

# Hello, ANDROID

Programowanie na platformę Google  
dla urządzeń mobilnych. WYDANIE III

*Przygotuj aplikację na najpopularniejszą platformę mobilną!*

Co czyni  
Androida wyjątkową  
platformą?

Jak zaprojektować  
interfejs użytkownika?



Jak skorzystać  
z zaawansowanych usług  
geolokalizacji i dostępu  
do sieci?

ED BURNETTE



## » Idź do

- Spis treści
- Przykładowy rozdział
- Skorowidz

## » Katalog książek

- Katalog online
- Zamów drukowany katalog

## » Twój koszyk

- Dodaj do koszyka

## » Cennik i informacje

- Zamów informacje o nowościach
- Zamów cennik

## » Czytelnia

- Fragmenty książek online

## » Kontakt

Helion SA  
ul. Kościuszki 1c  
44-100 Gliwice  
tel. 32 230 98 63  
e-mail: helion@helion.pl  
© Helion 1991–2011

## Hello, Android. Programowanie na platformę Google dla urządzeń mobilnych

Autor: Ed Burnette

Tłumaczenie: Krzysztof Sawka

ISBN: 978-83-246-3140-7

Tytuł oryginału: [Hello, Android: Introducing Google's Mobile Development Platform](#)

Format: 168×237, stron: 304



### Przygotuj aplikację na najpopularniejszą platformę mobilną!

- Co czyni Androida wyjątkową platformą?
- Jak zaprojektować interfejs użytkownika?
- Jak skorzystać z zaawansowanych usług geolokalizacji i dostępu do sieci?

Android to system operacyjny dla telefonów komórkowych, który swoją dynamiką wzrostu udziału w rynku przyćmił niejednego produkt. Według badań IDC dla Europy Zachodniej w ciągu niecałego roku Android zwiększył swój udział z 4 do 31 procent! Co jest tego przyczyną? Niezwykle precyzyjnie wykonana i doskonale działająca synchronizacja danych ze wszystkimi najważniejszymi usługami firmy Google, łatwy dostęp do internetu dzięki szybkiej przeglądarce oraz tysiące świetnych aplikacji, zarówno darmowych, jak i płatnych. A skoro mowa o aplikacjach, jak stworzyć własny produkt na tę dynamicznie rozwijającą się platformę? Jak wprowadzić ją na rynek i zarobić? Na te oraz wiele innych pytań związanych z Androidem odpowie Ci ta książka! Na samym wstępie poznasz trochę faktów związanych z powstawaniem samej platformy oraz zobaczysz, jakie nowości oferowały kolejne jej wersje. Wbrew pozorom wiedza ta przyda się przy tworzeniu aplikacji! Po krótkim wstępie historycznym dowiesz się, jak szybko przygotować stanowisko pracy, oraz poznasz kluczowe pojęcia z nim związane. W trakcie lektury nauczysz się projektować interfejs użytkownika, obsługiwać dane wejściowe czy multimedia. Niezwykle ważne są wiadomości zawarte w części trzeciej. Wykorzystanie tych informacji pozwoli Ci na stworzenie nowatorskiej aplikacji. Dostęp do sieci, usług geolokalizacyjnych, baz SQL i trójwymiarowej grafiki pozwoli Ci rozwinąć skrzydła! Książka ta jest długo wyczekiwaną pozycją, omawiającą wszystkie istotne zagadnienia związane z platformą Android, dlatego musisz ją mieć!

- Nowości w poszczególnych wersjach Androida
- Instalacja narzędzi oraz przygotowanie środowiska pracy
- Kluczowe pojęcia związane z wytwarzaniem aplikacji dla Androida
- Korzystanie z zasobów
- Projektowanie interfejsu użytkownika, tworzenie elementów menu
- Wykorzystanie motywów, tworzenie aktywnych tapet
- Grafika dwuwymiarowa oraz trójwymiarowa
- Obsługa danych wejściowych
- Wykorzystanie multimedii
- Przechowywanie danych na urządzeniu mobilnym
- Usługi geolokalizacji
- Wykorzystanie bazy danych SQLite
- Wielodotykowość
- Publikowanie aplikacji w serwisie Android Market

**Wykorzystaj potencjał i popularność platformy Android!**

---

# Spis treści

---

<b>Podziękowania</b> .....	<b>9</b>
<b>Przedmowa</b> .....	<b>11</b>
Co czyni Androida wyjątkowym? .....	11
Dla kogo ta książka jest przeznaczona? .....	13
Zawartość książki .....	13
Co nowego w Androidzie? .....	14
Zasoby w internecie .....	16
Przewijamy >> .....	16
<b>Część I Wprowadzenie do Androida</b> .....	<b>19</b>
<b>Rozdział 1. Szybki start</b> .....	<b>21</b>
1.1. Instalacja narzędzi .....	21
1.2. Tworzymy pierwszy program .....	27
1.3. Uruchomienie aplikacji w emulatorze .....	28
1.4. Uruchomienie aplikacji na prawdziwym telefonie .....	31
1.5. Przewijamy >> .....	32
<b>Rozdział 2. Kluczowe pojęcia</b> .....	<b>33</b>
2.1. Ogólna perspektywa .....	33
2.2. To żyje! .....	38
2.3. Składniki budulcowe .....	42
2.4. Korzystanie z zasobów .....	44
2.5. Stabilnie i bezpiecznie .....	45
2.6. Przewijamy >> .....	46

<b>Część II Podstawy Androida .....</b>	<b>47</b>
<b>Rozdział 3. Projektowanie interfejsu użytkownika .....</b>	<b>49</b>
3.1. Wprowadzenie do Sudoku .....	49
3.2. Projektowanie za pomocą deklaracji .....	51
3.3. Tworzenie ekranu powitalnego .....	51
3.4. Stosowanie alternatywnych zasobów .....	60
3.5. Implementacja okna informacyjnego .....	62
3.6. Wprowadzenie motywu .....	67
3.7. Dodajemy menu .....	68
3.8. Dodajemy ustawienia .....	70
3.9. Uruchamianie nowej gry .....	72
3.10. Usuwanie błędów .....	73
3.11. Wyjście z gry .....	75
3.12. Przewijamy >> .....	76
<b>Rozdział 4. Przegląd grafiki dwuwymiarowej .....</b>	<b>77</b>
4.1. Nauka podstaw .....	77
4.2. Dodawanie grafiki do gry Sudoku .....	82
4.3. Obsługa danych wejściowych .....	90
4.4. Reszta historii .....	96
4.5. Usprawnień nigdy za wiele .....	104
4.6. Przewijamy >> .....	105
<b>Rozdział 5. Multimedia .....</b>	<b>107</b>
5.1. Odtwarzanie dźwięku .....	107
5.2. Odtwarzanie plików wideo .....	113
5.3. Dodawanie dźwięków do gry Sudoku .....	116
5.4. Przewijamy >> .....	120
<b>Rozdział 6. Przechowywanie danych lokalnych .....</b>	<b>121</b>
6.1. Dodawanie opcji do aplikacji Sudoku .....	121
6.2. Kontynuowanie przerwanej gry .....	123
6.3. Zapamiętywanie bieżącej pozycji .....	125
6.4. Uzyskiwanie dostępu do wewnętrznego systemu plików .....	127
6.5. Uzyskiwanie dostępu do kart SD .....	128
6.6. Przewijamy >> .....	129
<b>Część III Ponad podstawami .....</b>	<b>131</b>
<b>Rozdział 7. Podłączenie do świata .....</b>	<b>133</b>
7.1. Przeglądanie za pomocą intencji .....	134
7.2. Sieć z widokami .....	138

7.3. Od obiektów JavaScript do obiektów Java i odwrotnie .....	142
7.4. Korzystanie z usług sieciowych .....	149
7.5. Przewijamy >> .....	161
<b>Rozdział 8. Umiejscawianie i wyczuwanie .....</b>	<b>163</b>
8.1. Lokacja, lokacja, lokacja .....	164
8.2. Ustawmy czujniki na maksimum .....	170
8.3. Widok z lotu ptaka .....	174
8.4. Przewijamy >> .....	179
<b>Rozdział 9. Zaprzęgnięcie bazy SQL do pracy .....</b>	<b>181</b>
9.1. Wprowadzenie do bazy danych SQLite .....	182
9.2. Język SQL dla początkujących .....	183
9.3. Witaj, bazo danych .....	185
9.4. Wiązanie danych .....	192
9.5. Stosowanie klasy ContentProvider .....	195
9.6. Implementacja klasy ContentProvider .....	198
9.7. Przewijamy >> .....	200
<b>Rozdział 10. Trójwymiarowa grafika w środowisku OpenGL .....</b>	<b>201</b>
10.1. Zrozumienie grafiki 3D .....	201
10.2. Poznajemy środowisko OpenGL .....	202
10.3. Utworzenie programu korzystającego ze środowiska OpenGL .....	204
10.4. Renderowanie scenarii .....	206
10.5. Budowanie modelu .....	210
10.6. Światła, kamera, ... .....	212
10.7. Akcja! .....	214
10.8. Nakładanie tekstury .....	215
10.9. A kuku! .....	218
10.10. Pomiar płynności .....	219
10.11. Przewijamy >> .....	221
<b>Część IV Następne pokolenie .....</b>	<b>223</b>
<b>Rozdział 11. Wielodotykowość .....</b>	<b>225</b>
11.1. Wstęp do zagadnienia wielodotykowości .....	225
11.2. Tworzenie aplikacji Dotyk .....	227
11.3. Zrozumienie zdarzeń związanych z dotykiem .....	230
11.4. Konfigurowanie procesu transformacji obrazu .....	233
11.5. Implementacja gestu przewijania .....	234
11.6. Implementacja gestu rozsuwania .....	235
11.7. Przewijamy >> .....	237

