# Navigation entre activités

# Plan

- Principes
- Navigation entre écrans
- Passage de données entre écrans
- Appeler d’autres applications
- Exécuter une action

# Principes

- La navigation entre activités sous Android est réalisé par des intents
- Chaque application est découpée en plusieurs activités
- Chaque activité est accessible à l’aide d’un intent
- Avec les intents, on définit la logique de navigation de l’application

# Navigation entre écrans

- Afin de pouvoir exécuter une activité, il faut utiliser la méthode startActivity
  - public void startActivity(Intent intent)
- Cette méthode utilise  un intent qui correspond à l'activité qu'on souhaite afficher
- Création d'un intent
  - Intent intent = new Intent(Context context, <Class?>ActivityToLauch)
  - startActivity(intent)

# Exercice 1

- Écrire une application qui crée trois activités avec trois layouts.
  - La première activité : l'activité principale affiche l'écran de la figure 1
  - La deuxième activité : affiche l'écran de la figure 2
  - La troisième activité : affiche l'écran de la figure 3
  - Quand on clique sur le bouton Nouvel Employé de la première activité, ça affiche l'activité 2 et quand on clique sur le bouton Liste des employés, ça affiche lactivité 3.

fig1


fig2

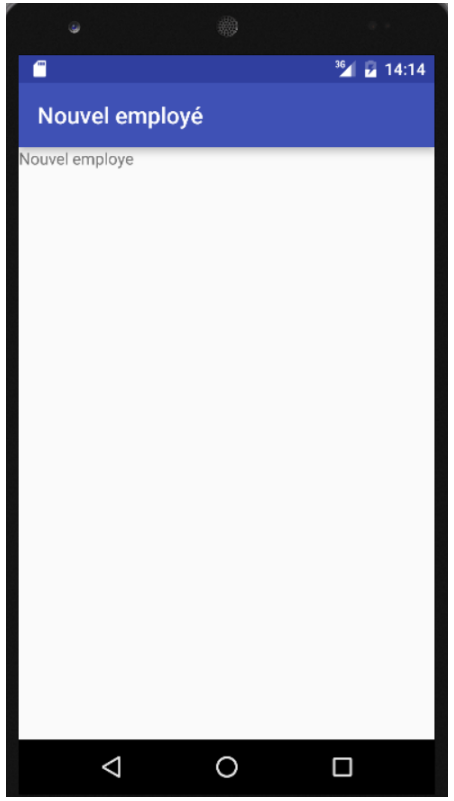
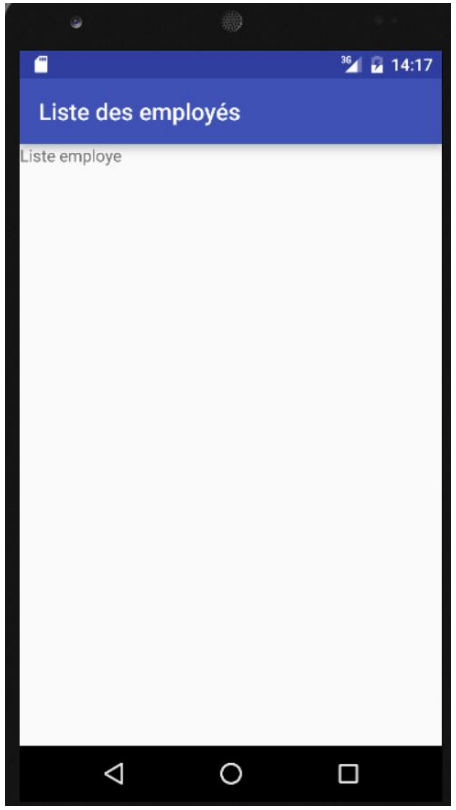



