Développer des interfaces utilisateurs

M213

Support du cours développement mobile

ETTAHERI Nizar
ettaheri.nizar@gmail.com

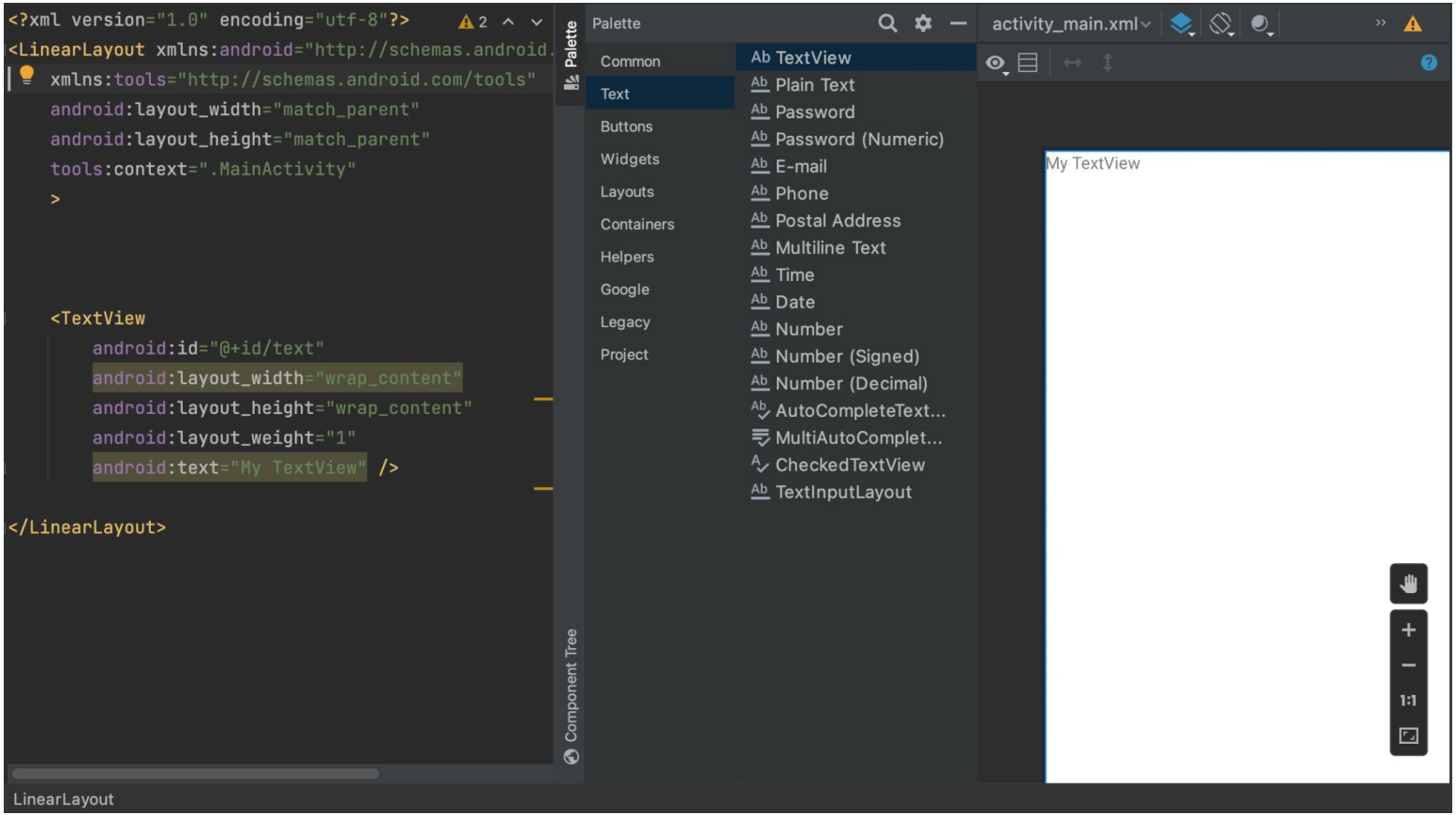
Rappel :

- **Implémenter des éléments graphiques**
- **Utiliser les Layouts**
- **Gérer les évènements**

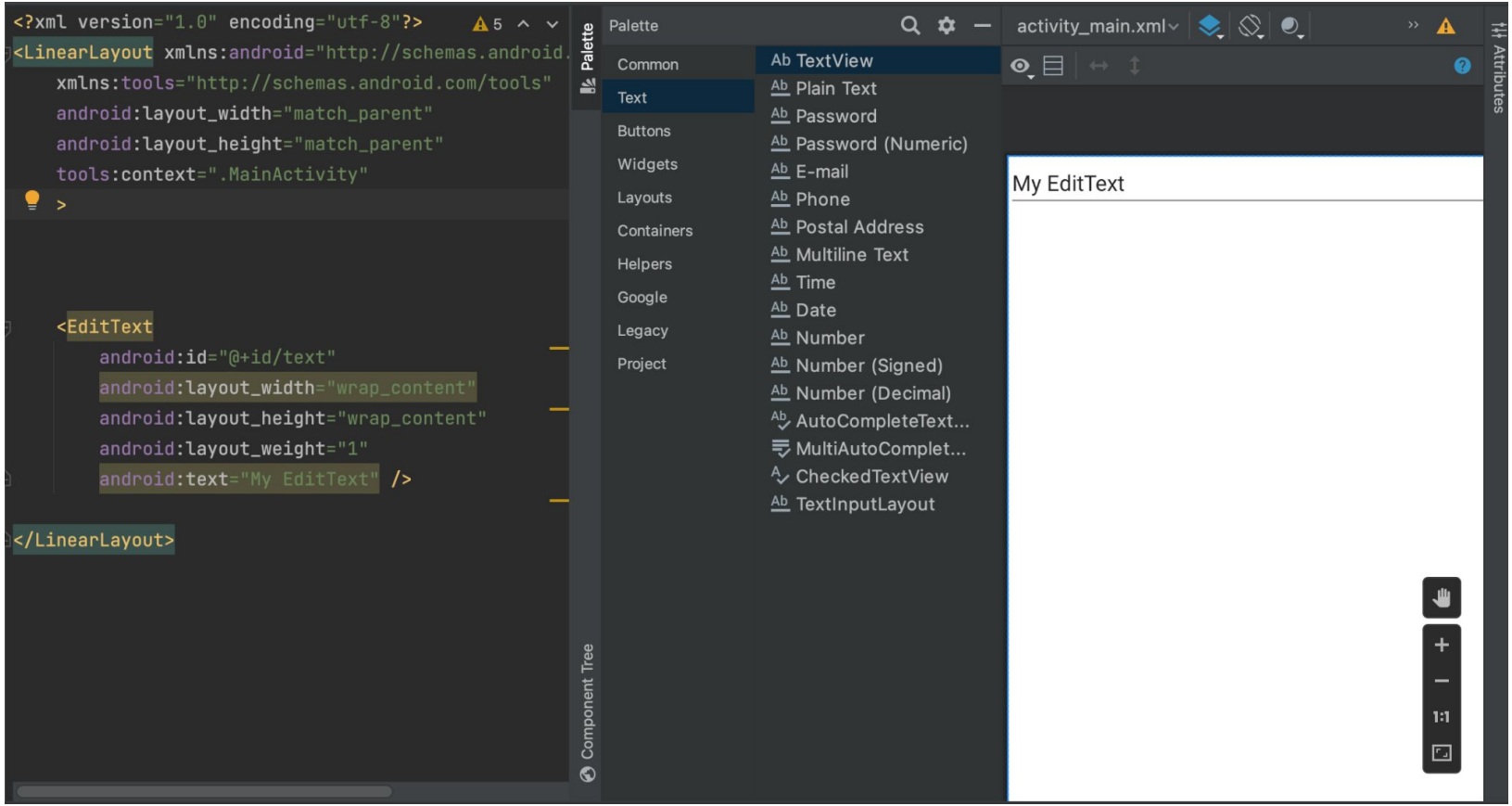
- **Implémenter des éléments graphiques**

Définition des Widgets (TextView, EditText, ImageView, …)

