TP 2

Les interfaces graphiques & Layout

Pré requis:

Il est indispensable de bien connaître la notion de Layout.

Objectifs:

- Utiliser les Layout Relatives et lineaires
- Utiliser les ConstraintLayout et le Layout Editor

Néto graphie :

https://developer.android.com/studio/write/layout-editor
https://developer.android.com/training/constraint-layout/
https://developer.android.com/topic/performance/vitals/render

Contenu du TP:

1. **Présentation du LayoutEditor**

   Une partie théorique présentant le ConstraintLayout et le LayoutEditor à travers des vidéos.

2. **CodeLabs & ConstraintLayout**

   Une première application à réaliser en suivant un CodeLabs d'Android.

3. **Construction du BugDroid d'Android**

   Une deuxième application à réaliser pour construire le BugDroid d'Android en utilisant : LinearLayout, RelativeLayout et ConstraintLayout.

4. **Optimisation du Rendu des interfaces utilisateurs**

   Les problèmes que peut causer la mauvaise conception des interfaces graphiques et comment on peut détecter ça en utilisant le mode « Debug GPU overdraw ».

Présentation du LayoutEditor 1

I.1. C'est quoi le ConstraintLayout ?

Le ConstraintLayout vous permet de créer des mises en page vastes et complexes avec une hiérarchie de vues à plat (pas de groupes de vues imbriquées).
Il est similaire au RelativeLayout dans la mesure où les vues sont agencées en fonction des relations entre les différentes vues, mais il est plus souple et plus facile à utiliser avec le LayoutEditor d'Android Studio ; il suffit d'un simple Drag & Drop pour tout construire.
Le ConstraintLayout est integré à partir de la version 3.0 d'Android Studio.
Pour savoir de plus sur le ConstraintLayout soit le lien suivant :: https://developer.android.com/training/constraint-layout/

I.2. Positionnement des vues

Pour définir la position d'une vue dans un ConstraintLayout, vous devez ajouter au moins une contrainte horizontale et une contrainte verticale mais parfois plus.
Chaque contrainte représente une connexion avec une autre vue ou le parent.

Lorsque vous déposez une vue dans le LayoutEditor, elle reste là où vous la mettez, même si elle ne présente aucune contrainte. Mais, lors de l'exécution elle sera à la position [0,0] (le coin supérieur gauche).

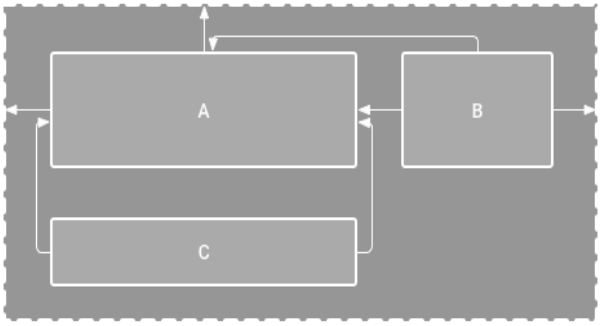


Figure 1: Vue C sans contrainte verticale

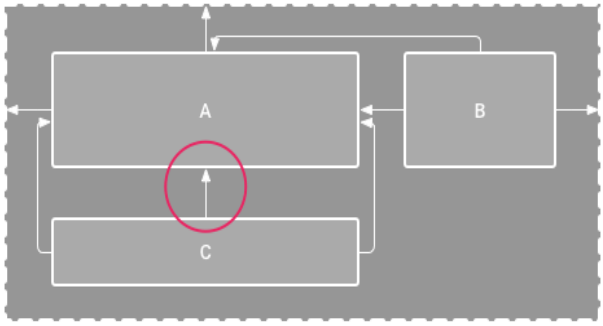


Figure 2: Vue C sous la vue A

Pour plus d'informations voir les vidéos fournies avec le cours :

- 1-Building interfaces with ConstraintLayout in Android Studio.mp4
- 2-The left side of a view is constrained to the left side of the parent.mp4
- 3-Adding a constraint that opposes an existing one.mp4
- 4-Adjusting the constraint bias.mp4

CodeLabs Google & 2

En plus du site officiel d'Android : https://developer.android.com/

Figure 3: Developer.android

Google offre à ses utilisateurs une autre expérience de codage pratique guidée, basée sur un tutoriel à travers des Codelabs : https://codelabs.developers.google.com/

Figure 4: https://codelabs.developers

La plupart des codelabs vous guideront tout au long du processus de création d'une petite application ou de l'ajout d'une nouvelle fonctionnalité à une application existante.

Dans le codelabs suivant, vous apprendrez à utiliser l'éditeur de mise en page Android Studio avec ConstraintLayout :

https://codelabs.developers.google.com/codelabs/constraint-layout/

⇨ Vous êtes demandés de réaliser ce codelab.

Construction du BugDroid 3

Le but de cet Exercice est de construire le BugDroid D'Android.

