



UNIWERSYTET
TECHNOLOGICZNO-HUMANISTYCZNY
im. Kazimierza Pułaskiego w Radomiu

APLIKACJE MOBILNE

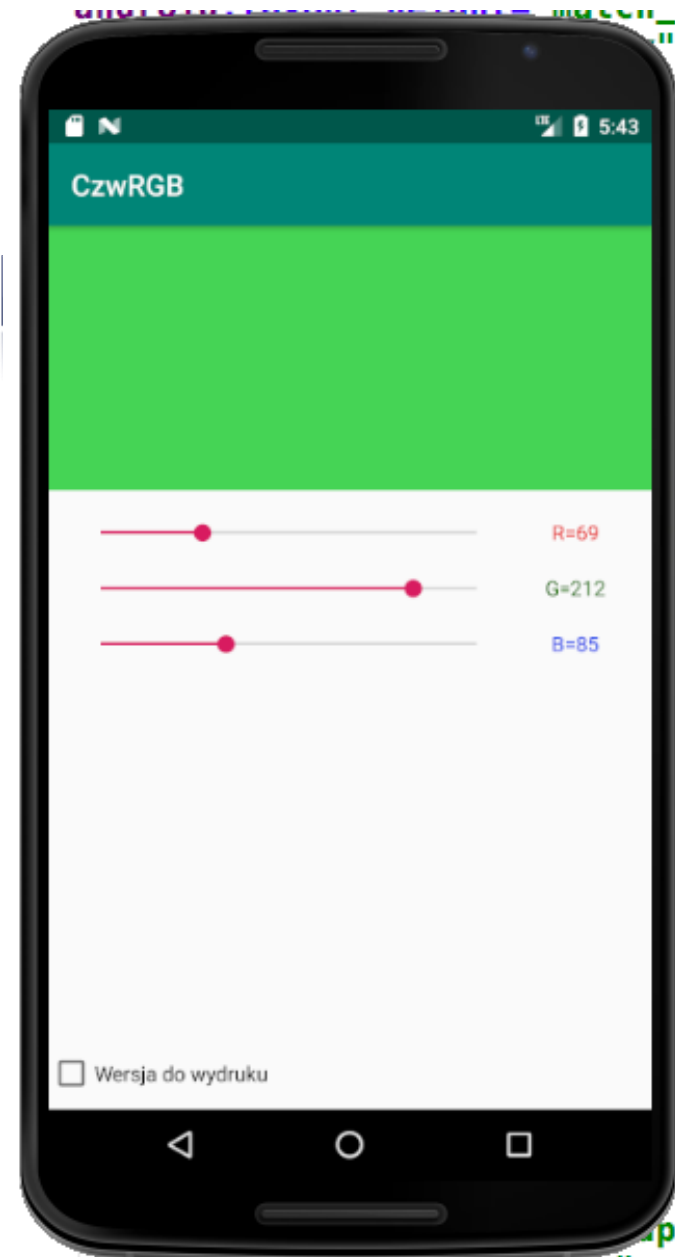
Instrukcja do ćwiczeń

ZADANIE PRAKTYCZNE

Celem ćwiczenia jest wykonanie testera kolorów RGB.

W programie możemy regulować trzy składowe koloru (Red, Green, Blue) za pomocą trzech suwaków.

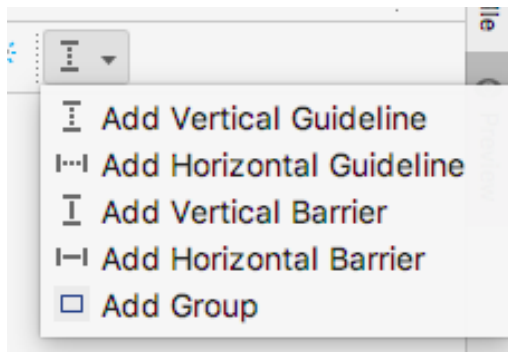
Wynikowy kolor pokazywany będzie na panelu powyżej suwaków.



