UPMC/Licence/Info/2I013
Groupe 3 – Rushdroid
Devoir sur table

22 Mars 2016

1

L'objectif de ce devoir est de décrire la réalisation pour Android d'un ≪jeu de couleur≫ proche de ceux vus en cours : approcher une couleur donnée à partir d'une autre couleur de départ en faisant varier ses composantes RVB (rouge, vert et bleu), en mélange additif. Les composantes d'une couleur sont des nombres variant entre 0 et 255 (un octet). Si les trois composantes valent 225, la couleur résultante est du blanc.

Le jeu est modélisé par la classe `Model` (document 2).

La réalisation Android comprend une *Application* appelée `ColorApp` et une unique *Activité*, appelée `ColorAct`. L'application `ColorApp` fournit la méthode `getModel()` qui donne l'instance du modèle du jeu.

Question 1 Donner le fichier `AndroidManifest.xml` de ce projet.

2

Le joueur dispose d'une couleur de départ : le blanc. Elle sera affichée dans l'interface de jeu comme couleur de fond d'un `TextView`. Nous l'appellerons `currentColorView`.

Le jeu propose un couleur à atteindre qui est également donnée comme couleur de fond d'un `TextView`. Nous l'appellerons `endColorView`.

Pour faire varier les composantes de sa couleur, le joueur dispose de trois ≪curseurs≫. Un pour chacune des couleurs primaires : rouge, vert et bleu. Les curseurs seront réalisés par trois vues que nous définirons : `RedCursorView`, `GreenCursorView` et `BlueCursorView`.

Question 2 Dessinez l'interface décrite dans le fichier `activity_color_x.xml` (document 1).

3

Dans l'interface graphique fournie par l'activité `ColorAct`, les actions sur les curseurs doivent modifier la couleur de fond de `currentColorView`. Lorsque cette couleur est suffisamment proche de celle de `endCursorView`, la partie est finie et le message de `endCursorView` devient `"Color Found !"`.

Le changement de couleur de `currentColorView` et de message de `endCursorView` sont réalisés par des méthodes de la classe `ColorAct` :

— `void updateCurrentColorView(Model m)` affecte à la couleur de fond de `currentColorView` la couleur courante du joueur ;
— `void updateEndColorView(Model m)` modifie le message de `endColorView` si la partie est finie.

Question 3a Quelle méthode faut-il surcharger pour initialiser les composants de l'interface de la partie ? Donnez la définition de cette méthode.

Question 3b Donnez la définition des méthodes `updateCurrentColorView` et `updateEndColorView`.

Rappel : dans une vue, la méthode `getContext()` donne l'instance de l'activité dont la vue fait partie.

4

Les trois curseurs `RedCursorView`, `GreenCursorView` et `BlueCursorsView` sont des sous classes de la classe `ColorCursorView` qui est elle-même sous classe de `SurfaceView`.

Un curseur représente une quantité de couleur (comprise entre 0 et 255) dont on peut faire varier la valeur en faisant glisser son doigt sur la surface du curseur.

La couleur associée à un curseur est donnée par la méthode `color()`, la quantité représentée par le curseur est donnée par la valeur du champ `value`, de type `int`.

Le dessin d'un curseur est un simple rectangle dont la hauteur est égale à la valeur de `value`.

L'action sur un curseur est divisée en trois phases aux quelles sont associées des actions :
— le doigt touche le curseur : on mémorise la position de départ du doigt ;
— le doigt glisse vers le haut ou le bas : on modifie l'état du modèle de jeu et la valeur du curseur ; on mémorise une nouvelle position de départ ;
— le doigt quitte le curseur ; on teste si la couleur obtenue est proche de celle à atteindre ; si oui, on modifie le message de `endCursor`.

Question 4a Donnez la (re)définition de la méthode `draw(Canvas c)` de la classe `ColorCursorView`.

Question 4b Donnez la (re)définition de la méthode `onTouchEvent(MotionEvent event)`.

Document 1 fichier `activity_color_x.xml` (sur 2 colonnes)

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <TextView
        android:id="@+id/title"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="30dp"
        android:text="The color X game" />

    <TextView
        android:id="@+id/currentColorView"
        android:textSize="50dp"
        android:text="Current Color"
        android:gravity="center"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="15dp"/>

    <LinearLayout
        android:layout_gravity="center"
        android:orientation="vertical"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content">

        <appdroid.colorxdroid.RedCursorView
            android:id="@+id/redView"
            android:layout_margin="10px"
            android:layout_width="80px"
            android:layout_height="255px" />

        <appdroid.colorxdroid.GreenCursorView
            android:layout_margin="10px"
            android:id="@+id/greenView"
            android:layout_width="80px"
            android:layout_height="255px" />

        <appdroid.colorxdroid.BlueCursorView
            android:layout_margin="10px"
            android:id="@+id/blueView"
            android:layout_width="80px"
            android:layout_height="255px" />

    </LinearLayout>

    <TextView
        android:id="@+id/endColorView"
        android:textSize="50dp"
        android:text="Color to find"
        android:gravity="center"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="15dp" />

</LinearLayout>
```

Document 2 Le modèle (interface)

```
public class Model {

    /** Construteur: couleur de départ = WHITE; couleur de fin aléatoire */
    Model()

    /** Couleur courante du joueur */
    int getCurrentColor()

    /** Couleur de fin, couleur à atteindre */
    int getEndColor()

    /** Modification de la couleur courante du joueur:
        c : composante à modifier (Color.RED ou Color.GREEN ou Color.BLUE
        diff : valeur de la modification (positive ou négative) */
    void changeColor(int c, int diff)

    /** Fin de partie: true si la couleur courante du joeur est proche de la
        couleur de fin */
    boolean endOfGame()

}
```