Android 5 : Lollipop

Présentation générale

- Intervenant : Raphaël Mina – raphael@appsolute.fr
  - Ingénieur INSA Lyon 2011
  - Directeur technique et gérant de société
  - Développeur Android depuis 2010 (Android 1.6 “Donut”)
- Société : Appsolute SARL – www.appsolute.fr
  - Créée en 2012, implantée à Villeurbanne
  - 15 personnes à temps plein (dont un ancien IEM)
  - CA 2014 : 600 k€, dont environ 30% de conception/développement Android
  - Plus de 100 projets édités à date

Plan

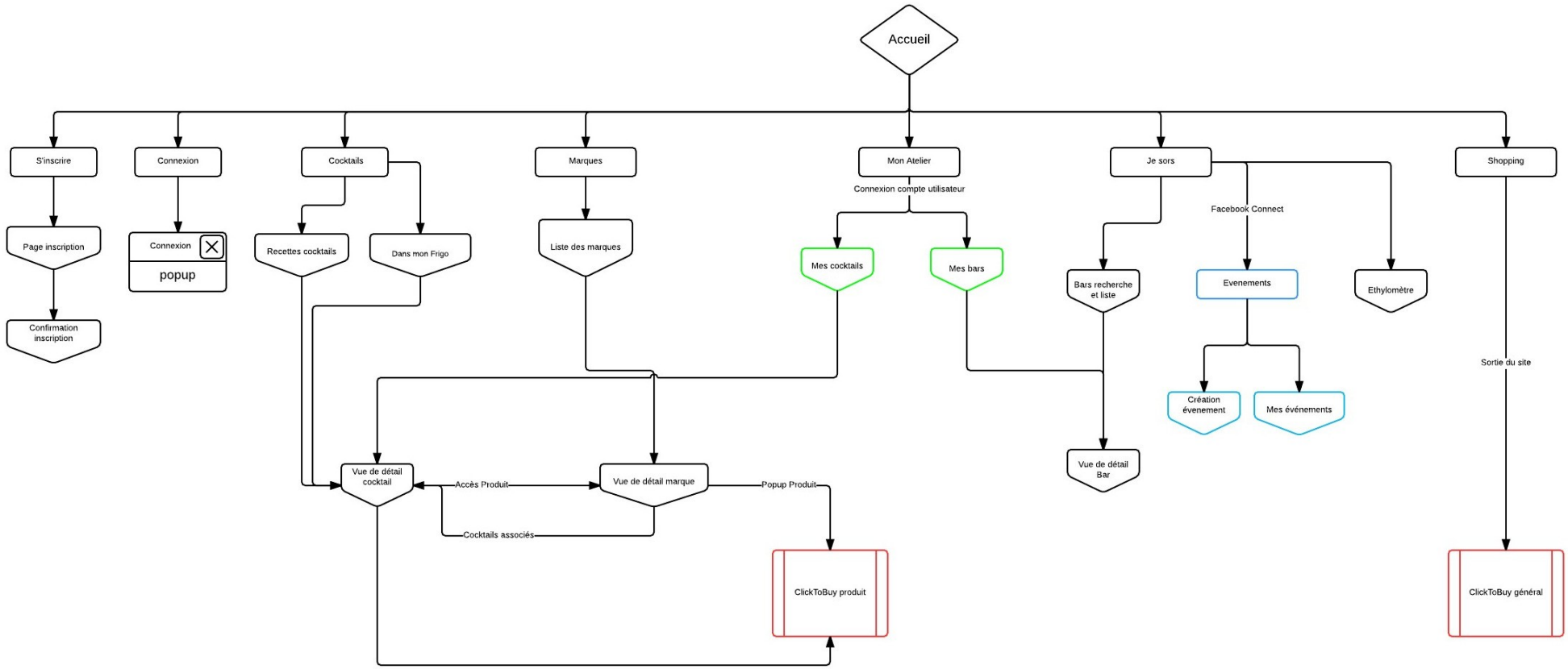
- Définition du design
- Le process du design dans une application mobile
- Les outils utilisée pour le design
- Rapide historique d'Android avant Lollipop
- La stratégie d'Android, et la fragmentation associée
- Le material design
- Les implémentations du material design sur Android
- Rappel de notions de bases sur l'intégration
- Implémentation du material design (c'est vous !)
- Les notifications et les changements sur Android 5
- Les autres nouveautés d'Android 5

Qu'est-ce que le design ?

- Design = conception
  - Bien au dela de la charte graphique (couleurs, typo...)
  - Définition des interactions utilisateurs
  - UI -> UX
- Compétences d'un designer mobile
  - Graphiste
  - Ergonome
  - Connaissance des patterns du système
  - **Intégrateur**
- Projet non désigné = 0% de chance de réussite
  - Insatisfaction du client
  - Problème de compréhension entre tous les acteurs (clients, designer, développeur)

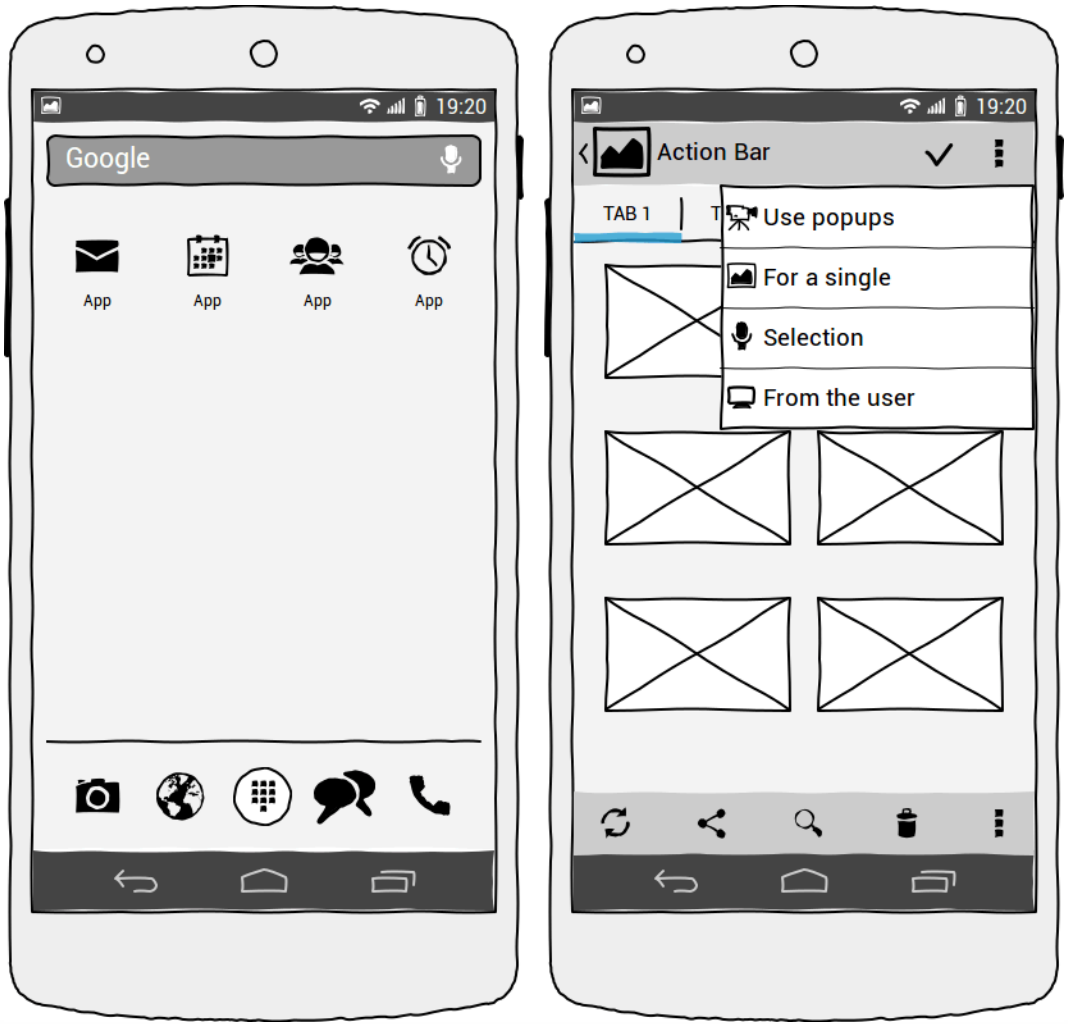
Design d'un projet mobile (1/5)