Przygotowanie wyglądu aplikacji

Edytujemy plik `activity_main.xml`

Rozpoczynamy od przygotowania linii pomocniczych (poziomej i pionowej). Linie pomocnicze najlepiej wstawić za pomocą narzędzia na oknie podglądu



Przygotowanie wyglądu aplikacji

W kodzie otrzymamy:

```
9
10
11 <android.support.constraint.Guideline
12     android:id="@+id/guideline"
13     android:layout_width="wrap_content"
14     android:layout_height="wrap_content"
15     android:orientation="horizontal"
16     app:layout_constraintGuide_percent="0.3" />
17
18 <android.support.constraint.Guideline
19     android:id="@+id/guideline2"
20     android:layout_width="wrap_content"
21     android:layout_height="wrap_content"
22     android:orientation="vertical"
23     app:layout_constraintGuide_percent="0.75" />
```

Położenie linii pomocniczych (przez edytor ustawione w jednostkach bezwzględnych) zastępujemy położeniem procentowym (jak na rys.)

Przygotowanie wyglądu aplikacji

Dodajemy panel View

Kolejnym krokiem jest dodanie kontrolki View, która posłuży do wyświetleni

```
<View  
    android:id="@+id/panel"  
    android:layout_width="0dp"  
    android:layout_height="0dp"  
    android:background="@color/colorPrimaryDark"  
    app:layout_constraintTop_toTopOf="parent"  
    app:layout_constraintLeft_toLeftOf="parent"  
    app:layout_constraintRight_toRightOf="parent"  
    app:layout_constraintBottom_toTopOf="@id/guideline"  
/>
```

Panel przyciągamy do lewego i prawo brzegu ekranu (parent) oraz poziomej linii pomocniczej (guadeline).

Wysokość i szerokość ustawimy na 0dp – aby panel rozciągnął się na cały wskazany obszar.

Pamiętajmy też o nadaniu ID (panel) oraz wybraniu jakiegoś koloru (będziemy widzieć panel na podglądzie)

Przygotowanie wyglądu aplikacji

Dodajemy suwaki - **SeekBar**

Kolejnym krokiem jest dodanie trzech suwaków (kontrolerek SeekBar)

```
35 <SeekBar
36     android:id="@+id/suwakR"
37     android:layout_width="0dp"
38     android:layout_height="wrap_content"
39     app:layout_constraintTop_toBottomOf="@id/guideline"
40     app:layout_constraintLeft_toLeftOf="parent"
41     app:layout_constraintRight_toRightOf="@id/guideline2"
42     android:layout_marginLeft="20dp"
43     android:layout_marginTop="20dp"
44     android:max="255"
45     />
46
47 <SeekBar ...>
48
49 <SeekBar ...>
```

Pierwszy suwak umieszczony jest pod poziomą linią pomocnicą (guideline), pomiędzy lewym brzegiem ekranu (parent) a pionową linią pomocniczą (guideline2). Kolejne suwaki umieszczamy pod poprzednimi.

Przygotowanie wyglądu aplikacji

```
47 <SeekBar
48     android:id="@+id/suwakG"
49     android:layout_width="0dp"
50     android:layout_height="wrap_content"
51     app:layout_constraintTop_toBottomOf="@id/suwakR"
52     app:layout_constraintLeft_toLeftOf="parent"
53     app:layout_constraintRight_toRightOf="@id/guideline2"
54     android:layout_marginLeft="20dp"
55     android:layout_marginTop="20dp"
56     android:max="255"
57 />
58
59 <SeekBar
60     android:id="@+id/suwakB"
61     android:layout_width="0dp"
62     android:layout_height="wrap_content"
63     app:layout_constraintTop_toBottomOf="@id/suwakG"
64     app:layout_constraintLeft_toLeftOf="parent"
65     app:layout_constraintRight_toRightOf="@id/guideline2"
66     android:layout_marginLeft="20dp"
67     android:layout_marginTop="20dp"
68     android:max="255"
69 />
```

Wartości „max” suwaków ustawiamy na 255

Pamiętajmy o nadaniu i ID – „suwakR” „suwakG”, „suwakB”

Przygotowanie wyglądu aplikacji

Dodajemy Opisy do suwaków – będą one wyświetlały wartości składowych koloru.

```
70
71
72 <TextView
73     android:id="@+id/opisR"
74     android:layout_width="wrap_content"
75     android:layout_height="wrap_content"
76     android:text="R=0"
77     android:textColor="#e60d0d"
78     app:layout_constraintLeft_toRightOf="@id/guideline2"
79     app:layout_constraintRight_toRightOf="parent"
80     app:layout_constraintBottom_toBottomOf="@id/suwakR"/>
```

Pierwszy element TextView umieszczony jest pomiędzy pionową linią pomocniczą (guideline2) a prawym brzegiem ekranu (parent). W pionie jest on wyrównany do odpowiadającego mu suwaka (suwakR) „Bottom_toBottomOf” – czyli „górze równo z górą”

Kolor tekstu ustawić można na kolor odpowiadający prezentowanej składowej.