fig3



# Solution : strings.xml

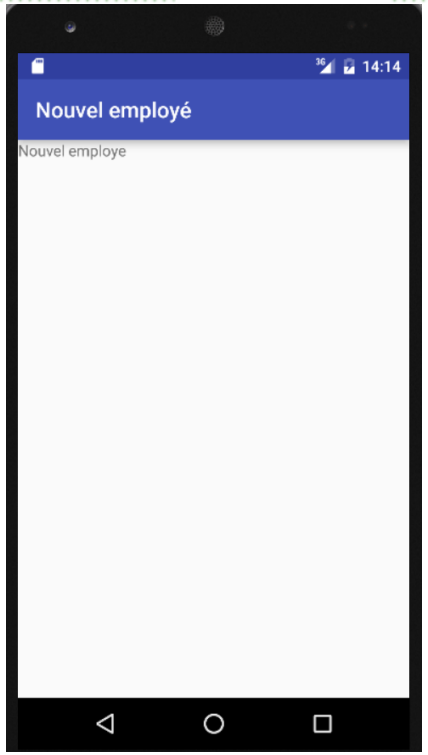
```
<resources>
    <string name="app_name">Gestion des employés</string>
    <string name="titreFenetreNouvelEmploye">Nouvel employé</string>
    <string name="titreFenetreListeEmploye">Liste des employés</string>
    <string name="text_button_Nouveau">Nouvel employé</string>
    <string name="text_button_List">Liste des employés</string>
</resources>
```

# Solution – main.xml (Layout de l'activité principale)

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="match_parent">

    <Button
        android:id="@+id/idButtonNouvelEmploye"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/text_button_Nouveau"/>
    <Button
        android:id="@+id/idButtonListeEmploye"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/text_button_List"/>


</LinearLayout>
```

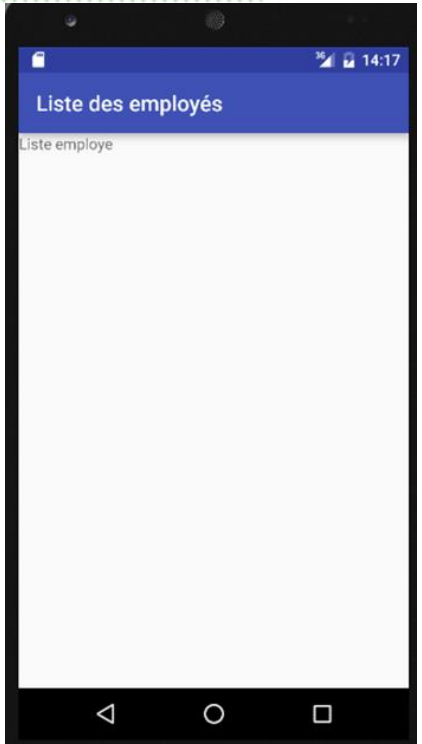
# Solution – nouvel_employe.xml (Layout de l'activité ActivityNouvelEmploye)

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.kfostine.navigationentreecrans.ActivityNouvelEmploye">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Nouvel employe" />
</RelativeLayout>
```



# Solution – liste_employe.xml (Layout de l'activité ActivityListeEmploye)

```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.kfostine.navigationentreecrans.ActivityListeEmploye">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Liste employe" />
</RelativeLayout>
```



# Solution : ActivityNouvelEmploye.java

```
package com.example.kfostine.navigationentreecrans;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class ActivityNouvelEmploye extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.nouveau_employe);
    }
}
```

# Solution : ActivityListeEmploye.java

```
package com.example.kfostine.navigationentreecrans;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class ActivityListeEmploye extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.liste_employe);
    }
}
```

# Solution : manifests/AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.kfostine.navigationentreecrans">
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".ActivityNouvelEmploye"
            android:label="@string/titreFenetreNouvelEmploye">
        </activity>
        <activity android:name=".ActivityListeEmploye"
            android:label="@string/titreFenetreListeEmploye">
        </activity>
    </application>
</manifest>
```

Activité principale

# Solution : MainActivity.java