<b>Rozdział 12. Trzy... dwa... jeden... start .....</b>	<b>239</b>
12.1. Witaj, widzenie .....	239
12.2. Aktywna tapeta .....	248
12.3. Przewijamy >> .....	259
<b>Rozdział 13. Napisz raz, testuj wszędzie .....</b>	<b>261</b>
13.1. Panie i Panowie, włączamy emulator .....	262
13.2. Tworzenie aplikacji dostosowanej do wielu wersji .....	263
13.3. Ewolucja wraz z interfejsami API .....	265
13.4. Parada błędów .....	271
13.5. Wszystkie ekrany duże i małe .....	273
13.6. Instalowanie aplikacji na karcie SD .....	274
13.7. Przewijamy >> .....	276
<b>Rozdział 14. Publikowanie aplikacji w serwisie Android Market .....</b>	<b>277</b>
14.1. Przygotowanie .....	277
14.2. Podpisywanie .....	279
14.3. Publikowanie .....	280
14.4. Aktualizowanie .....	281
14.5. Końcowe przemyślenia .....	282
<b>Dodatki .....</b>	<b>285</b>
<b>Dodatek A Języki i interfejsy API środowiska Java     kontra systemu Android .....</b>	<b>287</b>
A.1. Podzestaw językowy .....	287
A.2. Podzbiór biblioteki standardowej .....	289
A.3. Biblioteki pochodzące od niezależnych wydawców .....	291
<b>Dodatek B Bibliografia .....</b>	<b>293</b>
<b>Skorowidz .....</b>	<b>295</b>

---

## Rozdział 5.

# Multimedia

---

**S**wego czasu firma Apple wyprodukowała serię telewizyjnych reklam, na których widzimy sylwetki ludzkie tańczące obłądnie w rytm muzyki generowanej przez iPod<sup>1</sup>. Właśnie taki rodzaj podekscytowania chcemy wywołać w użytkownikach naszych aplikacji<sup>2</sup>. Dodanie muzyki, efektów dźwiękowych oraz filmów wideo może sprawić, że nasza aplikacja stanie się intensywniejsza oraz bardziej pociągająca niż w przypadku „suchej” aplikacji, zawierającej sam tekst i grafikę.

W tym rozdziale nauczymy się dodawać pliki multimedialne do naszej aplikacji. Nie chcemy, aby użytkownicy wykonywali jakieś tańce-połamańce na ulicy, jeżeli jednak odpowiednio przygotujemy aplikację, wywołamy przynajmniej uśmiech na ich twarzach.

### 5.1. Odtwarzanie dźwięku

*Dookoła panuje mroczna i niespokojna noc... Rozlega się gwizdek sędziego i wszyscy ruszają... Kibice szaleją, gdy drużyna Anwil Włocławek w ostatniej sekundzie zdobywa przewagę rzutem za trzy punkty...*

Dzięki dźwiękom otrzymujemy wskazówki dotyczące danego otoczenia oraz dostosowujemy emocje do zdarzeń. Uznajmy dźwięki za kolejny sposób na wtargnięcie do głowy użytkownika. Podobnie jak w przypadku grafiki służącej do przekazywania pewnych informacji użytkownikowi, dźwięki pozwalają wzmocnić i zwiększyć wyrazistość danych.

---

<sup>1</sup> Polscy czytelnicy mogą się zapoznać z jedną reklamą tej z serii na stronie <http://www.youtube.com/watch?v=NIHUz99l-eo> — przyp. tłum.

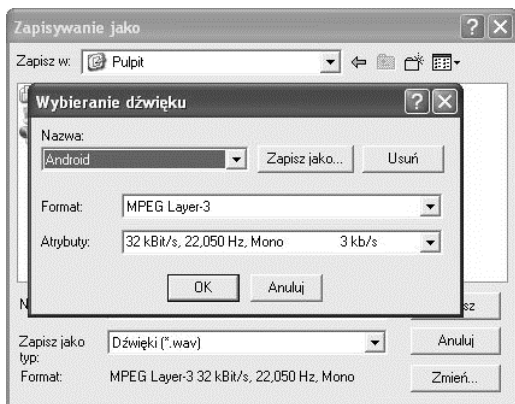
<sup>2</sup> Pomijając fakt, że osoby, które ukończyły 8. rok życia, nie mogą wyginać ciała w ukazany na reklamie sposób... poza momentami, gdy moje dzieci złośliwie umieszczają jaszczurkę w moich... cóż, taka mała dygresja.

Android obsługuje proces przetwarzania dźwięków i muzyki poprzez klasę MediaPlayer, znajdującą się w pakiecie android.media<sup>3</sup>. Wypróbujmy teraz prosty przykład, w którym dźwięk zostanie odtworzony po naciśnięciu przycisku na klawiaturze lub podkładce kierunkowej.

Rozpoczynamy od utworzenia projektu za pomocą następujących parametrów, wprowadzanych w oknie dialogowym *New Android Project*:

```
Project name: Audio
Build Target: Android 2.2
Application name: Audio
Package name: org.przyklad.audio
Create Activity: Audio
Min SDK Version: 8
```

Będzie nam teraz potrzebnych kilka dźwięków do odtwarzania. Dla naszego przykładu stworzyłem własne dźwięki za pomocą aplikacji Rejestrator dźwięku (*Start/Wszystkie Programy/Akcesoria/Rozrywka/Rejestrator dźwięku*) oraz niedrogich słuchawek z mikrofonem. Po ustawieniu odpowiedniego poziomu głośności nagrałem każdy plik, wybrałem opcje Plik/Zapisz jako..., następnie kliknąłem przycisk Zmień... i wybrałem format, który będzie akceptowany przez Androida (rysunek 5.1). Na serwerze ftp wydawnictwa Helion znajdziemy wszystkie wygenerowane przeze mnie dźwięki oraz działający kod źródłowy aplikacji.

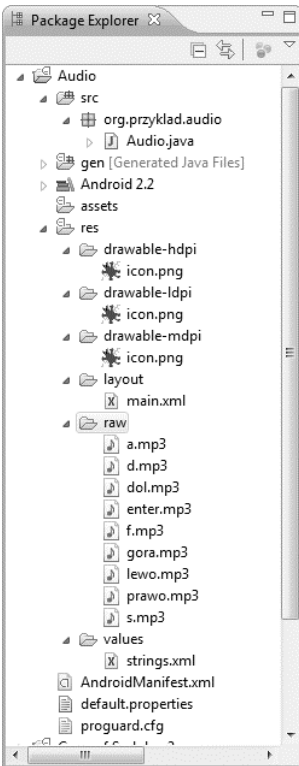


**Rysunek 5.1.** Efekty dźwiękowe zapisujemy w formacie zrozumiałym dla systemu Android

Skopiujemy pliki dźwiękowe do katalogu *res/raw* naszego nowego projektu. Jak pamiętamy z podrozdziału 2.4, „Korzystanie z zasobów” (strona 44), sam proces wklejenia pliku do katalogu *res* powoduje automatyczne stworzenie jego symbolu wewnątrz klasy *R*. Po skopiowaniu plików drzewo katalogów naszego projektu powinno przypominać zrzut prezentowany na rysunku 5.2.

<sup>3</sup> <http://d.android.com/guide/topics/media>





**Rysunek 5.2.** Pliki dźwiękowe kopiujemy do katalogu res/raw naszego projektu

Nadszedł teraz czas na wypełnienie aktywności Audio danymi. Najpierw zadeklarujemy pole nazwane mp, w którym będzie przechowywana instancja klasy MediaPlayer. W przypadku naszego programu tylko jedna instancja klasy MediaPlayer będzie aktywna w danym momencie.

**Audio/src/org/przyklad/audio/Audio.java**

```
package org.przyklad.audio;

import android.app.Activity;
import android.media.AudioManager;
import android.media.MediaPlayer;
import android.os.Bundle;
import android.view.KeyEvent;

public class Audio extends Activity {
    private MediaPlayer mp;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

```

        setVolumeControlStream(AudioManager.STREAM_MUSIC);
    }
}

```

Metoda `setVolumeControlStream()` informuje system, że po wciśnięciu przycisku zwiększenia lub zmniejszania głośności przez użytkownika w trakcie działania aplikacji, powinien zostać zaktualizowany poziom głośności muzyki i innych elementów multimedialnych tego programu, a nie sygnału dzwonka.

Musimy teraz zacząć przechwytywać zdarzenia dotyczące naciskania przycisków oraz odtwarzać stosowne dźwięki. Dokonujemy tego poprzez przesłonięcie metody `Activity.onKeyDown()`.

↳ `onKeyDown()`.

#### Audio/src/org/przyklad/audio/Audio.java

```

Wiersz 1    @Override
-          public boolean onKeyDown(int kodKlaw, KeyEvent zdarzenie) {
-              int idZas;
-              switch (kodKlaw) {
15          case KeyEvent.KEYCODE_DPAD_UP:
-              idZas = R.raw.gora;
-              break;
-          case KeyEvent.KEYCODE_DPAD_DOWN:
-              idZas = R.raw.do1;
10         break;
-          case KeyEvent.KEYCODE_DPAD_LEFT:
-              idZas = R.raw.lewo;
-              break;
-          case KeyEvent.KEYCODE_DPAD_RIGHT:
15         idZas = R.raw.prawo;
-              break;
-          case KeyEvent.KEYCODE_DPAD_CENTER:
-          case KeyEvent.KEYCODE_ENTER:
-              idZas = R.raw.enter;
20         break;
-          case KeyEvent.KEYCODE_A:
-              idZas = R.raw.a;
-              break;
-          case KeyEvent.KEYCODE_S:
25         idZas = R.raw.s;
-              break;
-          case KeyEvent.KEYCODE_D:
-              idZas = R.raw.d;
-              break;
30         case KeyEvent.KEYCODE_F:
-              idZas = R.raw.f;
-              break;
-          default:
-              return super.onKeyDown(kodKlaw, zdarzenie);
35         }
-
-          // Zwalnia zasoby wcześniejszej instancji MediaPlayer
-          if (mp != null) {
-              mp.release();

```

