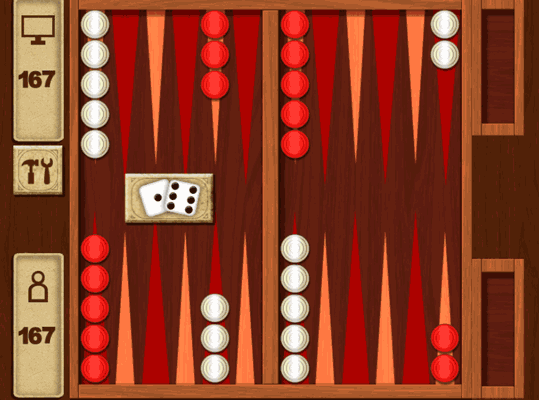
# Annexe 14– Glisser et déposer « Drag&Drop »

Dans cet exercice, on apprendra à :

* Utiliser des fichiers .xml pour représenter des formes et des tracés
* Gérer des événements OnTouchEvent et OnDragEvent afin de pouvoir glisser et déposer des objets d’un conteneur à un autre.

**Exercice : Un début de Backgammon**

Sans trop savoir comment y jouer, on a tous déjà vu un jeu de backgammon :



Nous allons reproduire l’apparence d’un tel jeu sans cependant coder la logique de ce dernier; le but est de pouvoir prendre une pastille et la déplacer dans une autre colonne.

1. ( *3 min* ) Créez un nouveau projet ( **API 24** )
2. ( *5 min* ) Utilisez- le fichier jeton.jpg et placez-le à l’endroit approprié pour les ressources images externes.

Quel est-il, où doit-on placer ces ressources-images ?

Pour placer les ressources-images comme jeton.jpg, vous devez les placer dans le répertoire approprié des ressources de votre projet. Habituellement, vous devriez les placer dans le répertoire res/drawable du projet. Si le répertoire drawable n'existe pas, vous pouvez le créer dans le dossier res..

1. ( *15 min* ) Toujours dans le même dossier qu’en 2, créez un nouveau fichier .xml vide ( **placez-vous sur le dossier drawable**, new 🡪 File ). Appelez-le background\_contenant.xml. Un peu comme on l’a fait dans l’extra de l’annexe 11B (<selector>), on va se servir d’une balise <shape> afin de développer un background intéressant pour nos futurs LinearLayout **( on aura un LinearLayout par colonne du jeu )**

Vous pouvez lire sur les objets <shape> ici :

<https://developer.android.com/guide/topics/resources/drawable-resource.html#Shape>

Quels 4 types de formes peut-on dessiner :

Rectangle (<shape android:shape="rectangle">)

Ovale (<shape android:shape="oval">)

Ligne (<shape android:shape="line">)

Anneau (<shape android:shape="ring">)

Oops, il n’y a pas de triangle, nous y reviendrons…codez le fichier .xml suivant dans votre fichier .xml ( comparez avec le fichier d’aide dont le lien est ci-dessus )  et appliquez-le comme background de vos colonnes :

*<?***xml version="1.0" encoding="UTF-8"***?>*<**shape xmlns:android="http://schemas.android.com/apk/res/android"  
 android:shape="rectangle"** >  
  
 <**stroke  
 android:width="2dp"  
 android:color="#FFFFFF"** />  
  
 <**gradient  
 android:angle="225"  
 android:endColor="#2ECCFA"  
 android:startColor="#000000"** />  
  
 <**corners  
 android:bottomLeftRadius="7dp"  
 android:bottomRightRadius="7dp"  
 android:topLeftRadius="7dp"  
 android:topRightRadius="7dp"** />  
  
</**shape**>



1. ( 10 min ) Créez un autre fichier .xml dans le dossier drawable ( background\_contenant\_selectionne.xml ) qui apparaîtra lorsque l’usager prendra ou déposera la pastille. Le fichier sera similaire au précédent mais faites-en sorte que le contour sera rouge.
2. ( 15 min ) Attaquez-vous maintenant au fichier de positionnement .xml présent dans le dossier layout. On veut reproduire l’image ci-haut. J’ai converti le ConstraintLayout présent par défaut par un LinearLayout avec 4 sous-conteneurs ).