Données fournies	Résultat final

Body.png

Hand.png

Head.png

Leg.png

1) Créez un nouveau projet à nommer BugDroid
2) Ajoutez les images Leg, Head, Body et Hand à votre projet (contenu du dossier Assests).
3) Construisez cette image en utilisant seulement les LinearLayout.
4) Ajoutez à votre projet un nouveau Layout et construisez cette image en utilisant seulement les RelativeLayout
5) Ajoutez à votre projet un nouveau Layout et construisez cette image en utilisant seulement les ConstraintLayout.

Optimisation du rendu des UI 4

VI.1. Problèmes liés aux interfaces graphiques

VI.1.1. Slow Rendering

Le rendu lent (slow rendering) est le problème de performance le plus courant.
Pour que l'interaction d'un utilisateur avec votre application soit fluide, votre application doit restituer les images en moins de 16 ms pour atteindre 60 images par seconde.
Si votre application souffre d'un rendu lent, le système est obligé de sauter des images et l'utilisateur percevra un bégaiement dans votre application. Nous appelons ça « Jank ».

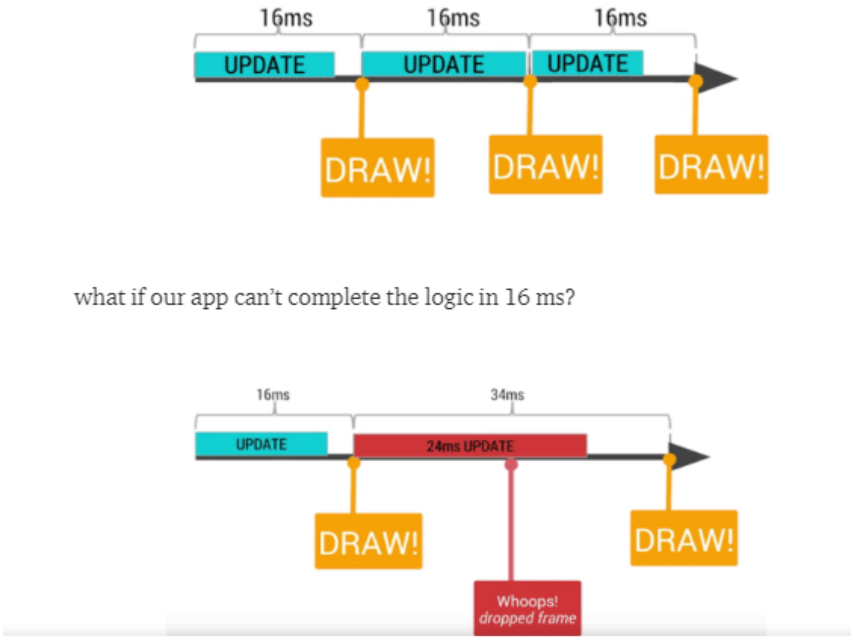


Figure 5: Slow Rendering

⇨ Notre application doit faire toute la logique pour mettre à jour l'écran en 16 ms.

VI.1.2. Overdraw

« Overdraw » signifie combien de fois un pixel sur l'écran a été redessiné dans une seule image.
Il se produit chaque fois que l'application demande au système de dessiner quelque chose par-dessus quelque chose d'autre.

GPU overdraw tool

Android offre un outil qui peint l'écran en différentes couleurs pour indiquer où se produit un overdraw, et combien.

Ces couleurs représentent combien de fois le pixel a été peint sur l'écran.

Comment l'activer?

Aller aux paramètres de votre téléphone/émulateur -> Options du développeur -> sélectionnez "Debug GPU overdraw".

Figure 6: Overdraw colors

VI.2. Travail à faire

Activez le mode Debug GPU overdraw et re-testez l'application précédente dans le cas d'utilisation de LinearLayout et dans le cas d'utilisation de LinearLayout (Modifiez les couleurs de fond des différents LinearLayout pour voir la différence)

Exercices d'application 5

1) Exercice 1: Calculatrice simple

L'objectif de cet exercice est de créer une application Android qui permet d'effectuer des opérations de calcul sur 2 entiers (addition, soustraction, multiplication et division).

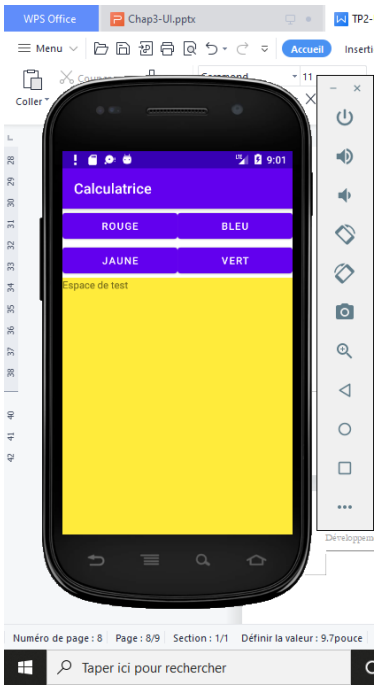
Pour se faire, suivre les étapes suivantes :