```
40         }  
-  
-         // Tworzy nową instancję MediaPlayer do odtwarzania dźwięku  
-         mp = MediaPlayer.create(this, idZas);  
-         mp.start();  
45  
-         // Wskazuje na to, że dany klawisz został obsłużony  
-         return true;  
-     }
```

Na pierwszym etapie omawiana metoda wybiera zasób w oparciu o wciśnięty klawisz. Następnie w wierszu 39. wywołujemy metodę `release()`; zatrzymuje ona aktualnie odtwarzane dźwięki oraz zwalnia wszelkie zasoby powiązane z instancją klasy `MediaPlayer`. Jeżeli zapomnimy to zrobić, program zostanie samoistnie zamknięty (więcej informacji znajdziemy w ramce).

W wierszu 43. wykorzystujemy metodę `create()` do utworzenia nowej instancji klasy `MediaPlayer`, przy okazji używamy wybranego zasobu dźwiękowego, a następnie wywołujemy metodę `start()` do rozpoczęcia odtwarzania. Metoda `start()` działa w sposób asynchroniczny, zatem program wraca do wykonywania dalszej części kodu, bez względu na długość trwania dźwięku. Jeżeli chcemy zostać powiadomieni o zakończeniu odtwarzania, możemy użyć metody `setOnCompletionListener()`.

### Gdy coś pójdzie nie tak

Osoby zajmujące się programowaniem multimedii odkryją szybko, że klasa `MediaPlayer` potrafi być narowistą bestią. W nowszych wersjach klasa ta została znacznie usprawniona, ciągle jednak potrafi zawiesić aplikację z najdrobniejszego powodu. Jedną z przyczyn takiego zachowania klasy `MediaPlayer` jest jej natywność, gdy na jej powierzchni znajduje się cienka warstwa środowiska Java. Natywny kod odtwarzacza został zoptymalizowany pod kątem wydajności, nie kontroli błędów.

Na szczęście silne zabezpieczenia procesów linuksowych w Androidzie znakomicie zapobiegają wszelkim uszkodzeniom w przypadku zawieszenia aplikacji. Emulator (lub urządzenie fizyczne) oraz inne aplikacje będą działały normalnie. Użytkownik będzie jedynie świadkiem zamknięcia pechowej aplikacji oraz prawdopodobnie ujrzy okno dialogowe zawierające komunikat o błędzie.

Jednakże na etapie tworzenia aplikacji posiadamy dostęp do o wiele bogatszych informacji diagnostycznych, pomagających określić, co się stało. W dzienniku systemowym Androida zostaną umieszczone komunikaty i tropy, które możemy przeglądać za pomocą widoku `LogCat` lub polecenia `adb logcat` (więcej informacji na ten temat zostało zawartych w podrozdziale 3.10, „Usuwanie błędów”, na stronie 73).



**Jasiu pyta...**

### **Jakie formaty dźwiękowe są obsługiwane przez system Android?**

Sytuacja prezentuje się następująco: pewne formaty są obsługiwane wyłącznie na papierze, inne na emulatorze, a jeszcze inne na urządzeniach fizycznych. W wersji papierowej Android obsługuje następujące formaty plików (ma się to zmienić wraz z nowymi wersjami platformy):

- ◆ **WAV** (nieskompresowana modulacja PCM),
- ◆ **AAC** (format stosowany w iPodach firmy Apple, niechroniony),
- ◆ **MP3** (MPEG-3),
- ◆ **WMA** (ang. *Windows Media Audio*),
- ◆ **AMR** (kodek odpowiedzialny za sygnał mowy),
- ◆ **OGG** (format firmy Vorbis)<sup>4</sup>,
- ◆ **MIDI** (instrumenty).

W rzeczywistości zauważyłem, że tylko formaty OGG, WAV i MP3 działają dobrze na emulatorze, a zatem jedynie takie rodzaje plików mogą zalecać do stosowania w procesie tworzenia aplikacji. Wydaje się, że natywnym formatem dźwiękowym w Androidzie jest 16-bitowy dźwięk stereo o częstotliwości próbkowania 44,1 kHz. Jednak pliki WAV spełniające wspomniane wymagania zajmują olbrzymią ilość miejsca, dlatego powinniśmy pozostać przy plikach OGG i MP3 (mono w przypadku mowy, stereo dla muzyki). Pliki OGG najlepiej się sprawdzają w przypadku krótkich dźwięków, na przykład efektów dźwiękowych w grach.

Trzymajmy się z daleka od niestandardowych częstotliwości, na przykład 8 kHz, ponieważ artefakty pojawiające się w czasie próbkowania bardzo deformują dźwięk. Najlepsze rezultaty osiągniemy przy częstotliwościach 11 kHz, 22 kHz i 44,1 kHz. Pamiętajmy, że chociaż telefon może posiadać malutki głośniczek, wielu użytkowników będzie podłączało słuchawki (tak jak w przypadku iPod'a), lepiej więc, żeby były słyszane dźwięki w wysokiej jakości.

Jeżeli uruchomimy teraz program i wciśniemy któryś ze zdefiniowanych klawiszy (na przykład *Enter* lub środkowy przycisk podkładki kierunkowej), powinniśmy usłyszeć dźwięk. Jeżeli z głośników nie wydobywa się żaden dźwięk, sprawdźmy, czy mamy włączone głośniki (nie śmiać się), lub zajrzyjmy do komunikatów widoku *LogCat*. Jeżeli uruchomiliśmy aplikację na urządzeniu nieposiadającym klawiatury fizycznej, podkładki kierunkowej lub manipulatora kulkowego, wciśniemy i przytrzymajmy przycisk *Menu*, dopóki nie pojawi się klawiatura dotykowa.

<sup>4</sup> <http://www.vorbis.com>



**Jasiu pyta...**

### **Jakie formaty wideo możemy oglądać na Androidzie?**

Poniżej prezentuję oficjalnie obsługiwane formaty:

- ◆ MP4 (format MPEG-4 o niskiej szybkości transmisji),
- ◆ H.263 (3GP),
- ◆ H.264 (AVC).

W przypadku wersji 1.5 systemu Android zalecanym formatem plików wideo jest H.263, ponieważ obsługuje go każda platforma sprzętowa oraz jest względnie wydajny pod kątem kodowania/dekodowania. Wykazuje on również kompatybilność z innymi urządzeniami, na przykład z iPhone'iem. Do konwersji z jednego formatu wideo na inny możemy stosować program QuickTime Pro<sup>5</sup>. W celu oszczędzenia miejsca używajmy w miarę niskiej rozdzielczości i prędkości transmisji, nie przesadzajmy jednak, gdyż możemy zupełnie utracić akceptowalną jakość obrazu.

Zauważmy, że w niektórych przypadkach efekt końcowy może brzmieć, jakby był przerywany lub opóźniany. W takim wypadku sprawdzmy inne formaty (na przykład OOG zamiast MP3) oraz mniejszą prędkość transmisji. Możemy również zbadać zastosowanie klasy SoundPool, która jawnie obsługuje kilka jednoczesnych strumieni dźwiękowych. W wersji Android 1.0 była ona pełna błędów oraz słabo udokumentowana, jednak od wersji 1.5 wydaje się być całkiem stabilna.

Naszą kolejną sztuczką będzie odtworzenie filmiku za pomocą jednej linijki kodu.

## **5.2. Odtwarzanie plików wideo**

Obraz wideo to coś więcej niż zbiór zdjęć wyświetlanych kolejno po sobie. Składa się na niego również dźwięk, który musi być ściśle zsynchronizowany z poszczególnymi klatkami.

Klasa MediaPlayer obsługuje pliki wideo w taki sam sposób, z jakim mieliśmy do czynienia w przypadku samego dźwięku. Jedyna różnica polega tu na konieczności utworzenia klasy Surface, na której odtwarzacz będzie rysował klatki. Do kontroli odtwarzania używamy metod start() i stop().

---

<sup>5</sup> <http://www.apple.com/quicktime/pro>

Nie zamierzam jednak pokazywać jeszcze jednego przykładu zawierającego klasę `MediaPlayer`, ponieważ istnieje łatwiejsza metoda wstawiania plików wideo do aplikacji: klasa `VideoView`. Aby zademonstrować jej działanie, stwórzmy za pomocą poniższych parametrów nowy projekt, zatytułowany *Wideo*:

```
Project name: Wideo
Build Target: Android 2.2
Application name: Wideo
Package name: org.przyklad.wideo
Create Activity: Wideo
Min SDK Version: 8
```

Zmieńmy układ graficzny (*res/layout/main.xml*) na następujący:

**Wideo1/res/layout/main.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >
    <VideoView
        android:id="@+id/wideo"
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:layout_gravity="center" />
</FrameLayout>
```

Teraz otwórzmy plik *Wideo.java* i zmienmy metodę `onCreate()` w poniższy sposób:

**Wideo1/src/org/przyklad/wideo/Wideo.java**

```
package org.przyklad.wideo;

import android.app.Activity;
import android.os.Bundle;
import android.widget.VideoView;

public class Wideo extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // Wypełnia widok danymi zasobu
        setContentView(R.layout.main);
        VideoView wideo = (VideoView) findViewById(R.id.wideo);

        // Wczytuje i uruchamia film
        wideo.setVideoPath("/data/przykladowewideo.3gp" );
        wideo.start();
    }
}
```

Metoda `setVideoPath()` otwiera dany plik, dopasowuje go do pojemnika przy jednoczesnym zachowaniu proporcji oraz rozpoczyna odtwarzanie.

Musimy teraz umieścić jakiś plik, który chcielibyśmy obejrzeć. W tym celu wpisujemy następujące polecenie:

```
C:\> adb push c:\kod\przykladowewideo.3gp /data/przykladowewideo.3gp
1649 KB/s (369870 bytes in 0.219s)
```

Plik *przykladowewideo.3gp* znajdziemy w pakiecie zawierającym kody źródłowe używane w tej książce, możemy również stworzyć swój własny plik. Zaprezentowany tu katalog (*/data*) został wprowadzony wyłącznie w celach pokazowych i w rzeczywistości nie powinien być używany do przechowywania plików multimedialnych.

Sposób ten działa wyłącznie w emulatorze, ponieważ w przypadku urządzeń fizycznych używany katalog jest chroniony.

Zauważmy, że Android raczej nie przejmuje się rozszerzeniem wstawianego pliku. Możemy również umieszczać i pobierać pliki za pomocą widoku *File Explorer* w perspektywie *Android*, uważam jednak, że do takich zadań lepiej nadaje się wiersz poleceń.