```
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Button cmdNouvelEmploye = (Button) findViewById(R.id.idButtonNouvelEmploye);
        Button cmdListeEmploye = (Button) findViewById(R.id.idButtonListeEmploye);
        cmdNouvelEmploye.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(MainActivity.this, ActivityNouvelEmploye.class);
                startActivity(intent);
            }
        });
        cmdListeEmploye.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(MainActivity.this, ActivityListeEmploye.class);
                startActivity(intent);
            }
        });
    }
}
```

# Passage de données entre écran

- Le passage de données entre différentes vues d'une application se fait à l'aide des extras (portés par les intents)
- Un extra est un couple clé/valeur utilisant le système de Bundle
- La méthode putExtra de Intent permet de passer plusieurs types de valeurs

# Exercice 2

- Créer une deux activités Android
- La première(Activité principale) affiche la figure 1 et demande à l'utilisateur son nom et prénom et quand on clique sur le bouton « Afficher » la 2e activité (fig2) est affichée avec le message « Bienvenue nom et prénom ».

fig1




fig2

# Solution : strings.xml

```
<resources>
    <string name="app_name">Passage de données entre activités</string>
    <string name="text_nom">Votre nom</string>
    <string name="text_prenom">Votre prénom</string>
    <string name="text_afficher">Afficher</string>
    <string name="text_bienvenue">Bienvenue</string>
</resources>
```

# Solution : main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="match_parent">

    <EditText
        android:id="@+id/idEditTextNom"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/text_nom"/>
    <EditText
        android:id="@+id/idEditTextPrenom"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/text_prenom"/>

    <Button
        android:id="@+id/idButtonAfficher"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/text_afficher"/>
</LinearLayout>
```

# Solution : bienvenue.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="16dp"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:paddingTop="16dp"
    tools:context="com.example.kfostine.passagedonneessimplesactivity.ActivityBienvenue">

    <TextView
        android:id="@+id/idTextViewBienvenue"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
       />
</RelativeLayout>
```

# Solution : MainActivity.java

```
public class MainActivity extends AppCompatActivity {
    Button btnAfficher;
    EditText txtNom;
    EditText txtPrenom;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        btnAfficher = (Button) findViewById(R.id.idButtonAfficher);
        btnAfficher.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                txtNom = (EditText) findViewById(R.id.idEditTextNom);
                txtPrenom = (EditText) findViewById(R.id.idEditTextPrenom);
                Intent intent = new Intent(MainActivity.this, ActivityBienvenue.class);
                intent.putExtra("lastName", txtNom.getText().toString());
                intent.putExtra("firstName", txtPrenom.getText().toString());
                startActivity(intent);
            }
        });
    }
}
```

# Solution : ActivityBienvenue.java

```
public class ActivityBienvenue extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);
        setContentView(R.layout.bienvenue);
        Intent intent = getIntent();
        String nom = intent.getStringExtra("lastName");
        String prenom = intent.getStringExtra("firstName");
        TextView textView = (TextView) findViewById(R.id.idTextViewBienvenue);
        String message = getString(R.string.text_bienvenue) + " " + nom + " " + prenom;
        textView.setText(message);

