

Kompendium wiedzy na temat platformy Android!



Android 3

tworzenie aplikacji

Satya Komatineni • Dave MacLean • Sayed Hashimi

Tytuł oryginału: Pro Android 3

Tłumaczenie: Krzysztof Sawka

ISBN: 978-83-246-3586-3

Polish edition copyright © Helion 2012
All rights reserved

Original edition copyright © 2011 by Satya Komatineni, Dave MacLean, and Sayed Y. Hashimi
All rights reserved

All rights reserved. No part of this work may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage or retrieval system, without the prior written permission of the copyright owner and the publisher.

Trademarked names, logos, and images may appear in this book. Rather than use a trademark symbol with every occurrence of a trademarked name, logo, or image we use the names, logos, and images only in an editorial fashion and to the benefit of the trademark owner, with no intention of infringement of the trademark. NFC Forum and the NFC Forum logo are trademarks of the Near Field Communication Forum.

The use in this publication of trade names, trademarks, service marks, and similar terms, even if they are not identified as such, is not to be taken as an expression of opinion as to whether or not they are subject to proprietary rights.

Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości lub fragmentu niniejszej publikacji w jakiegokolwiek postaci jest zabronione. Wykonywanie kopii metodą kserograficzną, fotograficzną, a także kopiowanie książki na nośniku filmowym, magnetycznym lub innym powoduje naruszenie praw autorskich niniejszej publikacji.

Wszystkie znaki występujące w tekście są zastrzeżonymi znakami firmowymi bądź towarowymi ich właścicieli.

Autor oraz Wydawnictwo HELION dołożyli wszelkich starań, by zawarte w tej książce informacje były kompletne i rzetelne. Nie biorą jednak żadnej odpowiedzialności ani za ich wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych lub autorskich. Autor oraz Wydawnictwo HELION nie ponoszą również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

Pliki z przykładami omawianymi w książce można znaleźć pod adresem:
<ftp://ftp.helion.pl/przyklady/and3ta.zip>

Wydawnictwo HELION
ul. Kościuszki 1c, 44-100 GLIWICE
tel. 32 231 22 19, 32 230 98 63
e-mail: helion@helion.pl
WWW: <http://helion.pl> (księgarnia internetowa, katalog książek)

Drogi Czytelniku!

Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres

<http://helion.pl/user/opinie/and3ta>

Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

- [Kup książkę](#)
- [Poleć książkę](#)
- [Oceń książkę](#)

- [Księgarnia internetowa](#)
- [Lubię to! » Nasza społeczność](#)

Spis treści

Przedmowa	21
O autorach	23
Informacje o redaktorze technicznym	25
Podziękowania	27
Słowo wstępne	29
Rozdział 1. Wprowadzenie do platformy obliczeniowej Android	31
Nowa platforma dla nowego typu komputera osobistego	32
Początki historii Androida	33
Zapoznanie się ze środowiskiem Dalvik VM	36
Stos programowy Androida	37
Projektowanie aplikacji użytkownika końcowego za pomocą zestawu Android SDK	38
Emulator Androida	38
Interfejs użytkownika na platformie Android	39
Podstawowe składniki Androida	40
Zaawansowane koncepcje interfejsu użytkownika	41
Składniki usług w Androidzie	43
Składniki multimediów oraz telefonii w Androidzie	43
Pakiety Java w Androidzie	44
Wykorzystanie zalet kodu źródłowego Androida	48
Przykładowe projekty zawarte w książce	49
Podsumowanie	49
Rozdział 2. Konfigurowanie środowiska programowania	51
Konfigurowanie środowiska	52
Pobieranie zestawu JDK 6	52
Pobieranie środowiska Eclipse 3.6	53
Pobieranie zestawu Android SDK	54
Okno narzędzi	56
Instalowanie narzędzi ADT	56
Przedstawienie podstawowych składników	58
Widok	58
Aktywność	59

Intencja	59
Dostawca treści	59
Usługa	59
AndroidManifest.xml	60
Urządzenia AVD	60
Witaj, świecie!	60
Wirtualne urządzenia AVD	65
Poznanie struktury aplikacji Androida	67
Analiza aplikacji Notepad	69
Wczytanie oraz uruchomienie aplikacji Notepad	69
Rozłożenie kodu na czynniki pierwsze	71
Badanie cyklu życia aplikacji	78
Usuwanie błędów w aplikacji	81
Uruchamianie emulatora	83
StrictMode	84
Odnosińki	89
Podsumowanie	89
Rozdział 3. Korzystanie z zasobów	91
Zasoby	91
Zasoby typu string	92
Zasoby typu layout	94
Składnia odniesienia do zasobu	95
Definiowanie własnych identyfikatorów zasobów do późniejszego użytku	97
Skompilowane oraz nieskompilowane zasoby Androida	98
Rodzaje głównych zasobów w Androidzie	99
Praca na własnych plikach zasobów XML	109
Praca na nieskompresowanych zasobach	111
Praca z dodatkowymi plikami	111
Przegląd struktury katalogów mieszczących zasoby	112
Zasoby a zmiany konfiguracji	112
Odnosińki	116
Podsumowanie	117
Rozdział 4. Dostawcy treści	119
Analiza wbudowanych dostawców Androida	120
Architektura dostawców treści	126
Implementowanie dostawców treści	139
Testowanie dostawcy BookProvider	150
Dodawanie książki	150
Usuwanie książki	150
Zliczanie książek	151
Wyświetlanie listy książek	151
Odnosińki	152
Podsumowanie	153

Rozdział 5. Intencje	155
Podstawowe informacje na temat intencji	155
Intencje dostępne w Androidzie	156
Przegląd struktury intencji	159
Intencje a identyfikatory danych URI	159
Działania ogólne	160
Korzystanie z dodatkowych informacji	161
Stosowanie składników	
do bezpośredniego przywoływania aktywności	162
Kategorie intencji	163
Reguły przydzielania intencji do ich składników	166
Działanie ACTION_PICK	169
Działanie ACTION_GET_CONTENT	171
Wprowadzenie do intencji oczekujących	172
Oдноśniki	173
Podsumowanie	174
Rozdział 6. Budowanie interfejsów użytkownika oraz używanie kontroltek	175
Projektowanie interfejsów UI w Androidzie	175
Programowanie interfejsu użytkownika wyłącznie za pomocą kodu	177
Tworzenie interfejsu użytkownika wyłącznie w pliku XML	179
Konstruowanie interfejsu użytkownika	
za pomocą kodu oraz języka XML	180
FILL_PARENT a MATCH_PARENT	182
Standardowe kontrolki Androida	182
Kontrolki tekstu	183
Kontrolki przycisków	187
Kontrolka ImageView	195
Kontrolki daty i czasu	197
Kontrolka MapView	200
Działanie adapterów	200
Zapoznanie się z klasą SimpleCursorAdapter	200
Zapoznanie się z klasą ArrayAdapter	202
Wykorzystywanie adapterów wraz z kontrolkami AdapterView	204
Podstawowa kontrolka listy — ListView	205
Kontrolka GridView	213
Kontrolka Spinner	215
Kontrolka Gallery	217
Tworzenie niestandardowych adapterów	218
Inne kontrolki w Androidzie	223
Style i motywy	224
Stosowanie stylów	224
Stosowanie motywów	227

Menedżery układu graficznego	227
Menedżer układu graficznego LinearLayout	228
Menedżer układu graficznego TableLayout	231
Menedżer układu graficznego RelativeLayout	235
Menedżer układu graficznego FrameLayout	237
Dostosowanie układu graficznego do konfiguracji różnych urządzeń	239
Usuwanie błędów i optymalizacja układów graficznych za pomocą narzędzia Hierarchy Viewer	242
Odnośniki	244
Podsumowanie	245
Rozdział 7. Praca z menu	247
Menu w Androidzie	247
Tworzenie menu	249
Praca z grupami menu	250
Odpowiedź na wybór elementów menu	251
Utworzenie środowiska testowego do sprawdzania menu	253
Praca z innymi rodzajami menu	259
Rozszerzone menu	259
Praca z menu w postaci ikon	259
Praca z podmenu	260
Zabezpieczanie menu systemowych	261
Praca z menu kontekstowymi	261
Praca z menu alternatywnymi	264
Praca z menu w odpowiedzi na zmianę danych	268
Wczytywanie menu poprzez pliki XML	268
Struktura pliku XML zasobów menu	268
Zapełnianie plików XML zasobów menu	269
Tworzenie odpowiedzi dla elementów menu opartych na pliku XML	270
Krótkie wprowadzenie do dodatkowych znaczników menu w pliku XML	271
Odnośniki	272
Podsumowanie	272
Rozdział 8. Praca z oknami dialogowymi	273
Korzystanie z okien dialogowych w Androidzie	274
Projektowanie okien alertów	274
Projektowanie okna dialogowego zachęty	276
Natura okien dialogowych w Androidzie	281
Przeprojektowanie okna dialogowego zachęty	282
Praca z zarządzanymi oknami dialogowymi	283
Protokół zarządzanych okien dialogowych	283
Przekształcenie niezarządzanego okna dialogowego na zarządzane okno dialogowe	283
Uproszczenie protokołu zarządzanych okien dialogowych	285

Praca z klasą Toast	293
Odnośniki	294
Podsumowanie	294
Rozdział 9. Praca z preferencjami i zachowywanie stanów	295
Badanie struktury preferencji	296
Klasa ListPreference	296
Widok CheckBoxPreference	305
Widok EditTextPreference	307
Widok RingtonePreference	308
Organizowanie preferencji	310
Programowe zarządzanie preferencjami	312
Zapisywanie stanu za pomocą preferencji	313
Odnośniki	314
Podsumowanie	315
Rozdział 10. Analiza zabezpieczeń i uprawnień	317
Model zabezpieczeń w Androidzie	317
Przegląd pojęć dotyczących zabezpieczeń	317
Podpisywanie wdrażanych aplikacji	318
Przeprowadzanie testów zabezpieczeń środowiska wykonawczego	324
Zabezpieczenia na granicach procesu	324
Deklarowanie oraz stosowanie uprawnień	325
Stosowanie niestandardowych uprawnień	326
Stosowanie uprawnień identyfikatorów URI	332
Odnośniki	334
Podsumowanie	335
Rozdział 11. Tworzenie i użytkowanie usług	337
Użytkowanie usług HTTP	337
Wykorzystanie modułu HttpClient do żądań wywołania GET	338
Wykorzystanie modułu HttpClient do żądań wywołania POST (przykład wieloczęściowy)	340
Parsery SOAP, JSON i XML	342
Obsługa wyjątków	343
Problemy z wielowątkowością	345
Zabawa z przekroczeniami limitu czasu	348
Stosowanie klasy HttpURLConnection	349
Używanie klasy AndroidHttpClient	349
Stosowanie wątków drugoplanowych (AsyncTask)	351
Obsługa zmian konfiguracji za pomocą klasy AsyncTask	357
Pobieranie plików za pomocą klasy DownloadManager	362
Stosowanie usług w Androidzie	367
Usługi w Androidzie	368
Usługi lokalne	369
Usługi AIDL	376

