Les Canvas Android [1]

Version du 23 septembre 2020, 10:09

Objectifs

Le but de cette séance est de vous permettre de réaliser des applications de jeux en 2D basées sur les Canvas.

▷ **Exercice 1** *Mise en place*

Un canvas peut être utilisé pour dessiner dans une image ou dans une vue. Pour l'utiliser dans une vue, il vous sera nécessaire de surcharger la méthode `onDraw()` de la vue, dans laquelle vous effectuerez vos actions avec le Canvas.

Il existe une vue spéciale, permettant d'utiliser un thread secondaire pour le rendu de la surface indépendamment de la hiérarchie de vue : la `SurfaceView`.

Questions :

1. En partant d'un squelette d'application simple, créez une classe `DrawingView` héritant de `SurfaceView` et implémentant `SurfaceHolder.Callback`.
2. Dans la classe `DrawingView` : créez un constructeur et surchargez les fonctions :
   — Créez un constructeur suivant le code :
   ```
   public DrawingView(Context context, AttributeSet attr) {
   super(context, attr);
   getHolder().addCallback(this);
   }
   ```
   — La fonction `surfaceChanged` permet d'effectuer des actions lors de la mise à jour de la vue. Il faut donc invalider la vue pour relancer le processus d'affichage.
   ```
   @Override
   public void surfaceChanged(SurfaceHolder arg0, int arg1, int arg2, int arg3) {
   // Permet de relancer le rendu de la surface s'il y a un changement
   this.invalidate();
   }
   ```
   — La méthode `onDraw(Canvas canvas)` sera la fonction dans laquelle seront réalisés les affichages.
3. Nous allons maintenant créer le thread pour le rendu :
   — créez une classe `DrawingThread` héritant de `Thread`. Votre classe aura 3 attributs privés : un `SurfaceHolder` et un `DrawingView` (attribués par la constructeur au thread) ainsi qu'un booléen d'exécution. Pour le booléen d'exécution,

1. Sujet réalisé par Nicolas Pavie (nicolas.pavie@unilim.fr)

n'oubliez pas de définir un setter. Ce booléen sera utilisé pour contrôler la boucle while (cf code dans la question suivante).

— Une fois votre Thread crée, surchargez la méthode `run()` suivant le code :

```
public void run() {
Canvas c;
while(myBoolean) {
c = null;
try {
c = mySurfaceHolder.lockCanvas(null);
synchronized(mySurfaceHolder) {
// ici, mettre les méthodes de modification des positions des objets
myDrawingView.postInvalidate();
// = remplacement de onDraw() (optimisation)
}
} finally {
if( c != null) {
mySurfaceHolder.unlockCanvasAndPost(c);
}
}
}
}
```

4. Dans la classe `DrawingView`, vous devez surcharger les méthodes de création et destruction de la surface d'affichage :
   — `public void surfaceCreated (SurfaceHolder holder);`
   — `public void surfaceDestroyed (SurfaceHolder holder);`
   Lors de la création de la surface, vous devez créer le `DrawingThread` et le démarrer (changer le booléen et utiliser la méthode `start()` du thread). Lors de la destruction de la surface, vous devez rechanger le booléen et utiliser la méthode de `join()` sur le thread. N'oubliez pas que votre thread peut renvoyer une exception (de type `InterruptedException`).
   Si vous utilisez `postInvalidate()` dans le thread, vous devez ajouter la fonction `setWillNotDraw(false)` à la création de la surface.
5. Si votre classe est fonctionnelle vous pouvez l'affecter à la hiérarchie de vues de votre application en passant par l'éditeur graphique. Pour cela, vous devez retrouver le layout affecté à votre activité (`R.layout.main_activity` par exemple), et modifier sa configuration qui se trouve dans `res > layout > main_activity.xml`.

En mode d'édition graphique, vous retrouverez votre `DrawingView` dans la catégorie "Containers → View" . Je vous recommande de supprimer les différents padding du layout parent et de régler la largeur et la hauteur de votre vue sur "match_parent".

▷ **Exercice 2** *L'affichage*

Le canvas vous permet de dessiner des formes géométriques ou des images. Cependant, pour le dessin géométrique, il est nécessaire de lui associer un style de dessin à travers l'objet `Paint`. Cet objet peut définir le style d'affichage, la couleur, l'épaisseur du contour etc.

Questions :

1. Définissez deux objets `Paint`, l'un pour afficher les objets en bleu remplis `(Style.FILL)` et l'autre les objets avec un contour rouge d'épaisseur 3 pixels.
2. Dans la fonction `onDraw()`, lancez l'affichage d'un rectangle bleu en haut à gauche de l'écran grâce à la méthode `drawRect()` du canvas.
3. Créez ensuite un damier noir et bleu qui remplisse toute la surface.
4. En utilisant le style contour, affichez un cercle rouge au milieu de l'écran.
5. Affichez une image (de préférence une petite image pour la suite). Pour cela, vous pouvez utilisez la classe statique `BitmapFactory` pour récupérer l'image depuis vos ressources et la sauvegarder dans un objet `Bitmap`.

▷ **Exercice 3** *Les Animations*

Nous allons maintenant animer une image dans le canvas. Pour cela nous allons jouer sur la position initiale du rectangle d'affichage de l'image (bord haut gauche du rectangle d'affichage).

Il nous faut plusieurs information pour mettre en place notre animation :

— la position initiale de la balle à l'écran.
— le vecteur 2D de déplacement (direction * vitesse).

Questions :

1. Commencez par déclarer les variables nécessaires à cette animation.
2. Dans la fonction `onDraw()`, initialisez puis modifiez la position courante de l'image.
3. Gérez le rebond contre le bord du cadre en testant la position courante.