    }
}
```

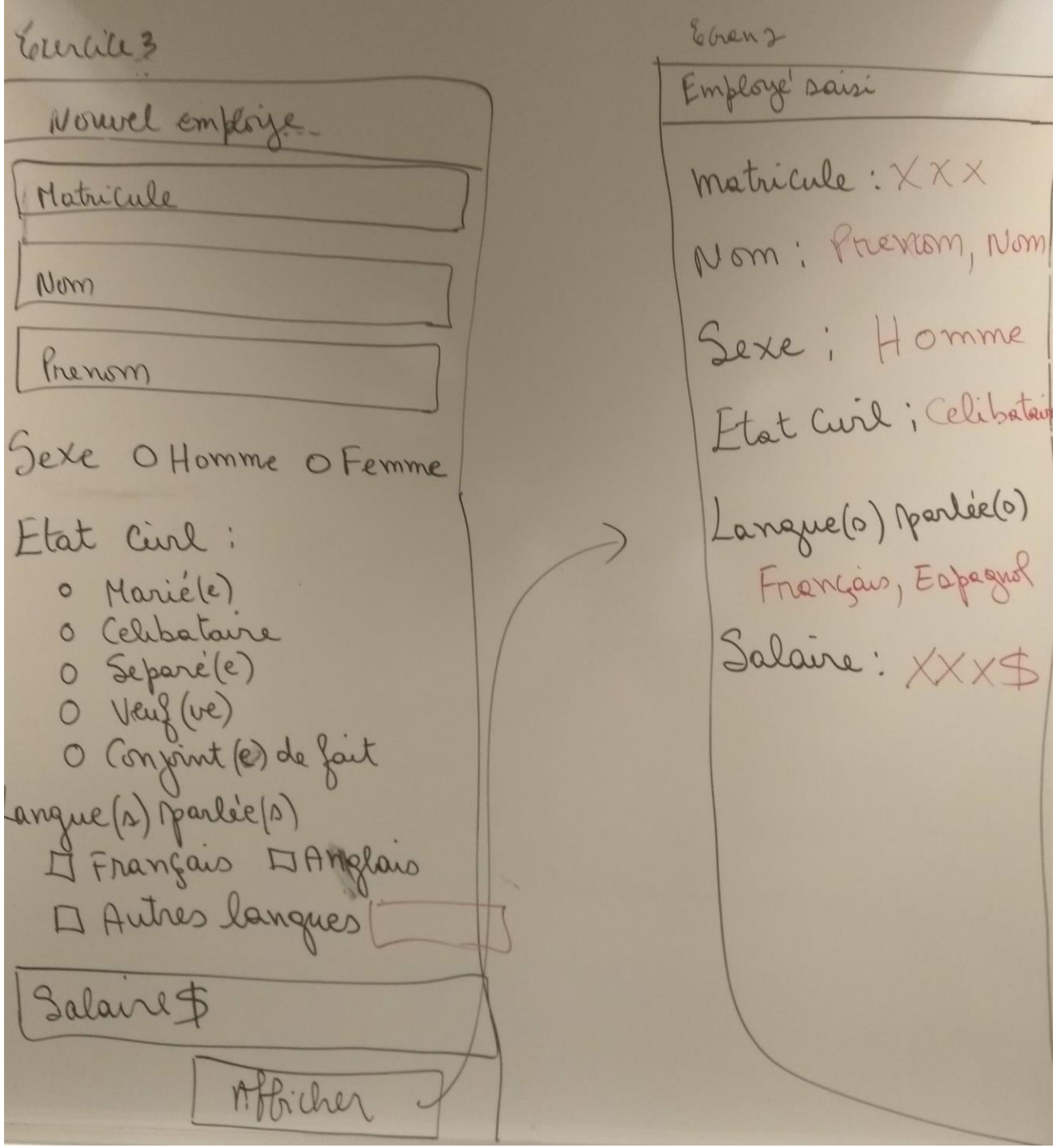
# Solution : manifests/AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.kfostine.passagedonneessimplesactivity">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="Passage de données entre activités"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".ActivityBienvenue"></activity>
    </application>

</manifest>
```

# Exercice 3

- Réaliser une application Android comportant deux activités (voir la diapo suivante :
  - L’activité principale demande à l’utilisateur de saisir les informations d’un nouvel employé puis de cliquer sur le bouton Afficher
  - Le bouton « afficher » appel une autre activité qui affiche les informations de l’employé

