Week 3 - oefening 3: Color Picker

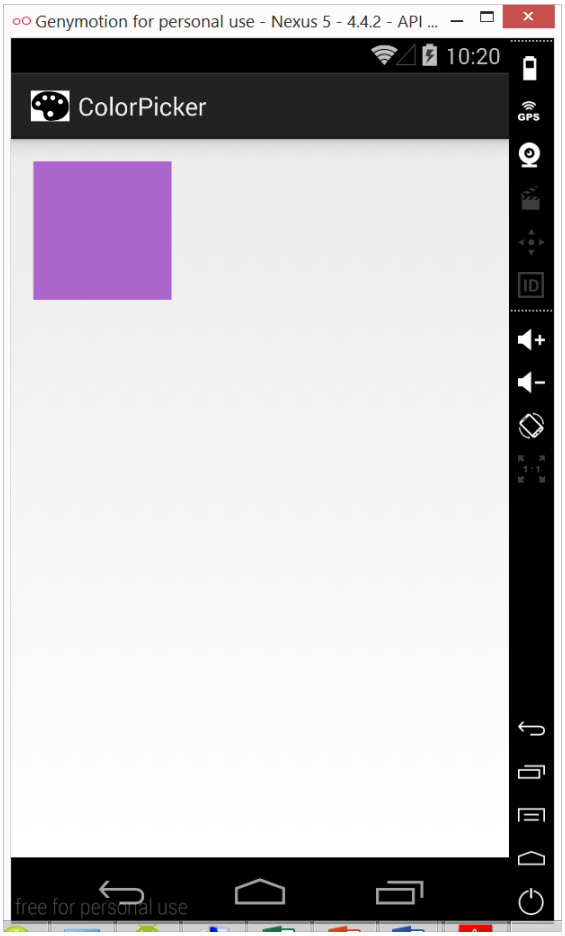
Doelstelling

Verschillende fasen in de lifecyle van een Activity met Custom-view registreren & coderen

Algemene beschrijving

Voor deze en volgende opgaves is het noodzakelijk dat specifieke opties voor ontwikkelaar op uw device of emulator ingesteld kunnen worden. Vóór android 4.2 zaten de ontwikkelaars opties gewoon in het menu, in 4.2 en hoger heeft Google dit verborgen onder 7 keer klikken op Build Number (te vinden onder Settings -> About Phone -> Buildnumber), omdat mensen verkeerde knoppen aanzetten en toen klaagden over problemen. Hierna kan u onder Settings ook 'Developer options' terugvinden.

We bouwen een activity met een basic custom-view (color picker). We wensen de state van de custom-view te bewaren en te herstellen wanneer de gebruiker op de 'home' buttom klikt.



Veel documentatie is te vinden op: developer.android.com

Voorbereiding

Maak een nieuw Android Application Project aan met volgende instellingen.

Application Name:	ColorPicker
Package Name:	be.howest.nmct.colorpicker
Target SDK:	21
Activity:	MainActivity – *layout: activity_main.xml*
Fragment:	ColorViewFragment – *layout: fragment_color_view.xml*
Launch icon:	Zie bijlage

Code voor de aanmaak van Custom-View 'ColorView'

Op volgende link vind je meer informatie over het aanmaken en beheer van custom-views:
http://developer.android.com/training/custom-views/index.html

De ColorView uit deze opgave is een view met een ingekleurde rechthoek. Wanneer de gebruiker op de colorview klikt, krijgt hij in een dialoogvenster een aantal kleuren (strings) te zien. Na keuze wordt de view opnieuw getekend: de rechthoek heeft nu een andere kleur.

Voor deze oefening kan je volgende werkwijze volgen:

1 Voeg onder dezelfde namespace als dat van jouw activity de klasse '**ColorView**' toe. Laat deze klasse erven van 'View' (Android.View). Op bovenstaande link vind je deze belangrijke info.

To allow the Android Developer Tools to interact with your view, at a minimum you must provide a constructor that takes a `Context` and an `AttributeSet` object as parameters. This constructor allows the layout editor to create and edit an instance of your view

Pas dit toe (tip: via Eclipse kan dit gegenereerd worden via menu-item 'Source > Generate Constructors from Superclass')

2 Volgende attributen voegen we aan de klasse toe:

```
private String color = "#FFFFFF";
private Paint paint;
private Rect rect;
```

3 Breid nu de constructor(en) verder uit zodat deze attributen 'paint' en 'rect' geïnitialiseerd en verder ingesteld worden:

- Voor de rechthoek stel je top en left in op 0, voor de lengte en breedte gebruiken we de waardes van de ColorView zelf (oproepen via getWidth(), getHeight()). De rechthoek op de ColorView wordt zo even groot als de ColorView zelf.