Definiowanie interfejsu usługi w języku AIDL	376
Implementowanie interfejsu AIDL	379
Wywoływanie usługi z poziomu aplikacji klienckiej	381
Przekazywanie usługom złożonych typów danych	385
Przykład aplikacji użytkowej korzystającej z usług	395
Interfejs Tłumacz Google	395
Stosowanie interfejsu Tłumacz Google	397
Odnośniki	405
Podsumowanie	405
Rozdział 12. Analiza pakietów	407
Pakiety i procesy	407
Szczegółowa specyfikacja pakietu	407
Przekształcanie nazwy pakietu w nazwę procesu	408
Tworzenie listy zainstalowanych pakietów	408
Usuwanie pakietu za pomocą aplikacji Package Browser	409
Jeszcze raz o procesie podpisywania pakietów	409
Zrozumienie koncepcji podpisów cyfrowych — scenariusz 1.	410
Zrozumienie koncepcji podpisów cyfrowych — scenariusz 2.	410
Wyjaśnienie koncepcji podpisów cyfrowych	410
Jak zatem tworzymy cyfrowy podpis	411
Implikacje wynikające z podpisywania plików	411
Współdzielenie danych pomiędzy pakietami	412
Natura współdzielonych identyfikatorów użytkownika	412
Schemat kodu wykorzystywanego przy współdzieleniu danych	413
Projekty bibliotek	414
Czym jest projekt bibliotek?	414
Twierdzenia dotyczące projektów bibliotek	414
Utworzenie projektu bibliotek	417
Tworzenie projektu testowego wykorzystującego projekt bibliotek	420
Odnośniki	425
Podsumowanie	426
Rozdział 13. Analiza procedur obsługi	427
Składniki Androida i wątkowanie	427
Aktywności działają w głównym wątku	428
Odbiorcy komunikatów działają w głównym wątku	429
Usługi działają w głównym wątku	429
Dostawcy treści działają w głównym wątku	429
Skutki posiadania pojedynczego głównego wątku	429
Pule wątków, dostawcy treści, składniki zewnętrznych usług	429
Narzędzia wątkowania — poznaj swój wątek	429
Procedury obsługi	431
Skutki przetrzymywania głównego wątku	432
Zastosowanie procedury obsługi	
do opóźnienia operacji w wątku głównym	432

Przykładowy kod źródłowy procedury obsługi opóźniającej	
przeprowadzanie operacji	433
Konstruowanie odpowiedniego obiektu Message	435
Wysyłanie obiektów Message do kolejki	435
Odpowiedź na metodę zwrotną handleMessage	436
Stosowanie wątków roboczych	436
Przywoływanie wątku roboczego z poziomu menu	437
Komunikacja pomiędzy wątkami głównym i roboczym	438
Szybki przegląd — jak działa wątek?	440
Klasy przykładowego sterownika procedury obsługi	441
Plik aktywności sterującej	442
Plik układu graficznego	444
Plik menu	445
Plik manifest	445
Czas życia składnika i procesu	446
Cykl życia aktywności	446
Cykl życia usługi	448
Cykl życia odbiorców komunikatów	448
Cykl życia dostawcy treści	448
Instrukcje dotyczące kompilowania kodu	449
Utworzenie projektu za pomocą pliku ZIP	449
Tworzenie projektu za pomocą listingów	449
Odnośniki	450
Podsumowanie	450

Rozdział 14. Odbiorcy komunikatów i usługi długoterminowe 453

Odbiorcy komunikatów	453
Wysyłanie komunikatu	454
Tworzenie prostego odbiorcy — przykładowy kod	454
Rejestrowanie odbiorcy komunikatów w pliku manifest	456
Wysyłanie komunikatu testowego	456
Wprowadzanie wielu odbiorców komunikatów	460
Projekt wykorzystujący odbiorców pozaprocesowych	462
Używanie powiadomień pochodzących od odbiorcy komunikatów	463
Monitorowanie powiadomień za pomocą menedżera powiadomień	463
Wysyłanie powiadomienia	464
Długoterminowi odbiorcy komunikatów i usługi	467
Protokół długoterminowego odbiorcy komunikatów	468
Klasa IntentService	469
Kod źródłowy klasy IntentService	470
Rozszerzanie klasy IntentService na odbiorcę komunikatów	472
Abstrakcja długoterminowej usługi wysyłającej komunikaty	472
Długoterminowy odbiorca komunikatów	474
Wyodrębnianie blokady przechodzenia	
w stan zatrzymania za pomocą klasy LightedGreenRoom	476

Oświetlony zielony pokój	478
Implementacja oświetlonego zielonego pokoju	478
Implementacja długoterminowej usługi	483
Szczegółowe informacje na temat usługi nietrwałej	484
Informacje dotyczące trwałej usługi	485
Odmiana nietrwałej usługi — ponownie dostarczane intencje	485
Definiowanie flag usługi w metodzie onStartCommand	485
Wybieranie odpowiedniego trybu usługi	485
Kontrolowanie blokady przechodzenia w stan zatrzymania z dwóch miejsc jednocześnie	486
Implementacja długoterminowej usługi	486
Testowanie długoterminowych usług	488
Instrukcje dotyczące kompilowania kodu	489
Utworzenie projektów za pomocą pliku ZIP	489
Utworzenie projektów za pomocą listingów	489
Odnośniki	491
Podsumowanie	492
Rozdział 15. Badanie menedżera alarmów	493
Podstawy menedżera alarmów — konfiguracja prostego alarmu	493
Uzyskanie dostępu do menedżera alarmów	494
Definiowanie czasu uruchomienia alarmu	494
Konfigurowanie odbiorcy dla alarmu	495
Utworzenie oczekującej intencji dostosowanej do alarmu	495
Ustawianie alarmu	496
Projekt testowy	497
Analiza alternatywnych wersji menedżera alarmów	503
Konfigurowanie powtarzalnego alarmu	503
Anulowanie alarmu	506
Praca z wieloma alarmami jednocześnie	508
Pierwszeństwo intencji w uruchamianiu alarmów	512
Trwałość alarmów	515
Twierdzenia dotyczące menedżera alarmów	515
Odnośniki	516
Podsumowanie	516
Rozdział 16. Analiza animacji dwuwymiarowej	517
Animacja poklatkowa	518
Zaplanowanie animacji poklatkowej	518
Utworzenie aktywności	519
Dodawanie animacji do aktywności	520
Animacja układu graficznego	523
Podstawowe typy animacji klatek kluczowych	524
Zaplanowanie środowiska testowego animacji układu graficznego	525

Utworzenie aktywności oraz widoku ListView	525
Animowanie widoku ListView	528
Stosowanie interpolatorów	531
Animacja widoku	533
Animacja widoku	533
Dodawanie animacji	536
Zastosowanie klasy Camera do symulowania głębi w obrazie dwuwymiarowym	539
Analiza interfejsu AnimationListener	541
Kilka uwag na temat macierzy transformacji	541
Odnośniki	542
Podsumowanie	543
Rozdział 17. Analiza usług wykorzystujących mapy i dane o lokalizacji	545
Pakiet do pracy z mapami	546
Uzyskanie klucza interfejsu API mapy od firmy Google	546
Klasy MapView i MapActivity	548
Dodawanie znaczników za pomocą nakładek	553
Pakiet do obsługi danych o położeniu geograficznym	559
Geokodowanie w Androidzie	559
Geokodowanie za pomocą wątków przebiegających w tle	563
Usługa LocationManager	566
Wyświetlanie informacji o położeniu za pomocą klasy MyLocationOverlay	574
Stosowanie alertów odległościowych	578
Odnośniki	583
Podsumowanie	583
Rozdział 18. Używanie interfejsów telefonii	585
Praca z wiadomościami SMS	585
Wysyłanie wiadomości SMS	585
Monitorowanie przychodzących wiadomości tekstowych	589
Praca z folderami wiadomości SMS	592
Wysyłanie wiadomości e-mail	593
Praca z menedżerem telefonii	594
Protokół inicjalizacji sesji (SIP)	597
Odnośniki	600
Podsumowanie	600
Rozdział 19. Używanie szkieletu multimedialnego	601
Stosowanie interfejsów API multimedialnych	601
Wykorzystywanie kart SD	602
Odtwarzanie multimedialnych	606
Odtwarzanie źródeł dźwiękowych	607
Odtwarzanie plików wideo	619

Rejestrowanie multimediiów	621
Analiza procesu rejestracji dźwięku za pomocą klasy MediaRecorder	622
Rejestracja dźwięków za pomocą klasy AudioRecord	626
Analiza procesu rejestracji wideo	630
Analiza klasy MediaStore	640
Rejestrowanie dźwięku za pomocą intencji	641
Dodawanie plików do magazynu multimediiów	644
Podłączenie klasy MediaScanner do całej karty SD	647
Odnośniki	647
Podsumowanie	648

Rozdział 20. Programowanie grafiki trójwymiarowej za pomocą biblioteki OpenGL	649
Historia i podstawy biblioteki OpenGL	650
OpenGL ES	651
Środowisko OpenGL ES a Java ME	652
M3G — inny standard grafiki trójwymiarowej środowiska Java	652
Podstawy struktury OpenGL	653
Podstawy rysowania za pomocą biblioteki OpenGL	654
Kamera i współrzędne	659
Tworzenie interfejsu pomiędzy standardem OpenGL ES a Androidem	663
Stosowanie klasy GLSurfaceView i klas pokrewnych	664
Implementacja klasy Renderer	664
Zastosowanie klasy GLSurfaceView z poziomu aktywności	667
Zmiana ustawień kamery	672
Wykorzystanie indeksów do dodania kolejnego trójkąta	675
Animowanie prostego trójkąta w bibliotece OpenGL	676
Stawianie czoła bibliotece OpenGL — kształty i tekstury	678
Rysowanie prostokąta	679
Praca z kształtami	680
Praca z teksturami	694
Rysowanie wielu figur geometrycznych	699
OpenGL ES 2.0	703
Powiązania środowiska Java z bibliotekami OpenGL ES 2.0	704
Etapy renderowania	707
Jednostki cieniujące	708
Kompilowanie jednostek cieniujących w programie	709
Uzyskiwanie dostępu do zmiennych jednostek cieniowania	711
Prosty trójkąt napisany w środowisku OpenGL ES 2.0	711
Dodatkowe źródła dotyczące środowiska OpenGL ES 2.0	715
Instrukcje związane z kompilowaniem kodu	715
Odnośniki	715
Podsumowanie	716