# Exercice 3


Exercice 3
Nouvel employé
Matricule
Nom
Prenom
Sexe O Homme O Femme
Etat Civil :
o Marié(e)
o Celibataire
o Séparé(e)
o Veuf (ve)
o Conjoint (e) de fait
Langue(s) parlée(s)
Français
Anglais
Autres langues
Salaire $
Afficher
Ecran 2
Employé saisi
Matricule : XXX
Nom : Prenom, Nom
Sexe : Homme
Etat Civil : Celibataire
Langue(s) parlée(s)
Français, Espagnol
Salaire : XXX$

# Passage de données (objet)

- Pour passer des objets, il faut utiliser une classe serialisées

```
public class User implements Serializable{
    private String user;
    private String password;
    ...
}
```

- Dans l'objet Intent, il faut appeler la méthode putExtra
  - User user = new User("user1", "p@ssword2017");
  - Intent intent = new Intent(this, NewActivity.this)
  - intent.putExtra("monUser", user);
- Dans l'activité cible, il faut récupérer l'objet avec la méthode :
  - Intent intent = getIntent();
  - User user = intent.getSerializableExtra("monUser");

# Exercice 4

- Modifier l’exercice 3 de manière à passer un objet Employe d’une activité à l’autre.

# Appel d’autres applications

- Android permet d’utiliser d’autres applications disponibles sur un appareil afin d’effectuer un traitement
- Par exemple, on peut appeler l’«application de paramètres d’Android» afin de demander à l’utilisateur d’activer son GPS.
- La manière la plus simple d’appeler une autre application est d’utiliser le PackageManager d’Android.

```
PackageManager pm = getPackageManager();
Intent intent = pm.getIntentForPackage("com.android.settings");
if(intent != null){
    startActivity(intent);
}
else{
    //Affichage d'une boite de dialogue
    Toast toast = Toast.makeText(this, "Package non trouvé", Toast.LENGTH_LONG);
    toast.show();
}
```

# Exercice 5

- Écrire un programme Android qui affiche un bouton « Afficher les paramètres », quand on clique sur le bouton, l'application appel l'application des paramètres d'Android dont le code est défini plus haut.

# Éxécuter une action

- Une application Android peut vous proposer une liste d'application en fonction du type d'action à exécuter
- Exemple
  - On aimerait ouvrir un fichier PDF avec une application installée sur l'appareil
  - On ne connait pas le nom de l'application, ni l'identifiant du package
  - Pour cela, il faut utiliser une autre spécificité des intents (**Une action**)

```
File file = new File("/sdcard/fichier.pdf");
Uri path = Uri.fromFile(file);
Intent intent = new Intent(Intent.ACTION_VIEW);
intent.setDataAndType(path, "application/pdf");
PackageManager  pm = getPackageManager();
ComponentName component = intent.resolveActivity(pm);

if(component == null){
    Toast toast = Toast.makeText(MainActivity.this,
            "Aucune application disponible pour ouvrir un fichier pdf", Toast.LENGTH_LONG);
    toast.show();
}
else{
    startActivity(intent);
}
```

# Exécuter une action

- **ACTION_VIEW** : permet d'afficher des données à l'utilisateur afin qu'il puisse faire une sélection
- On a spécifié le type de données ciblées et le fichier à ouvrir avec **setDataAndType**
- Pour savoir si au moins une application pouvant répondre à l'intent est disponible sur l'appareil, on utilise **resolveActivity**

- **NB.- Cet exercice sera réalisé dans le cadre du prochain chapitre**

# Personnalisation de l’interface graphique

# Plan

- Material Design
  - Thèmes
  - États des composants
  - Dégradé
  - Élévation
  - Ripple Effect
- Polices
- Icônes
- Animations
  - Tween
  - Frame
  - Transitions
- Gestion des événements
- Gestion de la rotation