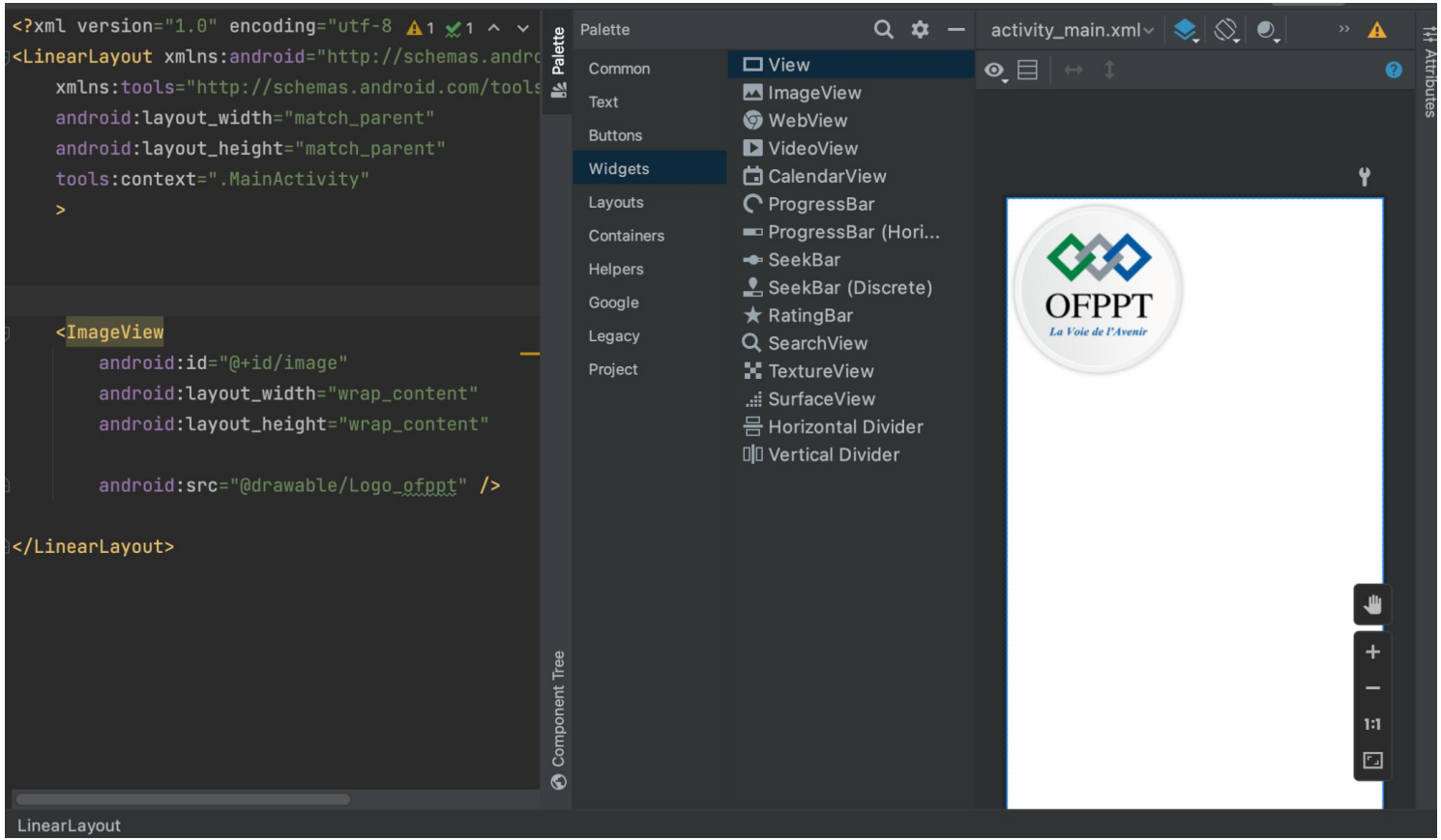
TextView


<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context=".MainActivity"
>
<TextView
android:id="@+id/text"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_weight="1"
android:text="My TextView" />
</LinearLayout>
Palette
Common
Text
Buttons
Widgets
Layouts
Containers
Helpers
Google
Legacy
Project
Ab TextView
Plain Text
Password
Password (Numeric)
E-mail
Phone
Postal Address
Multiline Text
Time
Date
Number
Number (Signed)
Number (Decimal)
AutoCompleteText...
MultiAutoComplet...
CheckedTextView
TextInputLayout
activity_main.xml
My TextView
Component Tree
LinearLayout

EditTextView

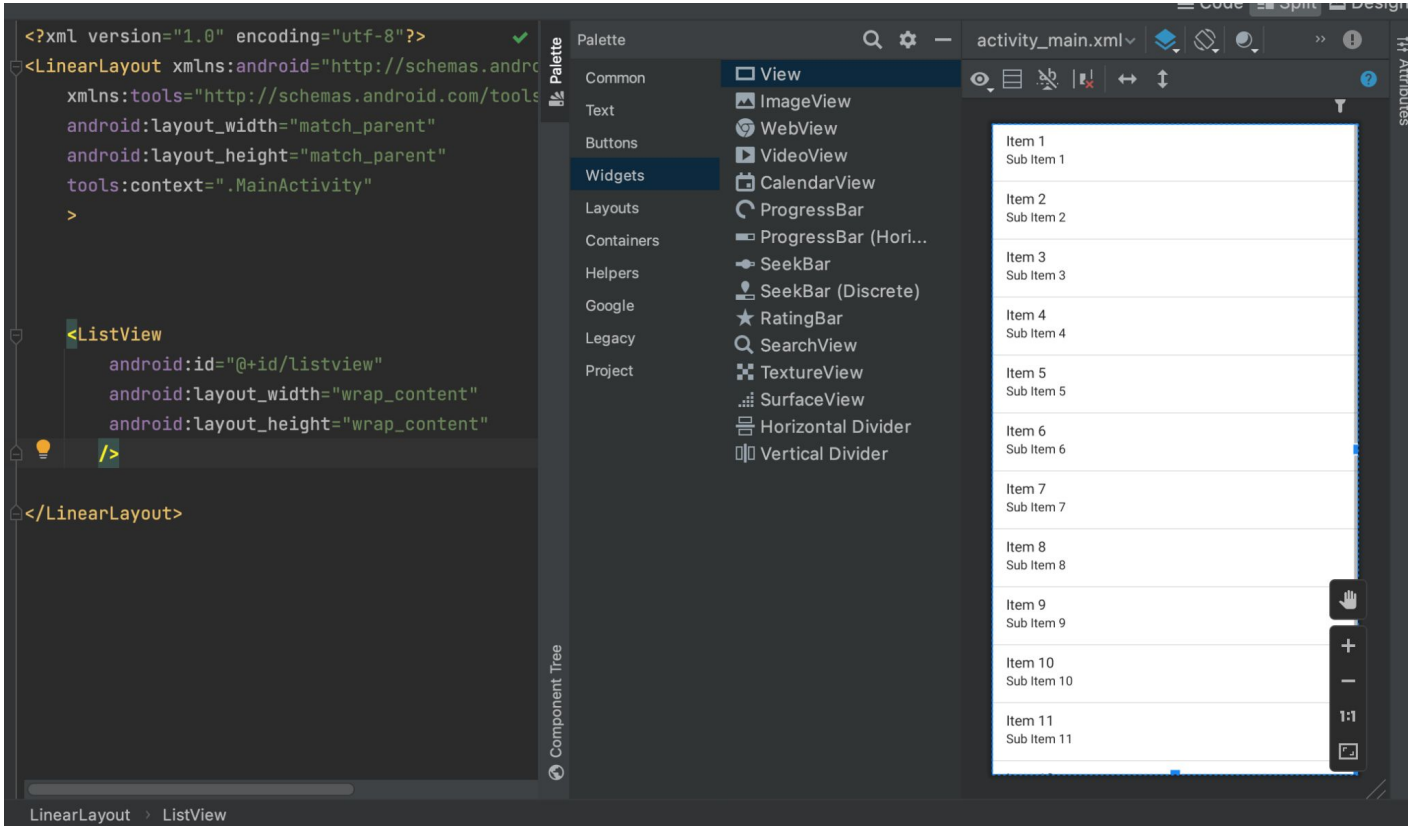


ImageView

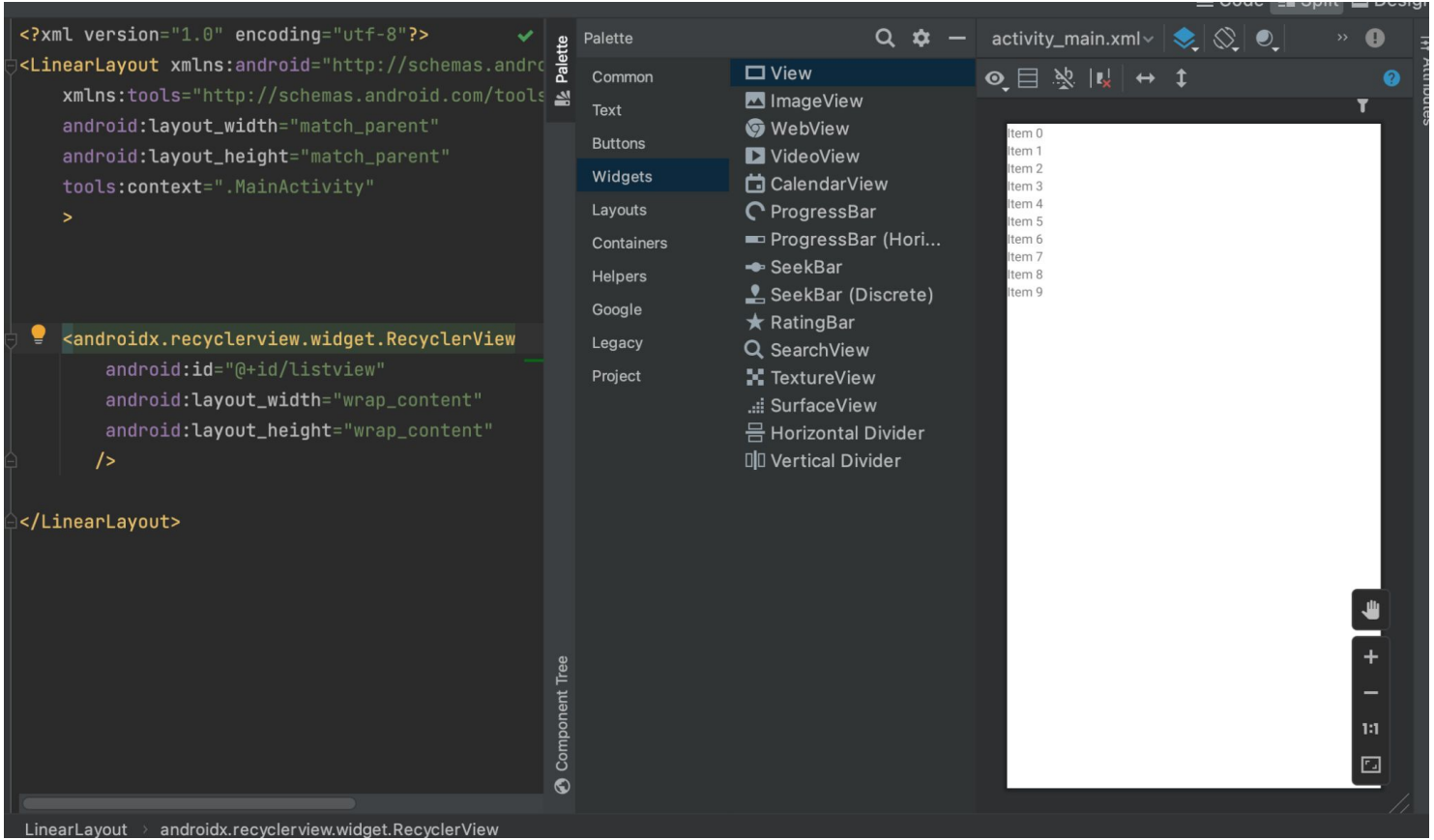

<?xml version="1.0" encoding="utf-8
<LinearLayout xmlns:android="http://schemas.andro
xmlns:tools="http://schemas.android.com/tools
android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context=".MainActivity"
>
<ImageView
android:id="@+id/image"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:src="@drawable/Logo_ofppt" />
</LinearLayout>
Palette
Common
Text
Buttons
Widgets
Layouts
Containers
Helpers
Google
Legacy
Project
View
ImageView
WebView
VideoView
CalendarView
ProgressBar
ProgressBar (Hori...
SeekBar
SeekBar (Discrete)
RatingBar
SearchView
TextureView
SurfaceView
Horizontal Divider
Vertical Divider
activity_main.xml
OFPPT
La Voie de l'Avenir
Component Tree
Attributes
LinearLayout