- Le workflow (schéma global)
  - Représentation schématique du process de navigation



Design d'un projet mobile (2/5)

- Les wireframes
  - Représentation schématique des vues de l'application

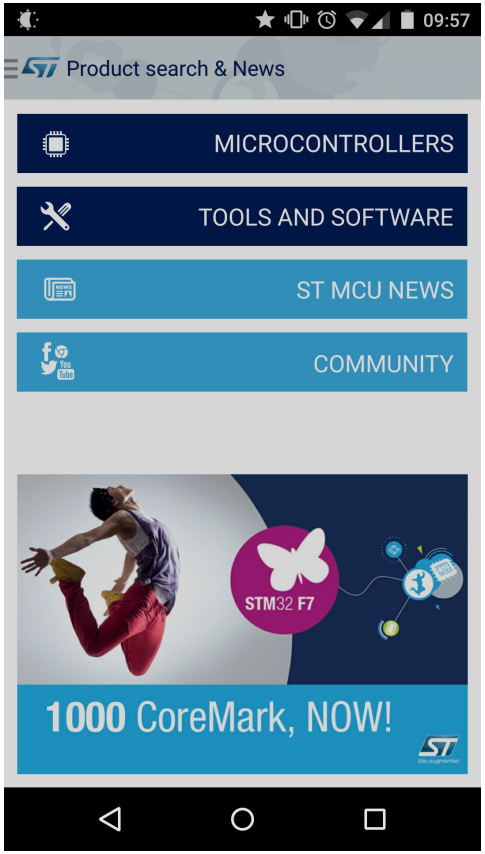


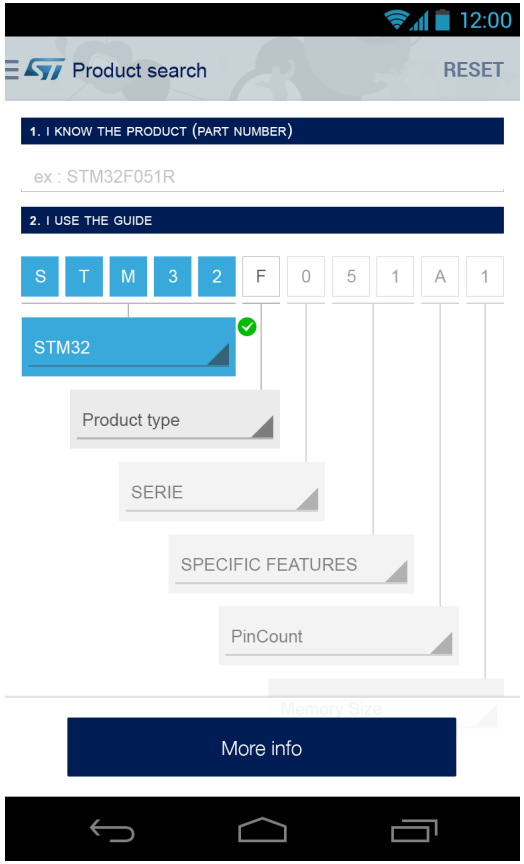
Design d'un projet mobile (3/5)

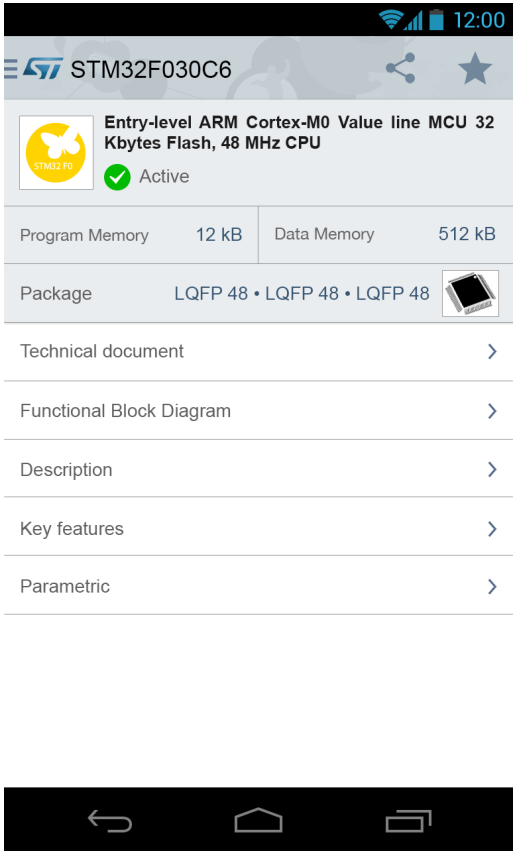
- La charte graphique
  - Reprendre l'existant ou créer de toute pièce
  - Exemple : charte graphique STMicroelectronics

Design d'un projet mobile (4/5)

- La création des vues
  - La partie la plus longue du travail du designer
  - Conception de toutes les vues de l'application





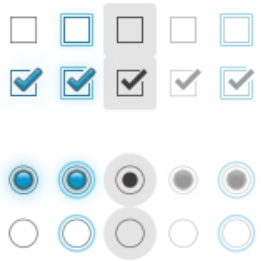


Design d'un projet mobile (5/5)

- L'arborescence globale et le storyboard fonctionnel
  - Arbre fonctionnel avec les vues
  - Spécifications fonctionnelles

Quels outils ?

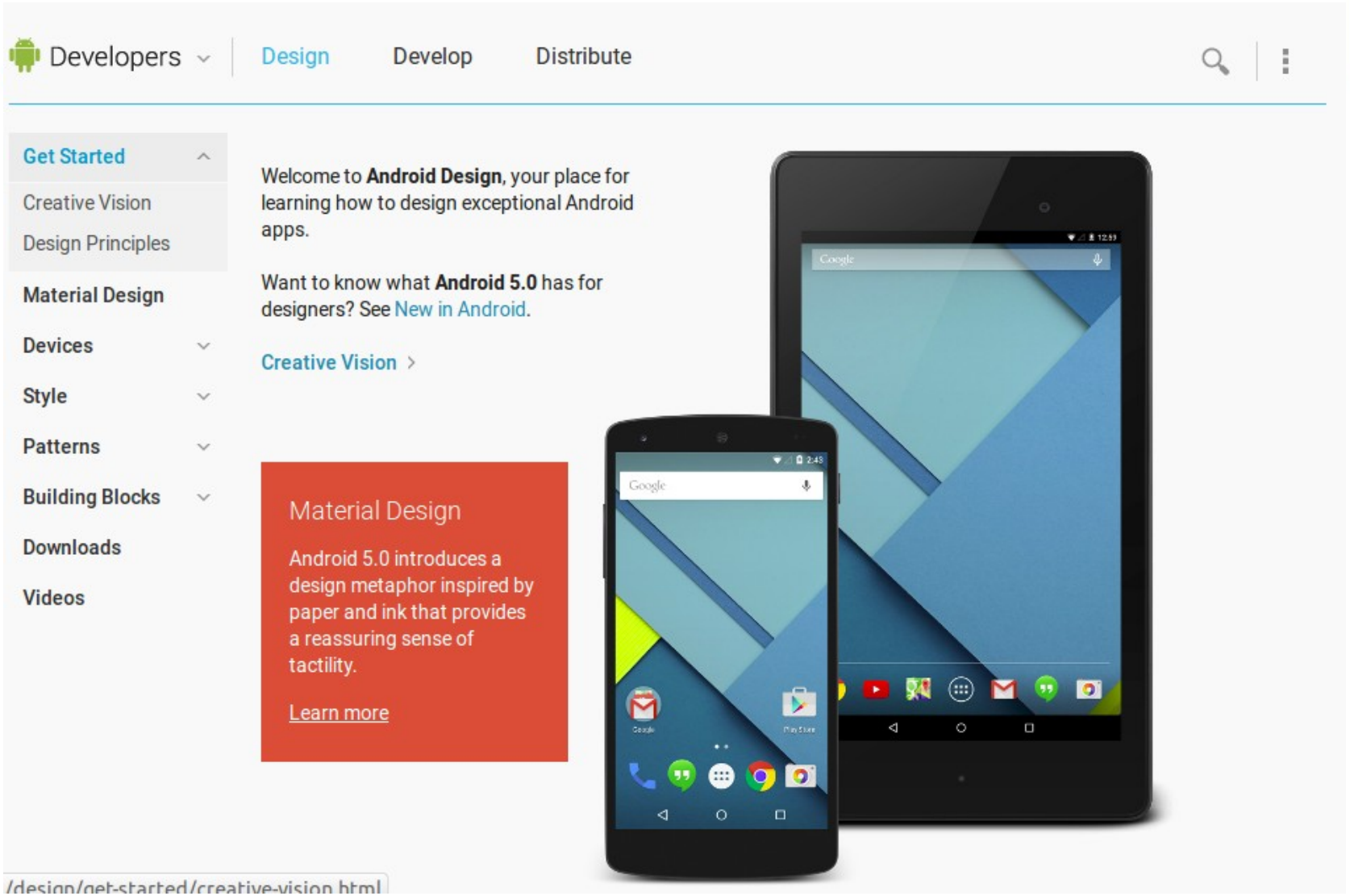
- Lucidcharts.com
  - logiciel web de conception de diagramme, incluant les mock ups Android
- Photoshop/GIMP
  - A partir des éléments téléchargeable fournis par Google sur http://developer.android.com/design/downloads/index.html