Rozdział 21. Badanie aktywnych folderów	717
Badanie aktywnych folderów	717
W jaki sposób użytkownik korzysta z aktywnych folderów	718
Tworzenie aktywnego folderu	722
Instrukcje dotyczące kompilowania kodu	733
Odnosińniki	733
Podsumowanie	734
Rozdział 22. Widżety ekranu startowego	735
Architektura widżetów ekranu startowego	736
Czym są widżety ekranu startowego?	736
W jaki sposób użytkownik korzysta z widżetów ekranu startowego?	736
Cykl życia widżetu	740
Przykładowy widżet	745
Definiowanie dostawcy widżetu	747
Definiowanie rozmiaru widżetu	748
Pliki związane z układem graficznym widżetu	749
Implementacja dostawcy widżetu	751
Implementacja modeli widżetów	753
Implementacja aktywności konfiguracji widżetu	761
Ograniczenia i rozszerzenia widżetów	764
Odnosińniki	765
Podsumowanie	766
Rozdział 23. Wyszukiwanie w Androidzie	767
Wyszukiwanie w Androidzie	768
Badanie procesu przeszukiwania globalnego w Androidzie	768
Włączanie dostawców propozycji do procesu wyszukiwania globalnego	774
Interakcja aktywności z przyciskiem wyszukiwania	777
Zachowanie przycisku wyszukiwania wobec standardowej aktywności	778
Zachowanie aktywności wyłączającej wyszukiwanie	786
Jawne wywoływanie wyszukiwania za pomocą menu	787
Wyszukiwanie lokalne i pokrewne aktywności	790
Uruchomienie funkcji type-to-search	797
Implementacja prostego dostawcy propozycji	798
Planowanie prostego dostawcy propozycji	798
Pliki implementacji prostego dostawcy propozycji	799
Implementacja klasy SimpleSuggestionProvider	799
Aktywność wyszukiwania dostępna w prostym dostawcy propozycji	803
Aktywność wywołania wyszukiwania	808
Użytkowanie prostego dostawcy propozycji	810
Implementacja niestandardowego dostawcy propozycji	813
Implementacja niestandardowego dostawcy propozycji	814
Pliki wymagane do implementacji projektu SuggestUrlProvider	814

Implementacja klasy SuggestUrlProvider	815
Implementacja aktywności wyszukiwania dla niestandardowego dostawcy propozycji	824
Plik manifest niestandardowego dostawcy propozycji	830
Korzystanie z niestandardowego dostawcy propozycji	831
Zastosowanie przycisków działania i danych wyszukiwania specyficznych dla aplikacji	835
Wykorzystanie przycisków działania w procesie wyszukiwania	835
Praca ze specyficznym dla aplikacji kontekstem wyszukiwania	838
Odnośniki	839
Wyszukiwanie w tabletach	840
Podsumowanie	840
Rozdział 24. Analiza interfejsu przetwarzania tekstu na mowę	841
Podstawy technologii przetwarzania tekstu na mowę w Androidzie	841
Używanie wyrażeń do śledzenia toku wypowiedzi	846
Zastosowanie plików dźwiękowych do przetwarzania tekstu na mowę	848
Zaawansowane funkcje silnika TTS	854
Konfiguracja strumieni audio	855
Stosowanie ikon akustycznych	855
Odtwarzanie ciszy	856
Wybór innych mechanizmów przetwarzania tekstu na mowę	856
Stosowanie metod językowych	857
Odnośniki	858
Podsumowanie	859
Rozdział 25. Ekryny dotykowe	861
Klasa MotionEvent	861
Obiekt MotionEvent	862
Wielokrotne wykorzystywanie obiektów MotionEvent	873
Stosowanie klasy VelocityTracker	874
Analiza funkcji przeciągania	876
Wielodotykowość	879
Funkcja wielodotykowości przed wersją 2.2 Androida	879
Funkcja wielodotykowości w systemach poprzedzających wersję 2.2	887
Obsługa map za pomocą dotyku	888
Gesty	891
Gest ściskania	891
Klasy GestureDetector i OnGestureListener	895
Niestandardowe gesty	898
Aplikacja Gestures Builder	898
Odnośniki	905
Podsumowanie	905

Rozdział 26. Czujniki	907
Czym jest czujnik?	907
Wykrywanie czujników	908
Jakie informacje możemy uzyskać na temat czujnika?	909
Pobieranie zdarzeń generowanych przez czujniki	911
Problemy pojawiające się podczas uzyskiwania danych z czujników ...	914
Interpretowanie danych czujnika	921
Czujniki oświetlenia	921
Czujniki zbliżeniowe	922
Termometry	922
Czujniki ciśnienia	923
Żyroskopy	923
Akcelerometry	924
Magnetometry	930
Współpraca akcelerometrów z magnetometrami	931
Czujniki orientacji w przestrzeni	931
Deklinacja magnetyczna i klasa GeomagneticField	938
Czujniki grawitacji	939
Czujniki przyśpieszenia liniowego	939
Czujniki wektora obrotu	939
Czujniki komunikacji bliskiego pola	939
Oдноśniki	950
Podsumowanie	951
Rozdział 27. Analiza interfejsu kontaktów	953
Koncepcja konta	954
Szybki przegląd ekranów związanych z kontami	954
Związek pomiędzy kontami a kontaktami	957
Wyliczanie kont	957
Aplikacja Kontakty	958
Wyświetlanie kontaktów	958
Wyświetlanie szczegółów kontaktu	959
Edytowanie szczegółów kontaktu	960
Umieszczanie zdjęcia powiązanego z kontaktem	962
Eksportowanie kontaktów	962
Różne typy danych kontaktowych	964
Analiza kontaktów	964
Badanie treści bazy danych SQLite	965
Nieprzetworzone kontakty	965
Tabela danych	967
Kontakty zbiorcze	968
view_contacts	971
contact_entities_view	971

Praca z interfejsem kontaktów	972
Eksploracja kont	972
Badanie kontaktów zbiorczych	980
Badanie nieprzetworzonych kontaktów	989
Przeglądanie danych nieprzetworzonego kontaktu	994
Dodawanie kontaktu oraz szczegółowych informacji o nim	998
Kontrola agregacji	1001
Konsekwencje synchronizacji	1002
Odnośniki	1002
Podsumowanie	1003
Rozdział 28. Wdrażanie aplikacji na rynek — Android Market i nie tylko	1005
Jak zostać wydawcą?	1006
Postępowanie zgodnie z zasadami	1006
Konsola programisty	1009
Przygotowanie aplikacji do sprzedaży	1012
Testowanie działania na różnych urządzeniach	1012
Obsługa różnych rozmiarów ekranu	1012
Przygotowanie pliku AndroidManifest.xml do umieszczenia w sklepie Android Market	1013
Lokalizacja aplikacji	1014
Przygotowanie ikony aplikacji	1015
Problemy związane z zarabianiem pieniędzy na aplikacjach	1016
Kierowanie użytkowników z powrotem do sklepu	1016
Usługa licencyjna systemu Android	1017
Przygotowanie pliku .apk do wysłania	1018
Wysyłanie aplikacji	1018
Korzystanie ze sklepu Android Market	1022
Alternatywy dla serwisu Android Market	1023
Odnośniki	1024
Podsumowanie	1024
Rozdział 29. Koncepcja fragmentów oraz inne pojęcia dotyczące tabletów	1025
Czym jest fragment?	1026
Kiedy należy stosować fragmenty?	1027
Struktura fragmentu	1027
Cykl życia fragmentu	1028
Przykładowa aplikacja ukazująca cykl życia fragmentu	1033
Klasy FragmentTransaction i drugoplanowy stos fragmentów	1042
Przejęcia i animacje zachodzące podczas transakcji fragmentu	1044
Klasa FragmentManager	1045
Ostrzeżenie dotyczące stosowania odniesień do fragmentów	1046
Klasa ListFragment i węzeł <fragment>	1047
Wywoływanie odrębnej aktywności w razie potrzeby	1051
Trwałość fragmentów	1054

Fragmety wyświetlające okna dialogowe	1054
Podstawowe informacje o klasie DialogFragment	1055
Przykładowa aplikacja wykorzystująca klasę DialogFragment	1060
Inne formy komunikowania się z fragmentami	1073
Stosowanie metod startActivity() i setTargetFragment()	1074
Tworzenie niestandardowych animacji	
za pomocą klasy ObjectAnimator	1075
Odnośniki	1078
Podsumowanie	1078
Rozdział 30. Analiza klasy ActionBar	1079
Anatomia klasy ActionBar	1080
Aktywność paska działania wyświetlającego zakładki	1081
Implementacja bazowych klas aktywności	1082
Wprowadzenie jednolitego zachowania klas ActionBar	1084
Implementacja obiektu nasłuchującego zdarzeń z zakładek	1087
Implementacja aktywności przechowującej pasek zakładek	1088
Przewijalny układ graficzny zawierający widok debugowania	1090
Pasek działania a interakcja z menu	1091
Plik manifest Androida	1093
Badanie aktywności przechowującej pasek zakładek	1093
Aktywność paska działania w trybie wyświetlania listy	1094
Utworzenie klasy SpinnerAdapter	1095
Utworzenie obiektu nasłuchującego listy	1095
Konfigurowanie paska działania w trybie wyświetlania listy	1096
Zmiany w klasie BaseActionBarActivity	1097
Zmiany w pliku AndroidManifest.xml	1097
Badanie aktywności zawierającej pasek działania	
w trybie wyświetlania listy	1098
Aktywność przechowująca standardowy pasek działania	1099
Aktywność przechowująca standardowy pasek działania	1100
Zmiany w klasie BaseActionBarActivity	1101
Zmiany w pliku AndroidManifest.xml	1101
Badanie aktywności przechowującej standardowy pasek działania	1102
Odnośniki	1102
Podsumowanie	1104
Rozdział 31. Dodatkowe zagadnienia związane z wersją 3.0 systemu	1105
Widżety ekranu startowego oparte na listach	1105
Nowe widoki zdalne w wersji 3.0 systemu	1106
Praca z listami stanowiącymi część widoku zdalnego	1107
Działający przykład	
— testowy widżet ekranu startowego oparty na liście	1121
Testowanie widżetu wyświetlającego listę	1130

Funkcja przeciągania	1131
Podstawowe informacje	
o funkcji przeciągania w wersji 3.0 Androida	1131
Przykładowa aplikacja prezentująca funkcję przeciągania	1133
Testowanie przykładowej aplikacji wykorzystującej funkcję przeciągania	1145
Odnośniki	1146
Podsumowanie	1147
Skorowidz	1149

Analiza animacji dwuwymiarowej

Animacja jest procesem pozwalającym wyświetlanemu na ekranie obiektowi na zmianę koloru, pozycji, rozmiaru lub orientacji w określonym przedziale czasowym. Animacje, które można wykorzystać w Androidzie, są bardzo praktyczne, zabawne, proste oraz są często wykorzystywane.

W wersji 2.3 i wcześniejszych Androida dostępne są trzy rodzaje animacji: animacja poklatkowa, która polega na rysowaniu serii klatek jedna po drugiej w regularnych odstępach czasu; animacja układu graficznego, w której są przetwarzane widoki osadzone w pojemniku, na przykład w tabeli na liście; a także animacja widoku, polegająca na animowaniu dowolnego widoku ogólnego przeznaczenia. Dwa ostatnie rodzaje należą do kategorii animacji klatek kluczowych (ang. *tweening*), która polega na interpolowaniu przez komputer klatek pośrednich pomiędzy klatkami kluczowymi.

Uwaga!

W wersji 3.0 Androida zmodernizowano mechanizm animacji, gdyż wprowadzono możliwość animowania elementów interfejsu użytkownika. Niektóre z nowych koncepcji, zwłaszcza dotyczące fragmentów, zostały omówione w rozdziale 29. Niniejszy rozdział ukończyliśmy przed wydaniem wersji 3.0 Androida, więc z powodu ograniczeń czasowych zajęliśmy się w nim jedynie elementami dostępnymi do wersji 2.3 systemu. W rozdziale 29. opisałeś kilka funkcji animacji dostępnych od wersji 3.0 Androida.