Jeszcze jedna sprawa: chcielibyśmy, aby odtwarzany film zajął cały ekran, włącznie z paskiem tytułowym oraz paskiem stanu. Aby to umożliwić, jedyną konieczną czynnością jest zdefiniowanie odpowiedniego motywu w pliku *AndroidManifest.xml*:

Videov1/AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="org.przyklad.wideo"
    android:versionCode="1"
    android:versionName="1.0" >
    <application android:icon="@drawable/icon"
        android:label="@string/app_name" >
        <activity android:name=".Wideo"
            android:label="@string/app_name"
            android:theme="@android:style/Theme.NoTitleBar.Fullscreen" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
    <uses-sdk android:minSdkVersion="3" android:targetSdkVersion="8" />
</manifest>
```

Gdy już przygotowujemy wszystkie pliki, po uruchomieniu programu powinniśmy ujrzeć i usłyszeć filmik (rysunek 5.3). Sprawdźmy, czy plik wideo będzie odtwarzany zarówno w orientacji pionowej, jak i poziomej. Voilà! Natychmiastowa, kinematograficzna przyjemność.

Usprawnijmy teraz nieco naszą grę Sudoku, dodając klimatyczną muzykę.



**Rysunek 5.3.** Dzięki klasie `VideoView` wstawianie plików wideo nie stanowi problemu

## 5.3. Dodawanie dźwięków do gry Sudoku

W tym podrozdziale wykorzystamy całą zdobytą dotychczas wiedzę na temat multimedii i dodamy do naszej gry Sudoku muzykę odtwarzaną w tle. Jeden utwór będzie grał na ekranie tytułowym, a inny usłyszymy w trakcie właściwej rozgrywki. W ten sposób nie tylko się dowiemy, w jaki sposób odtwarzać muzykę, lecz również rozpatrzemy pewne zagadnienia związane z cyklem życia.

Aby dodać muzykę do ekranu powitalnego, wystarczy, że przesłonimy następujące metody w klasie `Sudoku`:

`Sudokuw3/src/org/przyklad/sudoku/Sudoku.java`

```
@Override
protected void onResume() {
    super.onResume();
    Muzyka.play(this, R.raw.glowna);
}

@Override
protected void onPause() {
    super.onPause();
    Muzyka.stop(this);
}
```





Jasiu pyta...

### **Dlaczego filmik zostaje zrestartowany po zmianie orientacji wyświetlacza?**

Android domyślnie zakłada, że napisana aplikacja nie rozróżnia trybów orientacji ekranu. Aby wyświetlić ewentualne zmiany w zasobach, system niszczy i od podstaw odtwarza aktywność. Oznacza to, że metoda `onCreate()` zostaje ponownie wywołana, a to z kolei znaczy, iż plik wideo zostaje uruchomiony od początku (tak jakby cały przykład był pisany na bieżąco).

Takie zachowanie nie jest kłopotliwe w przypadku 90% programów, więc większość programistów nie musi się nim przejmować. W gruncie rzeczy jest to całkiem przydatny sposób na przetestowanie cyklu życia aplikacji oraz kodu odpowiedzialnego za zapis/odczyt jej stanu (podrozdział 2.2, „To żyje!”, na stronie 38). Jednakże istnieje kilka rozwiązań dla ambitnych, pozwalających na zoptimalizowanie przejścia pomiędzy trybem pionowym a poziomym.

Najprostszym rozwiązaniem jest implementacja metody `onRetainNonConfigurationInstance()` wewnątrz aktywności, dzięki czemu pewne dane zostaną zachowane podczas wywołań metod `onDestroy()` i `onCreate()`. W nowym wystąpieniu aktywności wykorzystujemy wtedy metodę `getLastNonConfigurationInstance()` do odzyskania tych danych. Możemy przechowywać wszelkie rodzaje informacji, nawet referencje do bieżącej intencji oraz uruchomionych wątków.

Bardziej zaawansowaną metodą jest zastosowanie właściwości `android:configChanges=` wewnątrz pliku `AndroidManifest.xml` w taki sposób, aby Android wiedział, którymi zmianami możemy się zająć. Na przykład, jeśli ustawimy jej wartość `keyboardHidden|orientation`, w momencie obrócenia klawiatury aktywność nie zostanie zniszczona i odtworzona. Zamiast tego system wywoła `onConfigurationChanged(Configuration)` z założeniem, że mamy sytuację pod kontrolą<sup>6</sup>.

Jeśli przypomnimy sobie, co przeczytaliśmy w podrozdziale 2.2, „To żyje!” (strona 38), metoda `onResume()` zostaje wywołana w momencie, gdy aktywność jest gotowa do interakcji z użytkownikiem. Jest to odpowiedni moment, aby uruchomić muzykę, zatem wstawiamy tutaj wywołanie `Muzyka.play()`. Wkrótce zdefiniujemy klasę `Muzyka`.

Odnosnik `R.raw.glowna` dotyczy pliku `res/raw/glowna.mp3`. Pliki dźwiękowe znajdziemy w projekcie `SudokuW3`, który można pobrać wraz z innymi przykładowymi projektami z serwera ftp wydawnictwa Helion.

<sup>6</sup> Szczegółowe informacje znajdziemy na stronie <http://d.android.com/reference/android/app/Activity.html#Configuration-Changes>.



Jasiu pyta...

### Czy nie powinniśmy wprowadzić muzyki do usługi przetwarzanej w tle?

Nie poświęciliśmy zbyt wiele miejsca klasie `Service`, mogliśmy się jednak na nią nakłnąć w różnych przykładach wykorzystujących muzykę, które można znaleźć w internecie. Zasadniczo klasa `Service` służy do rozpoczęcia procesu przetwarzanego w tle, który będzie działał nawet po zamknięciu aktywności. Usługi są podobne do demonów systemu Linux (ale nie identyczne). Jeżeli tworzymy odtwarzacz muzyki ogólnego przeznaczenia i chcemy, aby muzyka leciała podczas przeglądania poczty lub stron sieci Web, to zastosowanie klasy `Service` będzie jak najbardziej na miejscu. Jednakże w większości przypadków wolimy, aby muzyka zakończyła się w momencie zamknięcia programu, nie ma więc potrzeby, aby implementować klasę `Service`.

Metoda `onPause()` została wprowadzona w formie „wsparcia” dla metody `onResume()`. Android wstrzymuje bieżącą aktywność tuż przed rozpoczęciem kolejnej, więc w przypadku naszej gry po wciśnięciu przycisku *Nowa Gra* aktywność *Sudoku* zostanie wstrzymana, natomiast będziemy świadkami uruchomienia aktywności *Gra*. Również w przypadku naciśnięcia klawisza cofania lub powrotu do ekranu startowego zostanie wywołana metoda `onPause()`. We wszystkich wymienionych tu miejscach chcemy, aby nie grała muzyka tytułowa, zatem wywołujemy `Muzyka.stop()` w metodzie `onPause()`.

Przeprowadźmy podobną operację w aktywności *Gra*:

`Sudokuw3/src/org/przyklad/sudoku/Gra.java`

```
@Override
protected void onResume() {
    super.onResume();
    Muzyka.play(this, R.raw.gra);
}

@Override
protected void onPause() {
    super.onPause();
    Muzyka.stop(this);
}
```

### Ciekawostka o grze Sudoku

Istnieją dziesiątki odmian gry w Sudoku, jednak żadna nie zdołała zdobyć popularności porównywalnej z wersją oryginalną. Jedną z takich modyfikacji umożliwia grę na polu o rozmiarach 16×16 pól, danymi wejściowymi są natomiast cyfry zapisywane w systemie szesnastkowym. Inna wersja, znana jako *Gattai 5* lub *Samurai Sudoku*, zawiera pięć plansz o rozmiarach 9×9 pól, które łączą się ze sobą w narożnikach.

W porównaniu do klasy `Sudoku` wprowadziliśmy tu inny zasób dźwiękowy, `R.raw.gra` (`res/raw/gra.mp3`). Ostatnim fragmentem naszej dźwiękowej układanki jest klasa `Muzyka`, w której będzie zarządzana klasa `MediaPlayer`, wymagana do odtwarzania muzyki:

`Sudoku3/src/org/example/sudoku/Music.java`