- De paint (kleur) van de rechthoek stellen we in a.h.v. de string color. Je kan een string (hexadecimale waarde) naar een Color omzetten via de methode parseColor.

```
paint = new Paint();
paint.setColor(Color.parseColor(color));
```

- Stel een ClickListener in zodat we kunnen reageren als de gebruiker op de rechthoek klikt. In de OnClick-methode roep je de methode 'showColorDialog' aan (zie verder)

```
        setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                showColorDialog();
            }
        });
```

4 Override vervolgens de methode 'onDraw'. Doel is de rechthoek op het canvas te tekenen.

```
@Override
protected void onDraw(Canvas canvas) {
    super.onDraw(canvas);

    rect.set(0, 0, getWidth(), getHeight());
    canvas.drawRect(rect, paint);
}
```

5 Voeg een getter voor het attribuut color toe.

6 Voeg een setter voor het attribuut color. Hierin wijzigen we ook het attribuut paint (op dezelfde manier als hierboven). Roep op het einde de `invalidate`-methode aan om zo de view te laten hertekenen.

7 Voeg onder de map res\values\ het bestand 'color_choices.xml' toe. In dit bestand staat een array van strings opgesomd. Deze array stellen de verschillende keuzes voor de gebruiker voor.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="holo_colors">
        <item >Holo Blue</item>
        <item >Holo Purple</item>
        <item >Holo Green</item>
        <item >Holo Orange</item>
        <item >Holo Red</item>
    </string-array>
</resources>
```

8 Voeg de methode 'showColorDialog' toe: deze methode toont een dialogvenster waarlangs de gebruiker een kleur uit de een opgegeven lijstje kan kiezen. Bij het selecteren van een kleur wordt de methode 'selectColor' aan.

```
protected void showColorDialog() {
    AlertDialog.Builder builder = new AlertDialog.Builder(getContext());

    builder.setTitle("Pick a color")
           .setItems(R.array.holo_colors, new DialogInterface.OnClickListener() {
               public void onClick(DialogInterface dialog, int which) {
                   selectColor(which);
           }
    });

    builder.create().show();
}

protected void selectColor(int which) {
    switch (which) {
    case 0:
        setColor("#33B5E5");
        break;
    case 1:
        setColor("#AA66CC");
        break;
    case 2:
        setColor("#99CC00");
        break;
    case 3:
        setColor("#FFBB33");
        break;
    case 4:
        setColor("#FF4444");
        break;
    default:
        break;
    }
}
```

10 Gebruik nu de colorView in jouw fragment:

```
<be.howest.nmct.MyColorView
    android:id="@+id/myColorView1"
    android:layout_width="100dp"
    android:layout_height="100dp"
    android:layout_below="@id/textview_hello_world"
    android:layout_alignParentStart="true" />
```

Test de activity.

Bewaren van state van customview

Vooraf: ga eerst na welke informatie in situatie 2, 3 en 3bis (cfr. opgave 1) terug gezet wordt. Let op, in situatie 2 en 3 staat de vermelde developer optie (zie opgave 1) niet aan.

In deze opgave wensen we enkel het kleur bij te houden als de gebruiker op de button 'home' klikt. Bij het sluiten van de app via de Back-button hoeft de kleur niet bijgehouden te worden.

Merk op dat zowel in 3bis de kleur niet wordt bijgehouden!

Voor Activities:

Op de link http://developer.android.com/training/basics/activity-lifecycle/recreating.html lezen we het volgende:

By default, the system uses the `Bundle` instance state to save information about each `View` object in your activity layout (such as the text value entered into an `EditText` object). So, if your activity instance is destroyed and recreated, the state of the layout is restored to its previous state with no code required by you. However, your activity might have more state information that you'd like to restore, such as member variables that track the user's progress in the activity.

In tegenstelling tot opgave 2 kunnen we hier niet het kleur gaan reconstrueren op basis van andere (automatisch) bijgehouden informatie. Om dit wél te realiseren maken we gebruik van de **onSaveInstanceState**-methode en de **onRestoreInstanceState**-methode.

*To save additional data about the activity state, you must override the `onSaveInstanceState()` callback method. The system calls this method when the user is leaving your activity and passes it the `Bundle` object that will be saved in the event that your activity is destroyed **unexpectedly**. If the system must recreate the activity instance later, it passes the same `Bundle` object to both the `onRestoreInstanceState()` and `onCreate()` methods.*

Voor Fragments:

Voor Fragments is de situatie vrij analoog.
http://developer.android.com/guide/components/fragments.html

Also like an activity, you can retain the state of a fragment using a Bundle, in case the activity's process is killed and you need to restore the fragment state when the activity is recreated. You can save the state during the fragment's onSaveInstanceState() callback and restore it during either onCreate(), onCreateView(), or onActivityCreated(). For more information about saving state, see the Activities document.

Overschrijf de methodes onSaveInstanceState in het Fragment. Vul de methode 'onCreateView' verder aan. Baseer je op het voorbeeld in bovenstaande link om de getoonde kleur als een string te bewaren. Omgekeerd, om de kleur terug te zetten roep je methode 'setColor' aan uit de ColorView.

Alternatief (extra): een betere benadering is de state in de custom-view zelf te bewaren en te herstellen. Baseer je op het voorbeeld gezien in de theorieles. Overschrijf de methodes `onSaveInstanceState` en `onRestoreInstanceState` uit de klasse View.