dimension

`Pour plus d'infos :` http://developer.android.com/guide/practices/screens_support.html

6. Styles et Thèmes

A) Modifier le thème global : dans le dossier values-v14, fichier styles.xml, changer le parent :

```
Ex.
parent="android:Theme.Holo"
parent="android:Theme.Light"
```

B) **modifier des éléments de ces thèmes globaux** :

→ ajouter des items à l'intérieur de la définition des styles dans le fichier styles.xml :

Ex: `<item name="android:background">#FF0000</item>`

→ s'applique à toute l'activité, moins utile

C) **créer un style particulier:**

→ à l'image d'une CSS, définir un style et l'appliquer à un ou plusieurs widgets (View) plutôt que définir toutes les modifications de polices et de couleurs dans le fichier de positionnement XML

→ Ex:

dans le fichier styles.xml

```
<style name="modifierTexte">
        <item name="android:textSize">60sp</item>

</style>
```

appliquer le style dans le fichier de positionnement .xml , à un élément TextView par exemple :

```
<TextView
style="@style/modifierTexte"    />
```

→ On peut donc définir des styles et des thèmes dans chaque dossier, dépendant de la langue ou de la version du téléphone.