```
Wiersz 1  package org.przyklad.sudoku;
-
-         import android.content.Context;
-         import android.media.MediaPlayer;
5
-         public class Muzyka {
-             private static MediaPlayer mp = null;
-
-             /** Zatrzymuje stary utwór i rozpoczyna odtwarzanie nowego */
10         public static void play(Context kontekst, int zasob) {
-             stop(kontekst);
-             mp = MediaPlayer.create(kontekst, zasob);
-             mp.setLooping(true);
-             mp.start();
15        }
-
-             /** Zatrzymuje odtwarzanie muzyki */
-             public static void stop(Context context) {
-                 if (mp != null) {
20                     mp.stop();
-                     mp.release();
-                     mp = null;
-                 }
-             }
25    }
```

Metoda `play()` najpierw wywołuje metodę `stop()`, dzięki czemu wszelkie odtwarzanie bieżącej muzyki zostaje zatrzymane. Następnie za pomocą metody `MediaPlayer.create()` zostaje utworzona nowa instancja klasy `MediaPlayer`, której zostaje przekazany kontekst oraz identyfikator zasobów.

Po utworzeniu odtwarzacza wprowadzamy opcję zapętlenia muzyki i w końcu ją odtwarzamy. Metoda `start()` od razu pozwala aplikacji na przetwarzanie dalszej części kodu.

Rozpoczynająca się od wiersza 18. metoda `stop()` jest bardzo prosta. Po sprawdzeniu, czy rzeczywiście jest dostępna instancja klasy `MediaPlayer`, wywołujemy metody `stop()` i `release()`. O dziwo, metoda `MediaPlayer.stop()` zatrzymuje odtwarzanie muzyki. Z kolei metoda `release()` zwalnia zasoby systemowe powiązane z odtwarzaczem. Ponieważ są to natywne zasoby, nie możemy czekać, aż zwykły proces odzyskiwania pamięci środowiska Java wykona za nas to zadanie. Pozostawienie w spokoju metody `release()` stanowi świetny sposób na niespodziewane zawieszenie aplikacji (nie to, żeby mi się kiedykolwiek coś takiego przydarzyło; czytelnicy powinni jednak o tym pamiętać).

Teraz nadeszła pora na najlepszą część — zagrajmy w Sudoku po wprowadzeniu tych wszystkich zmian. Przetestujmy aplikację w każdy możliwy sposób, na przykład przechodząc pomiędzy różnymi aktywnościami, wciskając przyciski cofania i ekranu startowego na różnych etapach gry, obracając wyświetlacz, i tak dalej. Właściwe zarządzanie cyklem życia stanowi czasami nie lada kłopot, jednak użytkownicy aplikacji docenią włożony w nią wysiłek.

## 5.4. Przewijamy >>

W niniejszym rozdziale zajęliśmy się tematyką odtwarzania plików audio i wideo za pomocą pakietu Android SDK. Nie omawialiśmy procesu rejestracji multimediów, ponieważ większość programów nie wykorzystuje tej funkcji, jeżeli jednak chcemy stworzyć program będący pod tym względem wyjątkowym, możemy zajrzeć do dokumentacji klasy `MediaRecorder`<sup>7</sup>.

W rozdziale 6., „Przechowywanie danych lokalnych” (strona 121), poznamy kilka prostych metod zachowywania danych aplikacji pomiędzy jej wywołaniami. Jeżeli ta wiedza jest nam niepotrzebna, możemy od razu przejść do rozdziału 7., „Podłączenie do świata” (strona 133), aby dowiedzieć się co nieco o dostępie do sieci.

---

<sup>7</sup> <http://d.android.com/reference/android/media/MediaRecorder.html>

---

# Skorowidz

---

3GP, 113

## A

AAC, 12, 35, 112

adapter

SimpleCursorAdapter, 193

Adobe

AIR, 182

adres serwera proxy, 26

ADT, 26, 54

Android Development Toolkit, 24

Ajax, 174

akcelerometr, 172

aktualizacja

żądanie, 156

aktywność, 12, 13, 42, 52, 65, 79, 83, 84,  
122, 250

Audio, 109

Browser, 138

główna, 63, 204

kończenie, 75

referencja, 168

rejestracja, 43

Translator, 157

uruchamianie, 63

zamknięcie, 75

zdefiniowanie, 63

alfa

miara przezroczystości, 78

algorytm

dzielenia czasu, 288

time slicing, 288

Amazon, 133

AMD, 203

AMR

kodek odpowiedzialny za sygnał mowy,  
112

Android, 11, 12, 13, 15

BASE, 262

BASE\_1\_1, 262

Content Guidelines, 281

Cupcake, 14, 239, 262, 269, 271,  
272, 273

Device Dashboard, 15

Donut, 14, 262

Eclair, 14, 262, 270

Maintenance Release 1, 15, 248

MR1, 15

ECLAIR\_0\_1, 262

ECLAIR\_MR1, 262

Eclipse, 49

FroYo, 15, 29, 262

kod, 143

## Android

Market, 14, 17, 32, 38, 264, 276, 277, 278, 279, 280, 281

- Licensing, 280
- publikowanie, 277, 280
- serwis, 281
- wersja
  - darmowa, 283
  - płatna, 282

narzędzia programistyczne, 22

pakiet startowy, 23

platforma, 14

przestrzeń nazw XML, 55

SDK, 21, 24, 29, 42, 171, 279

- Components, 23
- Starter Package, 23

składniki, 23

software development kit, 21

wersja, 14, 261

- 1.0, 262
- 1.1, 15, 262
- 1.5, 14, 36, 113, 135, 227, 245, 262, 270, 271
- 1.6, 14, 225, 227, 262, 268
- 2.0, 14, 226, 262, 265, 266
- 2.0.1, 262
- 2.1, 15, 225, 248, 262, 274
- 2.2, 15, 29, 165, 175, 262, 264, 274, 276

animacja, 217

- wektorowa, 12

antialiasing, 12

API, 37

Apple, 22, 35, 107

- iPhone, 182
- MacBook, 173
- Safari, 35

architektura

- ARM, 12
- build target, 175
- docelowa, 175

komponentowa, 12

sprzętowa, 45

x86, 12

ARGB, 78

ASCII, 54

AT&T, 21

atrybut

android:installLocation=, 275

android:permission=, 249

android:scaleType=, 229

android:theme=, 229

android:versionCode=, 278

android:versionName=, 278, 282

AVC, 35, 113

AVD

Android Virtual Device, 28

architektura systemowa, 34

**B**

baza danych, 181

db4o, 200

instrukcja

- modyfikacji, 184

- zapytań, 184

Java SQL, 290

SQL, 12, 33, 35, 182

SQLite, 13, 181, 182, 183, 184, 185,

187, 189, 190, 200, 290

- licencja, 183

biblioteka

- graficzna, 79, 89, 105

Java

- 5 SE, 287

- 5 Standard Edition, 287

natywna, 34, 37

OpenGL ES, 77

podzbiór, 289

WebKit, 35

zawarta w Androidzie, 37

BlackBerry, 11

## bloczek

- budulcowy, 42

Bluetooth, 173

błąd, 272

- alert obiektu JavaScript, 145, 149

- aplikacja nie posiada zdefiniowanych

  - wymagań dotyczących poziomu
  - interfejsu API, 264

- Application does not specify an API

  - level requirement, 264

- brak urządzenia AVD, 28

- ClassNotFoundException, 176

- connection error, 25

- dwa palce, 235

- HTTPS SSL, 24

- kompilacji, 65

- komunikat, 57, 111

  - dziennika, 73

- strona internetowa jest niedostępna, 141

- usuwanie, 73

- wielodotykowość, 227

- zawieszenie, 137

Bornstein Dan, 36

BTS

- stacje bazowe sieci komórkowych, 164

**C**

C, 34

C++, 34

Carmack John, 203

certyfikat, 45, 129

cieniowanie, 209

- shader, 209

Compiz, 35

CSS

- Cascade Style Sheets, 67

- kaskadowe arkusze stylów, 67

cykl życia, 13, 37, 40, 43

- aktywności, 40

- procesu, 40

## czujnik

- Apple MacBook, 173

- falszywy, 173

- tablica, 172, 179

- testowanie, 172

- TYPE\_ACCELEROMETER, 171

- TYPE\_LIGHT, 171

- TYPE\_MAGNETIC\_FIELD, 171

- TYPE\_ORIENTATION, 171, 172

- TYPE\_PRESSURE, 171

- TYPE\_PROXIMITY, 171

- TYPE\_TEMPERATURE, 171, 172

- wartości, 172

- wywołanie, 171

**D**

Dalvik, 12, 287, 288

- virtual machine, 36

- VM, 36, 289

- wirtualna maszyna, 36

dane

- globalne, 43

- model, 192

- wejściowe, 90

- wiązanie, 192

DB2, 183

DDMS

- Dalvik Debug Monitor Service, 170

debuger, 73, 75

debugowanie, 31

delegacja, 266

demon, 43

Developer Console, 280, 282

- dialer telefoniczny, 38

dip, 59

- synonim jednostki dp, 59

DLL

- Data Definition Language, 183

- język definiowania danych, 183

dokowanie, 14

DOM, 290  
dostawca treści, 37, 42, 43  
dp  
  density-independent pixels, 59  
  piksele niezależne od gęstości, 59  
dpi, 59  
  dots per inch, 59  
  punkty na cal, 59  
Draw 9-patch, 242, 243  
DTD  
  schemat, 56  
dziennik  
  systemu, 231  
dźwięk  
  dodawanie, 116  
  format, 112  
  przetwarzanie, 108  
  rejestrator, 108

## E

Eclipse, 22, 51, 64, 65, 74, 75, 138, 170, 179, 182, 187, 240, 246, 252, 264, 265, 267, 268, 279  
  ADT, 24  
  Cupcake, 29  
  Donut, 29  
  Eclair, 29  
  FroYo, 29  
  Helios, 23  
  IDE for Java Developers, 22, 25  
  IDE for Java EE Development, 25  
  kompilacja, 28  
  konfiguracja, 24, 54  
  SDK, 22  
  wersja  
    3.3.1, 22  
    3.5, 24  
    3.6, 23  
  WST, 25  
  wtyczka, 24

edytor  
  XML, 65  
  graficzny, 54  
egzekutor, 150  
ekran  
  aktywności, 52  