Animację klatek kluczowych można wyjaśnić również w taki sposób, że *nie* wymaga ona rysowania klatki po klatce. Jeżeli możemy animować obiekt bez konieczności nakładania i powtarzania kolejnych klatek, to mamy do czynienia z techniką klatek kluczowych. Jeżeli na przykład dany obiekt znajduje się w punkcie A, a za 4 sekundy znajdzie się w punkcie B, możemy zmieniać jego położenie co sekundę i za każdym razem od nowa go rysować. Będziemy odnosić wrażenie, że obiekt ten porusza się z punktu A do punktu B.

Koncepcja jest taka, że znajomość początkowych i końcowych stanów animowanego obiektu pozwala grafikowi na zmianę pewnych aspektów tego obiektu w trakcie procesu animowania. Takim aspektem może być kolor, pozycja, rozmiar albo jakiś inny element. W komputerach jest to osiągnięte poprzez zmianę średnich wartości w regularnych odstępach czasu oraz ponowne rysowanie powierzchni.

W tym rozdziale zajmiemy się zagadnieniami animacji poklatkowej, układu graficznego oraz widoku — zaprezentujemy je z wykorzystaniem działających przykładów oraz poddamy je dogłębszej analizie.

Uwaga!

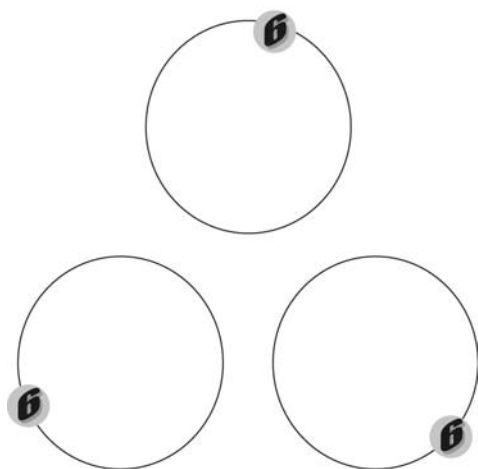
Na końcu rozdziału zamieściliśmy adres URL, z którego możemy pobrać projekty utworzone na potrzeby tego rozdziału i zaimportować je do środowiska Eclipse.

Animacja poklatkowa

Animacja jest prostym procesem polegającym na wyświetlaniu serii obrazów następujących po sobie w krótkich odstępach czasu, w wyniku czego powstaje wrażenie poruszającego lub zmieniającego się obiektu. W taki sposób działają projektory filmowe. Pokażemy przykładowy projekt, w którym zaprojektujemy obraz i zapiszemy go w formie serii oddzielnych klatek, różniących się od siebie w niewielkim stopniu. Następnie umieścimy ten zbiór obrazów w przykładowym kodzie umożliwiającym uruchomienie animacji.

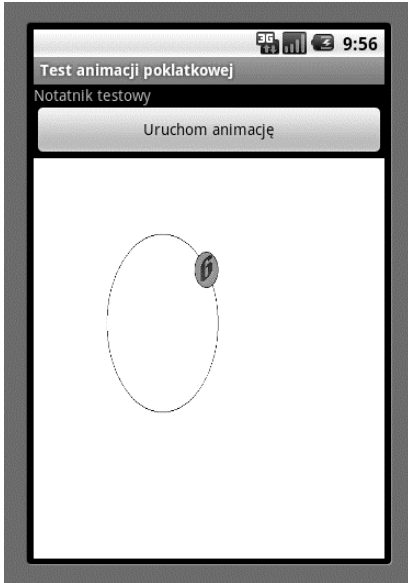
Zaplanowanie animacji poklatkowej

Przed rozpoczęciem pisania kodu należy najpierw zaplanować sekwencję animacji za pomocą rozrysowania jej na papierze. Przykład takiego planowania został zilustrowany na rysunku 16.1, przedstawiającym zbiór równowymiarowych okręgów, na których obwodach zostały umieszczone w różnych pozycjach kolorowe kółka. Można stworzyć zbiór takich rysunków przedstawiających okrąg oraz kółko umieszczone w różnych miejscach na obwodzie tego okręgu. Po zachowaniu siedmiu lub ośmiu takich klatek utworzymy animację symulującą ruch kółka po okręgu.



Rysunek 16.1. Etap projektowania animacji

Określmy sobie podstawowy człon nazwy takiego rysunku, na przykład `colored-ball`, a następnie zachowajmy utworzone rysunki w podkatalogu `/res/drawable`, żeby w przyszłości można było uzyskać do nich dostęp za pomocą identyfikatorów zasobów. Nazwa każdego pliku powinna zostać utworzona za pomocą wzoru `colored-ballN`, gdzie `N` jest numerem porządkowym klatki. Po utworzeniu animacji powinna ona wyglądać tak jak na rysunku 16.2.



Rysunek 16.2. Środowisko testowe animacji poklatkowej

Główny obszar aktywności jest wykorzystywany przez widok animacji. Wstawiliśmy przycisk uruchamiania i zatrzymywania animacji w celu obserwacji jej zachowania. W górnej części ekranu umieściliśmy również notatnik testowy, w którym można zapisywać wszelkie ważne zdarzenia podczas eksperymentowania z programem. Zobaczmy, w jaki sposób można utworzyć układ graficzny takiej aktywności.

Utworzenie aktywności

Rozpocznijmy od utworzenia prostego pliku XML układu graficznego w podkatalogu `/res/layout` (listing 16.1).

Listing 16.1. Plik XML układu graficznego do przykładu animacji poklatkowej

```
<?xml version="1.0" encoding="utf-8"?>
<!-- nazwa pliku: /res/layout/frame_animations_layout.xml -->
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<TextView android:id="@+id/textViewId1"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
```

```
        android:text="Notatnik testowy"
    />
    <Button
        android:id="@+id/startFABButtonId"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Uruchom animację"
    />
    <ImageView
        android:id="@+id/animationImage"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
    />
</LinearLayout>
```

Pierwszą kontrolką jest kontrolka notatnika testowego stanowiąca prosty widok `TextView`. Następnie dodajemy przycisk uruchamiania i zatrzymywania animacji. Ostatni jest widok `ImageView`, w którym będzie odtwarzana animacja. Po skonstruowaniu układu graficznego należy utworzyć aktywność wczytującą ten widok (listing 16.2).

Listing 16.2. Aktywność wczytująca widok `ImageView`

```
public class FrameAnimationActivity extends Activity
{
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.frame_animations_layout);
    }
}
```

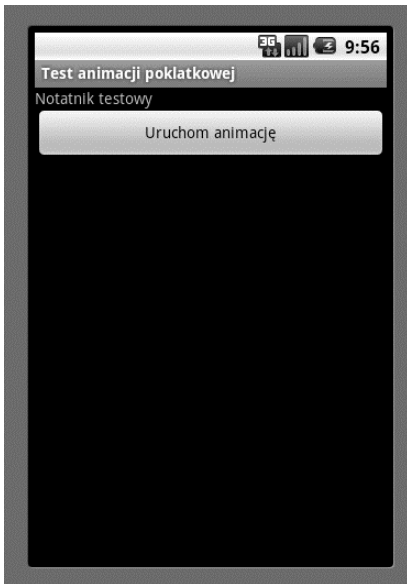
Taką aktywność będzie można uruchomić za pomocą dowolnego elementu menu, dostępnego w bieżącej aplikacji, poprzez wykonanie następującego kodu:

```
Intent intent = new Intent(inActivity, FrameAnimationActivity.class);
inActivity.startActivity(intent);
```

W tym momencie aktywność powinna wyglądać tak jak na rysunku 16.3.

Dodawanie animacji do aktywności

Po utworzeniu aktywności oraz układu graficznego pokażemy, w jaki sposób można do nich dodawać animację. W Androidzie animacja poklatkowa jest tworzona poprzez klasę `AnimationDrawable` z pakietu graficznego. Można stwierdzić po nazwie, że będzie się ona zachowywać jak każdy inny obiekt rysowany, mogący stanowić tło dla dowolnego widoku (na przykład mapy bitowe tła są reprezentowane jako elementy `Drawable`). Klasa `AnimationDrawable` poza tym, że należy do kategorii `Drawable`, może pobierać listę innych obiektów tego typu (na przykład obrazów) i wyświetlać je w określonych interwałach czasowych. W rzeczywistości klasa ta jest cienką osłoną wokół obsługi animacji zapewnianej przez bazową klasę `Drawable`.



Rysunek 16.3. Aktywność animacji poklatkowej

Wskazówka

Klasa `Drawable` uruchamia animację w ten sposób, że pojemnik lub widok wywołuje klasę `Runnable`, która w istocie przerysowuje obiekt `Drawable` za pomocą innego zestawu parametrów. Zwróćmy uwagę, że nie musimy znać takich szczegółów wewnętrznej implementacji, żeby korzystać z klasy `AnimationDrawable`. Jednak w przypadku bardziej złożonych wymagań można zajrzeć do kodu źródłowego klasy `AnimationDrawable`, aby znaleźć wskazówki do napisania własnych protokołów animacji.

Żeby móc skorzystać z klasy `AnimationDrawable`, należy najpierw umieścić zestaw zasobów typu `Drawable` (na przykład zbiór obrazów) w podkatalogu `/res/drawable`. Gwoli ścisłości, umieścimy tam osiem podobnych, lecz nie identycznych obrazów omówionych w punkcie „Zaplanowanie animacji poklatkowej”. Następnie utworzymy plik XML definiujący listę klatek (listing 16.3). Także ten plik musi zostać umieszczony w podkatalogu `/res/drawable`.

Listing 16.3.

```
<animation-list xmlns:android="http://schemas.android.com/apk/res/android"
    android:oneshot="false">
    <item android:drawable="@drawable/colored_ball1" android:duration="50" />
    <item android:drawable="@drawable/colored_ball2" android:duration="50" />
    <item android:drawable="@drawable/colored_ball3" android:duration="50" />
    <item android:drawable="@drawable/colored_ball4" android:duration="50" />
    <item android:drawable="@drawable/colored_ball5" android:duration="50" />
    <item android:drawable="@drawable/colored_ball6" android:duration="50" />
    <item android:drawable="@drawable/colored_ball7" android:duration="50" />
    <item android:drawable="@drawable/colored_ball8" android:duration="50" />
</animation-list>
```

Uwaga!

Podczas przygotowywania listy obrazów musimy pamiętać o pewnych ograniczeniach klasy `AnimationDrawable`. Przed rozpoczęciem animacji klasa ta wczytuje wszystkie obrazy do pamięci. Podczas testowania przykładowego projektu na emulatorze wyposażonym w wersję systemu 2.3 liczba klatek większa od 6 przekraczała pojemność pamięci przydzielonej dla aplikacji. W zależności od środowiska testowego być może będziemy musieli ograniczyć liczbę klatek. Aby rozwiązać ten problem, musimy bezpośrednio skorzystać z funkcji animacyjnych klasy `Drawable` i wprowadzić własny mechanizm. Niestety, klasa `Drawable` nie została szczegółowo omówiona w tym wydaniu książki. Proponujemy wizytę na stronie www.androidbook.com, gdyż planujemy zaktualizować jej zawartość w niedługim czasie.