Les 4 conteneurs représentant les colonnes seront des LinearLayout verticaux; leur background sera la forme rectangulaire présente dans le fichier du #3 ( **on y accède avec @drawable/nom du fichier .xml représentant la shape** )

À l’intérieur des colonnes, ajoutez un ImageView représentant la pastille que vous avez dessiné. Centrez les balles et placez-les en bas des conteneurs LinearLayout.

**30 min - EXTRA : Si vous voulez des triangles comme fond de vos LinearLayout.**

La « shape » triangle n’existant pas, on peut se rabattre sur des VectorDrawables …

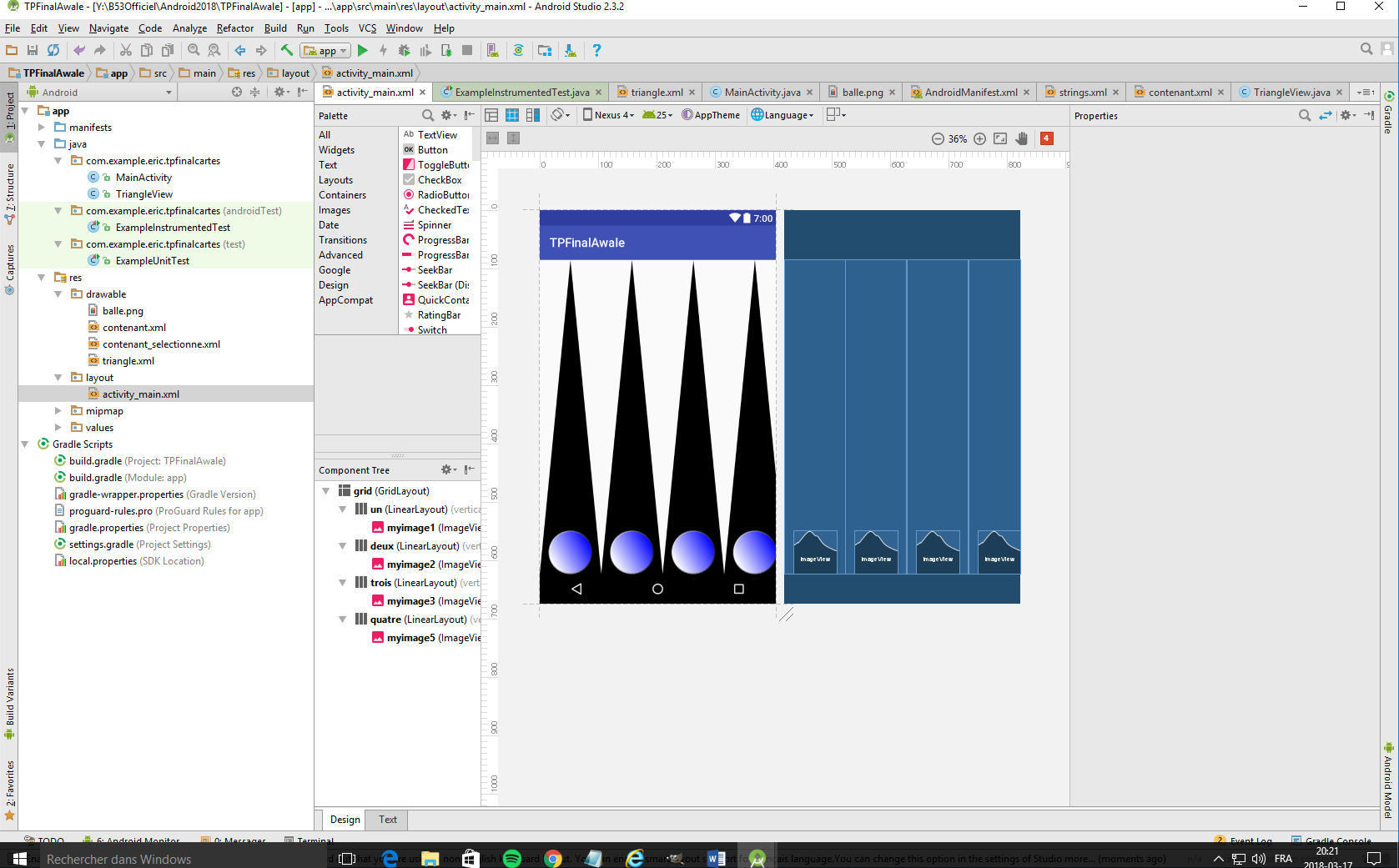
1. Créer un autre fichier .xml dans le dossier drawable
2. Codez-y ceci :
3. *<?***xml version="1.0" encoding="utf-8"***?>*<**vector xmlns:android="http://schemas.android.com/apk/res/android"  
    android:height="100dp"  
    android:width="100dp"  
    android:viewportHeight="100"  
    android:viewportWidth="100"** >  
    <**group  
    android:name="triableGroup"**>  
    <**path  
    android:name="triangle"  
    android:fillColor="#000000"  
    android:pathData="\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ "** />  
    </**group**>  
   </**vector**>