- Sketch (Mac only)
  - Kit complet avec outil d'export pour multi-résolution

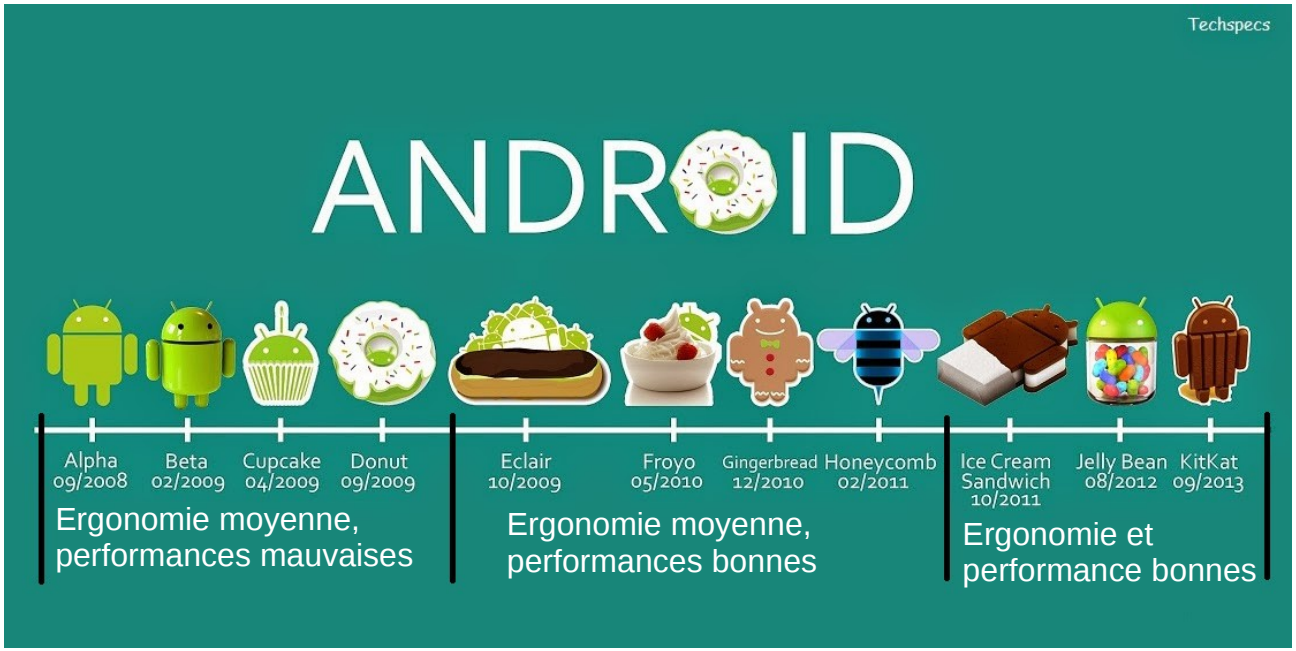
La Bible du design Android

- http://developer.android.com/design/index.html



Android : avant Lollipop

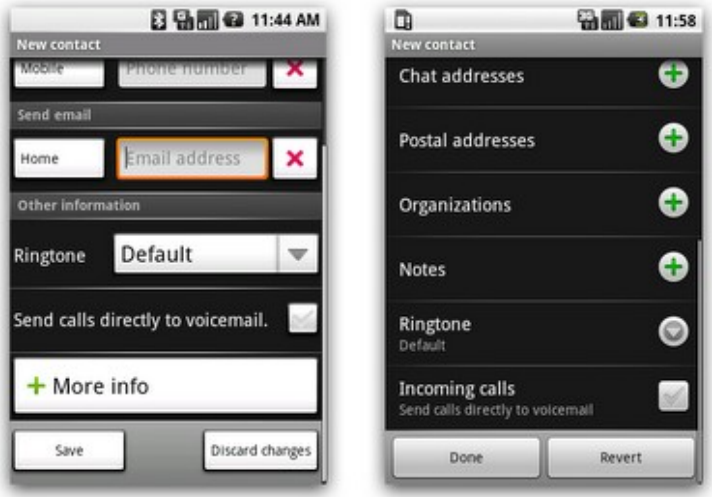
- Android ne s'est pas fait en un jour



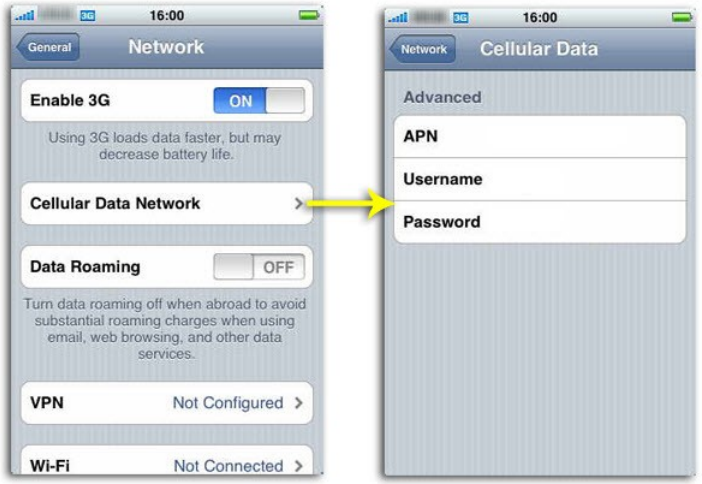
Pour une évolution complète des fonctionnalités d'Android, de 2008 à aujourd'hui :
http://www.theverge.com/2011/12/7/2585779/android-history

Une stratégie à l'opposée de celle d'iOS (Apple)

Exemple en 2009



Android est moche...
mais multitache



iOS (3.2) est beau...
mais monotache

Android a supporté très tôt des fonctionnalités comme le multitasking, le NFC, l'USB, le disque externe, initialement au détriment du design.
A l'inverse, le design est capital pour Apple et iOS.

Des périphériques toujours plus nombreux...

En plus des smartphones et tablettes et marque multiples, Android TV, Android Auto, Android wear, Chromecast...

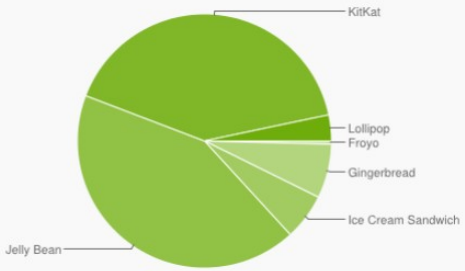
... et de fait, une fragmentation importante

Platform Versions

This section provides data about the relative number of devices running a given version of the Android platform.

For information about how to target your application to devices based on platform version, read Supporting Different Platform Versions.