# Material Design

- Le Material Design est l'une des grandes nouveautés d'Android Lollipop et devient le thème par défaut pour Android et pour l'univers de Google.
- L'objectif de Google est de créer des lignes directrices (au niveau design) pour des applications quelle que soit la plateforme(smartphone, tablette, …)
- Ceci permet d'unifier l'expérience utilisateur sur toutes les plateformes utilisées
- Le Material Design se base sur un environnement 3D, chaque objet aura 3 coordonnées (x, y, z)
  - L'axe z représente l'élévation de l'objet dans son conteneur, qui, combinée avec les ombres, donnera une multitude d'effets possibles
- Pour plus de détails sur le Matérial Design, consultez le site :
  - http://www.google.com/design/spec/material-design/

# Thèmes

- La personnalisation d'une application passe par l'utilisation des thèmes
- Un thème (style) spécifie l'aspect visuel des différents éléments qui composent une application en définissant leurs propriétés visuelles (couleur, taille, espacement interne, taille de polices, etc.)
- Afin de rendre les différentes applications cohérentes entre elles et avec le thème général de l'OS, Android fournit trois thèmes, inspirés du Material Design qu'on peut utiliser ou surcharger dans une application :
  - Dark Material theme (thème sombre)
  - Light Material theme (thème clair)
  - Light Material theme with Dark ActionBar (thème clair avec une barre d'actions sombre)

# Utilisation de thèmes Android

- On peut utiliser l'un des thèmes Android directement dans une application.
- Pour cela, spécifier dans la balise application du manifeste le thème qu'on veut utiliser

```
<application
    ...
    android:theme ="@android:style/Theme.Material.Light.DarkActionBar"
>
```

# Exercice 6

- Écrire une application qui affiche bonjour, qui utilise les thèmes suivants (en 3 versions)
  - Dark Material theme (thème sombre)
  - Light Material theme (thème clair)
  - Light Material theme with Dark ActionBar (thème clair avec une barre d’actions sombre)

# Surcharger le thème Android

- On peut surcharger un thème Android en créant un fichier styles.xml dans le dossier values

```
<resources>
    <style name="AppTheme" parent="android:Theme.Material.Light.DarkActionBar">
        <!-ajouter des attributs pour personnaliser le thème -->
    </style>
</resources>
```

- Pour créer un thème sans en surcharger un autre, il faut enlever l'attribut parent. Ce qui est déconseillé, il vaut mieux surcharger un thème Android et le personnaliser.

# Gérer les anciennes version

- Les anciennes versions d’Android (inférieures à 5) ne disposent pas du thème Material Design. Pour gérer ces versions, il faut créer plusieurs dossiers values dans lesquels sont stockées les thèmes
- Le dossier values contiendra le thème compatible avec les anciennes versions et le dossier values-v21 contiendra le thème Material Design

# Exercice 7

- Écrire une application  Android qui surcharge le thème clair avec une barre d’action Sombre pour modifier la couleur de la barre d’action pour la mettre en rouge.
- À gérer les vieilles version d’Android dans votre application.

# État des composants

- Chaque composant d'interface d'Android (un bouton par exemple) peut posséder plusieurs états :
  - Normal : représente l'état général d'un composant avec aucune interaction utilisateur
  - Pressé : état représentant le composant cliqué
  - Sélectionné : Représente l'état du composant une fois sélectionné
  - Désactivé

# États des composants (implémentation)

- Pour implémenté les différents états d'un bouton(background), il faut créer un fichier XML dans lequel on spécifie le comportement d'un bouton dans ses différents états.
- La brique XML utilisé pour spécifier ce comportement est la balise selector.
- Créer un fichier XML dans le dossier drawable

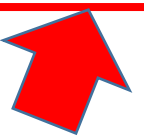
```
<xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:drawable="@color/green" android:state_pressed="true" />
    <item android:drawable="@color/white" />
</selector>
```

- Chaque item correspond à un état du bouton

# État des composants

- Il suffit ensuite de spécifier dans la déclaration de l'élément cible (dans un fichier layout) dans l'attribut background, le nom du fichier xml qui se trouve dans drawable sans l'extension.

```
<button
    android:layout_width ="match_parent"
    android:layout_height = "wrap_content"
    android:text = "@string/btn"
    android:background = "@drawable/btn_background_selector"
/>
```