Każda klatka wskazuje na jeden z rysunków określony przez jego identyfikator zasobu. Znacznik `animation-list` zostaje przekształcony do obiektu `AnimationDrawable`, reprezentującego zbiór obrazów. Musimy teraz umieścić klasę `Drawable` jako zasób tła dla widoku `ImageView`. Zakładając, że nazwaliśmy ten plik `frame_animation.xml` i umieściliśmy go w podkatalogu `/res/drawable`, możemy zastosować poniższy kod do ustanowienia klasy `AnimationDrawable` jako tła widoku `ImageView`:

```
view.setBackgroundResource(Resource.drawable.frame_animation);
```

Dzięki tej linii kodu Android rozpoznaje identyfikator zasobu `Resource.drawable.frame_animation` jako zasób XML i zgodnie z nim tworzy odpowiedni obiekt Java `AnimationDrawable`, zanim ustawi go jako tło. Gdy już będziemy mieli tło, możemy uzyskać dostęp do tego obiektu `AnimationDrawable` poprzez wprowadzenie instrukcji `get` do widoku `View` w następujący sposób:

```
Object backgroundObject = view.getBackground();  
AnimationDrawable ad = (AnimationDrawable)backgroundObject;
```

Po umieszczeniu obiektu klasy `AnimationDrawable` możemy wprowadzić metody `start()` i `stop()` służące do uruchamiania i zatrzymywania animacji. Poniżej zaprezentowaliśmy dwie inne istotne metody tego obiektu:

```
setOneShot();  
addFrame(drawable, duration);
```

Metoda `setOneShot()` odtwarza animację jeden raz i potem ją zatrzymuje. Metoda `addFrame()` dodaje nową klatkę za pomocą obiektu `Drawable` i konfiguruje czas jej wyświetlania. Działanie tej metody przypomina funkcję znacznika XML `android:drawable`.

Teraz musimy złożyć wszystkie fragmenty kodu w całość, aby otrzymać środowisko testowe animacji poklatkowej (listing 16.4).

Listing 16.4. Pełny kod środowiska testowego animacji poklatkowej

```
public class FrameAnimationActivity extends Activity {  
    @Override  
    public void onCreate(Bundle savedInstanceState)  
    {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.frame_animations_layout);  
        this.setupButton();  
    }  
    private void setupButton()  
    {
```



```
Button b = (Button)this.findViewById(R.id.startFABButtonId);
b.setOnClickListener(
    new Button.OnClickListener(){
        public void onClick(View v)
        {
            parentButtonClicked(v);
        }
    });
}
private void parentButtonClicked(View v)
{
    animate();
}
private void animate()
{
    ImageView imgView = (ImageView)findViewById(R.id.animationImage);
    imgView.setVisibility(ImageView.VISIBLE);
    imgView.setBackgroundResource(R.drawable.frame_animation);

    AnimationDrawable frameAnimation =
        (AnimationDrawable) imgView.getBackground();

    if (frameAnimation.isRunning())
    {
        frameAnimation.stop();
    }
    else
    {
        frameAnimation.start();
    }
}
}
} //eof-class
```

Metoda `animate()` lokalizuje widok `ImageView` w bieżącej aktywności i przypisuje mu tło `AnimationDrawable` rozpoznane przez zasób `R.drawable.frame_animation`. Następnie kod odczytuje ten obiekt i przeprowadza proces animowania. Przycisk uruchamiania i zatrzymywania jest skonfigurowany w ten sposób, że jego naciśnięcie w trakcie odtwarzania animacji zatrzyma ją; jeżeli animacja jest zatrzymana, jego naciśnięcie spowoduje jej uruchomienie.

Zauważmy, że jeśli przypiszemy parametrowi listy animacji `OneShot` wartość `true`, animacja wykona tylko jeden cykl. Jednak nie można dokładnie przewidzieć, kiedy się to stanie. Chociaż animacja zostaje zakończona po wyświetleniu ostatniego obrazu, nie otrzymamy żadnego informującego o tym komunikatu. Z tego powodu nie istnieje żaden bezpośredni sposób wywołania kolejnej czynności w odpowiedzi na zakończenie animacji.

Pomimo tej niedogodności można uzyskać wspaniałe efekty wizualne poprzez wyświetlanie po kolei serii obrazów w prostym procesie animacji poklatkowej.

Animacja układu graficznego

Podobnie jak w przypadku animacji poklatkowej, animacja układu graficznego jest bardzo prosta. Jak sama nazwa wskazuje, animacja tego typu jest przeznaczona dla pewnego rodzaju widoków,

ułożonych w określony sposób. Stosowana jest ona na przykład w przypadku widoków `ListView` oraz `GridView`, które są dwiema powszechnie implementowanymi kontrolkami w systemie Android. W szczególności animacja układu graficznego jest używana do dodawania efektów graficznych, zmieniających sposób wyświetlania elementów umieszczonych w wymienionych widokach. Tak naprawdę może być ona stosowana wobec wszystkich kontrolki wywodzących się z klasy `ViewGroup`.

W przeciwieństwie do animacji poklatkowej, animacja układu graficznego nie jest generowana poprzez powtarzanie klatek. Zamiast tego są zmieniane w czasie różne właściwości widoku. Każdy widok w Androidzie zawiera macierz transformacji, która odwzorowuje widok wyświetlony na ekranie. Poprzez zmianę takiej macierzy na różne sposoby można przeprowadzić procesy skalowania, obracania i przemieszczania (translacji) tego widoku. Na przykład poprzez zmianę przezroczystości widoku w skali od 0 do 1 otrzymujemy tak zwaną animację typu `alpha`.

Podstawowe typy animacji klatek kluczowych

Poniżej prezentujemy nieco bardziej szczegółowo podstawowe rodzaje animacji klatek kluczowych (ang. *tweening*):

- **Animacja skali.** Ten typ animacji umożliwia powiększanie lub zmniejszanie widoku w osi *x* oraz w osi *y*. Można także określić punkt zwrotny, wokół którego będzie odtwarzana animacja.
- **Animacja rotacyjna.** Dzięki niej można obracać widok wokół punktu zwrotnego o określony kąt.
- **Animacja translacyjna.** Służy ona do przesuwania widoku wzdłuż osi *x* lub *y*.
- **Animacja typu alfa.** Służy do zmieniania przezroczystości widoku.

Animacje tego typu są definiowane w postaci plików XML umieszczonych w podkatalogu `/res/anim`. Listing 16.5 prezentuje krótki przykład, pomagający zrozumieć, w jaki sposób te animacje są definiowane.

Listing 16.5. Animacja skali zdefiniowana w pliku XML, umieszczonym w podkatalogu `/res/anim/scale.xml`

```
<set xmlns:android="http://schemas.android.com/apk/res/android"
    android:interpolator="@android:anim/accelerate_interpolator">
    <scale
        android:fromXScale="1"
        android:toXScale="1"
        android:fromYScale="0.1"
        android:toYScale="1.0"
        android:duration="500"
        android:pivotX="50%"
        android:pivotY="50%"
        android:startOffset="100" />
</set>
```

Wszystkie wartości parametrów w tym pliku animacji zostają określone „od – do”, ponieważ musimy określić wartości początkowe i końcowe animacji.

Każda z animacji dopuszcza również możliwość korzystania z interpolatorów czasu w postaci argumentów. Interpolatory zostaną omówione na końcu podrozdziału związanego z animacją układu graficznego, teraz jednak wystarczy wiedzieć, że są one odpowiedzialne za szybkość zmian argumentów w trakcie przetwarzania animacji.

Po utworzeniu tego pliku deklarowania animacji możemy powiązać animację z układem graficznym, dzięki czemu elementy składowe układu graficznego będą animowane.

Uwaga!

W tym miejscu warto wspomnieć, że każda z tych animacji jest reprezentowana jako klasa Java w pakiecie `android.view.animation`. Dokumentacja każdej z tych klas nie tylko opisuje jej metody języka Java, lecz również dopuszczalne argumenty XML dla każdego typu animacji.

Skoro już naszkicowaliśmy zarys rodzajów animacji układu graficznego wystarczający do ich chociażby podstawowego zrozumienia, zajmijmy się projektowaniem przykładu.

Zaplanowanie środowiska testowego animacji układu graficznego

Za pomocą prostego zestawu `ListView` w aktywności można przetestować wszystkie omówione przez nas koncepcje animacji układu graficznego. Po utworzeniu widoku `ListView` można do niego dołączyć animację, co spowoduje jej przetworzenie wobec każdego elementu tego widoku.

Załóżmy, że chcemy utworzyć animację skali, która powiększa widok od zera do oryginalnego rozmiaru w osi *y*. Możemy to sobie wyobrazić wizualnie jako linijkę tekstu, która najpierw przypomina poziomą linię, a następnie zostaje powiększona do właściwego rozmiaru czcionki.

Można tę animację dołączyć do widoku `ListView`. Kiedy to zrobimy, każdy element tej listy będzie wyświetlany za pomocą tej animacji.

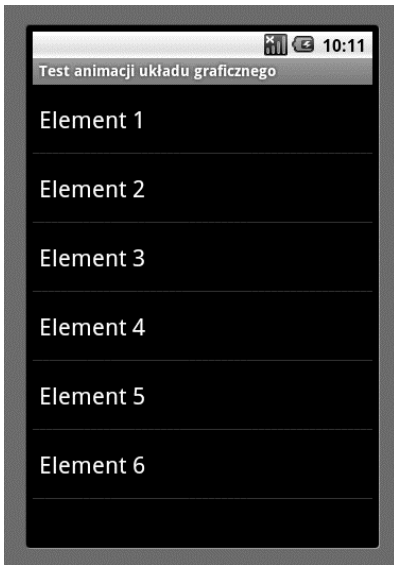
Możemy dodać kilka parametrów, które urozmaicą podstawową animację, na przykład animowanie listy od góry do dołu lub odwrotnie. Parametry te są definiowane w klasie pośredniej, zachowującej się jak mediator pomiędzy konkretnym plikiem XML animacji a widokiem listy.

Istnieje możliwość zdefiniowania zarówno animacji, jak i mediatora w pliku XML umieszczonym w podkatalogu `/res/anim`. Gdy już utworzymy taki pośredniczący plik XML, możemy go wykorzystać w postaci danych wejściowych dla widoku `ListView` w jego własnym pliku definicji XML. Gdy ta podstawowa konfiguracja będzie już działać, będziemy zmieniać animacje, żeby przekonać się, w jaki sposób wpływają one na wyświetlanie elementów widoku `ListView`.

Zanim rozpoczniemy ćwiczenie, przyjrzymy się, jak widok `ListView` będzie wyglądał po zakończeniu animacji (rysunek 16.4).

Utworzenie aktywności oraz widoku `ListView`

Rozpoczniemy od utworzenia układu graficznego XML dla widoku `ListView` przedstawionego na rysunku 16.4, dzięki czemu możliwe będzie wczytanie tego układu graficznego w prostej aktywności. Na listingu 16.6 został umieszczony taki nieskomplikowany układ graficzny z zaimplementowanym widokiem `ListView`. Taki plik należy umieścić w podkatalogu `/res/layout`. Zakładając, że nazwa pliku brzmi `list_layout.xml`, kompletna ścieżka do niego będzie wyglądała następująco: `/res/layout/list_layout.xml`.