- Présentation des Widgets avancés (ListView, RecylerView, …)

ListView


<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.andro
xmlns:tools="http://schemas.android.com/tools
android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context=".MainActivity"
>
<ListView
android:id="@+id/listview"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
/>
</LinearLayout>
Palette
Common
Text
Buttons
Widgets
Layouts
Containers
Helpers
Google
Legacy
Project
View
ImageView
WebView
VideoView
CalendarView
ProgressBar
ProgressBar (Hori...
SeekBar
SeekBar (Discrete)
RatingBar
SearchView
TextureView
SurfaceView
Horizontal Divider
Vertical Divider
activity_main.xml
Item 1
Sub Item 1
Item 2
Sub Item 2
Item 3
Sub Item 3
Item 4
Sub Item 4
Item 5
Sub Item 5
Item 6
Sub Item 6
Item 7
Sub Item 7
Item 8
Sub Item 8
Item 9
Sub Item 9
Item 10
Sub Item 10
Item 11
Sub Item 11
Component Tree
LinearLayout › ListView

RecycleView



- **Utiliser les Layouts**

TP : (TextView, EditText, ImageView, …)

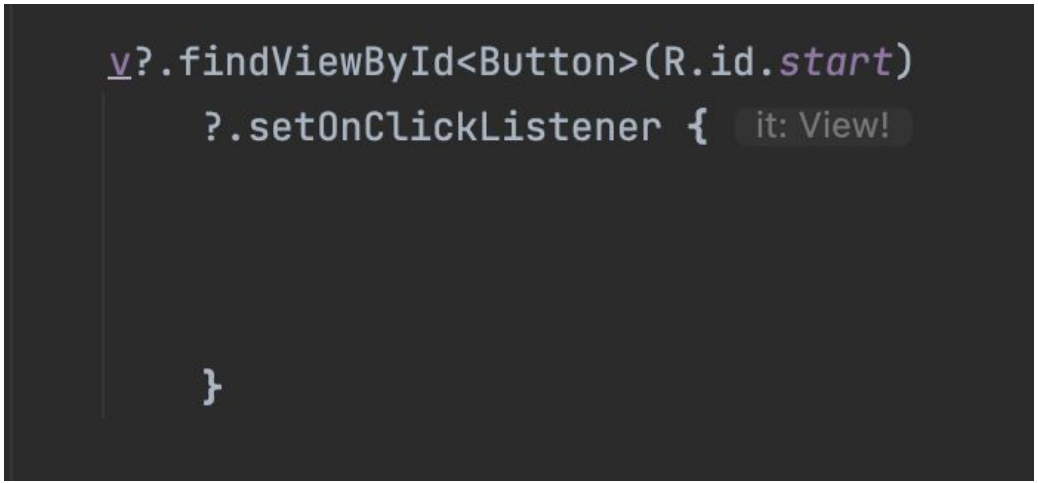
TP : Widgets avancés (ListView, RecylerView, …)

- **Gérer les évènements**

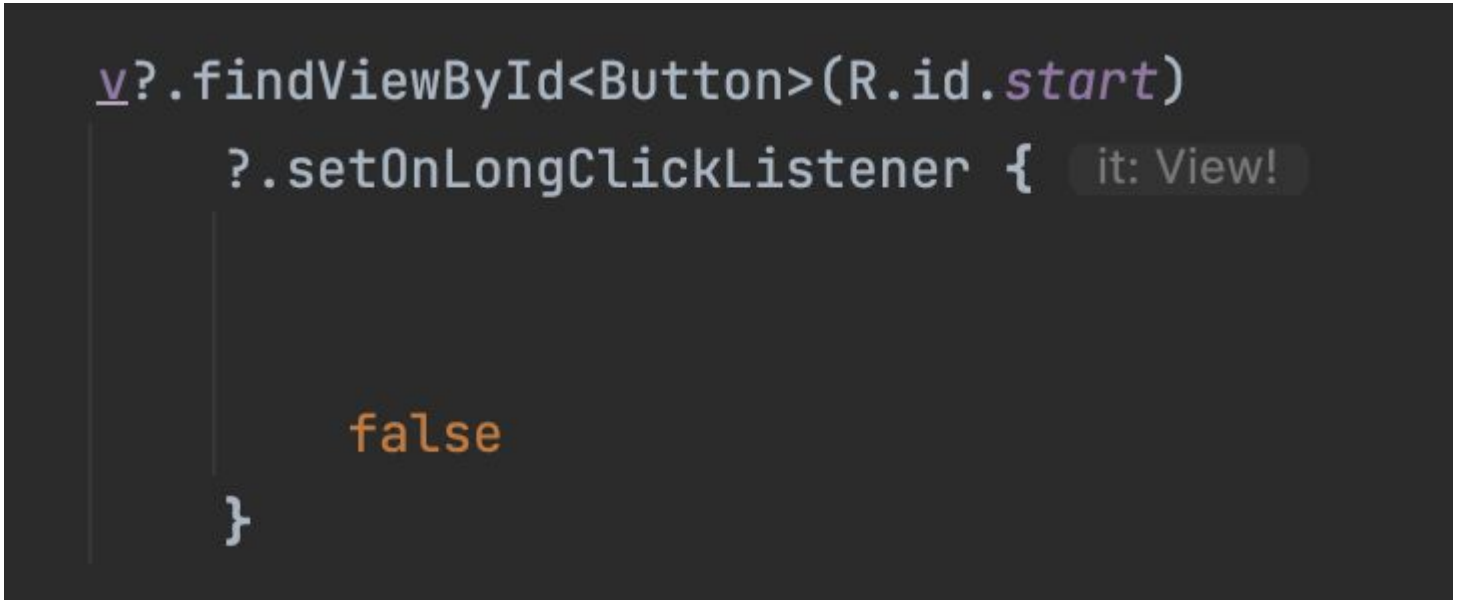
- Définition d’un évènement
- Présentation des types des événements (Click Listener, Touch Listener, Focus Change, Long Click, …)

TP : Présentation des types des événements (Click Listener, Touch Listener, Focus Change, Long Click, …)

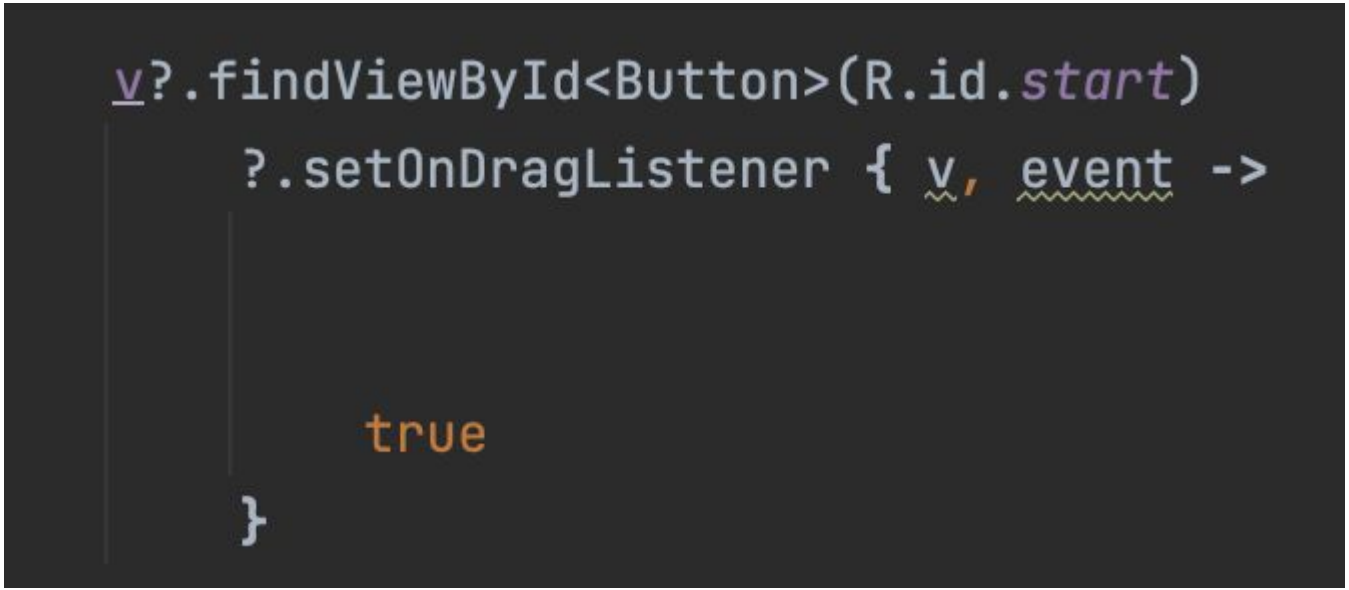
OnClickListener



OnLongClickListener



OnDragListener



```
v?.findViewById<Button>(R.id.start)
    ?.setOnDragListener { v, event ->


        true
    }
```