**Nom du fichier xml de drawable**

**Les selections ne s'appliquent que sur des drawables (image ou couleur de fond d'un composant)**

# État des composants

- android:state_pressed : clic sur un élément
- android:state_focused : prise de focus d'un élément
- android:state_hovered : survol d'un élément
- android:state_selected : sélection d'un élément
- android:state_checkable : survient quand l'élément cible peut-être coché
- android:state_checked : survient quand l'élément cible est coché
- android:state_enabled : survient quand l'élément cible est utilisable (reçoit les interactions de l'utilisateur)
- android:state_activated : utiliser quand l'élément cible est activé

# Exercice 8

- Écrire une application Android qui comporte un bouton de couleur de fond blanche.
- Quand on sélectionne le bouton la couleur devient verte

# Dégradé

- On peut créer un dégradé avec android (comme couleur de fond par exemple) en utilisant les balises shape et gradiant
- La balise shape permet de créer des formes (rectangle, cercle, ligne, …)

```
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle|oval|line|ring">
```

# Dégradé

- La balise gradient permet de créer des dégradés suivant la forme définies par la balise shape (fichier à créer dans le drawable)

```
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle|oval|line|ring">
        <gradient
            android:angle = "integer"
            android:centerX = "integer"
            android:centerY = "integer"
            android:centerColor = "integer"
            android:endColor = "color"
            android:gradientRadius = "integer"
            android:startColor = "color"
            android:type = "linear|radial|sweep"
            android:useLevel = "true|false"
        />
</shape>
```

# Dégradé

- android:angle : représente l'angle du dégradé. Les valeurs doivent être un multiple de 45 (0 représente un gradient de gauche à droite et 90 de bas en haut)
- android:centerX et android:centerY : représente le centre du gradient (X/Y). Cette valeur est comprise entre 0 et 1.0
- android:centerColor : la couleur de centre du gradé
- android:startColor : La couleur de départ du dégradé
- android:endColor: La couleur de fin du dégradé
- android:gradiantRadius : Représente le rayon du dégradé
- android:type : représente le type du dégradé
- android:useLevel : vrai si ce gradiant est utilisé comme un LevelListDrawable, faux sinon (prise en compte d'un niveau dans le remplissage du drawable)
  - Voir le lien suivant pour un exemple :
  - http://developer.android.com/reference/android/graphics/drawable/LevelListDrawable.html

# Exercice 9

- Créer une dégradé en couleur de fond vert d’une activité (à reproduire le code des diapos suivantes)

# Solution (fichier drawable/gradient.xml)

- Clic droit sur le dossier drawable
- New /Drawable resource File
- File name : gradient
- Saisir le code de la diapo suivante

# Solution (fichier drawable/gradient.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android" android:shape="rectangle">
    <gradient
            android:angle = "90"
            android:endColor = "#59de8e"
            android:startColor = "#33f240"
        />
</shape>
```

# Solution : Fichier layout.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="16dp"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:paddingTop="16dp"
    android:background="@drawable/gradiant"
    tools:context="com.example.kfostine.exemplegradient.MainActivity"
    >

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!" />
</RelativeLayout>
```

# Élévation

- La notion d’élévation est introduite avec Android 5.0.
- Elle permet de modifier la position z d’un élément et de lui donner ainsi un effet d’ombre.
- L’attribut utilisée pour les éléments est :
  - android:elevation=« 3dp »

# Exercice 10

- Créer une application avec un bouton, ajouter un effet ombre de 2dp pour ce bouton.