  dotknięcia, 174  
  dotykowy, 90  
  gęstość, 59, 274  
  pikseli, 274  
  orientacja, 274  
  powitalny, 51  
  rodzaj, 274  
  rozdzielczość, 274  
  rozmiar, 274  
  startowy, 39, 225, 239, 244, 246, 257, 258, 259  
  tapeta, 257  
  tło, 255  
emulacja  
  czujnika, 173  
emulator, 13, 28, 32, 52, 60, 66, 74, 111, 128, 170, 172, 173, 262, 263, 264, 277, 279  
  Androida, 31  
  em15, 271  
  em16, 264, 265  
  telefonu, 21, 31  
  wielodotykowość, 231

## F

fabrykacja, 266  
Firefox, 35  
Flash, 12  
format  
  binarny, 55  
FPS  
  frames per second, 219  
funkcja  
  Copy Protection, 280  
  window.alert(), 146



**G**

gadżet, 38  
 Garns Howard, 50, 83  
 Gennick Jonathan, 200  
 gest  
   przeciągnięcie, 226  
   przewijania  
     implementacja, 234  
   rodzaje, 226  
   rozsuwanego przybliżania, 226  
   rozsuwania  
     implementacja, 235  
     stuknięcie, 226  
 GestureWorks, 237  
 gęstość, 14  
 Goetz Brian, 162  
 Google, 11, 15, 21, 36, 43, 133, 150, 160, 172, 174, 175, 178, 247, 272, 274, 275, 279, 280  
   APIs, 175  
   Checkout, 280  
   Chrome, 35, 133  
   GWT, 287  
   I/O, 274  
   Maps, 173, 174, 175, 176  
   Tłumacz, 151, 153, 158  
   Web Toolkit, 287  
 GPS, 12, 45, 163, 164, 165, 179  
   dostawca, 170  
   Global Positioning System, 164  
   odbiornik, 165  
   procesor, 164, 172  
   sygnał, 164  
 grafika  
   3D, 201  
   dwuwymiarowa, 12, 13, 35, 77, 201  
   OpenGL, 290  
   trójwymiarowa, 13, 35, 77, 201, 202, 204, 221  
     biblioteka, 213

**H**

H.263, 35, 113  
 H.264, 35, 113  
   AVC, 12  
 Hipp Richard, 182  
 hotspot wifi, 164  
 HTC, 21  
 HTML, 35, 143, 151  
   łącza, 146  
   odwrócony ukośnik, 63  
   tabela, 55  
   widok, 67  
   znacznik, 63

**I**

id Software, 203  
 IDE, 13, 21, 22  
   integrated development environment, 13  
   zintegrowane środowisko  
     programistyczne, 13  
 identyfikator  
   dla przycisku, 59  
   URI, 196, 197, 198  
   użytkownika, 45, 129  
   zasobu, 59  
 in  
   cale, 59  
   rozmiar zmierzony miarką, 59  
 instalacja  
   narzędzi programistycznych, 21  
 instrukcja  
   case, 92  
   Eclipse IDE Pocket Guide, 32  
   importu, 64, 69, 72  
 Intel, 203  
 intencja, 42, 43  
   anonimowa, 65  
   nazwana, 65  
   prywatna, 65  
   publiczna, 65  
 IntencjaPrzeglądarki, 134, 136, 139, 141

## interfejs

- API, 32, 43, 265, 274, 287, 289
  - 1.1, 14
  - 1.5, 14
- AIDL, 195
- Android Interface Definition
  - Language, 195
- bazy danych, 290
- Google Maps, 175, 279
- język definiowania interfejsu
  - w systemie Android, 195
- niestandardowy, 43
- OpenGL E 2.0, 15
- poziom
  - 1, 268
  - 3, 264
  - 5, 268
  - 8, 264
- preferencji, 121, 123
- raportów, 227
- tłumacza, 150
- BaseColumns, 187
- Callable, 150
- graficzny, 80
- komputerowy, 59
- LocationListener, 168
- OnClickListener, 65
- publiczny, 37
- Runnable, 150
- sensoryczny, 171, 173
- sieciowy, 134
  - środowiska Java, 149
- skalowanie, 59
- użytkownika, 13, 17, 35, 39, 40, 41, 42, 44, 52, 72, 136, 137, 144, 147, 149, 154, 155, 156, 159, 225, 289
  - aktualizowanie, 159
  - ekran, 52
  - elementy, 138
  - projektowanie, 49
- ViewBinder, 200

## instalowanie

- narzędzi, 21
  - na karcie SD, 274
- interpolator animacji, 95
- iPad, 273
- iPhone, 11, 112, 113, 203, 273
- iPod, 107, 112

**J**

- Java, 15, 52, 111
  - 2 Platform Standard Edition 5.0, 289
  - 5, 150, 187
  - 5.0+, 22
  - biblioteki, 36
  - Concurrency in Practice, 162
  - Dalvik VM, 36
  - JDK, 22
  - język, 11, 12, 13
  - ME, 37
  - Mobile Edition, 11, 37
  - SE, 14, 37
  - Standard Edition, 14, 37
    - 5.0, 289
  - standardowy zasób, 44
  - wersja
    - 6, 288
    - 7, 288
  - wtyczka, 23
- JavaScript, 147, 161
  - bazy danych Java SQL, 290
- JDK
  - SE 6.0, 22
- JetBrains IDEA, 22
- język, 274
  - C#, 13
  - DDL, 183
  - HTML, 51
  - HTML5, 14
  - Java, 13, 287, 288, 289
  - JavaScript, 143, 174, 266, 287
  - Ruby, 266

- SQL, 183, 184, 200
- XML, 45, 51, 52, 55
  - analiza składni, 290, 291
- JRE
  - Java Runtime Environment, 22
- JSON
  - JavaScript Object Notation, 161
- JSR, 209, 239
  - Java Specification Request, 203
- K**
- Kanał alfa, 219
- karta
  - odczyt danych, 129
  - SD, 29, 128, 129, 274, 275, 276
  - Secure Digital, 29, 128
  - wirtualna, 29, 128
- katalog
  - assets, 145
  - gen., 53
  - instalacyjny pakietu SDK, 23
  - nazywanie, 44
  - prywatny aplikacji, 129
  - res, 44, 53, 108, 145
  - roboczy, 25
  - tools, 23
  - tymczasowy, 23
- Khronos Group, 203
- klasa, 52
  - Activity, 40, 192, 198
  - Adapter, 155
  - AlarmManager, 247
  - AlertDialog, 63
  - android.R, 54
  - AppWidgetProvider, 243
  - Bundle, 126, 127
  - Canvas, 77, 79
  - Color, 77, 78
  - ContentProvider, 195, 196, 198
    - implementacja, 198
  - Context, 43, 127
  - Cursor, 191
  - Dialog, 63
  - Direction.CW, 80
  - Drawable, 80, 81, 82
  - dziedziczenie, 63, 187
  - Engine, 257
  - ExecutorService, 150
  - FontMetrics, 89
  - Future, 150
  - GLSurfaceView, 205, 208, 256
  - Handler, 147
  - HttpURLConnection, 161
  - ImageView, 85, 233, 271, 272
  - IntencjaPrzeglądarki, 136
  - Intent, 65, 125, 139
  - klawiatura, 97, 100
  - ListActivity, 192, 193, 194
  - LocationListener, 166
  - LocationManager, 166
  - Log, 73
  - MapActivity, 177
  - MapView, 175, 176, 177, 178
  - MediaPlayer, 108, 109, 111, 113, 114, 119
  - MediaRecorder, 179
  - MenuInflater, 69
  - MotionEvent, 265, 266, 267
  - MyLocationOverlay, 178
  - nadrzędna, 127
  - Paint, 79
  - Path, 80
  - PathEffect, 80
  - PreferenceActivity, 122
  - R, 44, 53, 54, 108
  - rejestrwanie, 71
  - RendererGL, 206, 214, 255, 258
  - renderowania, 206
  - ResourceManager, 78
  - SensorManager, 171
  - setImageMatrix(), 271
  - siatka, 85
  - SQLiteOpenHelper, 185

- klasa
  - Surface, 113, 257
  - TextView, 67
  - Thread, 150
  - Translator, 153, 158, 193
  - UriMatcher, 199
  - VideoView, 114
  - View, 79, 126, 139
  - WallpaperService, 250, 257
  - WebView, 67, 138, 139, 141
  - wewnętrzna, 65
- klauzula
  - BY, 190
  - GROUP, 190
  - HAVING, 190
  - WHERE, 190
- klawiatura, 84, 96, 97, 98, 99, 100
  - dostępność, 274
  - dotykowa, 14
  - numeryczna, 60
  - obracanie, 117
  - programowa, 135
- klawisz
  - /, 135
  - .com, 135
  - Ctrl+F11, 60, 263
  - Enter, 93, 137, 141
  - Esc, 66
  - Fn+Ctrl+F11, 60
  - kierunkowe, 62
  - Menu, 122
  - NumLock, 263
  - wirtualny, 14
- klucz, 44
  - Maps API, 279
- kod
  - bajtowy, 287
  - dwuliterowy języka, 274
  - HTML, 149
  - Java, 51, 78, 84, 143, 147
  - JavaScript, 143, 147, 148
  - jawny, 11, 21, 33
  - konstruktora, 186
  - MCC, 274
  - metody, 168
  - MNC, 274
  - odtworzacza, 111
  - pisanie, 51
  - pośredni, 36
  - wywołujący, 64
  - XML, 51, 54
  - źródłowy, 11, 16, 33
    - Androida, 272
    - bibliotek, 287
- kodek, 12
  - multimediów, 35
  - odpowiedzialny za sygnał mowy, 112
- kolor
  - alfa, 78
  - ARGB, 78
  - definiowanie, 58
  - RGB, 78
  - składowe, 78
- kompas, 175
- kompatybilność, 14, 29
- kompilacja, 44, 268
  - kodu, 96
- kompilator, 44
  - (aapt), 55
  - Java, 287
  - zasobów, 55
- komunikacja
  - Inter-Process Communication, 195
  - IPC, 195
  - międzyprocesowa, 195
- komunikat
  - widoku LogCat, 112
- konstruktor, 185
  - obiektu, 193
  - widoku, 84
- kontroler
  - kulkowy, 90
  - Wii, 173

## kontrolka

- Button, 135
- EditText, 135
- LinearLayout, 140
- MapView, 174, 175, 177
- Spinner, 152
- standardowa, 177
- TableLayout, 62
- TableRow, 152
- TextView, 165, 188
- WebView, 142, 144

## kreator

- New Android Project, 134
- widżetów, 240

## kształt, 81

- shape, 81

## kursor

- położenie, 125
- pozycja, 126
- wyboru, 86

**L**

## Lee Doug, 150

## licencja

- publiczna GNU, 200

## liczby

- stałoprzecinkowe, 210
- zmiennoprzecinkowe, 210

## Linux, 21, 23, 25, 33, 34, 36, 38, 45, 127, 203

- LiMo, 11
- Mobile, 11
- powłoka, 34
- system plików, 34

## lista

- języków, 152
- ostatnio aktualizowanych programów, 282
- Recently Updated, 282
- wyboru, 72

## Locale, 163

**Ł**

## łącze

- Upload Application, 280