Version	Codename	API	Distribution
2.2	Froyo	8	0.4%
2.3.3 - 2.3.7	Gingerbread	10	6.9%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	5.9%
4.1.x	Jelly Bean	16	17.3%
4.2.x		17	19.4%
4.3		18	5.9%
4.4	KitKat	19	40.9%
5.0	Lollipop	21	3.3%

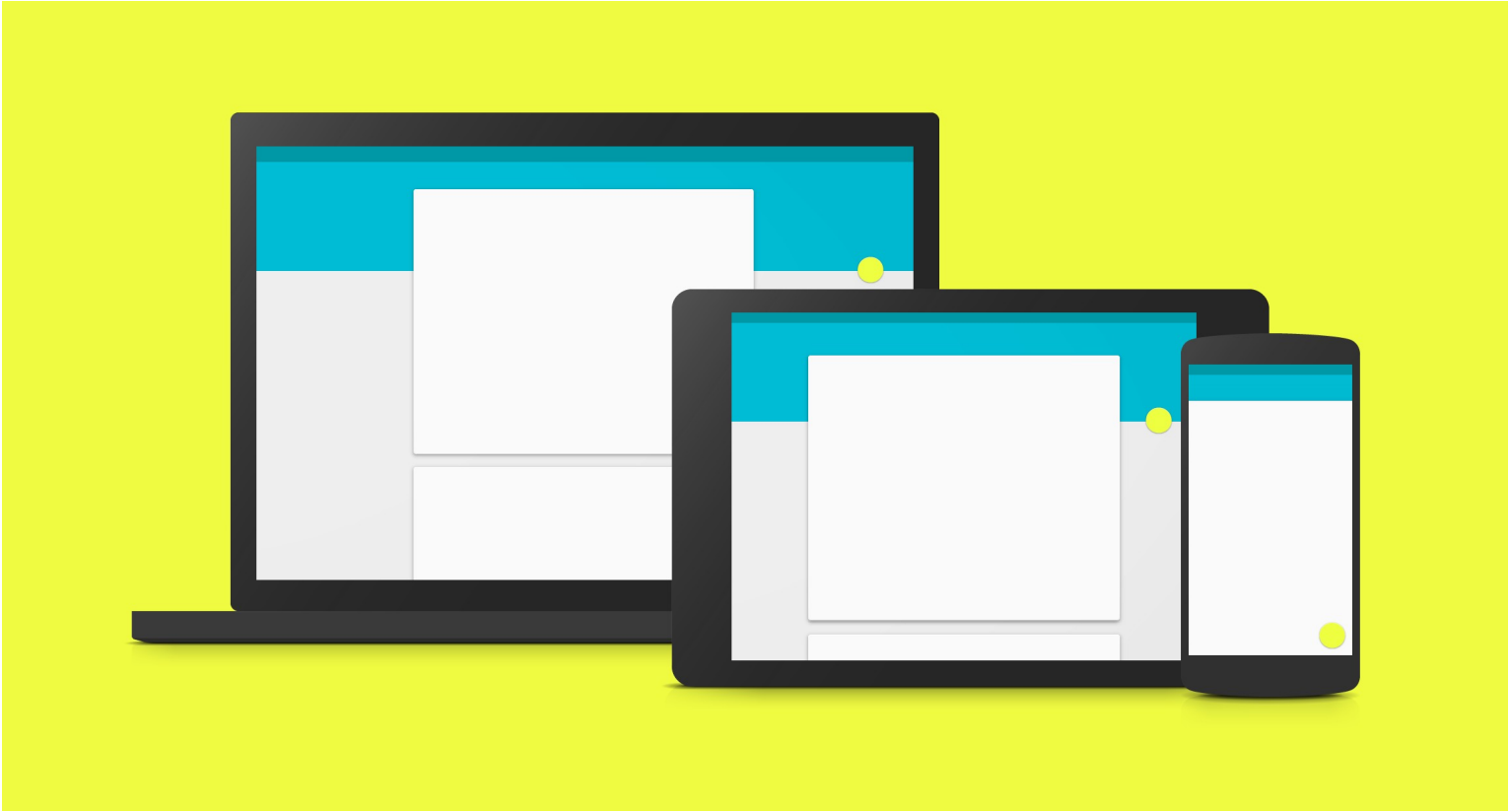


A titre de comparaison, iOS 8 est présent sur 78% des périphériques iOS et iOS 7 20%
Chiffres en live sur :
http://developer.android.com/about/dashboards/index.html

Un vrai casse-tête pour les développeurs !

Une réponse partielle à la fragmentation

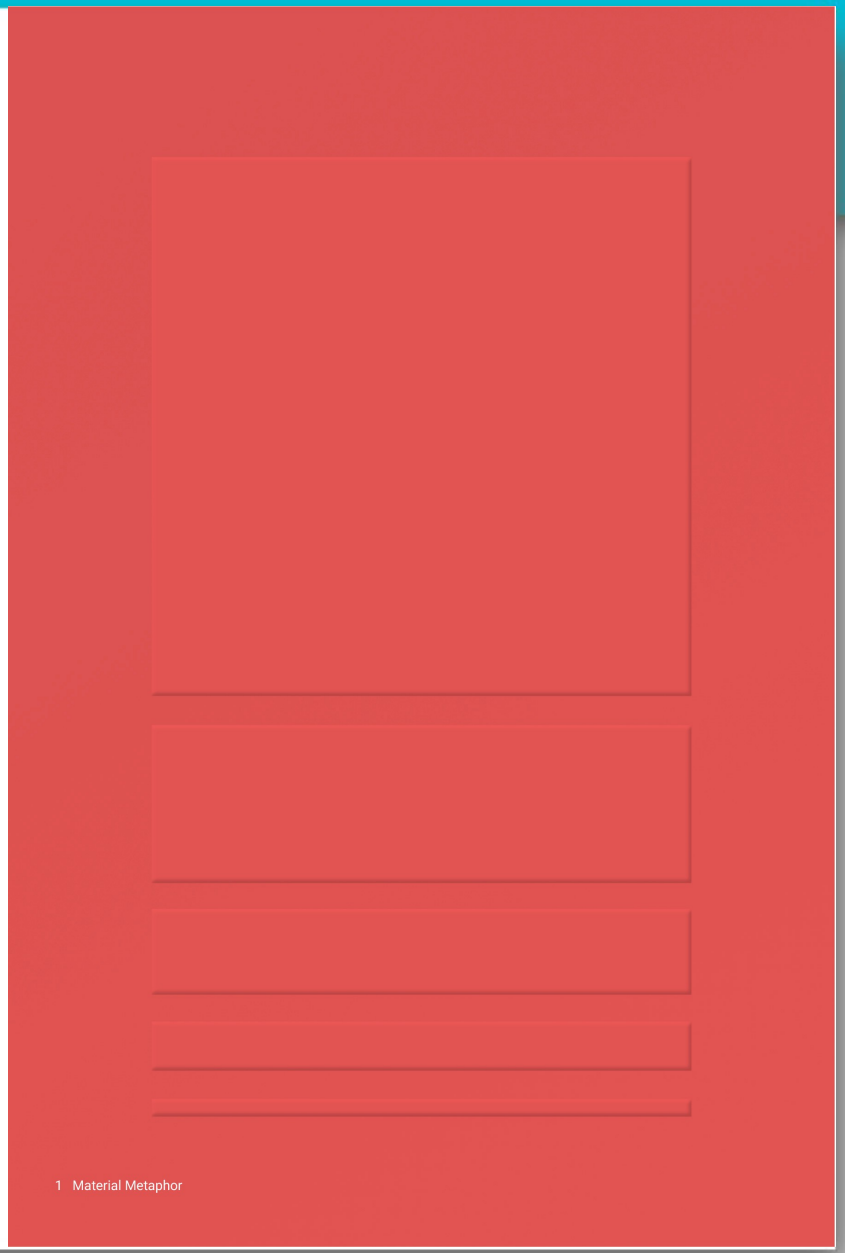
- L'introduction dans Android 5 Lollipop d'une nouvelle norme de design pour tous les périphériques : le material design



Material design : la métaphore matérielle

- La métaphore matérielle

La base du material design est l'analogie avec la matière, que l'on déplace, assemble et sépare, superpose...
Le but est de reproduire le côté tactile du réel tout en profitant du contournement des règles physiques permises par le virtuel.