Rysunek 16.4. Animowana lista ListView

Listing 16.6. Plik XML układu graficznego definiujący widok ListView

```
<?xml version="1.0" encoding="utf-8"?>
<!-- nazwa pliku: /res/layout/list_layout.xml -->
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >

    <ListView
        android:id="@+id/list_view_id"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        />
</LinearLayout>
```

Listing 16.6 przedstawia prosty menedżer `LinearLayout` z umieszczonym wewnątrz niego prostym widokiem `ListView`. Powinniśmy jednak skorzystać z okazji i wyjaśnić pewną rzecz dotyczącą definicji widoku `ListView`, która jest dość marginalnie powiązana z treścią rozdziału. Jeżeli Czytelnik będzie pracował na aplikacji Notepad lub innych przykładowych programach, zauważy zapewne, że identyfikator widoku `ListView` jest przeważnie określany jako `@android:id/list`. Zgodnie z informacjami z rozdziału 3. odniesienie `@android:id/list` wskazuje na identyfikator predefiniowany w przestrzeni nazw `android`. Pytanie brzmi: kiedy należy stosować odniesienie `android:id`, a kiedy nasz własny identyfikator, na przykład `@+id/list_view_id`?

Identyfikatora `@android:id/list` używamy jedynie w przypadku, gdy aktywnością jest `ListActivity`. W przypadku tej aktywności zakłada się, że widok `ListView`, określony przez ten predefiniowany identyfikator, jest dostępny do wczytania. W tym wypadku używamy raczej

aktywności ogólnego przeznaczenia, a nie `ListActivity`, i musimy własnoręcznie wypełnić w jawny sposób widok `ListView`. W związku z tym nie ma żadnych ograniczeń co do rodzaju identyfikatora, który ma reprezentować tę listę. Jednak można także wykorzystać odniesienie `@android:id/list`, ponieważ nie stwarza to żadnego konfliktu z powodu braku aktywności `ListActivity`.

To taka mała dygresja, warto jednak o niej pamiętać podczas tworzenia własnych widoków `ListView` poza aktywnością `ListActivity`. Gdy już posiadamy układ graficzny wymagany dla aktywności, możemy napisać kod odpowiedzialny za wczytanie tego pliku układu graficznego, dzięki czemu zostanie wygenerowany interfejs użytkownika (listing 16.7).

Listing 16.7. Kod aktywności odpowiedzialnej za animację układu graficznego

```
public class LayoutAnimationActivity extends Activity
{
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.list_layout);
        setupListView();
    }
    private void setupListView()
    {
        String[] listItems = new String[] {
            " Element 1", " Element 2", " Element 3",
            " Element 4", " Element 5", " Element 6",
        };

        ArrayAdapter listItemAdapter =
            new ArrayAdapter(this
                ,android.R.layout.simple_list_item_1
                ,listItems);
        ListView lv = (ListView)this.findViewById(R.id.list_view_id);
        lv.setAdapter(listItemAdapter);
    }
}
```

Niektóre fragmenty kodu widocznego na listingu 16.7 są oczywiste, ale inne nie. Pierwsza część kodu w zwykły sposób wczytuje widok na podstawie wygenerowanego identyfikatora układu graficznego `R.layout.list_layout`. Naszym zadaniem jest wypełnienie widoku `ListView` z tego układu graficznego sześcioma elementami. Te elementy tekstowe zostały wczytane do tablicy. Musimy ustanowić adapter danych wobec widoku `ListView`, żeby te elementy mogły zostać wyświetlone.

Aby utworzyć wymagany adapter, musimy określić, w jaki sposób każdy z elementów będzie wstawiany podczas wyświetlania listy na ekranie. Układ graficzny określamy za pomocą predefiniowanego układu, znajdującego się w strukturze Androida. W naszym przykładzie układ graficzny wyznaczono następująco:

```
android.R.layout.simple_list_item_1
```

Innymi dostępnymi układami graficznymi widoku dla tych elementów są:

```
simple_list_item_2
simple_list_item_checked
simple_list_item_multiple_choice
simple_list_item_single_choice
```

Można zajrzeć do dokumentacji Androida, aby się dowiedzieć, jak te układy graficzne wyglądają i jak się zachowują. Teraz możemy wywołać tę aktywność za pomocą dowolnego przycisku menu w aplikacji po wstawieniu następującego kodu:

```
Intent intent = new Intent(inActivity, LayoutAnimationActivity.class);
inActivity.startActivity(intent);
```

Jednak — podobnie jak w przypadku wywołań innych aktywności — musimy zarejestrować aktywność `LayoutAnimationActivity` w pliku `AndroidManifest.xml`, jeżeli powyższe wywołanie aktywności ma zadziałać. Poniżej umieściliśmy potrzebny do tego kod:

```
<activity android:name=".LayoutAnimationActivity"
android:label="Testowa aktywność widoku animacji"/>
```

Animowanie widoku ListView

Po przygotowaniu środowiska testowego (listingi 16.6 i 16.7) Czytelnik dowie się, w jaki sposób wstawiać animację skali do widoku `ListView`. Spójrzmy, jak animacja ta zostaje zdefiniowana w pliku XML (listing 16.8).

Listing 16.8. Definiowanie animacji skali w pliku XML

```
<set xmlns:android="http://schemas.android.com/apk/res/android"
android:interpolator="@android:anim/accelerate_interpolator">
  <scale
    android:fromXScale="1"
    android:toXScale="1"
    android:fromYScale="0.1"
    android:toYScale="1.0"
    android:duration="500"
    android:pivotX="50%"
    android:pivotY="50%"
    android:startOffset="100" />
</set>
```

Jak już wcześniej wspomnieliśmy, pliki definiujące animacje są przechowywane w podkatalogu `/res/anim`.

Przetłumaczymy te atrybuty XML na język polski.

Wagi `from` i `to` są wskaźnikami początku oraz zakończenia procesu powiększania. W naszym wypadku powiększanie rozpoczyna się od wartości 1 i takie pozostaje dla osi x . Oznacza to, że element nie będzie powiększany ani zmniejszany w tej osi.

Jednak w przypadku osi y powiększanie rozpoczyna się od wartości 0.1 i dąży do 1.0. Innymi słowy, na początku animacji rozmiar obiektu stanowi jedną dziesiątą jego naturalnego rozmiaru, do którego dąży w czasie trwania animacji.

Cała operacja skalowania zajmie 500 milisekund.

Środek działania znajduje się w połowie drogi obydwu osi (50%).

Wartość `startOffset` odnosi się do czasu (wyrażonego w milisekundach), po którym animacja zostanie uruchomiona.

Węzeł nadrzędny animacji skali wskazuje na zestaw animacji, który dopuszcza wprowadzenie większej liczby animacji. Omówimy również tego rodzaju przykład. Na razie jednak mamy do dyspozycji tylko jedną animację w zestawie.

Nazwijmy ten plik `scale.xml` i umieśćmy go w podkatalogu `/res/anim`. Nie jesteśmy na razie gotowi, żeby wstawić ten plik XML animacji jako argument w widoku `ListView`; widok ten wymaga jeszcze jednego pliku XML, który będzie zachowywał się jak pośrednik pomiędzy widokiem a zestawem animacji. Kod pliku XML, w którym zaimplementowane jest takie powiązanie, został pokazany na listingu 16.9.

Listing 16.9. Definicja dla pliku XML stanowiącego kontroler układu graficznego

```
<layoutAnimation xmlns:android="http://schemas.android.com/apk/res/android"
    android:delay="30%"
    android:animationOrder="reverse"
    android:animation="@anim/scale" />
```

Również ten plik należy umieścić w podkatalogu `/res/anim`. W naszym przykładzie zakładamy, że plik nosi nazwę `list_layout_controller`. Po przyjrzeniu się definicji pliku pośredniczącego zrozumiemy, dlaczego jest on niezbędny.

W pliku tym zostaje określone, że animacja tej listy powinna przebiegać w odwrotnym porządku oraz że animacja każdego elementu będzie opóźniona o 30% względem całkowitego czasu trwania animacji. Znajduje się tu również odniesienie do pliku animacji — `scale.xml`. Zauważmy również, że w kodzie jest użyte odniesienie do tego pliku `@anim/scale` zamiast jego nazwy.

Gdy już posiadamy wymagane pliki XML z danymi wejściowymi, pokażemy, w jaki sposób należy zaktualizować definicję XML widoku `ListView`, żeby obejmowała ona animację XML jako argument. Najpierw przejrzymy dotychczas utworzone pliki XML:

```
// pojedyncza animacja skali
/res/anim/scale.xml
```

```
// plik pośredniczący
/res/anim/list_layout_controller.xml
```

```
// plik układu graficznego widoku aktywności
/res/layout/list_layout.xml
```

Gdy te pliki są gotowe, musimy zmodyfikować plik XML układu graficznego `list_layout.xml` w taki sposób, żeby widok `ListView` wskazywał plik `list_layout_controller.xml` (listing 16.10).

Listing 16.10. Zaktualizowany kod pliku `List_Layout.xml`

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <ListView
```

```
android:id="@+id/list_view_id"  
android:persistentDrawingCache="animation|scrolling"  
android:layout_width="fill_parent"  
android:layout_height="fill_parent"  
android:layoutAnimation="@anim/list_layout_controller" />  
/>
```

```
</LinearLayout>
```

Zmienione wiersze zostały wyróżnione pogrubioną czcionką. Kluczowym znacznikiem jest `android:layoutAnimation`, który wskazuje pośredniczący plik XML definiujący kontroler układu graficznego za pomocą znacznika `layoutAnimation` (listing 16.9). Z kolei znacznik `layoutAnimation` odnosi się do animacji, w naszym wypadku animacji skali zdefiniowanej w pliku *scale.xml*.

Android zaleca także wstawienie znacznika `persistentDrawingCache`, który optymalizuje animację i przesuwanie. Więcej informacji na jego temat można znaleźć w dokumentacji środowiska Android SDK.

Po zaktualizowaniu pliku *list_layout.xml* zgodnie z listingiem 16.10 wtyczka ADT środowiska Eclipse automatycznie przekompiluje pakiet, uwzględniając wprowadzone zmiany. Gdybyśmy teraz uruchomili aplikację, zobaczylibyśmy, że animacja skali jest przeprowadzana na każdym elemencie. Zdefiniowaliśmy czas trwania animacji na 500 milisekund, zatem ujrzemy wyraźnie zmianę skali podczas rysowania obiektu.

Możemy już eksperymentować z innymi rodzajami animacji. Sprawdźmy teraz animację typu alfa. W tym celu utworzymy plik */res/anim/alpha.xml* i umieścimy w nim treść listingu 16.11.

Listing 16.11. Plik *alpha.xml* do testowania animacji typu alfa