OnTouchListener

```
v?.findViewById<Button>(R.id.start)
    ?.setOnTouchListener { view, motionEvent ->

        false
    }
```

OnHoverListener

```
v?.findViewById<Button>(R.id.start)
    ?.setOnHoverListener { view, motionEvent ->

        false
    }
```

OnKeyListener

```
v?.findViewById<Button>(R.id.start)
    ?.setOnKeyListener { view, i, keyEvent ->

        false
    }
```

OnFocusChangeListener

```
v?.findViewById<Button>(R.id.start)
    ?.setOnFocusChangeListener { view, b ->


    }
```

android.View

```
OnClickListener                  // clic
OnLongClickListener              // clic long
OnDragListener                   // glissement
OnTouchListener                  // touché
OnHoverListener                  // survol
OnKeyListener                    // frappe de clavier
OnAttachStateChangeListener      // changement de l'état d'attachement
OnLayoutChangeListener           // changement du layout
OnCreateContextMenuListener      // création du menu contextuel
OnFocusChangeListener            // changement du focus
OnGenericMotionListener          // un mouvement (mouse, pen, finger, ...)
OnSystemUiVisibilityChangeListener // changement de la visibilité de
                                 // la barre d'état
```

- **Coder des custom view**

- **Manipulation des fichiers des styles**

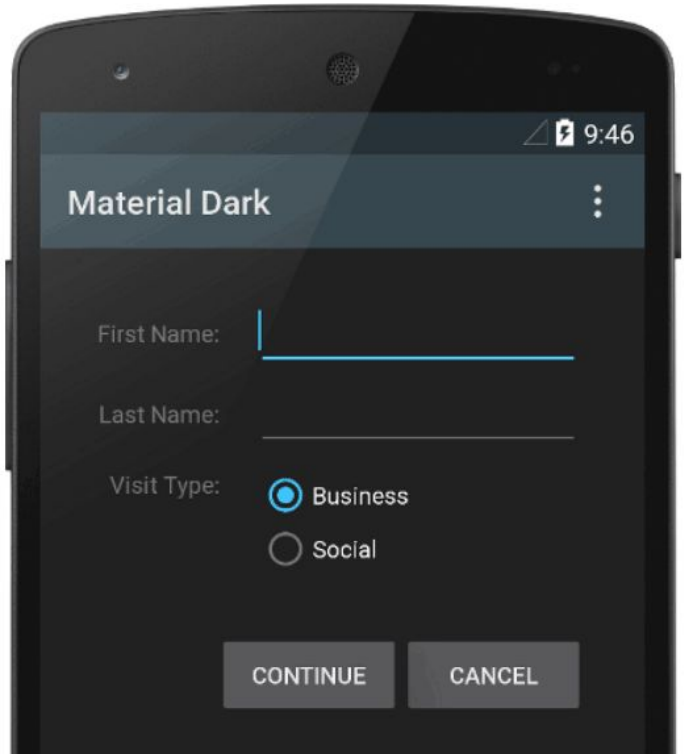
Styles et thèmes

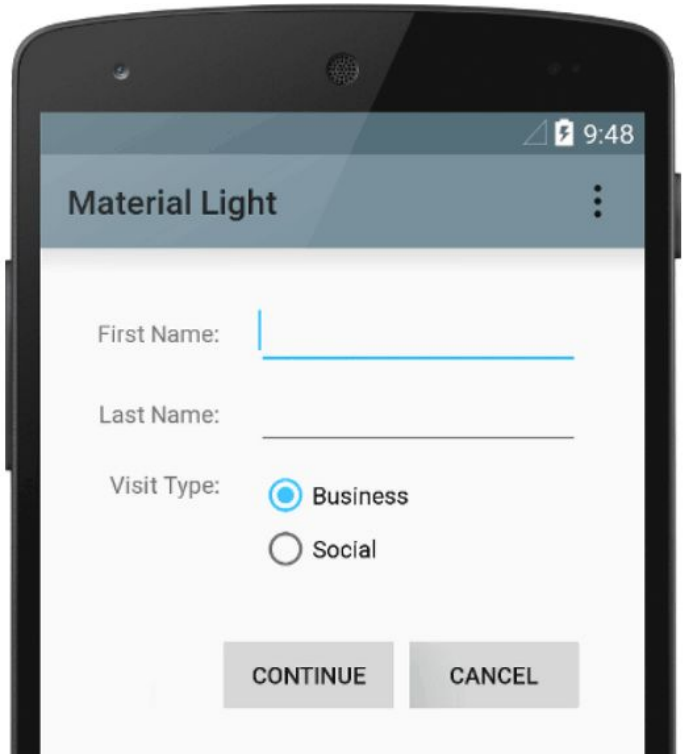
Les styles et les thèmes sur Android vous permettent de séparer les détails de la conception de votre application de la structure et du comportement de l'interface utilisateur, comme les feuilles de style dans la conception Web.

Un *style* est une collection d'attributs qui spécifient l'apparence d'un seul fichier View. Un style peut spécifier des attributs tels que la couleur de la police, la taille de la police, la couleur d'arrière-plan et bien plus encore.

Un *thème* est un ensemble d'attributs appliqués à une application, une activité ou une hiérarchie de vues entière, et pas seulement à une vue individuelle. Lorsque vous appliquez un thème, chaque vue de l'application ou de l'activité applique chacun des attributs du thème qu'elle prend en charge. Les thèmes peuvent également appliquer des styles à des éléments non visibles, tels que la barre d'état et l'arrière-plan de la fenêtre.

Les styles et les thèmes sont déclarés dans un fichier de ressources de style dans res/values/, généralement nommé styles.xml.


9:46
Material Dark
First Name:
Last Name:
Visit Type:
Business
Social
CONTINUE
CANCEL


9:48
Material Light
First Name:
Last Name:
Visit Type:
Business
Social
CONTINUE
CANCEL

Thèmes versus Styles

Les thèmes et les styles présentent de nombreuses similitudes, mais ils sont utilisés à des fins différentes. Les thèmes et les styles ont la même structure de base : une paire clé-valeur qui mappe les *attributs* aux *ressources* .

Un style spécifie les attributs d'un type de vue particulier. Par exemple, un style peut spécifier les attributs d'un bouton. Chaque attribut que vous spécifiez dans un style est un attribut que vous pouvez définir dans le fichier de mise en page. En extrayant tous les attributs d'un style, il est facile de les utiliser et de les maintenir sur plusieurs widgets.

Un thème définit une collection de ressources nommées qui peuvent être référencées par des styles, des mises en page, des widgets, etc. Les thèmes attribuent des noms sémantiques, comme colorPrimary, aux ressources Android.

Créer et appliquer un style

Pour créer un nouveau style ou thème, ouvrez le res/values/styles.xml fichier de votre projet. Pour chaque style que vous souhaitez créer, suivez ces étapes :

1. Ajoutez un <style>élément avec un nom qui identifie de manière unique le style.
2. Ajoutez un <item>élément pour chaque attribut de style que vous souhaitez définir.
   Le namedans chaque élément spécifie un attribut que vous utiliseriez autrement comme attribut XML dans votre mise en page. La valeur de l' <item>élément est la valeur de cet attribut.

Par exemple, si vous définissez le style suivant :

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="GreenText" parent="TextAppearance.AppCompat">
        <item name="android:textColor">#00FF00</item>
    </style>
</resources>
```

Vous pouvez appliquer le style à une vue comme suit :

```
<TextView
   style="@style/GreenText"
   ... />
```

Chaque attribut spécifié dans le style est appliqué à cette vue si la vue l'accepte. La vue ignore simplement tous les attributs qu'elle n'accepte pas.

Remarque : Seul l'élément auquel vous ajoutez l' **style**attribut reçoit ces attributs de style ; les vues enfant n'appliquent pas les styles. Si vous souhaitez que les vues enfants héritent des styles, appliquez plutôt le style avec l' **android:theme**attribut.

Étendre et personnaliser un style

Lorsque vous créez vos propres styles, vous devez toujours étendre un style existant à partir du framework ou de la bibliothèque de support afin de maintenir la compatibilité avec les styles d'interface utilisateur de la plate-forme. Pour étendre un style, spécifiez le style que vous souhaitez étendre avec l' parentattribut. Vous pouvez ensuite remplacer les attributs de style hérités et en ajouter de nouveaux.

Par exemple, vous pouvez hériter de l'apparence du texte par défaut de la plateforme Android et la modifier comme suit :

```
<style name="GreenText" parent="@android:style/TextAppearance">
    <item name="android:textColor">#00FF00</item>
</style>
```

Cependant, vous devez toujours hériter de vos styles d'application principaux de la bibliothèque de support Android. Les styles de la bibliothèque de support assurent la compatibilité avec Android 4.0 (API niveau 14) et supérieur en optimisant chaque style pour les attributs d'interface utilisateur disponibles dans chaque version. Les styles de la bibliothèque de support ont souvent un nom similaire au style de la plate-forme, mais avec AppCompatinclus.

Pour hériter des styles d'une bibliothèque ou de votre propre projet, déclarez le nom du style parent *sans* la @android:style/partie indiquée ci-dessus. Par exemple, l'exemple suivant hérite des styles d'apparence de texte de la bibliothèque de support :

```
<style name="GreenText" parent="TextAppearance.AppCompat">
    <item name="android:textColor">#00FF00</item>
</style>
```

Vous pouvez également hériter de styles (sauf ceux de la plate-forme) en étendant le nom d'un style avec une notation par points, au lieu d'utiliser l' parentattribut. Autrement dit, préfixez le nom de votre style avec le nom du style dont vous souhaitez hériter, séparé par un point. Vous ne devez généralement le faire que lorsque vous étendez vos propres styles, et non les styles d'autres bibliothèques. Par exemple, le style suivant hérite de tous les styles du GreenTextstyle ci-dessus, puis augmente la taille du texte :