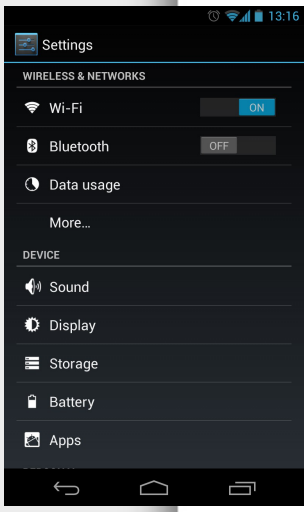


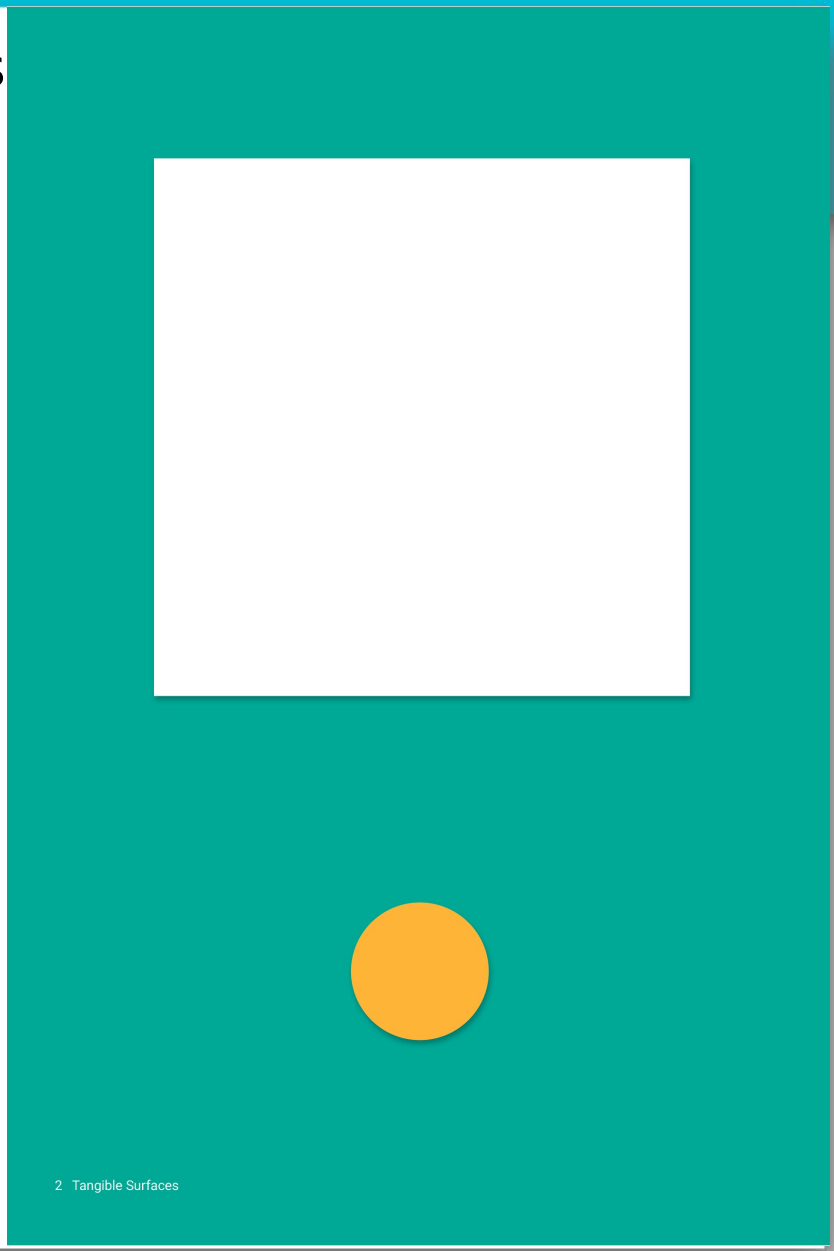
Material design : la métaphore matérielle

- Les surfaces et les bordures

L'utilisateur intéragit avec des surfaces très contrastées, dont les bordures sont clairement visibles. L'identification des différentes composante d'une interface se fait immédiatement, l'interface se “lit” en un clin d'oeil.

La différence est fondamentale par rapport à Holo Dark / Holo Light, beaucoup moins constrastée et moins cloisonnée.

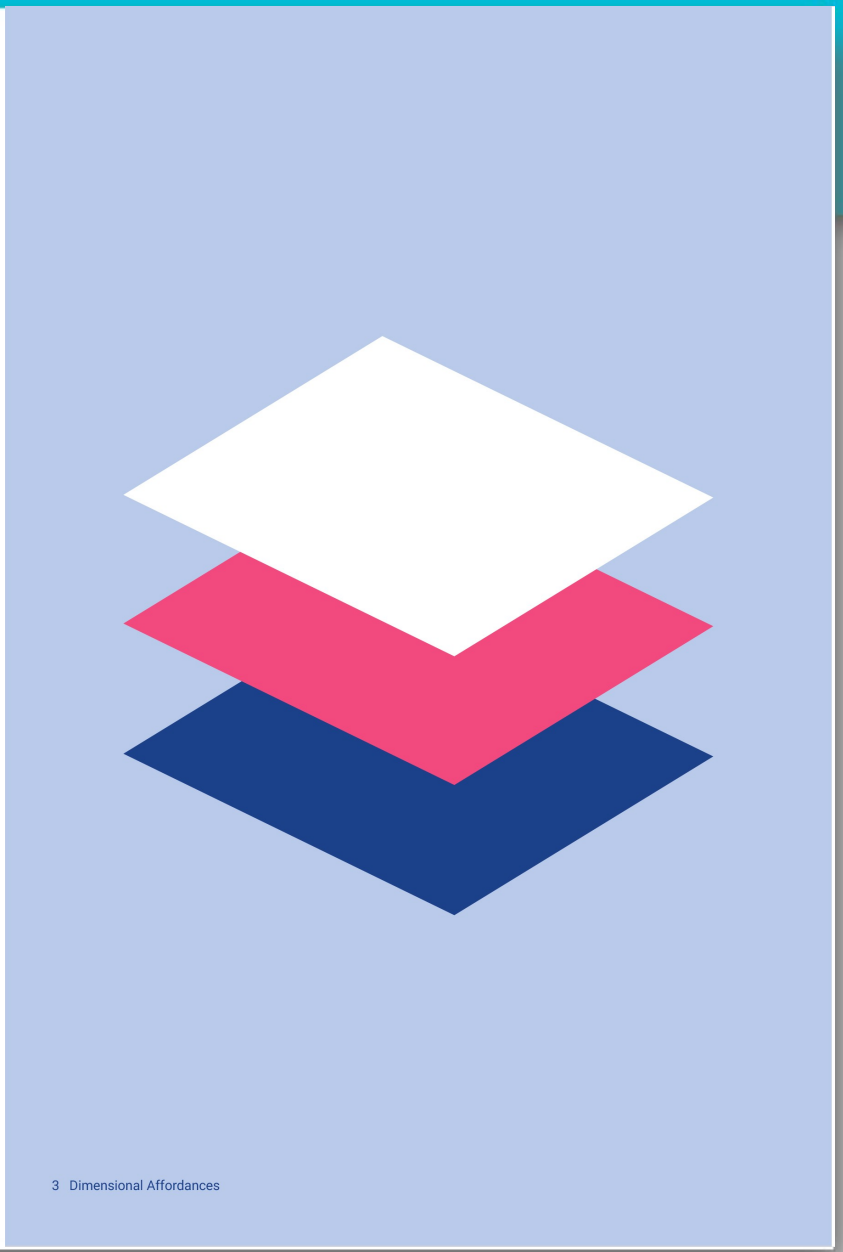




Material design : la métaphore matérielle

- L'élévation

La lumière, les surfaces et le mouvements définissent comment les objets intéragissent. L'utilisation intensive de **l'ombre portée** permet de hiérarchiser clairement les différents composants..
Cela se traduit dans Android par la propriété **elevation** d'une vue.



Material design : la métaphore matérielle

- Un design adaptatif

Quelque soit le support sur lequel le contenu est projeté, les principes du material design sont les mêmes. Les éléments sont réarrangés en fonction de l'espace disponible et de la hiérarchie du contenu.

4 One Design

Material design : la métaphore matérielle

- La graisse et l'espace

  La graisse (typographique) et la taille du texte, le contraste important entre les couleurs souvent vives, et l'espace blanc intentionnel permettent de mettre en évidence le contenu.

  Le principe de base est le même que “style suisse”.