```
<alpha xmlns:android="http://schemas.android.com/apk/res/android"  
    android:interpolator="@android:anim/accelerate_interpolator"  
    android:fromAlpha="0.0" android:toAlpha="1.0" android:duration="1000" />
```

Animacja typu alfa jest odpowiedzialna za kontrolę zmiany nasycenia kolorów. W tym przykładzie w ciągu 1000 milisekund (1 sekundy) kolor z przezroczystego staje się w pełni nasycony. Dobrze jest ustawić czas trwania animacji na co najmniej 1 sekundę, w przeciwnym wypadku zmiana nasycenia będzie trudna do zaobserwowania.

W przypadku zmiany animacji pojedynczego elementu musimy zmienić również treść pliku pośredniczącego (listing 16.9), żeby wskazywała plik z nową animacją. Poniżej pokazaliśmy sposób wskazywania z animacji skali na animację typu alfa:

```
<layoutAnimation xmlns:android="http://schemas.android.com/apk/res/android"  
    android:delay="30%"  
    android:animationOrder="reverse"  
    android:animation="@anim/alpha" />
```

Zmieniony wiersz w tym kodzie wyróżniono pogrubioną czcionką. Spróbujmy teraz stworzyć animację łączącą zmianę położenia ze zmianą gradientu nasycenia koloru. Listing 16.12 przedstawia przykładowy kod takiej animacji.

Listing 16.12. Połączenie animacji translacyjnej z animacją typu alfa w zestawie animacji

```
<set xmlns:android="http://schemas.android.com/apk/res/android"
  android:interpolator="@android:anim/accelerate_interpolator">
  <translate android:fromYDelta="-100%" android:toYDelta="0"
  android:duration="500" />
  <alpha android:fromAlpha="0.0" android:toAlpha="1.0"
  android:duration="500" />
</set>
```

Zwróćmy uwagę, w jaki sposób określiliśmy dwie animacje w zestawie animacji. Animacja translacyjna będzie przesuwała tekst z góry na dół w wydzielonym dla niego obszarze wyświetlania. Animacja typu alfa będzie powodować zmianę gradientu nasycenia koloru od przezroczystego do całkowicie nasyconego podczas przesuwania tekstu w dół. Wartość 500 czasu trwania animacji pozwoli użytkownikowi obserwować w wygodny sposób zmianę. Oczywiście znowu będzie trzeba zmienić plik pośredniczący `layoutAnimation`, tak żeby znalazło się w nim odniesienie do nowego pliku. Zakładając, że nazwą pliku zawierającego połączone animacje jest `/res/anim/translate-alpha.xml`, plik `layoutAnimation` będzie wyglądał następująco:

```
<layoutAnimation xmlns:android="http://schemas.android.com/apk/res/android"
  android:delay="30%"
  android:animationOrder="reverse"
  android:animation="@anim/translate-alpha" />
```

Zobaczmy, w jaki sposób można używać animacji rotacyjnej (listing 16.13).

Listing 16.13. Plik XML animacji rotacyjnej

```
<rotate xmlns:android="http://schemas.android.com/apk/res/android"
  android:interpolator="@android:anim/accelerate_interpolator"
  android:fromDegrees="0.0"
  android:toDegrees="360"
  android:pivotX="50%"
  android:pivotY="50%"
  android:duration="500" />
```

Kod z listingu 16.13 spowoduje wykonanie jednego pełnego obrotu przez każdy element tekstowy wokół środka tego elementu. Czas trwania 500 milisekund całkowicie wystarczy, żeby obserwator dostrzegł animację. Podobnie jak w poprzednich przypadkach, tak i teraz muszą zostać zmodyfikowane pliki XML kontrolera animacji oraz układu graficznego `ListView`, a aplikacja musi zostać ponownie uruchomiona, żeby animacja zadziałała.

Omówiliśmy już podstawowe pojęcia dotyczące animacji układu graficznego, począwszy od prostego pliku animacji, a skończywszy na powiązaniu go poprzez plik pośredniczący `layoutAnimation` z widokiem `ListView`. Ta wiedza wystarczy, żeby ujrzeć animowane efekty. Musimy omówić jednak jeszcze jedno pojęcie dotyczące animacji układu graficznego — interpolatory.

Stosowanie interpolatorów

Interpolatory określają, w jaki sposób dana właściwość, na przykład gradient koloru, zmienia się względem czasu. Czy będzie się ona zmieniała w sposób liniowy, czy w sposób wykładniczy? Czy rozpocznie się szybko, lecz będzie zwalniała z biegiem czasu? Zastanówmy się nad przykładem animacji typu alfa z listingu 16.11:

```
<alpha xmlns:android="http://schemas.android.com/apk/res/android"
    android:interpolator="@android:anim/accelerate_interpolator"
    android:fromAlpha="0.0" android:toAlpha="1.0" android:duration="1000" />
```

Animacja rozpoznaje zastosowany interpolator — w tym przypadku `accelerate_interpolator`. Istnieje odpowiedni obiekt Java, służący do definiowania tego interpolatora. Poza tym zwróćmy uwagę, że określiliśmy ten interpolator jako odniesienie do zasobów. Oznacza to, że musi istnieć plik odpowiadający identyfikatorowi `anim/accelerate_interpolator`, w którym opisany jest ten obiekt języka Java oraz jego dodatkowe parametry. Tak jest w istocie. Przyjrzyjmy się definicji pliku XML, do którego odniesieniem jest identyfikator `@android:anim/accelerate_interpolator`:

```
<accelerateInterpolator
xmlns:android="http://schemas.android.com/apk/res/android"
factor="1" />
```

Plik ten można odnaleźć w następującym podkatalogu pakietu Android:

```
/res/anim/accelerate_interpolator.xml
```

Znacznik XML `accelerateInterpolator` odpowiada następującemu obiektowi środowiska Java:

```
android.view.animation.AccelerateInterpolator
```

W dokumentacji języka Java dotyczącej tej klasy można zobaczyć, jakie znaczniki XML są dla niej dostępne. Zadaniem tego interpolatora jest zapewnienie współczynnika powielania danego przedziału czasowego w oparciu o krzywą hiperboliczną. Widać to w kodzie źródłowym interpolatora:

```
public float getInterpolation(float input)
{
    if (mFactor == 1.0f)
    {
        return (float)(input * input);
    }
    else
    {
        return (float)Math.pow(input, 2 * mFactor);
    }
}
```

Każdy interpolator w inny sposób implementuje metodę `getInterpolation`. W naszym przypadku, jeśli interpolator zostanie skonfigurowany tak, że współczynnik będzie wynosił 1.0, zostanie zwrócony kwadrat tego współczynnika. W przeciwnym razie zostanie zwrócona potęga danych wejściowych, które będą nadal skalowane przez ten współczynnik. Jeżeli zatem wartość współczynnika będzie wynosiła 1.5, zamiast funkcji kwadratowej ujrzymy funkcję sześcienną.

Poniżej wypisaliśmy listę obsługiwanych interpolatorów:

```
AccelerateDecelerateInterpolator
AccelerateInterpolator
CycleInterpolator
DecelerateInterpolator
LinearInterpolator
AnticipateInterpolator
AnticipateOvershootInterpolator
BounceInterpolator
OvershootInterpolator
```

Żeby zaprezentować potencjalną elastyczność interpolatorów, przyjrzyjmy się pokrótce obiektowi `BounceInterpolator`, powodującemu „podskakiwanie” elementu (to znaczy jego naprzemienny ruch w górę i w dół) do samego końca poniższej animacji:

```
public class BounceInterpolator implements Interpolator {
    private static float bounce(float t) {
        return t * t * 8.0f;
    }

    public float getInterpolation(float t) {
        t *= 1.1226f;
        if (t < 0.3535f) return bounce(t);
        else if (t < 0.7408f) return bounce(t - 0.54719f) + 0.7f;
        else if (t < 0.9644f) return bounce(t - 0.8526f) + 0.9f;
        else return bounce(t - 1.0435f) + 0.95f;
    }
}
```

Zachowanie tych interpolatorów zostało omówione pod poniższym adresem:

<http://developer.android.com/reference/android/view/animation/package-summary.html>

W dokumentacji języka Java wymienione są również znaczniki XML, pozwalające na kontrolowanie każdej z tych klas. Jednak z dokumentacji trudno wywnioskować przeznaczenie każdego typu interpolatora. Najlepiej jest samemu wypróbować wszystkie interpolatory i sprawdzić skutki ich działania. Pod poniższym adresem można również przejrzeć kod źródłowy:

<http://android.git.kernel.org/?p=platform%2Fframeworks%2Fbase.git&a=search&h=HEAD&st=grep&s=BounceInterpolator>

Na tym zakończymy wywody poświęcone animacji układu graficznego. Przejdziemy teraz do trzeciej części animowania, poświęconej programowaniu animacji widoku.

Animacja widoku

Skoro zapoznaliśmy się już z animacją poklatkową oraz animacją układu graficznego, możemy zająć się animacją widoku — najbardziej skomplikowanym rodzajem animacji. Stosowana jest w niej technika animowania dowolnego widoku poprzez kontrolowanie macierzy transformacji, służącej do wyświetlania widoku.

Animacja widoku

Widok wyświetlany przez Androida przechodzi przez macierz transformacji. W aplikacjach graficznych macierze transformacji służą do przekształcenia w jakiś sposób widoku. Proces ten polega na przetłumaczeniu wejściowego zestawu współrzędnych pikseli i kombinacji kolorów na nowy zestaw. Po przeprowadzeniu transformacji ujrzymy obraz zmieniony pod względem rozmiaru, pozycji, orientacji lub koloru.

Te przekształcenia można przeprowadzić za pomocą aparatu matematycznego, mnożąc w określony sposób wejściowy zestaw współrzędnych przez wartości macierzy transformacji, dzięki czemu powstanie nowy zestaw współrzędnych. Poprzez zmianę macierzy transformacji wpływamy na wygląd widoku.

Macierz, która *nie* zmienia widoku podczas tego mnożenia, nazywana jest macierzą jednostkową. Transformację przeważnie rozpoczynamy od macierzy jednostkowej i kolejno wprowadzamy serie transformacji rozmiaru, pozycji i orientacji. Następnie za pomocą macierzy końcowej rysujemy widok.

Android odsłania taką macierz transformacji widoku poprzez umożliwienie zarejestrowania obiektu animacji wobec tego widoku. Obiekt animacji będzie posiadał procedurę wywołania, dzięki której uzyska dostęp do tej macierzy i w określony sposób zmieni jej wartości, co pociągnie za sobą zmianę wyświetlania widoku. Zajmiemy się teraz tym procesem.

Rozpocznijmy tworzenie przykładowego projektu od zaplanowania animacji widoku. Na początek zapełnimy aktywność kilkoma elementami w widoku `ListView`, podobnie jak miało to miejsce w podrozdziale „Animacja układu graficznego”. Następnie w górnej części ekranu umieścimy przycisk powodujący uruchomienie animacji `ListView` (rysunek 16.5). Widoczne są zarówno lista elementów, jak i przycisk, żadna animacja nie została jednak jeszcze uruchomiona. Do tego będzie służył utworzony przycisk.



Rysunek 16.5. Aktywność animacji widoku

Po kliknięciu przycisku *Uruchom animację* powinien się pojawić mały widok pośrodku ekranu, który następnie stopniowo będzie się powiększał aż do wypełnienia zarezerwowanej dla niego przestrzeni. Zaprezentujemy kod, który nam to umożliwi. Na listingu 16.14 został pokazany kod pliku XML układu graficznego, nadający się do zastosowania w aktywności.

Listing 16.14. Plik XML układu graficznego dla aktywności animacji widoku

