# Annexe 6 – Dessin personnalisé avec Android

Nous avons vu comment dessiner une interface-utilisateur à l’aide de certains widgets; néanmoins, on doit parfois dessiner des éléments qui ne sont pas des composantes pré-établies. On dit que c’est du dessin personnalisé ( custom drawing )

1 Démarrez un nouveau projet avec AndroidStudio.

2. Nous n’utiliserons pas le fichier de positionnement XML car on doit dessiner sans utiliser des composantes. On a cependant besoin d’une composante vierge sur lequel on pourra dessiner ( i.e une surface de dessin ).

On peut définir cette zone à l’aide d’une classe interne à notre Activité. Cette nouvelle classe devra être une sous-classe de la classe View ; on peut donc dire qu’on développe un nouveau type de widget !

ConstraintLayout par défaut

Instance d’une sous-classe de la classe View où on va « dessiner »

|  |  |
| --- | --- |
| Java pur | Java avec Android |
| On ferait une sous-classe de JPanel dans une classe interne | On fait une sous-classe de View dans une classe interne |

3. Tel que demandé par AndroidStudio, codez un constructeur pour la nouvelle classe interne SurfaceDessin similaire à ceci pour le moment :

**public** SurfaceDessin(Context context) {  
 **super**(context);

…. D’autres initialisations à venir…

}

4. On doit redéfinir la méthode onDraw de la classe View; c’est dans cette méthode qu’on va dessiner. **Pour le moment**, on va laisser la méthode vide.

5. Ce n’est pas suffisant de définir une classe interne à l’activité pour qu’elle soit ajoutée au ConstraintLayout, conteneur de notre activité…on doit  :

1. Crée un objet de notre SurfaceDessin

2. lui donner une taille et une position (LayoutParams)

3. ajouter la surface de dessin sur le ConstraintLayout (main)

Notes sur setLayoutParams :

Cliquez ici pour entrer du texte.

Densité : Cliquez ici pour entrer du texte.

Pour plus d’infos : <https://developer.android.com/training/multiscreen/screendensities>

Ajoutez-donc notre surface de dessin au ConstraintLayout parent. Donnez-lui une couleur de fond pour qu’on puisse bien l’identifier lorsqu’on roule l’interface.

6**. De retour dans le code .java de notre surface de dessin,** on doit définir les objets dont on aura besoin pour dessiner un cercle.

La marche à suivre est la suivante par rapport à ce qu’on faisait avec les JFrames :

|  |  |
| --- | --- |
| Java pur | Java avec Android |
| Crayon 🡪 objet Graphics | Type de crayon ( couleur, largeur, style ) 🡪 objet Paint qu’on doit créer |
| Dessiner dans la méthode paintComponent | Dessiner dans la méthode onDraw |
| Dessiner une forme 🡪 à l’aide du crayon ( objet Graphics ) passé en paramètre à la méthode paintComponent | Dessiner une forme à l’aide de l’objet Canvas passé en paramètre à la méthode onDraw |

Donc, déclarez un objet Paint ( package android.graphics ) et créez-le dans le constructeur. Il sert à savoir comment on va dessiner ( couleur, largeur du trait, etc )

Dans la méthode protected void onDraw ( Canvas canvas )et, à l’aide des méthodes de l’API, faites en sorte :

* Appliquez l’antialias
* Assignez la couleur verte
* Dessiner un cercle plein de rayon = 80 pixels

Pour plus d’infos : [https://developer.android.com/develop/ui/views/layout/custom-views/custom-drawing#draw](https://developer.android.com/develop/ui/views/layout/custom-views/custom-drawing#23draw)

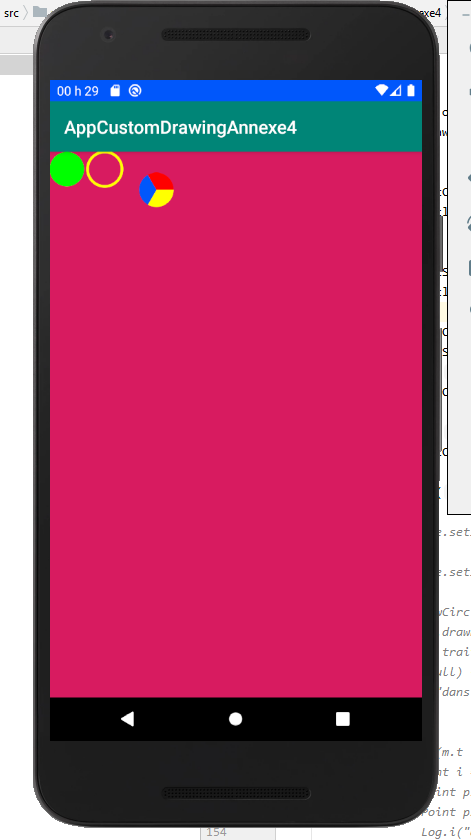
Prenez des notes sur les paramètres à passer :Cliquez ici pour entrer du texte.

6. Dessinez maintenant un cercle vide en :

* Appliquant l’antialias
* Assignant la couleur jaune
* Changeant le style de trait de l’objet Paint pour Paint.Style.STROKE ( Le style par défaut des objets Paint est Paint.Style.FILL )
* Changeant la largeur du trait



7. Faites à présent un graphique circulaire avec un / objet Paint. Il sera composé de trois sections de 120 degrés chacune de couleur différente que vous pouvez définir dans le fichier colors.xml



Prendre des notes sur les paramètres : Cliquez ici pour entrer du texte.