```
<style name="GreenText.Large">
    <item name="android:textSize">22dp</item>
</style>
```

Appliquer un style comme thème

Vous pouvez créer un thème de la même manière que vous créez des styles. La différence réside dans la façon dont vous l'appliquez : au lieu d'appliquer un style avec l' styleattribut sur une vue, vous appliquez un thème avec l' android:themeattribut sur la <application>balise ou une <activity>balise dans le AndroidManifest.xmlfichier.

Par exemple, voici comment appliquer le thème "sombre" de la conception matérielle de la bibliothèque de support Android à l'ensemble de l'application :

```
<manifest ... >
  <application android:theme="@style/Theme.AppCompat" ... >
  </application>
</manifest>
```

Et voici comment appliquer le thème "lumière" à une seule activité :

```
<manifest ... >
  <application ... >
    <activity android:theme="@style/Theme.AppCompat.Light" ... >
    </activity>
  </application>
</manifest>
```

Personnaliser le thème par défaut

Lorsque vous créez un projet avec Android Studio, il applique un thème de conception de matériau à votre application par défaut, tel que défini dans le styles.xmlfichier de votre projet. Ce AppThemestyle étend un thème de la bibliothèque de support et inclut des remplacements pour les attributs de couleur utilisés par les éléments clés de l'interface utilisateur, tels que la barre d'application et le bouton d'action flottant (le cas échéant). Vous pouvez ainsi personnaliser rapidement la conception des couleurs de votre application en mettant à jour les couleurs fournies.

Par exemple, votre styles.xmlfichier devrait ressembler à ceci :

```
<style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
    <!-- Customize your theme here. -->
    <item name="colorPrimary">@color/colorPrimary</item>
    <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
    <item name="colorAccent">@color/colorAccent</item>
</style>
```

Notez que les valeurs de style sont en fait des références à d'autres ressources de couleur , définies dans le res/values/colors.xmlfichier du projet. C'est donc le fichier que vous devez éditer pour changer les couleurs. Mais avant de commencer à modifier ces couleurs, prévisualisez vos couleurs avec l' outil de couleur de matériau . Cet outil vous aide à choisir des couleurs dans la palette de matériaux et à prévisualiser leur apparence dans une application.

Une fois que vous connaissez vos couleurs, mettez à jour les valeurs dans res/values/colors.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <!--   color for the app bar and other primary UI elements -->
    <color name="colorPrimary">#3F51B5</color>

    <!--   a darker variant of the primary color, used for
          the status bar (on Android 5.0+) and contextual app bars -->
    <color name="colorPrimaryDark">#303F9F</color>

    <!--   a secondary color for controls like checkboxes and text fields -->
    <color name="colorAccent">#FF4081</color>
</resources>
```

Et puis vous pouvez remplacer les autres styles que vous voulez. Par exemple, vous pouvez modifier la couleur d'arrière-plan de l'activité comme suit :

```
<style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
    ...
    <item name="android:windowBackground">@color/activityBackground</item>
</style>
```

Personnaliser les styles de widgets

Chaque widget du framework et de la bibliothèque de support a un style par défaut. Par exemple, lorsque vous stylisez votre application à l'aide d'un thème de la bibliothèque de support, une instance de Buttonest stylisée à l'aide du Widget.AppCompat.Buttonstyle. Si vous souhaitez appliquer un style de widget différent à un bouton, vous pouvez le faire avec l' styleattribut dans votre fichier de mise en page. Par exemple, ce qui suit applique le style de bouton sans bordure de la bibliothèque :

```
<Button
   style="@style/Widget.AppCompat.Button.Borderless"
   ... />
```

Et si vous souhaitez appliquer ce style à tous les boutons, vous pouvez le déclarer dans vos thèmes buttonStylecomme suit :

```
<style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
    <item name="buttonStyle">@style/Widget.AppCompat.Button.Borderless</item>
    ...
</style>
```

- **Utilisation des fichiers ressources**

Nous pouvons distinguer plusieurs types de ressources :

- les fichiers de ressources de l'application (images, chaînes de caractères, layout, xml) qui dépendent du contexte (langue française, taille de l'écran, mode portrait/paysage…) ;
- les bases de données ;
- les fichiers préférences ;
- les fichiers pour la lecture et la lecture/écriture sans contexte.

Toutes ces ressources se trouvent dans votre application sous res/.

Fichiers de ressources

Ces types de ressources sont celles utilisées par l'application et interprétées par le système Android. Elles sont de quatre types :

- res\drawable pour les images ;
- res\raw pour les ressources brutes, les fichiers sans contexte dont nous parlerons au chapitre suivant ;
- res\values pour les chaînes de caractères, les couleurs, les tableaux et les dimensions ;
- res\xml pour les fichiers XML.
-

Les chaînes simples s'utilisent, soit dans le code :

```
getString(R.string.simpleString)
```

soit dans le `layout.xml` pour déclarer un composant en utilisant son identifiant :

```
<TextView
  android:id="@+id/simpleStringTV"
  android:layout_width="fill_parent"
  android:layout_height="wrap_content"
  android:text="@string/hello"
  />
```

Les chaînes avec paramètres s'utilisent avec un formateur.

Dans le fichier *strings.xml*, nous avons déclaré la chaîne :

```
<!--String with parameters ($s for string, $d for decimal (any decimal:integer, double, float...)-->
    <string name="param_message">Hello, i love %1$s, i hate %2$s, i am %3$d years old</string>
```

Nous la manipulons ainsi en Java :

```
// Then instanciate a string with paremeter
String
parameterString=String.format(getString(R.string.param_message),"peace","war",2);
// And set the text view
TextView textViewWithParam=(TextView)findViewById(R.id.parameterStringTV);

    textViewWithParam.setText(parameterString);
```

Les chaînes peuvent être regroupées sous forme de tableaux :

Sélectionnez

```
<!--String array-->
 <string-array name="i_like_array">
    <item>chocolate</item>
    <item>television</item>
    <item>internet</item>
    <item>nicotine</item>
    <item>hug</item>
    <item>Santa Claus</item>
 </string-array>
```

Dans le code, celui-ci se manipule ainsi :

```
// Instanciate the resources object:
Resources resources=getResources();

    String[] planets = resources.getStringArray(R.array.i_like_array);
```

Les chaînes peuvent aussi posséder un genre singulier ou pluriel :

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <plurals name="numberOfSongsAvailable">
    <item quantity="one">One song found.</item>
    <item quantity="other">%d songs found.</item>
  </plurals>

    </resources>
```

et s'utilisent dans le code ainsi :

```
int count = getNumberOfHuman();
Resources res = getResources();
    String humans = res.getQuantityString(R.plurals.singleOrPlural, count);
```

Il faut faire attention aux signes ` et "

```
<string name="good_example">"This'll work"</string>
<string name="good_example_2">This\'ll also work</string>
    <string name="bad_example">This doesn't work</string>
```

Les image

Les formats acceptés sont PNG et JPEG (GIF n'est pas recommandé).

Il suffit de déposer les fichiers image dans res\drawable pour qu'ils soient référencés par le système.

Ainsi l'image res/drawable/myimage.png peut être utilisée,

soit dans le code :

```
Resources res = getResources();

        Drawable drawable = res.getDrawable(R.drawable.myimage);
```

soit dans la description des IHM dans le fichier de layout :

```
<ImageView
   android:layout_height="wrap_content"
   android:layout_width="wrap_content"

        android:src="@drawable/myimage" />
```

- **Intégration de Material design**

Material Design, c'est quoi ?

Tout d'abord, je vais vous dire c'est quoi le Material design.

Le Material Design est un ensemble de règles de design proposées par Google et qui s'appliquent à l'interface graphique des logiciels et applications. Il est utilisé notamment à partir de la version 5.0 du système d'exploitation Android.


12:30
Sasha Phillips
On my way.
12:30 pm
Josh Kenny
You: Sounds good.
11:41 am
Roommates
Andrea: Going to Kevin's tonight?
10:28 am
Peter Cheng
You: What time?
9:52 am
Movie Night
Russell sent a sticker.
Mon
Lisa Benet
How was your trip?
Mon
Anthony Tran
Thanks.
12:30
Sasha Phillips
Active now
12:30 PM
On my way.
Should be there in 5 minutes.
OK, just finishing up a few things.
I'm here. Can you come outside?
Be right there!
Write a message
Aa

Google a présenté le Material Design pour la première fois lors de la conférence Google I/O, le 25 juin 2014. En misant sur les motifs « carte », déjà utilisés dans Google Now, ces règles de design mettent l'accent sur une utilisation accrue des mises en page basées sur une grille, des animations et des transitions, des effets de profondeur tels l'éclairage et les ombres. Selon Google ce nouveau langage de design est basé sur le papier et l'encre.

Le designer Matías Duarte explique que « contrairement au vrai papier, notre matériau numérique peut s'étirer et se modifier de manière intelligente. Le matériau contextuel a une surface physique et des bords. Les superpositions et les ombres donnent des informations sur ce que vous pouvez toucher »

Si vous vous comprenez bien, le material design proposé par l'entreprise Américaine Google, n'est pas seulement pour les Android, mais pour tout logiciel et application. Dans notre cas nous allons l'utiliser dans Android, dans le but de construire des interfaces plus beaux et plus simples.

Grâce aux éléments que compose ce nouveau style de design, vous allez faire des interfaces avec des cards, ombres, animations et plein des trucs géniaux.

Comment intégrer dans Android Studio ?

Android studio est connu comme l'IDE officiel pour le développement des applications Android. Il contient des outils nécessaire pour un bon développement : un éditeur de code intelligent, interface graphique, un émulateur Android, et bien plus d'autres outils. Je vais publier le plus tôt possible un article sur l'installation de ce merveillleux logiciel.

Sans plus tarder, si vous avez déjà installé Android Studio, c'est déjà une bonne chose. Au moment de la rédaction de cet article, Android Studio est à la version 3.6.1, télécharger la dernière version d'Android studio sur le site officiel.

Si vous n'avais pas encore de projet, créez un nouveau projet via Fichier > Nouveau > Nouveau Projet

Dès que votre projet est prêt, vous pouvez ajouter les dépendences ou les libraires des Material Design dans le fichier Gradle de votre application se trouvant sur : build.gradle (Module: app)

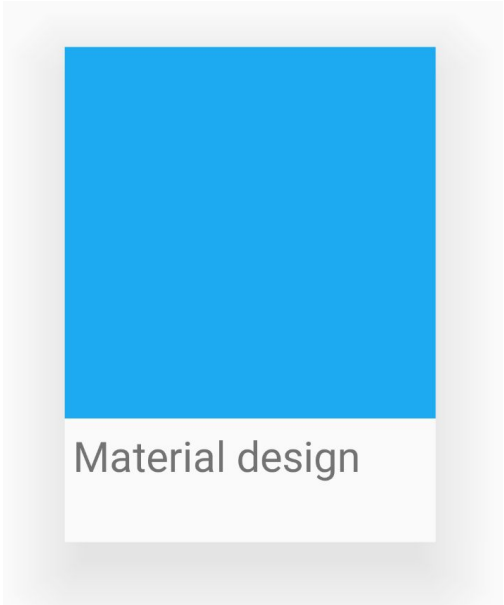
```
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])