Material design : la métaphore matérielle

- Couleur, surface, iconograp

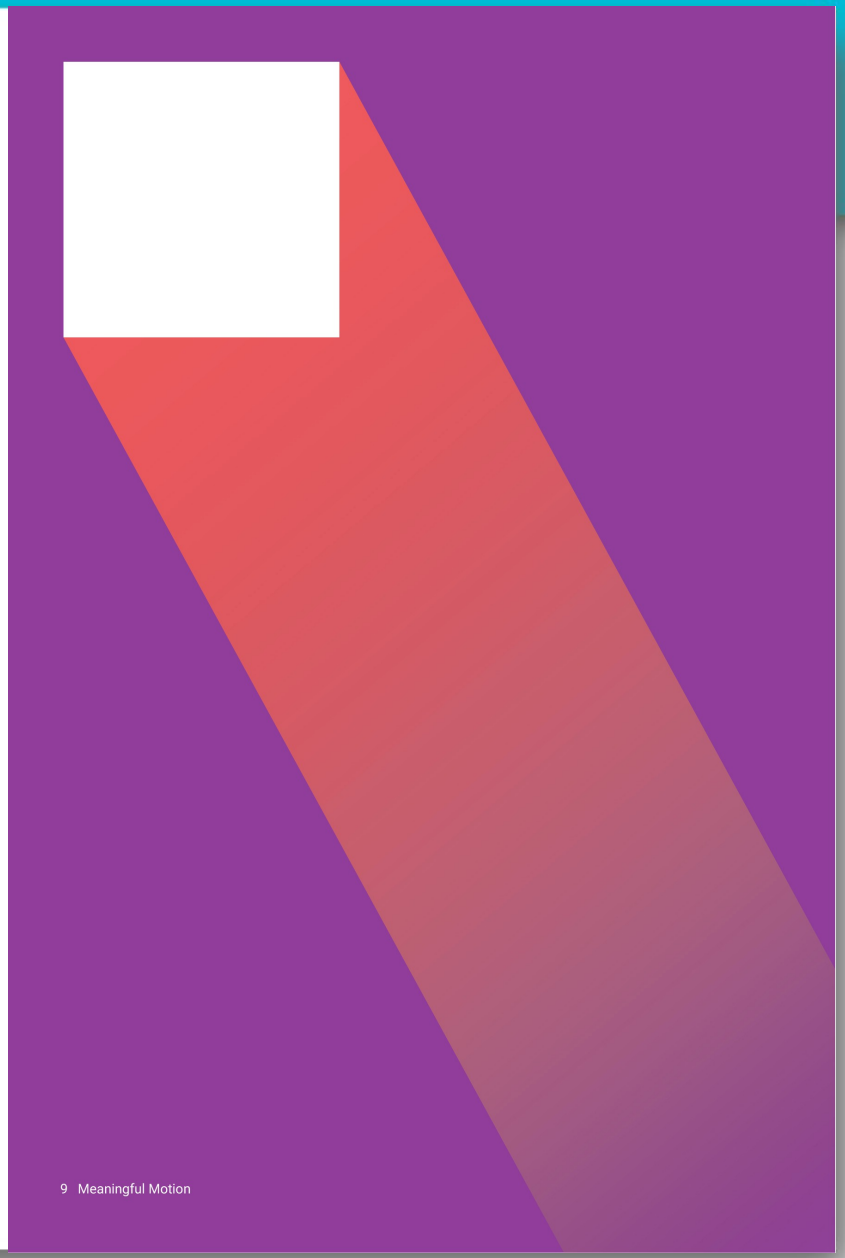
L'action de l'utilisateur est au coeur du material design, contrairement aux interfaces statiques. Les actions les plus importantes transforme complètement les vues.

L'animation est présente même lorsque l'action ne débouche sur aucun changement (exemple sur le circular reveal).

Material design : la métaphore matérielle

- L'importance du mouvement

  Le mouvement est au coeur du material design. Les éléments communs entre deux vues doivent avoir un mouvement qui apporte du sens (voir les shared elements).



Résumé de la métaphore matérielle

- La métaphore matérielle
  https://www.youtube.com/watch?v=Q8TXgCzxEnw

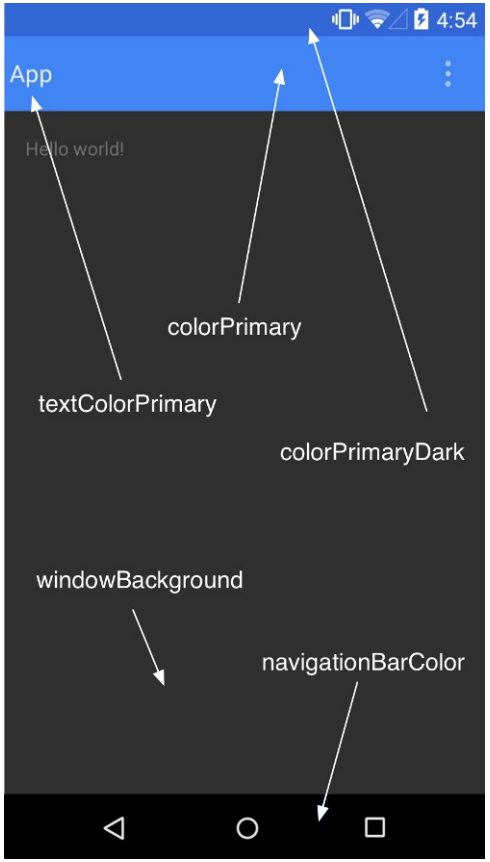
Les implémentations dans Android

- Déclarer le thème générale de l'application en matérial

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <!-- Fichiers styles.xml -->
    <style name="AppTheme" parent="android:Theme.Material.Light">

        <!-- Personnalisation des autres éléments -->

    </style>
</resources>
```

Les implémentations dans Android

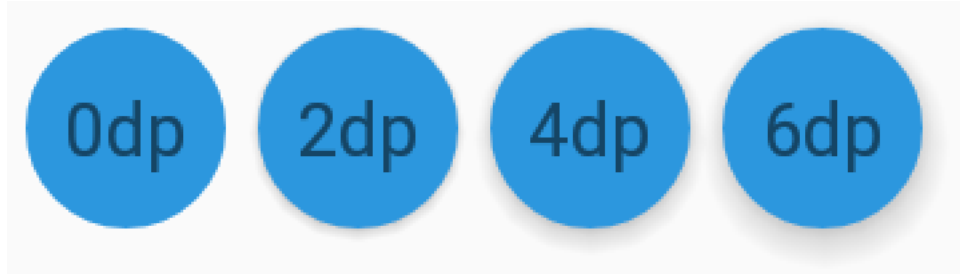
- Personnaliser les couleurs du thèmes
  - Changer les valeurs de `android:colorPrimary`, `android:textColorPrimary` etc. dans le fichier styles.xml



Les implémentations dans Android

- Définir l'”elevation” d'une vue