Przygotowanie wyglądu aplikacji

```
80
81
82 <TextView
83     android:id="@+id/opisG"
84     android:layout_width="wrap_content"
85     android:layout_height="wrap_content"
86     android:text="G=0"
87     android:textColor="#207005"
88     app:layout_constraintLeft_toRightOf="@id/guideline2"
89     app:layout_constraintRight_toRightOf="parent"
90     app:layout_constraintBottom_toBottomOf="@id/suwakG"/>
91
92 <TextView
93     android:id="@+id/opisB"
94     android:layout_width="wrap_content"
95     android:layout_height="wrap_content"
96     android:text="B=0"
97     android:textColor="#0d32e9"
98     app:layout_constraintLeft_toRightOf="@id/guideline2"
99     app:layout_constraintRight_toRightOf="parent"
100    app:layout_constraintBottom_toBottomOf="@id/suwakB"/>
```

Podobnie ustawiamy dwa kolejne opisy.

Pamiętać należy o nadaniu ID – „opisR”, „opisG”, opisB”

Na tym etapie możemy przetestować interface aplikacji i przejść do jego oprogramowania

Przygotowanie wyglądu aplikacji

Zawartość pliku `activity_main.xml` :

- po „zwinieciu” kodów kontroltek – powinna mieć poniższą strakturę


```
1  <?xml version="1.0" encoding="utf-8"?>
2  <android.support.constraint.ConstraintLayout xmlns:android="http://schemas.a
3      xmlns:app="http://schemas.android.com/apk/res-auto"
4      xmlns:tools="http://schemas.android.com/tools"
5      android:layout_width="match_parent"
6      android:layout_height="match_parent"
7      tools:context=".MainActivity">
8
9      <android.support.constraint.Guideline...>
15
16      <android.support.constraint.Guideline...>
22
23      <View...>
33
34      <SeekBar...>
45
46      <SeekBar...>
57
58      <SeekBar...>
69
70      <TextView...>
79
80      <TextView...>
89
90      <TextView...>
99
100 </android.support.constraint.ConstraintLayout>
```

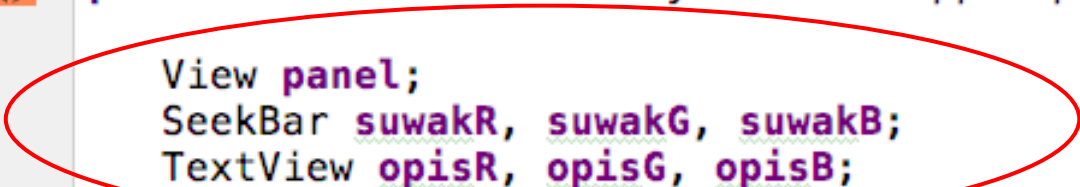
Na tym etapie możemy przetestować interfejs aplikacji i przejść do jego oprogramowania

Przygotowanie kodu aplikacji

Edytujemy plik MainActivity.java

Rozpoczynamy od dodania pól (zmiennych) które reprezentowały będą w kodzie przygotowane wcześniej kontrolki.

```
10
11  public class MainActivity extends AppCompatActivity {
12
13     View panel;
14     SeekBar suwakR, suwakG, suwakB;
15     TextView opisR, opisG, opisB;
```



Umieszczamy je wewnątrz klasMainActivity, ale nie wewnątrz żadnej z jej metod.