    implementation 'androidx.appcompat:appcompat:1.1.0'
    implementation 'androidx.recyclerview:recyclerview:1.1.0'
    implementation 'androidx.legacy:legacy-support-v4:1.0.0'
    implementation 'androidx.cardview:cardview:1.0.0'
    implementation 'com.google.android.material:material:1.1.0'
```

Le plus important de tous : `implementation 'com.google.android.material:material:1.1.0'`, car il contient à lui seul beaucoup des composants comme des Boutons, des ViewPager, et bien plus encore.

Après avoir ajouté les lignes suivantes, cliquer sur Sync Now pour faire un Build de votre projet.

Et voilà, tout reste maintenant à passer au bon codage plus simple et avec des meilleurs outils.

un exemple :



un Joli Card avec du text.

xml :

```
<com.google.android.material.card.MaterialCardView
    android:layout_width="300dp"
    android:elevation="5dp"
    android:layout_height="400dp">
    <LinearLayout
        android:orientation="vertical"
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <ImageView
            android:background="#03A9F4"
            android:layout_width="match_parent"
            android:layout_height="300dp"/>

        <TextView
            android:padding="7dp"
            android:text="Material design"
            style="@style/TextAppearance.AppCompat.Display1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"/>
    </LinearLayout>

</com.google.android.material.card.MaterialCardView>
```

- **Maîtrise des Canvas**

Préparer son crayon / pinceau Pour dessiner sur un Canvas
il faut utiliser un pinceau/crayon Classe Paint

LA CLASSE PAINT

La classe Paint est utilisée pour déterminer la nature du pinceau utilisé pour dessiner sur un Canvas

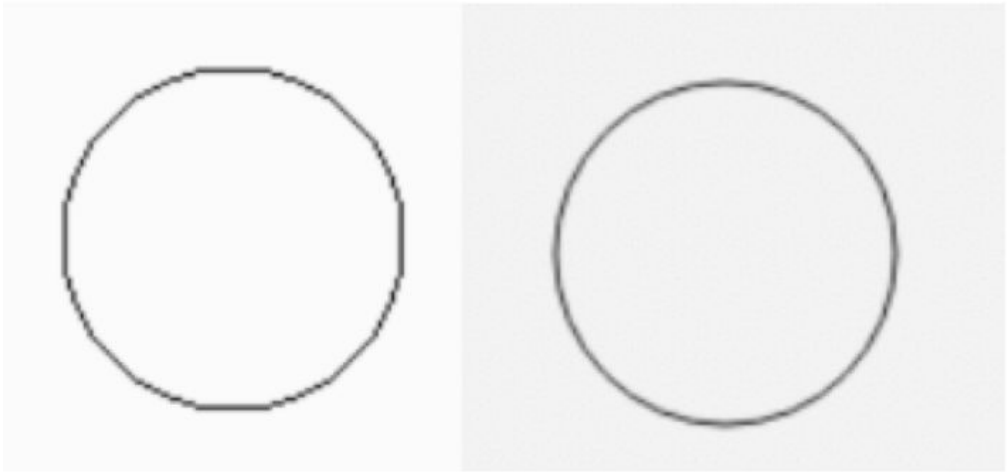
Déclaration :
Paint paint = new Paint();

Type de pinceaux : Qualité de la couleur (en cas de dégradé par exemple) :

paint.setDither(false/true);

Qualité du pinceau (lissage ou pixélisé) :

paint.setAntiAlias(false/true);



Couleur du pinceau :

paint.setColor(Color.BLUE);

Choix de la couleur :

```
0xTTRRGGBB
Color.parseColor("blue")
Color.parseColor("#TTRRGGBB")
Color.parseColor("#AA000099")
...
```

TT : Transparence
RR : Rouge
GG : Vert
BB : Bleur

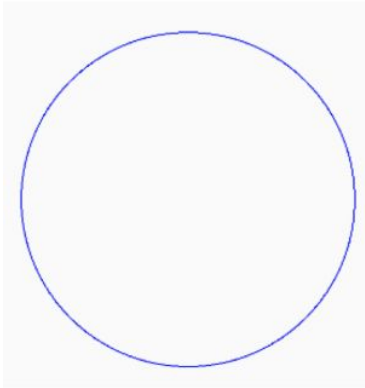
00 → Invisible
FF → Non transpa

Type de pinceaux :

Nature du trait :

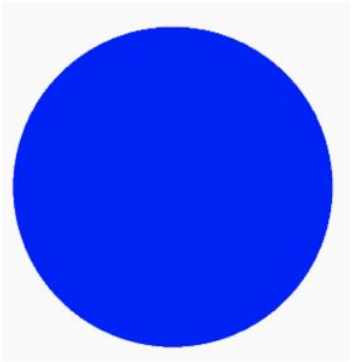
- Contour

paint.setStyle(Style.STROKE);

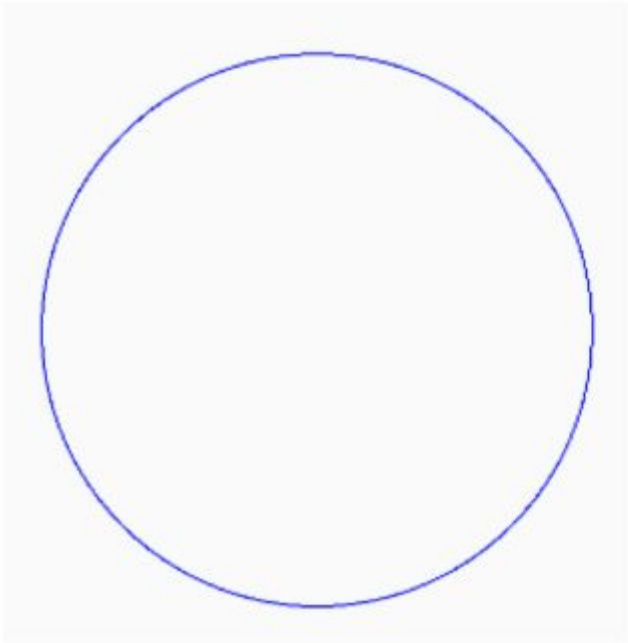


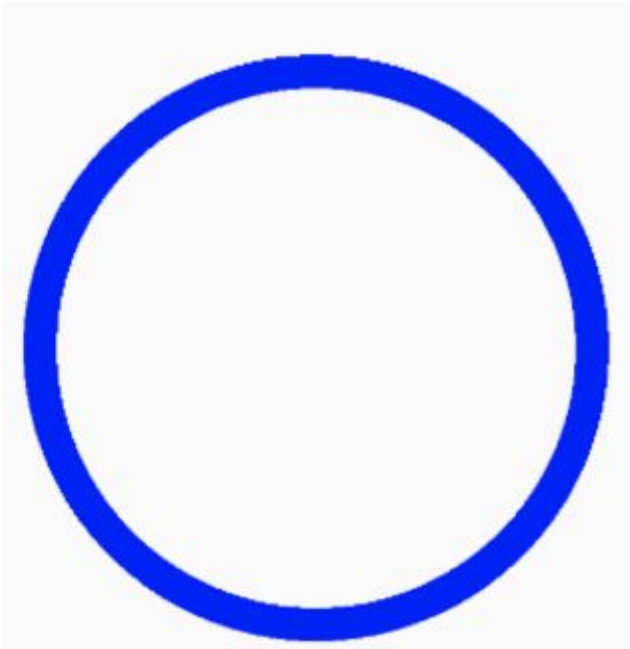
- Remplissage

paint.setStyle(Style.FILL);



Taille du pinceau : paint.setStrokeWidth(4.0f);

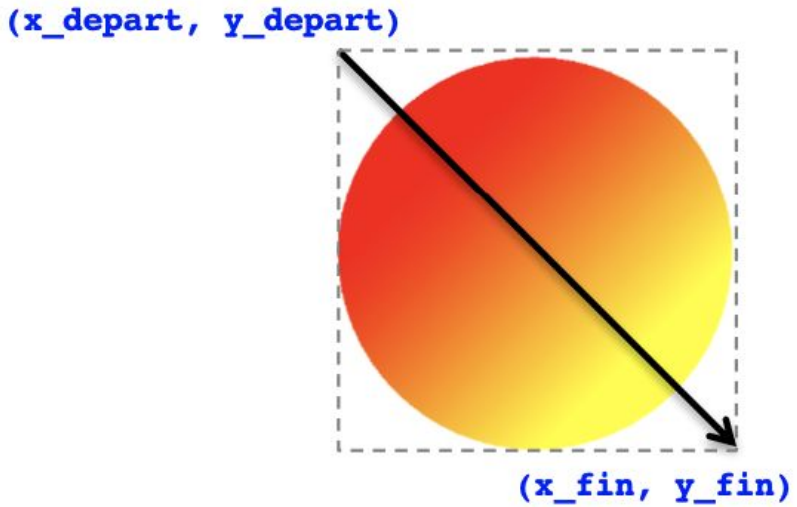




Type de pinceaux :

Dégradé (linéaire) :