```
<TextView
    android:id="@+id/my_textview"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/next"
    android:background="@color/white"
    android:elevation="5dp" />
```



Les implémentations dans Android

- Définir la translation
  - Valeur définie sur trois axes : X, Y, et Z. On l'utilise souvent pour augmenter la hauteur en Z d'un élément au clic, mais possible avec n'importe quelle propriété
  - Implémentation simple grâce à un object `objectAnimator` dans un `selector`

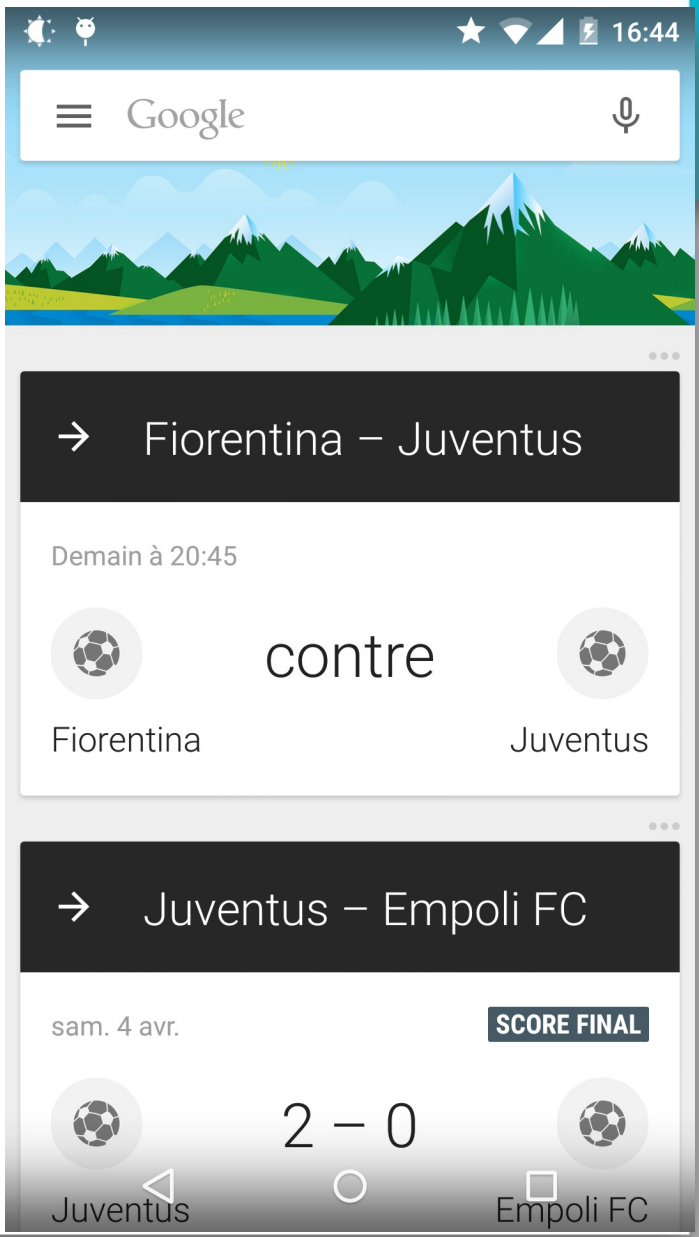
```
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <!-- Fichier transition_animator.xml -->
    <item android:state_pressed="true">
        <set>
            <objectAnimator android:propertyName="translationZ"
                android:duration="@android:integer/config_shortAnimTime"
                android:valueTo="20dp"
                android:valueType="floatType"/>

            <!-- Définition possible d'autres éléments à alimenter -->
        </set>
    </item>
```

Les implémentations dans Android

- Mettre en place des cardviews
  - La cardview est un élément de design prêt à l'emploi, qui permet de mettre en place des coins arrondis et une ombre.

```
<android.support.v7.widget.CardView
    android:id="@+id/card_view"
    android:layout_width="200dp"
    android:layout_height="200dp"
    card_view:cardCornerRadius="3dp">
    ...
</android.support.v7.widget.CardView>
```

  - Utilisez les `recyclerview` pour optimiser les performances de liste



Les implémentations dans Android

- Personnaliser les transitions
  - Plusieurs animations de base sont présentes comme `explode`, `slide` et `fade`
  - Définition au niveau global

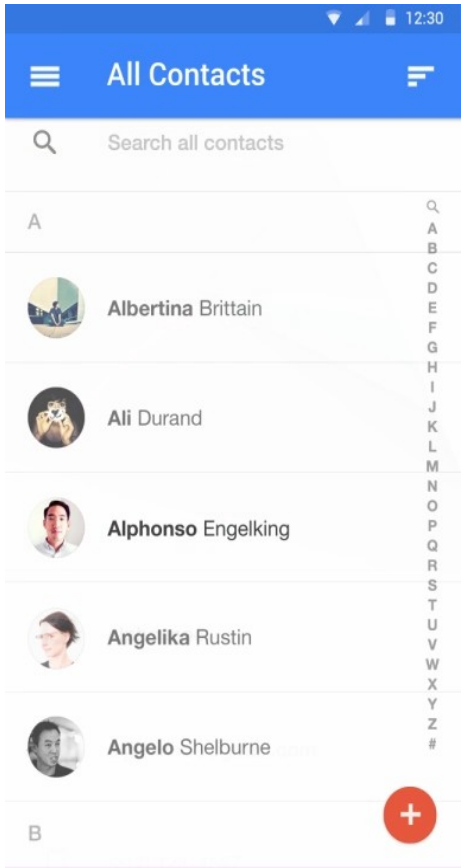
```
<!-- definition des transitions entrée et sortie -->
<item name="android:windowEnterTransition">@android:transition/explode</item>
<item name="android:windowExitTransition">@android:transition/explode</item>
```

  - Définition dans l'activité

```
getWindow().requestFeature(Window.FEATURE_CONTENT_TRANSITIONS);
getWindow().setExitTransition(new Explode());
```

Les implémentations dans Android

- Définir des shared elements
  - Eléments communs entre deux vues, possibilité de définir des animations entre les vues



- Définir un nom commun dans chacun des layouts avec la propriété `android:transitionName`

```
//Récupérer la vue à passer
profilImage=  findViewById(R.id.image_profil);

//Créer l'intent pour lancer la nouvelle activité
Intent intent = new Intent(this,ChangePictureActivity.class);

// Définir l'objet commun en reprenant le transitionName du XML
ActivityOptions options = ActivityOptions
        .makeSceneTransitionAnimation(this, profilImage, "imagetopass");

// Lancer la nouvelle activité
startActivity(intent, options.toBundle());
```

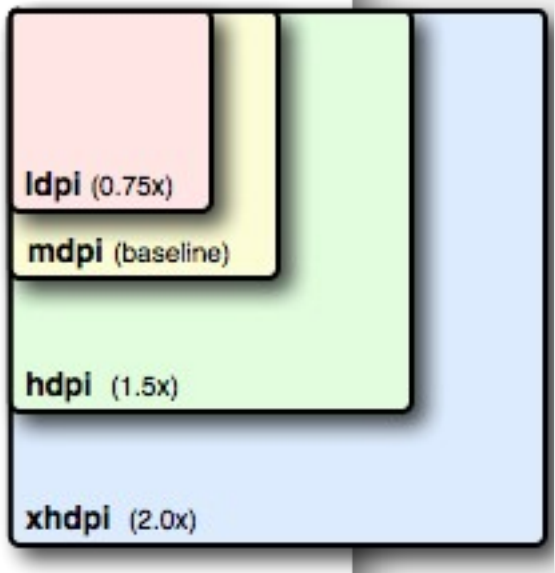
Notes complémentaires

- Le clipping
  - Donne la possibilité de changer facilement la forme d'une vue
  - Utile pour faire des animations

```
private class CircleOutlineProvider extends ViewOutlineProvider {
    @Override
    public void getOutline(View view, Outline outline) {
        outline.setOval(0, 0, view.getWidth(), view.getHeight());
    }
}


    mOutlineProviderCircle = new CircleOutlineProvider();