Uwaga: Jeżeli typ którejkolwiek zmiennej (View, SeekBar...) będzie zaznaczony na czerwono – najprawdopodobniej brak jej importu – ustaw na nim kursor i kliknij ALT + ENTER – to powinno rozwiązać problem

Przygotowanie kodu aplikacji

Kolejnym krokiem jest „połączenie” przygotowanych zmiennych z wpisami w pliku XML. Robimy to za pomocą funkcji `findViewById()`. Jej parametrem jest ID, które nadaliśmy naszym kontrolkom.

```
19
20
21
22
23
24
25
26
27
28
29
30
```

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    panel = (View)findViewById(R.id.panel);
    suwakR = (SeekBar)findViewById(R.id.suwakR);
    suwakG = (SeekBar)findViewById(R.id.suwakG);
    suwakB = (SeekBar)findViewById(R.id.suwakB);
    opisR = (TextView)findViewById(R.id.opisR);
    opisG = (TextView)findViewById(R.id.opisG);
    opisB = (TextView)findViewById(R.id.opisB);
}
```

Polecenia te umieszczamy najczęściej wewnątrz metody `onCreate` aktywności – czyli wykonane będą podczas jej tworzenia

Przygotowanie kodu aplikacji

Kolejnym krokiem jest przygotowanie „słuchacza zdarzeń” czyli metody, która zareaguje na przesunięcie suwaka.

```
SeekBar.OnSeekBarChangeListener sluchaczSuwakow =  
    new SeekBar.OnSeekBarChangeListener() {  
        @Override  
        public void onProgressChanged(SearchBar seekBar, int progress, boolean fromUser) {  
            rysuj();  
        }  
        @Override  
        public void onStartTrackingTouch(SearchBar seekBar) {  
        }  
        @Override  
        public void onStopTrackingTouch(SearchBar seekBar) {  
        }  
    };