# Animation

- Les animations représentent des effets qu'on peut appliquer à plusieurs éléments d'une interface
- Android possède deux types d'animation :
  - Tween animation : Qui affecte les propriétés des vues (taille, position, opacité)
  - Frame animation : Qui affecte plusieurs images dans une même vue

# Tween animation

- Cette animation permet d’effectuer des transitions (rotation, opacité, mouvement
- Elle est située dans un dossier XML se trouvant dans le dossier anim, qui est à créer dans le dossier values.
- Le fichier XML représentant ce type d’animation peut contenir les balises suivantes :
  - set
  - alpha
  - scale
  - translate
  - rotate

# Tween animation : Les balises du fichier XML

- set : Le fichier doit commencer avec cette balise
  - Chaque balise set représente un groupe d'animation
  - On peut imbriquer des balises set pour créer un groupe d'animation
- alpha : l'animation utilise la transparence des éléments
- scale : l'animation sert à redimensionner des éléments.
  - On peut spécifier le centre utilisé pour redimensionner l'élément
- Translate : Représente une translation verticale ou horizontale d'un élément
- Rotate : l'animation sert à effectuer une rotation

# Exercice 11

- Créer une application Android vec une vue représentant une image view (image du joueur Tony Kroos de Real Madrid)
- Créer une animation servant à faire une rotation de 360 degré sur l'image

# Exercice 11 – Solution – Fichier Layout

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="match_parent">
    <ImageView
        android:id="@+id/image"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:src="@drawable/toni_kroos"/>
</LinearLayout>
```

# Exercice 11 – Solution- fichier res/anim/my_animation.xml

- Créer le dossier anim dans res
- Créer un « android resource file » dans anim et le nommer my_animation.xml

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android"
    android:shareInterpolator="false">
    <rotate
        android:duration="2000"
        android:fromDegrees="0"
        android:pivotX="50%"
        android:pivotY="50%"
        android:repeatCount="3"
        android:toDegrees="360"
        />
</set>
```

# Exercice 11 – Solution : MainActivity

```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.layout_principal);
        ImageView image = (ImageView) findViewById(R.id.image);
        Animation myAnimation = AnimationUtils.loadAnimation(this, R.anim.my_animation);
        image.startAnimation(myAnimation);
    }
}
```

# Frame Animation – Exercice 12

- Écrire une animation Android permettant d'afficher une suite d'image dans un ordre prédéfini (Utiliser les images des joueurs de Real Madrid disponible dans Léa)

# Solution – Fichier layout_principal.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="match_parent">
    <ImageView
        android:id="@+id/image"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"/>

</LinearLayout>
```

# Solution : Fichier res/drawable/frame_animation.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<animation-list xmlns:android="http://schemas.android.com/apk/res/android"
    android:oneshot="false">
    <item
        android:drawable="@drawable/cristiano_ronaldo" android:duration="400" />
    <item
        android:drawable="@drawable/gareth_baie" android:duration="600" />
    <item
        android:drawable="@drawable/isco" android:duration="400" />
    <item
        android:drawable="@drawable/keylor_navas" android:duration="500" />
    <item
        android:drawable="@drawable/marcelo_vieira" android:duration="600" />
    <item
        android:drawable="@drawable/sergio_ramos" android:duration="400" />
    <item
        android:drawable="@drawable/toni_kroos" android:duration="500" />
</animation-list>
```

# Solution : Fichier MainActivity.java

```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.layout_principal);
        ImageView marketImage = (ImageView) findViewById(R.id.image);
        marketImage.setBackgroundResource(R.drawable.frame_animation);

        final AnimationDrawable marketAnimation = (AnimationDrawable) marketImage.getBackground();
        marketImage.post(new Runnable(){
            public void run(){
                if(marketAnimation != null){
                    marketAnimation.start();
                }
            }
        });
    }
}
```

# Gestion de la rotation

- On peut construire une application  avec une présentation différente entre le mode portrait et le mode paysage du téléphone. Pour cela, il faut utiliser :
  - Le dossier layout pour le mode portrait
  - Le dossier layout_land pour le mode paysage

# Exercice 13

- Écrire une application qui affiche en mode portrait l'écran 1 et en mode paysage l'écran 2

Fig 1.