    View buttonImageChange=  findViewById(R.id.btn_image_change);
    buttonImageChange.setOutlineProvider(mOutlineProviderCircle);
    buttonImageChange.setClipToOutline(true);
```

Avant de coder : DIP = WTF ?

- Définition
  - DIP = density independant pixel
  - Densité = quantité de pixel dans une surface donnée
  - Pixel “virtuel”, qui est l'unité à utiliser dans toutes les dimensions (sauf taille de texte)
  - La taille physique dépent du périphérique :
    - En 160 dpi (MDPI), 1 DIP = 1 Pixel
    - En 240 dpi (HDPI), 1 DIP = 1,5 Pixel
    - En 320 dpi (XHPI), 1 DIP = 2 Pixels
    - En 480 dpi (XXHDPI), 1 DIP = 3 Pixels



Let's code !

- Vérifier qu'Android Studio est correctement configuré
  - Lancer le SDK Manager et télécharger les éléments de l'API 22
- Créer un nouveau projet, appliquer le material design en suivant les consignes de notre designer !
  - Modifier la palette de couleur
  - Mettre en place le layout, en utilisant le clipping sur le bouton de changement d'image.
  - Mettre en place l'elevation sur le bouton de changement d'image
  - Mettre en place une transition au clic sur le bouton pour afficher l'image dans une nouvelle activité, centrée, sur fond noir
- Utiliser les animations et custom animation
- Utiliser les cardviews

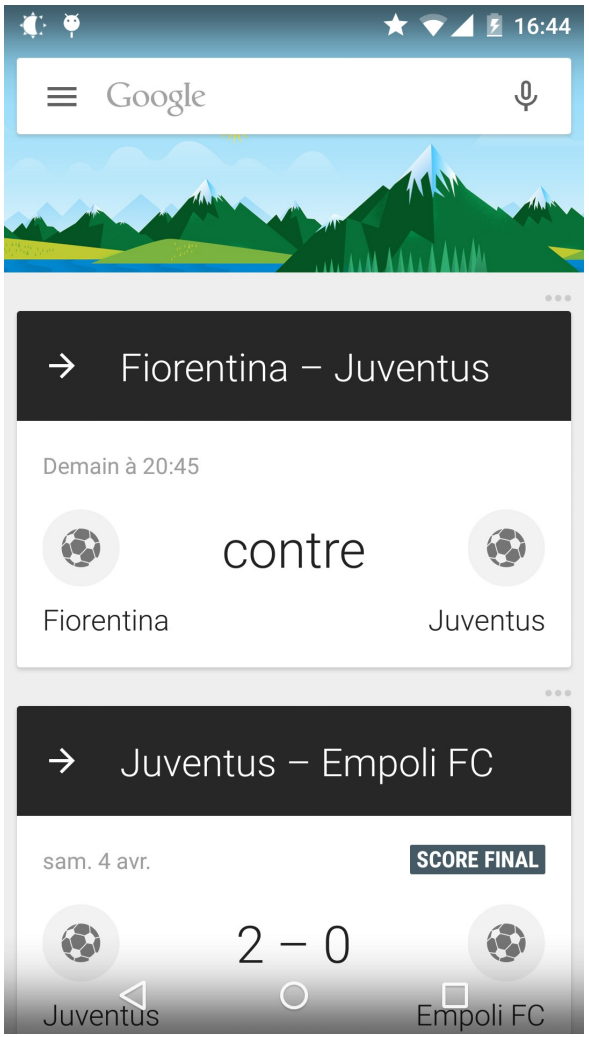
Partie 1/3

- Reproduire l'interface :
  - Résolution type : Nexus 5, XXHDPI
  - Utiliser les dimensions du fichier view_spec (attention, dimensions en pixels, certaines inutiles !)
  - Le bouton bleu doit être clippé, avec une valeur initiale d'élévation et une valeur de translationZ
  - **Vous passerez le test du pixel perfect !**
  - bit.ly/1c1bfmi

Partie 2/3

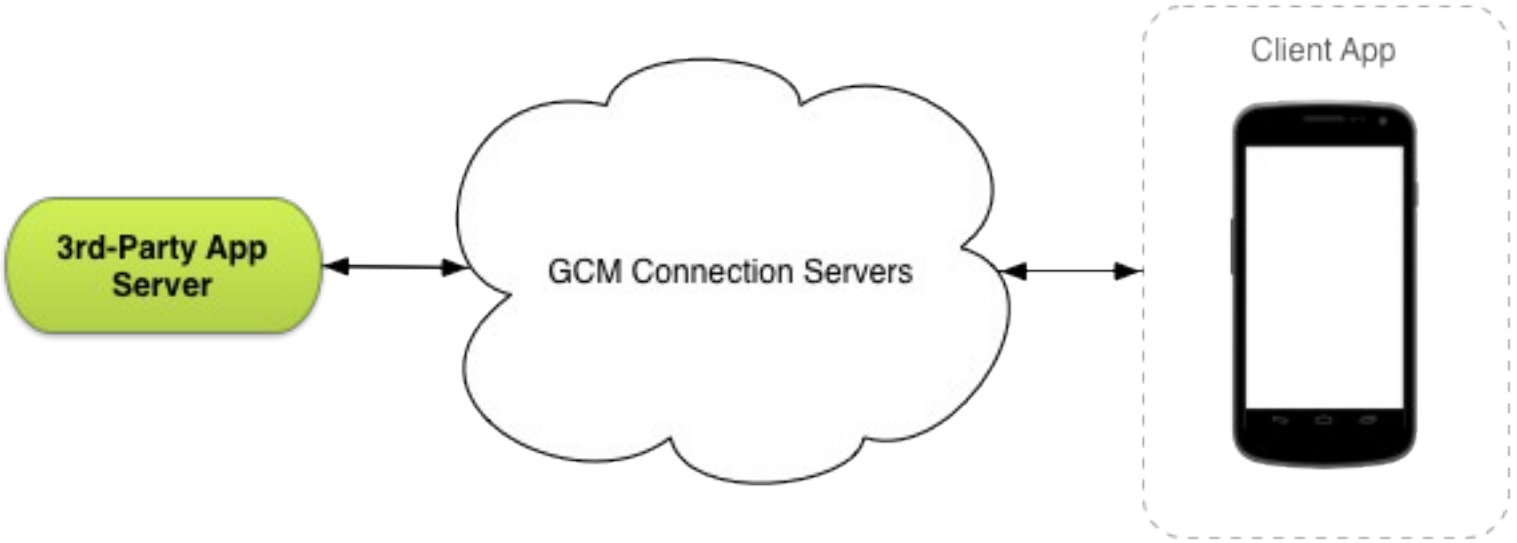
- Créer une seconde activité, qui se contente d'afficher au milieu de l'écran la même image, sans aucun élément, en la définissant comme un shared element.
- Utiliser une transition de votre choix

- Créer un autre projet, dans lequel vous allez reproduire les cardview disponibles ci-dessous. Utiliser un radius de 5dip et une elevation de 4dp.



Les notifications

- Notifications != push !



Les notifications

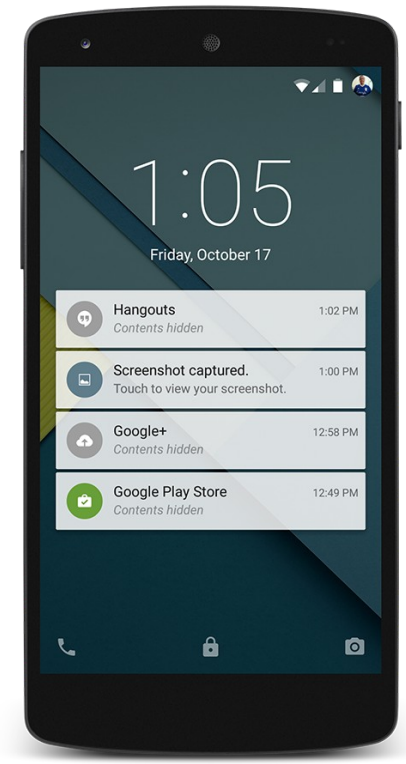
Android Lollipop introduit les “Heads-up notifications”

- Permet d'afficher une notification sans interrompre l'application en cours
- Peut avoir des boutons d'action
- Peut être personnalisé

Les notifications

Android Lollipop affiche également les notifications sur l'écran de verrouillage (locked-screen notifications)

- Le contenu des notifications est affichée ou non suivant la visibilité : privée, publique, ou secrète.



Les notifications

```
Notification notification = new Notification.Builder(context)
    // Show controls on lock screen even when user hides sensitive content.
    .setVisibility(Notification.VISIBILITY_PUBLIC)
    .setSmallIcon(R.drawable.ic_stat_player)
    // Add media control buttons that invoke intents in your media service
    .addAction(R.drawable.ic_prev, "Previous", prevPendingIntent) // #0
    .addAction(R.drawable.ic_pause, "Pause", pausePendingIntent)  // #1
    .addAction(R.drawable.ic_next, "Next", nextPendingIntent)     // #2
    // Apply the media style template
    .setStyle(new Notification.MediaStyle()
    .setShowActionsInCompactView(1 /* #1: pause button */)
    .setMediaSession(mMediaSession.getSessionToken())
    .setContentTitle("Wonderful music")
    .setContentText("My Awesome Band")
    .setLargeIcon(albumArtBitmap)
    .build();
```

L'ajout d'une vibration ou d'une sonnerie permet de rendre la notification “heads-up”, en utilisant par exemple `.setVibrate(100)`

Les autres nouveautés d'Android 5 Lollipop

- - changement dans les webview (WebGL notamment). La webview se met à jour via Google Play
- - 64 bit support (également dans le NDK)
- - Concurrent documents on Overview (chrome tab = un onglet)
- - Camera 2 API :
- - Open GL 2 (tesselation notamment)
- - Project Volta : optimisation de la batterie -> Lien vers Battery Historian. Job Scheduler