```

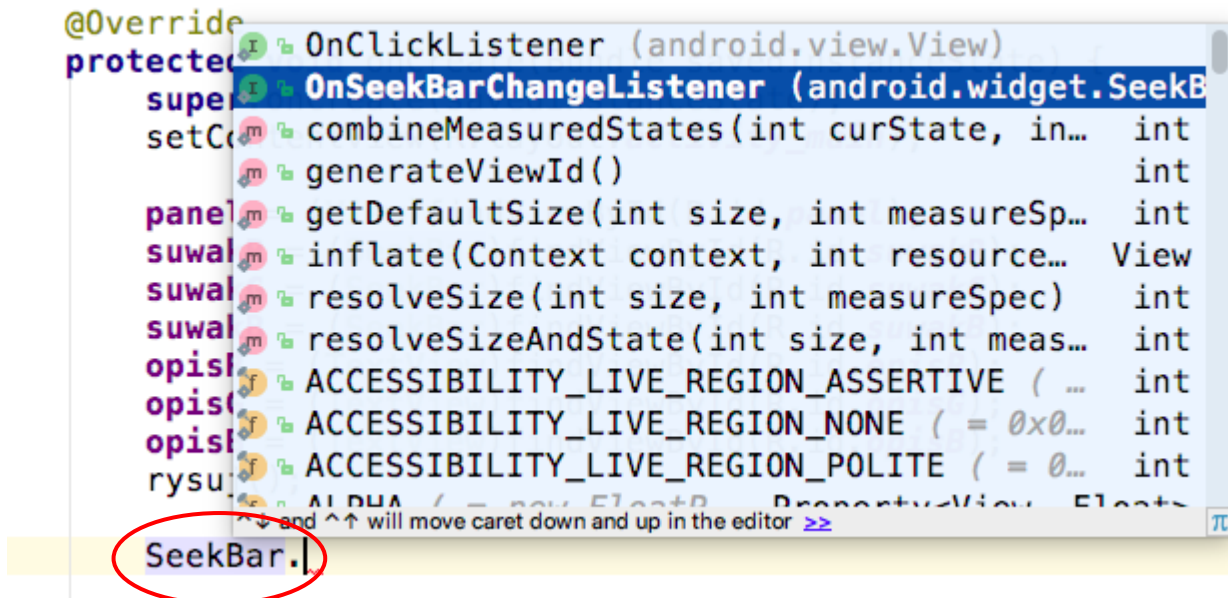
Słuchacza zdarzeń tworzymy najczęściej wewnątrz metody onCreate aktywności.

Tworzenie słuchacza – patrz następne



Przygotowanie kodu aplikacji

Słuchacz zdarzeń (a właściwie słuchacze – bo mamy do dyspozycji różne dla różnych kontrolki i różnych zdarzeń) to dosyć skomplikowane fragmenty kodu. Nie warto uczyć się ich na pamięć – lepiej skorzystać z pomocy kreatora.

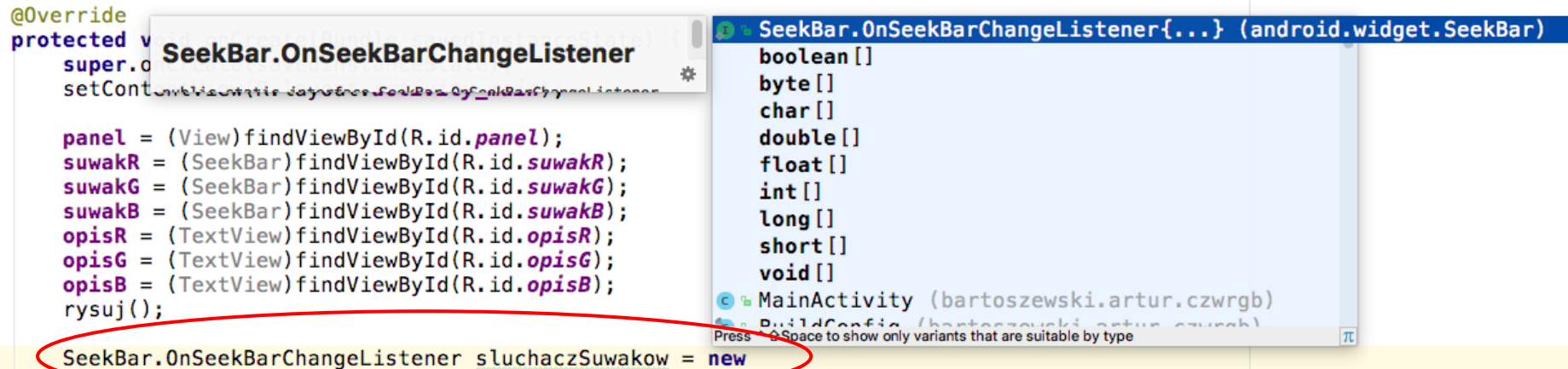


1. Wpisujemy nazwę klasy SeekBar. (z kropką) i wybieramy z listy „OnSeekBarChangeListener”



Przygotowanie kodu aplikacji

2. Następnie wybieramy nazwę słuchacza,
3. Piszemy „= new” i za pomocą Ctrl + Spacja wywołujemy podpowiedzi.
4. Wybieramy kreator



The screenshot shows an IDE with two windows. The left window displays Java code for an Android application. The right window shows a list of suggestions for the `SeekBar.OnSeekBarChangeListener` interface. A red circle highlights the line `SeekBar.OnSeekBarChangeListener sluchaczSuwakow = new` in the left window, indicating the point where the user is typing and expecting suggestions.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    panel = (View)findViewById(R.id.panel);
    suwakR = (SeekBar)findViewById(R.id.suwakR);
    suwakG = (SeekBar)findViewById(R.id.suwakG);
    suwakB = (SeekBar)findViewById(R.id.suwakB);
    opisR = (TextView)findViewById(R.id.opisR);
    opisG = (TextView)findViewById(R.id.opisG);
    opisB = (TextView)findViewById(R.id.opisB);
    rysuj();

    SeekBar.OnSeekBarChangeListener sluchaczSuwakow = new
```