**M**

## Mac OS X, 21, 22, 23, 25, 26, 182, 203

## macierz transformacji, 233

## manifest, 66

## manipulator

- kulkowy, 62, 91
- trackball, 91

## mapa bitowa, 81

- bitmap, 81

## Mattson Justin, 274

## mechanizm

- synchronizacji, 288

## menedżer

- aktywności, 37, 39
- lokacji, 37
- powiadomień, 37
- powierzchni, 35
- zasobów, 37, 54

## menu

- kontekstowe, 68
- lista, 68
- opcji, 68
- w języku XML, 69

## metoda

- Activity.onKeyDown(), 110
- Activity.onSaveInstanceState(), 126
- addJavascriptInterface(), 142, 143
- addPreferencesFromResource(), 71
- close(), 289
- Context.getExternalFilesDir(), 129
- create(), 111
- createSnapshot(), 142
- deleteFile(), 127
- dismiss(), 100
- enableCompass(), 178
- enableMyLocation(), 178
- fileList(), 127

## metoda

- finalize(), 289
- findViewById(), 64, 177
- finish(), 75
- getAction(), 232, 267
- getBestProvider(), 167
- getColor(), 78
- getColumnIndexOrThrow(), 191
- getController(), 177
- getHeight(), 84
- getIntExtra(), 125
- getLastNonConfigurationInstance(), 117
- getLong(), 191
- getMenuInflater(), 69
- getPointerCount(), 233, 266, 267, 269
- getPointerId(int), 267
- getReadableDatabase(), 190
- getResources(), 78
- getSettings(), 142
- getString(), 191
- getSystemService(), 166, 171
- getWidth(), 84, 105
- getWritableDatabase(), 189, 197
- getX(), 233, 267, 270
- getX(int), 267, 269
- getY(), 233, 267, 270
- getY(int), 267, 269
- glColor4f(), 210
- glColor4x(), 210
- glDisable(), 207
- glEnable(), 207
- Handler.post(), 159
- insertOrThrow(), 189
- invalidate(), 92, 94
- isValid(), 99
- lista, 141
- loadData(), 142
- loadDataWithBaseURL(), 142
- loadUrl(), 140, 142, 143, 148, 149
- Log.d(), 74
- Log.e(), 74
- Log.i(), 74
- Log.v(), 74
- Log.w(), 74
- Log.wtf(), 74
- managedQuery(), 198
- MediaPlayer.create(), 119
- moveToNext(), 191
- obliczUzytePola(), 84
- onClick(), 65, 72, 125
- onCommand(), 259
- onCreate(), 52, 64, 81, 84, 114, 117, 125, 129, 136, 137, 140, 154, 166, 186, , 189, 188, 197, 250, 254
- onCreate(Bundle), 40
- onDestroy(), 41, 117
- onDraw(), 79, 84, 86, 90, 92, 94, 104, 123
- onDrawFrame(), 208, 214, 220, 258
- onKeyDown(), 91, 92, 93, 99
- onLocationChanged(), 168
- onOffsetsChanged(), 258
- onOptionsItemSelected(), 69
- onPause(), 40, 42, 118, 124, 167, 171, 205
- onProviderDisabled(), 168
- onProviderEnabled(), 168
- onRestart(), 41
- onRestoreInstanceState(), 127
- onResume(), 40, 117, 118, 167, 205
- onRetainNonConfigurationInstance(), 117
- onSaveInstanceState(), 40, 127
- onSaveInstanceState(Bundle), 41
- onSensorChanged(), 171
- onSizeChanged(), 84, 86, 91
- onStart(), 40, 250
- onStartCommand(), 250
- onStatusChanged(), 168
- onStop(), 41
- onSurfaceChanged(), 207, 256
- onSurfaceCreated(), 214, 255
- onSurfaceDestroyed(), 256
- onTouch(), 229, 230

- onTouchEvent(), 93
- onTrackballEvent(), 91
- onVisibilityChanged(), 257
- onXX(), 40
- openFileInput(), 128
- openFileOutput(), 128
- Paint.setColor(), 79
- play(), 119, 123
- postTranslate(), 234
- query(), 190
- rawQuery(), 190
- registerListener(), 171
- release(), 111, 119
- RendererGL.onSurfaceCreated(), 214
- requestLocationUpdates(), 167
- runOnFirstFix(), 178
- setBackgroundResource(), 81
- setBuiltInZoomControls(), 177, 226
- setContentView(), 52, 64, 137
- setDownloadListener(), 142
- setId(), 127
- setItems(), 73
- setOnClickListener(), 64, 65, 149
- setOnCompletionListener(), 111
- setSatellite(), 177
- setVideoPath(), 114
- setVolumeControlStream(), 110
- setWebChromeClient(), 142, 147
- setWebViewClient(), 142
- start(), 111, 113, 119
- startActivity(), 83, 137
- startManagingCursor(), 190
- stop(), 119
- stopLoading(), 142
- super.onDestroy(), 255
- superklasy, 252
- środowiska JavaScript, 143
- terminate(), 289
- verifyPointerIndex(), 269
- View.onDraw(), 79
- window.alert(), 148
- wywołana z klasy, 127

- Microsoft, 174, 203
- MIDI
  - instrumenty, 112
- MIME, 199
- mm
  - milimetry, 59
  - rozmiar zmierzony miarką, 59
- model
  - Document Object Model, 143
  - DOM, 143
  - obiektowy model dokumentu, 143
- moduł
  - payment processor, 280
  - przetwarzania opłat, 280
- Motorola, 21
- motyw, 67
- Mozilla Firefox, 182
- MP3, 12, 35
  - MPEG-3, 112
- MP4
  - MPEG-4, 35
  - MPEG-4 o niskiej szybkości transmisji, 113
- MySQL, 183

## N

- nawigacja
  - dostępność, 274
- nazwa pakietu, 52
- NDK, 36
  - Native Development Kit, 209
- NetBeans, 22
- NinePatch, 81
- Nintendo
  - DS, 170
  - Wii, 163
- NKD
  - Native Development Toolkit, 36
  - zestaw narzędzi umożliwiających tworzenie bibliotek natywnych, 36

Nokia, 203  
 Number Place, 83  
 Nvidia, 203

## O

obiekt

- Button, 144
- Cursor, 193
- Drawable, 242
- Handler, 147
- Java, 142, 143, 147
- JavaScript, 142, 143
- JButton, 51
- JFrame, 51
- LocationListener, 167
- MapController, 177
- nasłuchujący, 73
- POJO, 147
- powiązany z kodem JavaScript, 143
- R.layout.main, 52
- Runnable, 147
- SimpleCursorAdapter, 193, 200
- Spinner, 155, 157, 158
- Surface, 256
- TextView, 144, 147
- WebChromeClient, 147
- WebSettings, 142
- window.android, 146
- XmlHttpRequest, 174

obraz

- NinePatch, 242
- PNG, 242

obszar

- extraData, 100

odnośnik

- do danych, 54
- R.raw.glowna, 117

OGG format firmy Vorbis, 112

okno

- dialogowe
- New Project, 240
- informacyjne, 62

- Open Handest Alliance, 11, 21
- OpenGL, 12, 33, 35, 77, 200, 201, 202, 203, 204, 205, 206, 207, 208, 209, 210, 212, 214, 216, 217, 218, 220, 248, 256
  - dla osadzonych systemów, 203
  - ES, 105, 201, 203, 205, 209
  - ES 1.0, 209
  - ES 2.0, 209
  - for Embedded Systems, 203
  - interakcja, 254
  - opcje środowiska, 207, 208
  - wersja 1.3, 209

OpenIntents, 173

Oracle, 22, 183

orientacja

- panoramiczna, 60
- pionowa, 60
- pozioma, 60, 62

ostrosłup widzenia

- view frustum, 202

oświetlenie, 212

- ambient, 213
- diffuse, 213
- dookolne, 213
- kierunkowe, 213
- rozproszone, 213
- specular, 213
- tekstura, 215

Owens Mike, 200

## P

pakiet

- .apk, 279
- android.graphics, 77, 105
- android.media, 108
- android.view.View, 65
- com.google.android.maps, 178
- Java
  - nazewnictwo, 278
- java.lang.management, 289
- java.net.HttpURLConnection, 153



- java.util.concurrent, 150, 288, 290
- javax.security.auth.kerberos, 290
- javax.security.sasl, 290
- nieobsługiwany, 290
- org.przyklad.dotyk, 267
- pamięć
  - flash, 127
  - karta SD, 15
  - zewnętrzna, 15
- parametr
  - array.length, 102
  - trud, 124
- PayPal, 280
- PHP, 182
- platforma
  - Android, 13
  - Java, 22
  - mobilna, 11, 21
  - projektowa, 11
- plik
  - .apk, 279, 282
  - .class, 36
  - .dex, 36
  - .jar, 36
  - AndroidManifest.xml, 32, 45, 57, 65, 67, 71, 75, 84, 115, 117, 141, 151, 165, 175, 198, 229, 241, 245, 249, 264, 275, 278, 282
  - bazy danych, 182
  - Dotyk.java, 228
  - dźwiękowy, 109
  - indeks.html, 145, 146, 149
  - klasy, 36
  - main.xml, 52, 54
  - MP3, 112
  - OGG, 112
  - OpenGL.java, 204
  - pamięci flash, 121
  - R.java, 53, 65
  - strings.xml, 57
  - tablice.xml, 155
  - układu graficznego, 144, 175
  - WAV, 112
  - wewnętrzny system, 127
  - wideo, 113, 115, 117
  - wklejanie do katalogu, 108
  - XML, 54, 81, 84, 95, 127
    - zasobu, 53, 56
      - XML, 78
    - zewnętrzny system, 129
    - źródłowy, 16
- plynność
  - frames per second, 219
- poczta email, 38
- podkładka kierunkowa, 112
- pole
  - Activity, 240
  - widzenia, 202
- polecenie
  - adb, 182
  - adb logcat, 74, 111
- powłoka
  - overlay, 178
- poziom
  - level, 81
- proces linuksowy, 40
- program
  - Translator, 193
  - usługa, 250
- programowanie
  - równoległe, 150
- projektowanie
  - deklaracyjne, 51, 52
  - interfejs użytkownika, 49
  - proceduralne, 51, 52
  - za pomocą deklaracji, 51
- przeciążnięcie, 226
- przedrostek
  - @android, 67
- przeglądarka
  - sandbox, 143
- PrzeglądarkaLokalna, 134, 144, 146, 148
- przycisk
  - cofania, 39, 75

PSP PlayStation Portable, 219

pt

1/72 cala, 59

punkty, 59

px

piksele, 59

punkty na ekranie, 59

## Q

QuickTime Pro, 113

## R

refleksja, 266

renderowanie, 206, 208

Re-Translate Pro, 280

rozdzielczość, 59

wymiary niezależne, 59