Il vous reste à remplir le “pathData”…lisez le document suivant afin d’y parvenir ( sections commandes et example ) :

<https://medium.com/@ali.muzaffar/understanding-vectordrawable-pathdata-commands-in-android-d56a6054610e>

<https://dev.to/brightdevs/creating-simple-vector-drawables-in-android-studio-bbm>

En utilisant le nouveau fichier .xml dans votre fichier de positionnement comme fond des LinearLayout, vous devriez obtenir  :



De la même manière qu’avec les rectangles, produisez un second xml représentant le triangle sélectionné avec un triangle avec un contour rouge

1. Nous devons maintenant coder notre Activité ( 35 min )

La technique employée pour faire du « drag & drop » s’appuie sur deux interfaces-écouteurs :

OnDragListener : les sources seront les conteneurs dans lesquels on va déplacer les images ( les LinearLayouts )

View.OnTouchListener : les sources seront ce qu’on veut déplacer ( les ImageViews )

À l’aide d’une boucle, inscrivez les différentes sources aux écouteurs cités ci-haut ( la méthode getChildAt sera utile ! )

1. L’écouteur OnTouchListener comprend la méthode onTouch.

* Quand on touche la balle , on doit démarrer le démarche en appelant la méthode **startDragAndDrop** ( startDrag pour les téléphones API23 et moins ) . Lisez dans l’API sur cette méthode pour connaître les paramètres à y passer
* Nous devons rendre invisible la balle à son origine une fois le processus de « drag & drop » commencé; utilisez la méthode setVisibility sur l’objet que vous vous apprêtez à « dragger ».

1. On doit à présent coder la méthode onDrag faisant partie de l’interface View.OnDragListener ; rappelez-vous que la vue en paramètre est la source, donc dans notre cas les LinearLayouts.

Lisez les informations dans l’API relatives au processus de « Drag & Drop » :

<https://developer.android.com/guide/topics/ui/drag-drop.html#AboutDragging>

…en particulier la section « The drag & drop Process » où on parle des 4 états du processus. Ces 4 états peuvent être identifiés par des constantes retournées par la méthode getAction() sur l’objet événementiel DragEvent :

|  |  |
| --- | --- |
| **État du processus** | **Ce qu’on doit faire dans cet exercice** |
| « started » démarré | Rien à accomplir |
| « continuing » entré / sorti | Quand on entre, changer le background de la source avec notre drawable background\_selectionne /triangleselectionne…quand on sort de la source remettre le background contenant/triangle |
| « dropped » relâché | Étape importante :  Dans le touchListener tantôt nous n’avons que rendu la pastille invisible et nous trainons son « ombrage » ( shadow ) . Nous devons :   * Récupérer la pastille de départ à partir de l’objet événementiel * Récupérer son parent afin de l’enlever définitivement de ce conteneur * Récupérer la source actuelle ( le LinearLayout de destination ) * Y ajouter la pastille récupérée * La mettre visible avec la méthode setVisibility |
| « ended » terminé | Changer le background pour le background\_contenant |

\*\*\* Pour charger un Drawable en Java :

getResources().*getDrawable*(R.drawable.contenant\_selectionne);

BRAVO !!!