SeekBar.OnSeekBarChangeListener{...} (android.widget.SeekBar)

- boolean[]
- byte[]
- char[]
- double[]
- float[]
- int[]
- long[]
- short[]
- void[]

MainActivity (bartoszewski.artur.czwrgb)

BuildConfig (bartoszewski.artur.czwrgb)

Press Space to show only variants that are suitable by type

Kreator utworzy kod słuchacza zdarzeń (patrz następny slajd)



Przygotowanie kodu aplikacji

```
SeekBar.OnSeekBarChangeListener słuchaczSuwakow = new SeekBar.OnSeekBarChangeListener() {  
    @Override  
    public void onProgressChanged(SearchBar seekBar, int progress, boolean fromUser) {  
    }  
  
    @Override  
    public void onStartTrackingTouch(SearchBar seekBar) {  
    }  
  
    @Override  
    public void onStopTrackingTouch(SearchBar seekBar) {  
    }  
};
```

Uwaga: słuchacz zdarzeń stanowi klasę wbudowaną – należy zakończyć go średnikiem



Przygotowanie kodu aplikacji

Teraz należy oprogramować wybrana metodę słuchacza zdarzeń.
Do wyboru mamy trzy:

1. Metoda wykonywana przy każdej zmianie wartości suwaka – czyli „w czasie rzeczywistym” podczas jego przesuwania

```
@Override  
public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {  
  
}
```

2. Metoda wykonywana raz przy starcie ruchu suwakiem

```
@Override  
public void onStartTrackingTouch(SeekBar seekBar) {  
  
}
```

3. Metoda wykonywana raz po zakończeniu przesuwania suwaka

```
@Override  
public void onStopTrackingTouch(SeekBar seekBar) {  
  
}
```

Przygotowanie kodu aplikacji

Ponieważ program nie wykonuje czasochłonnych obliczeń wynik obserwować będziemy na bieżąco – w tym celu oprogramujemy metodę „onProgressChange()”

```
SeekBar.OnSeekBarChangeListener sluchaczSuwakow =  
    new SeekBar.OnSeekBarChangeListener() {  
        @Override  
        public void onProgressChanged(SearchBar seekBar, int progress, boolean fromUser) {  
            rysuj();  
        }  
        @Override  
        public void onStartTrackingTouch(SearchBar seekBar) {  
        }  
        @Override  
        public void onStopTrackingTouch(SearchBar seekBar) {  
        }  
    }:  
};
```

W metodzie „onProgressChange()” umieszczone zostało wywołanie metody „rysuj()” jest to nasza własna metoda (żadna z metod systemowych), którą należy teraz utworzyć.



Przygotowanie kodu aplikacji

```
50 private void rysuj() {  
51  
52     int kolor = Color.argb( alpha: 255,  
53         suwakR.getProgress(),  
54         suwakG.getProgress(),  
55         suwakB.getProgress());  
56  
57     panel.setBackgroundColor(kolor);  
58  
59     opisR.setText("R="+String.valueOf(suwakR.getProgress()));  
60     opisG.setText("G="+String.valueOf(suwakG.getProgress()));  
61     opisB.setText("B="+String.valueOf(suwakB.getProgress()));  
62  
63 }  
64 }
```

W metodzie „`rysuj()`” ustawiamy kolor jako „`argb()`” – pierwszy parametr to przezroczystość, trzy kolejne to składowe koloru.

Składowe koloru pobieramy z suwaków za pomocą `getProgress()`.

Wartości pobrane z suwaków używamy także jako opis (na polach tekstowych)

Finalnie kod pliku „MainActivity.java”
wyglądać powinien następująco:

```
11 public class MainActivity extends AppCompatActivity {
12     View panel;
13     SeekBar suwakR, suwakG, suwakB;
14     TextView opisR, opisG, opisB;
15
16     @Override
17     protected void onCreate(Bundle savedInstanceState) {
18         super.onCreate(savedInstanceState);
19         setContentView(R.layout.activity_main);
20
21         panel = (View) findViewById(R.id.panel);
22         suwakR = (SeekBar) findViewById(R.id.suwakR);
23         suwakG = (SeekBar) findViewById(R.id.suwakG);
24         suwakB = (SeekBar) findViewById(R.id.suwakB);
25         opisR = (TextView) findViewById(R.id.opisR);
26         opisG = (TextView) findViewById(R.id.opisG);
27         opisB = (TextView) findViewById(R.id.opisB);
28         rysuj();
29         SeekBar.OnSeekBarChangeListener sluchaczSuwakow =
30             new SeekBar.OnSeekBarChangeListener() {
31                 @Override
32                 public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {
33                     rysuj();
34                 }
35                 @Override
36                 public void onStartTrackingTouch(SeekBar seekBar) {
37                 }
38                 @Override
39                 public void onStopTrackingTouch(SeekBar seekBar) {
40                 }
41             };
42         suwakG.setOnSeekBarChangeListener(sluchaczSuwakow);
43         suwakR.setOnSeekBarChangeListener(sluchaczSuwakow);
44         suwakB.setOnSeekBarChangeListener(sluchaczSuwakow);
45     }
46     private void rysuj()
47     {
48         int kolor = Color.argb(255,
49             suwakR.getProgress(),
50             suwakG.getProgress(),
51             suwakB.getProgress());
52         panel.setBackgroundColor(kolor);
53         opisR.setText("R="+String.valueOf(suwakR.getProgress()));
54         opisG.setText("G="+String.valueOf(suwakG.getProgress()));
55         opisB.setText("B="+String.valueOf(suwakB.getProgress()));
56     }
57 }
```

Literatura