```
paint.setShader( new LinearGradient( x_depart, y_depart ,
                        x_fin, y_fin, couleur1, couleur2, Shader.TileMode.CLAMP ) );
```



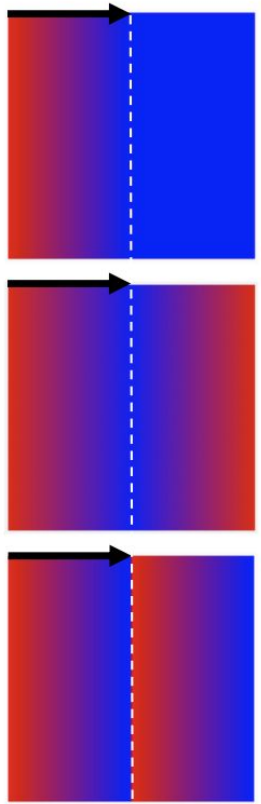
Type de pinceaux :

Dégradé (linéaire) :

¢ Shader.TileMode.CLAMP

¢ Shader.TileMode.MIRROR

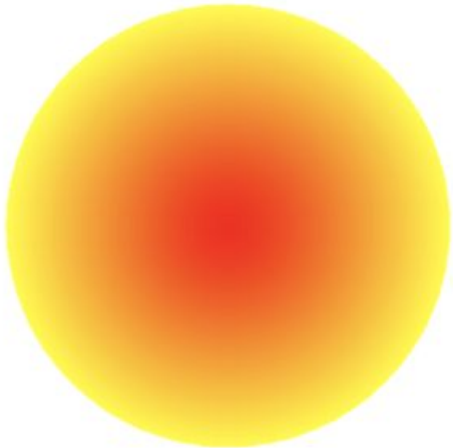
¢ Shader.TileMode.REPEAT



Type de pinceaux :

Dégradé (radial) :

paint.setShader(new RadialGradient(float x_centre, float y_centre, rayon, couleur1, couleur2, Shader.TileMode.CLAMP));



Type de pinceaux :

Dégradé (linéaire) :

paint.setShader(new SweepGradient(x_centre , y_centre, couleur1, couleur2));

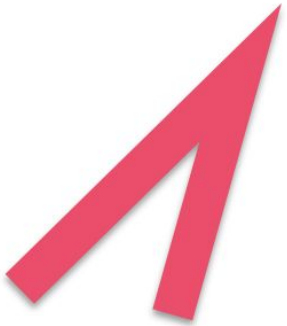
Type de pinceaux :

Transparence paint.setAlpha(x);

x = 0 .. 255

Type de pinceaux :

Nature des connexions entre les lignes paint.setStrokeCap(Paint.Cap.SQUARE);



Paint.Cup.BUTT

Onglet

Paint.Cup.ROUND

Arrondi

Paint.Cup.SQUARE

Plaque

Il est possible de dessiner sur un Bitmap

1. Création d'un Bitmap 100x100 : Bitmap bitmap = Bitmap.createBitmap(100, 100, Bitmap.Config.ARGB_8888);

2. Créer le Canvas du Bitmap : Canvas canvas = new Canvas(); canvas.setBitmap(bitmap); ou : Canvas canvas = new Canvas(bitmap);

3. Dessiner sur le Canvas du bitmap : canvas.drawRect(…

4. Afficher le Bitmap à l'aide d'un composant (View) : imageView.setImageBitmap(bitmap);

Dessiner une ligne : entre (x1, y1) et (x2, y2)

canvas.drawLine(0, 0, 100, 0, paint);

Dessiner un rectangle : entre (x1, y1) et (x2, y2)

canvas.drawRect(10, 10, 110, 110, paint);

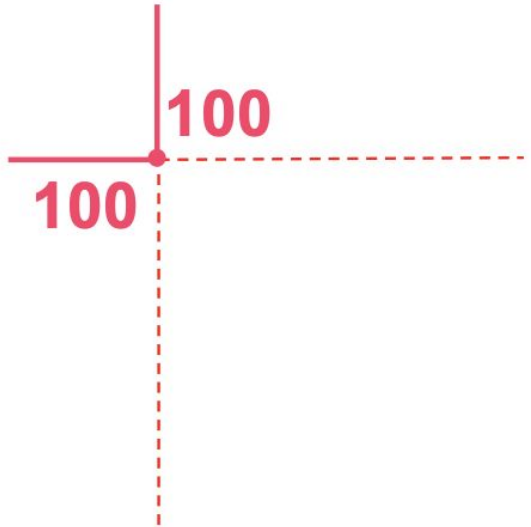
Dessiner un rectangle : entre (x1, y1) et (x2, y2)

canvas.drawOval(10, 10, 110, 110, paint);

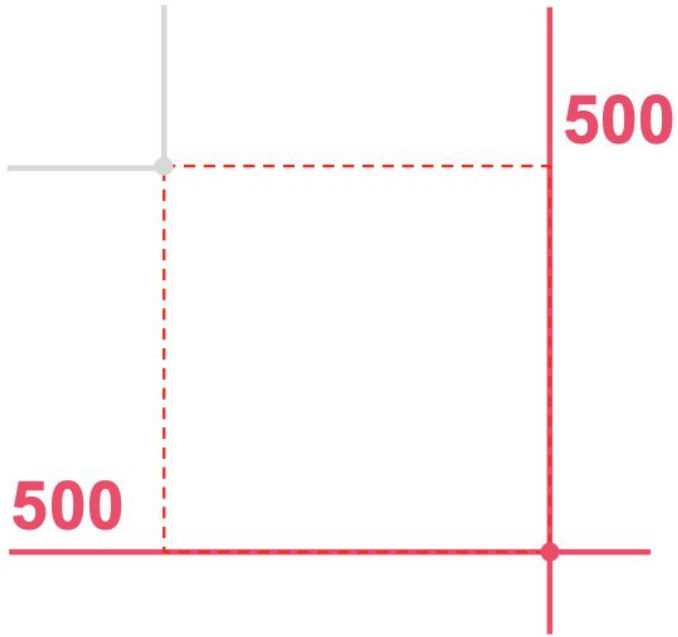
Dessiner un cercle : au centre (x1, y1) et de rayon r

canvas.drawCircle(300, 300, 250, paint);

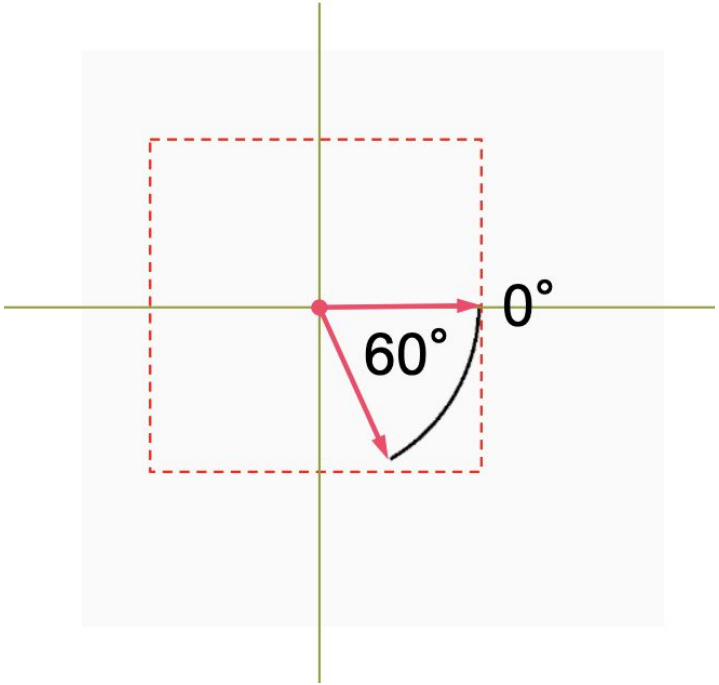
Dessiner un arc canvas.drawArc(100, 100, 500, 500, 0, 60, false, paint1);



Dessiner un arc canvas.drawArc(100, 100, 500, 500, 0, 60, false, paint1);



Dessiner un arc canvas.drawArc(100, 100, 500, 500, **0**, **60**, false, paint1);

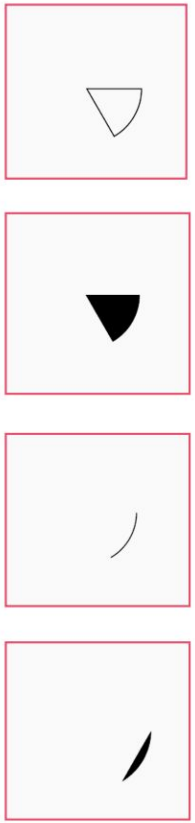


Dessiner un arc 1 canvas.drawArc(100, 100, 500, 500, 0, 60, true, paint1);

Dessiner un arc 2 canvas.drawArc(100, 100, 500, 500, 0, 60, true, paint2);

Dessiner un arc 3 canvas.drawArc(100, 100, 500, 500, 0, 60, false, paint1);

Dessiner un arc 4 canvas.drawArc(100, 100, 500, 500, 0, 60, false, paint2);



Dessiner du texte :

```
canvas.drawText("Bonjour", 100, 100, paint);
```

Paramètres du texte :

```
paint.setTextSize(22);
paint.setTextAlign(Paint.Align.LEFT);
paint.setFakeBoldText(true);
```

Dessiner un Bitmap :

```
Bitmap bitmap = ...
```

ou une image des ressources :

```
Bitmap bitmap = BitmapFactory.decodeResource(getResources(),
R.drawable.nom_de_image);

canvas.drawBitmap(bitmap, 100, 100, null);
```

Pour les images pixélisées Pas possible de redimensionner l'image

Dessiner un Drawable :