rzutowanie, 201

## S

Safari, 133

Samsung, 203

SDK, 264, 265

Setup, 23

wersja zestawu, 274

Sensor Simulator, 173

sieć

bezprzewodowa, 164

Web, 134, 138, 140, 141, 143, 148

z widokami, 138

Silicon Graphics, 202

silnik

bazy danych, 35

przeglądarki, 35, 140, 148

SQL, 184

klasa, 200

skala

scale, 81

Skype, 182

słowo

kluczowe

synchronized, 288

smartfon, 12

Solaris, 182

Sony, 203

PlayStation, 170

Portable, 219

sp, 59

piksele niezależne od skali, 59

scale-independent pixels, 59

SQL

Server, 183

Structured Query Language, 182

strukturalny język zapytań, 182

SQLite, 35

stała

ACTION\_POINTER\_DOWN, 266

stan

instancji, 41, 42

proces zachowania, 42

przerwania, 42

state, 81

trwały, 40

zakończenia, 42

zatrzymania, 42

zmiany, 40

sterownik

USB, 24

stuknięcie, 225, 226

Swing, 51

Symbian, 11, 182, 203

symulator, 173

synchronizacja, 14

system

szesnastkowy, 54

## Ś

środowisko uruchomieniowe, 36, 37

**T**

tablica, 102, 288  
   typ, 83  
   wartości zmiennopozycyjnych, 172  
 tabliczka dotykowa, 12  
 tapeta  
   aktywna, 14, 15, 248  
   live wallpaper, 248  
 TCP/IP, 133  
 tekst  
   parametry, 58  
 tekstura  
   nakładanie, 215  
   współrzędne, 217  
 telefon  
   Apple iPhone, 35, 133, 226  
   Nokia S60, 35  
   orientacja, 42  
   pamięć wewnętrzna, 274, 275  
   położenie, 60  
   ustawienia, 163  
 tłumaczenie  
   maszynowe, 150  
 Torvalds Linus, 33  
 trackball, 12  
 translator, 157, 159, 287  
 Translator, 134, 151, 152, 153, 154, 156,  
   157, 158, 159  
   aktualizacja, 280  
 triangulacja, 12, 165  
 tryb  
   rysowania, 212  
 tworzenie podklas, 266  
 typ  
   Bundle, 83  
   byte, 288  
   char, 288  
   double, 288  
   float, 288  
   int, 288  
   long, 288

MIME, 199  
 Object, 288  
 Parcelable, 83  
 Serializable, 83  
 short, 288  
 String, 288

**U**

układ  
   FrameLayout., 55  
   graficzny, 55, 84, 85, 151, 152  
     dostosowanie, 60  
     tryb panoramiczny, 60  
     własny, 63  
   layout, 55  
   LinearLayout, 55  
   RelativeLayout, 55  
   TableLayout, 55  
 Unicode, 289  
 Unix, 43  
 uprawnienia, 45  
   <uses-permission>, 141  
   ACCESS\_COARSE\_LOCATION.,  
     45  
   ACCESS\_FINE\_LOCATION., 45  
   INTERNET., 45  
   READ\_CONTACTS., 45  
   RECEIVE\_SMS., 45  
   WRITE\_CONTACTS., 45  
 URI  
   adres, 196  
   identyfikator, 196, 197, 198  
   jednolity identyfikator zasobu, 137  
   uniform resource identifier, 137  
 urządzenie  
   AVD, 28, 30, 262, 263, 264, 273  
   czujniki, 171  
   Droid, 31  
   em16, 263  
   em22, 29  
   mobilne, 11, 12, 14, 55

- urządzenie
    - Nexus, 31
    - ograniczona ilość pamięci, 36
    - parametry techniczne, 104
    - przenośne, 11
  - USB
    - kabel, 32
    - port, 31
  - usługa, 42, 43
    - ustawienia, 70
  - UTF-8, 54
- V**
- Vista, 35
  - Vorbis, 112
- W**
- warstwa, 33
    - aplikacji, 38
    - layer, 81
    - sprzętowa, 33
    - szkieletu aplikacji, 37
    - widżetów, 38
  - wartość
    - true, 123
  - WAV
    - nieskompresowana modulacja PCM, 112
  - Web Tools, 26
  - WebKit, 133, 138
  - wideo
    - formaty, 113
    - odtworzenie, 113
    - restartowanie, 117
  - widok
    - Emulator Control, 170, 179
    - File Explorer, 115, 182
    - LinearLayout, 152
    - LogCat, 57, 74, 111, 112
    - Package Explorer, 267, 279
    - TableLayout, 151
    - WebView, 139, 140, 141, 143, 145, 148
    - widżetu, 52
    - WidokPrzeglądarki, 139, 140
  - widżet, 14, 15, 38, 67, 80, 225, 239, 244, 259, 275
    - aktualizacja, 242, 246, 247
    - definicja, 241
    - ekran startowy, 245
    - gadżet, 38
    - kreator, 240
    - niewielki widok aplikacji, 239
    - potomny, 56
    - rozmiar, 242
    - standardowy, 85
  - wielodotykowość, 14, 225, 226, 227, 228, 237, 262, 265, 270
    - bezruch, 231
  - wiersz poleceń, 115
  - Windows, 21, 22, 23, 25, 31, 38
    - 32-bity, 23
    - 64 bity, 23
    - Mobile, 11
  - WMA
    - Windows Media Audio, 112
  - WST
    - standardowe narzędzia sieci Web, 25
    - Web Standard Tools, 25
  - wstęga trójkątów
    - triangle strips, 212
  - WWW, 138
  - wyjątek
    - VerifyError, 265
  - wyświetlacz
    - LCD, 219
    - orientacja, 12
    - rozdzielczość, 12
  - wziernik
    - viewport, 201

**X**

XML

edytor, 65

**Y**

Yahoo!, 133, 174

**Z**

zabezpieczenie 12, 34, 45

luki, 143

zakładka

strings.xml, 56

Target, 263

zapora sieciowa, 26

zarządzanie, 12

pamięcią, 34

procesami, 34

zasób, 37, 44

alternatywny, 60

przyrostki, 53, 54, 59, 62

bitmapa, 44

domyślny, 53

egzemplarz, 52

identyfikator, 44, 52, 73, 78, 137

informacja niebędąca kodem, 44

klucz, 44

odnośniki, 44

zlokalizowany ciąg znaków, 44

zdarzenie

ACTION\_MOVE, 232

ACTION\_POINTER\_DOWN, 232

ACTION\_POINTER\_UP, 232

zestaw

bundle, 121

stanów instancji, 121

zmiana kontekstu, 288

zmienna, 102

app\_name, 57

macierz, 234

PATH., 23

srod, 233

start, 233

staryDyst, 233

this, 65

xOffset, 258

znacznik

&lt;/manifest&gt;, 264

&lt;activity&gt;, 65

&lt;activity&gt;, 241

&lt;application&gt;, 141

&lt;application&gt;):, 198

&lt;manifest&gt;, 275

&lt;table&gt;, 151

&lt;tr&gt;, 151

&lt;uses-library&gt;, 175, 176

XML, 55

**Ż**

źródło

danych, 173



**Android to system operacyjny dla telefonów komórkowych**, który swoją dynamiką wzrostu udziału w rynku przyćmił niejeden produkt. Według badań IDC dla Europy Zachodniej w ciągu niecałego roku Android zwiększył swój udział z 4 do 31 procent! Co jest tego przyczyną? Niezwykłe precyzyjnie wykonana i doskonale działająca synchronizacja danych ze wszystkimi najważniejszymi usługami firmy Google, łatwy dostęp do internetu dzięki szybkiej przeglądarce oraz tysiące świetnych aplikacji, zarówno darmowych, jak i płatnych. A skoro mowa o aplikacjach, jak stworzyć własny produkt na tę dynamicznie rozwijającą się platformę? Jak wprowadzić go na rynek i zarobić?

Na te oraz wiele innych pytań związanych z Androidem odpowie Ci ta książka! Na samym wstępie poznasz trochę faktów związanych z powstawaniem samej platformy oraz zobaczysz, jakie nowości oferowały kolejne jej wersje. Wbrew pozorom wiedza ta przyda się przy tworzeniu aplikacji! Po krótkim wstępie historycznym dowiesz się, jak szybko przygotować stanowisko pracy, oraz poznasz kluczowe pojęcia z nim związane. W trakcie lektury nauczysz się projektować interfejs użytkownika, obsługiwać dane wejściowe czy multimedia. Niezwykle ważne są wiadomości zawarte w części trzeciej. Wykorzystanie tych informacji pozwoli Ci na stworzenie nowatorskiej aplikacji. Dostęp do sieci, usług geolokalizacyjnych, baz SQL i trójwymiarowej grafiki pozwoli Ci rozwinąć skrzydła! Książka ta jest długo wyczekiwaną pozycją, omawiającą wszystkie istotne zagadnienia związane z platformą Android, dlatego musisz ją mieć!

- Nowości w poszczególnych wersjach Androida
- Instalacja narzędzi oraz przygotowanie środowiska pracy
- Kluczowe pojęcia związane z wytwarzaniem aplikacji dla Androida
- Korzystanie z zasobów
- Projektowanie interfejsu użytkownika
- Wykorzystanie motywów
- Tworzenie elementów menu
- Grafika dwuwymiarowa oraz trójwymiarowa
- Obsługa danych wejściowych
- Wykorzystanie multimediów
- Przechowywanie danych na urządzeniu mobilnym
- Usługi geolokalizacji
- Wykorzystanie bazy danych SQLite
- Wielodotykowość
- Tworzenie aktywnych tapet
- Publikowanie aplikacji w serwisie Android Market

Wykorzystaj potencjał i popularność platformy Android!

W katalogu 6607



Katalogiarna internetowa:  
<http://helion.pl>



Zamówienia telefoniczne:  
**0 801 339900**



**0 601 339900**



**Helion**

Sprzedaż i promowanie publikacji:  
• <http://helion.pl/promocje>  
Książki naukowe i artystyczne  
• <http://helion.pl/bestsellery>  
Zamów informacje o nominacjach  
• <http://helion.pl/nomwest>

Helion SA  
ul. Rakowiecki 1c, 44-100 Gliwice  
tel.: 32 230 98 63  
e-mail: [helion@helion.pl](mailto:helion@helion.pl)  
<http://helion.pl>

**helion.pl**  
Kolegium  
Intelektualne

Cena 57,00 zł

ISBN 978-83-246-3140-7



9 788324 631407

Informatyka w najlepszym wydaniu