1) Implémentez l'interface suivante en utilisant les LinearLayout sachant que les id des différents champs sont les suivants :
   - *EditText(entier 1) => texte1*
   - *EditText(entier 2) => texte2*
   - *RadioButton (+) => r1*
   - *RadioButton (-) => r2*
   - *RadioButton (*) => r3*
   - *RadioButton (/) => r4*
   - *Button => btn*
   - *TextView(Résultat) => res*
2) Implémentez l'événement Onclick relatif au click sur le bouton « = » permettant de :

   - Vérifier que les 2 EditText ne sont pas vides => Utilisez la méthode **Boolean isEmpty()** .
   - Si l'un des EditText est vide affichez dans un Toast le message « Champs vide » si non le résultat calculé sera affiché au niveau du TextView Résultat.

2) Exercice 2: Les couleurs

1) Ajoutez à votre projet une nouvelle activité qu'on nommera «couleur»
2) Créez l'interface suivante:



Cette interface contient 4 Button et un TextView

Le clic sur chacun des boutons engendre le changement de la couleur de l'arriére plan du TextView comme le montre la figure prise lors du clic su le bouton jaune.

3) Implémentez les différents événements onclick.

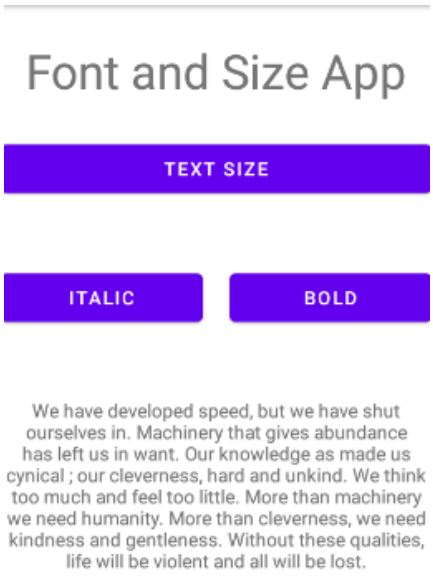
Utilisez les méthodes suivantes:

```
int c=getResources().getColor(R.color.nom_couleur);
nom_view.setBackgroundColor(c);
```

3) Exercice 3: Font

1) Ajoutez à votre projet une nouvelle activité qu'on nommera «fontstyle»

2) Créez l'interface principale suivante:



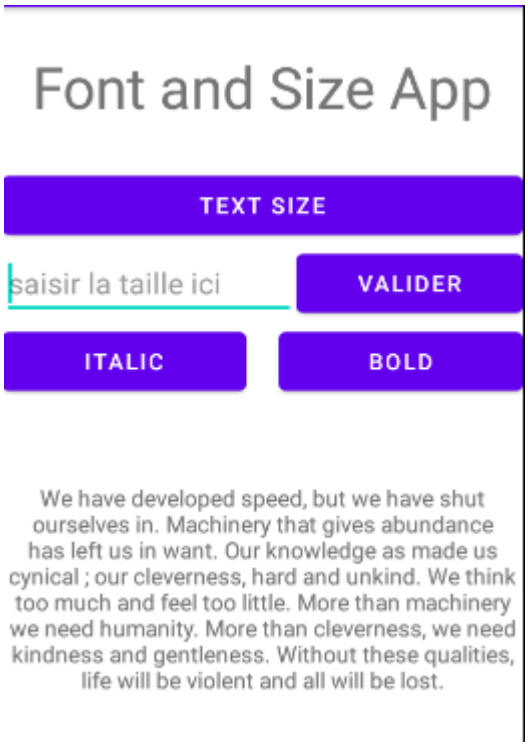
Cette interface contient:

- TextView: contenant la chaine «Font and size app»
- Un bouton «Text Size»
- Un bouton «Italic»
- Un bouton «Bold»
- Un TextView contenant la citation suivante:

```
«We have developed speed, but we have shut ourselves in.
Machinery that gives abundance has left us in want.
Our knowledge as made us cynical ; our cleverness, hard and unkind.
We think too much and feel too little. More than machinery we need humanity.
More than cleverness, we need kindness and gentleness.
Without these qualities, life will be violent and all will be lost.»
```

Cette citation est à déclarer au niveau du fichier strings.xml

3) Button «text size»:



Le clic sur ce bouton engendre l'affichage d'un LinearLayout contenant les éléments suivants:

- EditText
- Button «valider»

4) Button «valider»:

Le clic sur ce bouton permettra de changer la taille du texte contenu dans de la citation(la nouvelle taille est celle saisie au niveau de l'EditText)

Utilisez les méthodes suivantes

```
Nom_view.setTextSize(float taille);
Nom_view.setVisibility(View.VISIBLE);
```

5) Button Italic et Button Bold

Le clic sur ce bouton engendre la modification du style du texte de la citation.

Vous pouvez utiliser la méthode suivante:

```
Vien_name.setTypeface(Typeface.defaultFromStyle(Typeface.BOLD));
```