```
Drawable drawable = getResources().getDrawable(R.drawable.img, null);
drawable.setBounds(10, 10, 200, 200); drawable.draw(canvas);
```

Dessiner une figure (path) :

```
Path path = new Path();
path.reset();
path.moveTo(x_1, y_1);
path.arcTo(...);
path.cubicTo(...);
path.quadTo(...);
... path.close();

canvas.drawPath(path, paint);
```

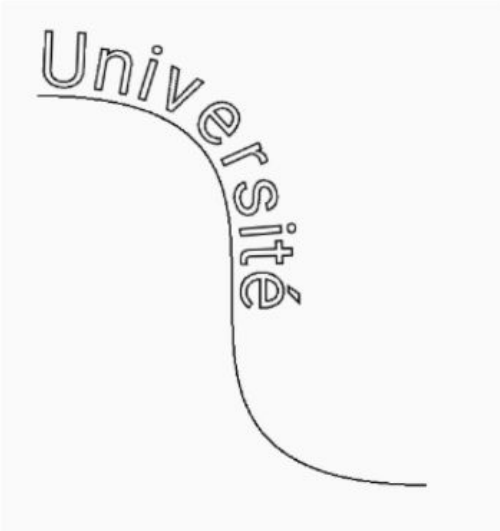
```
Path path = new Path();
path.reset();
path.moveTo(30 , 0);
path.lineTo(70, 0);
path.quadTo(100,0,100,50);
path.quadTo(70,50,70,100);
path.lineTo(30, 100);
path.arcTo(-30,70,30,130, 0, -90, false);
path.lineTo(0, 30);
path.cubicTo(30,30,0,0, 30, 0);

canvas.drawPath(path, paint);
```

LA CLASSE RECTF

```
 RectF rectF = new RectF();

rectF.left = 100;
rectF.top = 100;
rectF.right = 500;
rectF.bottom = 500;

ou RectF rectF = new RectF(100, 100, 500, 500);


canvas.drawRect(rectF, paint);
```

DRAWTEXTONPATH

```
Paint paint = new Paint();
paint.setColor(Color.BLACK);
paint.setStyle(Paint.Style.STROKE);
paint.setStrokeWidth(2);
paint.setTextSize(80);

Path path = new Path();
path.reset();
path.moveTo(100,100);
path.cubicTo(500, 100, 100, 500, 500, 500);
canvas.drawPath(path, paint);

canvas.drawTextOnPath("Université", path, 0, -10, paint);
```



https://github.com/NizarETH/MyCanvas

https://lottiefiles.com/animation/circle

- **Les animations (Définition, utilisation et maîtrise)**

Animations personnalisées et effets de transition entre les fragments dans Android

Naviguer entre les fragments peut parfois être déroutant. Avec les animations de transition personnalisées, si un utilisateur se dirige vers une nouvelle destination, nous pouvons ajouter l'animation de transition respective et descriptive. De plus, lorsque vous revenez au fragment précédent, nous pouvons animer l'action.

Nous pouvons ajouter des transitions personnalisées pour animer l'apparition et la suppression des fragments de dialogue. De plus, nous pouvons inclure des éléments de transition partagés dans notre application pour ouvrir une image dans une nouvelle destination.

Les animations de transition améliorent généralement l'expérience utilisateur (UX) de l'application, ce qui permet de fidéliser les utilisateurs.

Transition animations

Les animations de transition peuvent être de 4 types :

- **Entrez** - apportant un nouveau fragment à NavHostFragment.
- **Quitter** - suppression du fragment actuellement affiché de NavHostFragment.
- **Pop Enter** - lors du retour, cela ramènera le fragment précédent.
- **Pop Exit** - cela sortira du fragment pour laisser de la place au fragment précédent pour être visible.

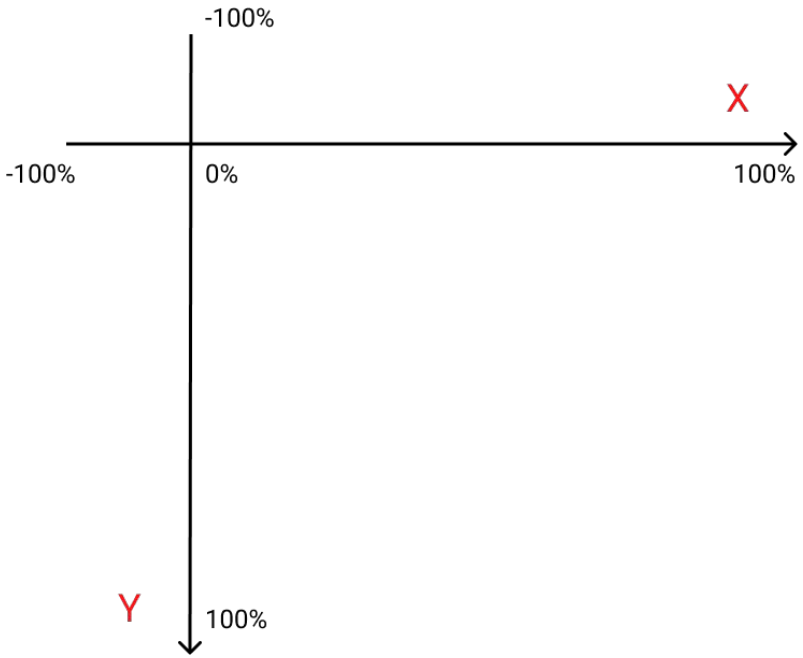
Nous pouvons également définir une transition d'élément partagé qui anime le mouvement d'une image cliquée vers une nouvelle destination. Ceci est utile lorsque vous avez des images et que vous souhaitez accéder aux détails d'une image particulière. L'image se développe dans une nouvelle destination.

Avec les dialogues, nous pouvons animer leur mouvement lors de leur affichage et lors de leur fermeture. Nous pouvons définir `slide-up` et faire `slide-down` la transition des animations.

Lors de la traduction de différents éléments, c'est-à-dire de gauche, de droite, de haut et de bas, nous pouvons utiliser les attributs suivants dans nos fichiers de ressources d'animation :

- `fromXDelta`- indique à partir de quelle valeur de l'axe X nous passons.
- `toXDelta`- indique jusqu'à quelle position dans l'axe X.
- `fromYDelta`- indique à partir de quelle valeur de l'axe Y.
- `toYDelta`- indique à quelle valeur de l'axe Y nous passons.
- `duration`en millisecondes - c'est le temps nécessaire pour qu'une animation se produise.

Graphique d'animation de transition



Création de transitions

Saisie d'un fragment

Cela implique de mettre en vue un nouveau Fragment. Ce nouveau Fragment entrera par le côté gauche. Nous devons donc créer une anim appelée `from_left`. Pour ce faire, cliquez avec le bouton droit sur le répertoire anim et sélectionnez `new >> animation layout file`.

```
<set xmlns:android="http://schemas.android.com/apk/res/android">

   <translate android:fromXDelta="-100%" android:toXDelta="0%"
android:duration="700"/>

</set>
```

Revenir au fragment précédent

Tout d'abord, nous devrons supprimer le fragment actuellement affiché. Nous allons créer une animation appelée `to_left`qui supprimera le Fragment vers le côté gauche.

```
<?xml version="1.0" encoding="utf-8"?>

<set xmlns:android="http://schemas.android.com/apk/res/android">

    <translate android:fromXDelta="0%" android:toXDelta="-100%"
android:duration="700"/>

</set>
```

TP
(Custom animation Fragments)

- **Présentation et création des différents effets d’animation**,

Déplacer une vue avec animation

Les objets à l'écran devront souvent être repositionnés. Cela peut se produire en raison de l'interaction de l'utilisateur ou d'un traitement effectué en coulisses. Au lieu de mettre à jour immédiatement la position des objets, ce qui le ferait clignoter d'une zone à l'autre, vous devez utiliser une animation pour le déplacer de la position de départ à sa position finale.

Android fournit des moyens qui vous permettent de repositionner vos objets de vue à l'écran, tels que ObjectAnimator . Vous pouvez fournir la position finale sur laquelle vous souhaitez que l'objet se pose, ainsi que la durée de l'animation. Vous pouvez combiner cela avec des interpolateurs temporels pour contrôler l'accélération ou la décélération de l'animation.

Changer la position de la vue avec ObjectAnimator

L' ObjectAnimatorAPI fournit un moyen simple de modifier les propriétés d'une vue avec une durée spécifiée. Il contient des méthodes statiques pour créer des instances de ObjectAnimatorselon le type d'attribut que vous animez. Lors du repositionnement de vos vues à l'écran, vous utiliserez les attributs translationXet .translationY

Voici un exemple de ObjectAnimatorqui déplace la vue à 100 pixels de la gauche de l'écran en 2 secondes :

```
ObjectAnimator animation = ObjectAnimator.ofFloat(view, "translationX", 100f);
animation.setDuration(2000);
animation.start();
```

Cet exemple utilise la ObjectAnimator.ofFloat() méthode puisque les valeurs de traduction doivent être des flottants. Le premier paramètre est la vue que vous souhaitez animer. Le deuxième paramètre est la propriété que vous animez. Comme la vue doit être déplacée horizontalement, la translationXpropriété est utilisée. Le dernier paramètre est la valeur finale de l'animation. Étant donné que cette valeur est de 100, ce sera à ce nombre de pixels de la gauche de l'écran.

La méthode suivante spécifie combien de temps l'animation doit prendre en millisecondes. Dans cet exemple, l'animation durera 2 secondes (2000 millisecondes).

La dernière méthode provoque l'exécution de l'animation qui mettra à jour la position de la vue à l'écran.

Pour plus d'informations sur l'utilisation de ObjectAnimator, consultez Animation avec ObjectAnimator .

Code :

https://www.tutorialspoint.com/android/android_animations.htm

C’est la fin du module

Merci !