```
<?xml version="1.0" encoding="utf-8"?>
<!-- Ten plik jest umieszczony w /res/layout/list_layout.xml -->
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
```

```

<Button
    android:id="@+id/btn_animate"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Uruchom animację"
/>
<ListView
    android:id="@+id/list_view_id"
    android:persistentDrawingCache="animation|scrolling"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
/>
</LinearLayout>

```

Pogrubiona czcionka ma zwrócić uwagę Czytelnika na lokalizację oraz nazwę pliku. Ten układ graficzny składa się z dwóch części: pierwsza z nich to przycisk *btn_animate*, służący do uruchomienia animacji widoku; drugą jest widok *ListView*, w naszym przypadku nazwany *list_view_id*.

Skoro mamy już układ graficzny dla aktywności, możemy utworzyć samą aktywność, żeby wyświetlić widok i skonfigurować przycisk *Uruchom animację* (listing 16.15).

Listing 16.15. Kod dla aktywności animacji widoku przed rozpoczęciem animacji

```

public class ViewAnimationActivity extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.list_layout);
        setupListView();
        this.setupButton();
    }
    private void setupListView()
    {
        String[] listItems = new String[] {
            "Element 1", "Element 2", "Element 3",
            "Element 4", "Element 5", "Element 6",
        };
        ArrayAdapter listItemAdapter =
            new ArrayAdapter(this
                , android.R.layout.simple_list_item_1
                , listItems);
        ListView lv = (ListView)this.findViewById(R.id.list_view_id);
        lv.setAdapter(listItemAdapter);
    }
    private void setupButton()
    {
        Button b = (Button)this.findViewById(R.id.btn_animate);
        b.setOnClickListener(
            new Button.OnClickListener(){
                public void onClick(View v)
                {

```

```
        //animateListView();
    }
});
}
```

Kod przedstawiony dla aktywności animacji widoku z listingu 16.15 bardzo przypomina kod aktywności animacji układu graficznego z listingu 16.7. W podobny sposób wczytaliśmy widok i wstawiliśmy sześć elementów tekstowych do widoku `ListView`. Skonfigurowaliśmy przycisk w taki sposób, żeby wywoływał metodę `animateListView()` po kliknięciu. Na razie jednak oznaczymy ten fragment komunikatem, dopóki nasz przykład nie zadziała.

Aktywność możemy wywołać tuż po jej zarejestrowaniu w pliku *AndroidManifest.xml*:

```
<activity android:name=".ViewAnimationActivity"
    android:label="Aktywność testowa animacji widoku">
```

Po przeprowadzeniu procesu rejestracji możemy wywołać aktywność animacji widoku za pomocą dowolnego przycisku menu w aplikacji, korzystając z poniższego fragmentu kodu:

```
Intent intent = new Intent(this, ViewAnimationActivity.class);
startActivity(intent);
```

Po uruchomieniu programu pojawi się ekran pokazany na rysunku 16.5.

Dodawanie animacji

W tym ćwiczeniu naszym celem jest dodanie animacji do widoku `ListView`, widocznego na rysunku 16.5. W tym celu potrzebujemy klasy wywodzącej się z pakietu `android.view.animation.Animation`. Następnie musimy przesłonić metodę `applyTransformation`, aby można było zmodyfikować macierz transformacji. Nazwijmy tę klasę `ViewAnimation`. Po jej utworzeniu możemy przeprowadzić w klasie `ListView` następującą czynność:

```
ListView lv = (ListView)this.findViewById(R.id.list_view_id);
lv.startAnimation(new ViewAnimation());
```

Pójdźmy dalej. Przyjrzyjmy się kodowi źródłowemu klasy `ViewAnimation` i zastanówmy się, jaki rodzaj animacji chcemy otrzymać (listing 16.16).

Listing 16.16. Kod źródłowy klasy `ViewAnimation`

```
public class ViewAnimation extends Animation
{
    @Override
    public void initialize(int width, int height, int parentWidth,
        int parentHeight)
    {
        super.initialize(width, height, parentWidth, parentHeight);
        setDuration(2500);
        setFillAfter(true);
        setInterpolator(new LinearInterpolator());
    }
    @Override
    protected void applyTransformation(float interpolatedTime, Transformation t)
```

```

{
    final Matrix matrix = t.getMatrix();
    matrix.setScale(interpolatedTime, interpolatedTime);
}
}

```

Metoda zwrotna `initialize` informuje nas o wymiarach widoku. W niej są również inicjalizowane wszelkie parametry animacji. W naszym przykładzie skonfigurowaliśmy czas trwania na 2500 milisekund (2,5 sekundy). Sprawimy także, że wynik końcowy animacji pozostanie niezmieniony po jej zakończeniu, a to za sprawą przypisania parametrowi `FillAfter` wartości `true`. W dodatku określiliśmy, że nasz interpolator jest liniowy, co oznacza, że animacja zmienia się stopniowo od początku do końca. Wszystkie wymienione właściwości pochodzą z bazy klasy `android.view.animation.Animation`.

Część zasadnicza animacji jest przeprowadzana w metodzie `applyTransformation`. Szkielet Androida będzie ją bez przerwy wywoływał w celu symulowania animacji. Za każdym wywołaniem tej metody zmienia się wartość parametru `interpolatedTime`. Zmienia się ona w zakresie od 0 do 1 w zależności od tego, w jakim momencie się znajdujemy podczas 2,5-sekundowego cyklu animacji, ustawionego na etapie jej inicjalizacji. Kiedy wartość parametru `interpolatedTime` wynosi 1, znajdujemy się na końcu animacji.

Naszym kolejnym zadaniem jest zmiana macierzy transformacji, dostępnej poprzez obiekt transformacji `t`, umieszczony w metodzie `applyTransformation`. Najpierw należy uzyskać dostęp do macierzy i zmienić jej wartości. Po narysowaniu nowego widoku zadziała również zmodyfikowana macierz. W dokumentacji interfejsów API dotyczącej klasy `android.graphics.Matrix` można znaleźć opis wielu metod dostępnych w obiekcie `Matrix`:

<http://developer.android.com/reference/android/graphics/Matrix.html>

W kodzie z listingu 16.16 zmianą macierzy transformacji zajmuje się poniższy wiersz:

```
matrix.setScale(interpolatedTime, interpolatedTime);
```

Metoda `setScale` zawiera dwa parametry — są to współczynniki skali w osiach x oraz y . Ponieważ wartości parametru `interpolatedTime` mieszczą się w zakresie od 0 do 1, można go zastosować bezpośrednio w postaci współczynnika skali.

Zatem na początku animacji współczynnik ten wynosi 0 w obydwu kierunkach. W połowie przebiegu animacji osie x oraz y będą miały wartość 0,5. Po zakończeniu animacji widok będzie miał pełny rozmiar, ponieważ obydwa współczynniki skali będą miały wartości równe 1. W wyniku tego widok `ListView` jest na początku animacji niewielki i powiększa się do standardowego rozmiaru.

Na listingu 16.17 został zaprezentowany kompletny kod źródłowy aktywności `ViewAnimation` → `Activity` zawierającej animację.

Listing 16.17. Kod źródłowy aktywności animacji widoku wraz z animacją

```

public class ViewAnimationActivity extends Activity {

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

```

```
        this.setupButton();
    }
    private void setupListView()
    {
        String[] listItems = new String[] {
            "Element 1", "Element 2", "Element 3",
            "Element 4", "Element 5", "Element 6",
        };

        ArrayAdapter listItemAdapter =
            new ArrayAdapter(this
                ,android.R.layout.simple_list_item_1
                ,listItems);
        ListView lv = (ListView)this.findViewById(R.id.list_view_id);
        lv.setAdapter(listItemAdapter);
    }
    private void setupButton()
    {
        Button b = (Button)this.findViewById(R.id.btn_animate);
        b.setOnClickListener(
            new Button.OnClickListener(){
                public void onClick(View v)
                {
                    animateListView();
                }
            });
    }
    private void animateListView()
    {
        ListView lv = (ListView)this.findViewById(R.id.list_view_id);
        lv.startAnimation(new ViewAnimation());
    }
}
```

Po uruchomieniu kodu z listingu 16.17 Czytelnik zobaczy coś dziwnego. Widok `ListView`, zamiast równomiernie powiększać się od środka ekranu, rozrasta się od lewego górnego rogu. Wynika to z faktu, że operacje macierzy transformacji mają swój początek właśnie w lewym górnym rogu ekranu. Chcąc uzyskać zamierzony efekt, musimy najpierw przesunąć cały widok w taki sposób, żeby jego środek pokrywał się ze środkiem animacji (w lewym górnym rogu). Następnie wprowadzamy macierz i z powrotem przenosimy widok na właściwe miejsce.

Na listingu 16.18 wstawiliśmy przerobiony kod z listingu 16.16 i zaznaczyliśmy najistotniejsze elementy.

Listing 16.18. Animacja widoku wykorzystująca metody `preTranslate` i `postTranslate`

```
public class ViewAnimation extends Animation {
    float centerX, centerY;
    public ViewAnimation3(){}

    @Override
    public void initialize(int width, int height, int parentWidth, int parentHeight) {
        super.initialize(width, height, parentWidth, parentHeight);
        centerX = width/2.0f;
        centerY = height/2.0f;
    }
}
```



```

        setDuration(2500);
        setFillAfter(true);
        setInterpolator(new LinearInterpolator());
    }
    @Override
    protected void applyTransformation(float interpolatedTime, Transformation t) {
        final Matrix matrix = t.getMatrix();
        matrix.setScale(interpolatedTime, interpolatedTime);
        matrix.preTranslate(-centerX, -centerY);
        matrix.postTranslate(centerX, centerY);
    }
}

```

Metody `preTranslate` oraz `postTranslate` konfiguruje macierz przed operacją skalowania oraz po tej operacji. Jest to proces równoważny utworzeniu zespołu trzech macierzy transformacji. Następujący kod:

```

matrix.setScale(interpolatedTime, interpolatedTime);
matrix.preTranslate(-centerX, -centerY);
matrix.postTranslate(centerX, centerY);

```

jest równoważny instrukcjom:

```

przejdź do innego środka
skaluj
przejdź do oryginalnego środka

```

Taki wzorzec metod `pre` i `post` jest stosowany bardzo często. Podobne wyniki można osiągnąć za pomocą innych metod klasy `Matrix`, ta technika jest jednak najpopularniejsza — a do tego jest zwięzła. Pozostałe techniki również zostaną omówione pod koniec rozdziału.

Co ważniejsze, klasa `Matrix` umożliwia nie tylko skalowanie widoku, lecz również przenoszenie go za pomocą metod `translate` oraz zmianę jego orientacji za pomocą metod `rotate`. Można sprawdzić te metody i przekonać się, jak wyglądają ich efekty. W rzeczywistości wszystkie animacje omówione w podrozdziale „Animacja układu graficznego” są implementowane wewnętrznie za pomocą metod klasy `Matrix`.

Zastosowanie klasy `Camera` do symulowania głębi w obrazie dwuwymiarowym

Pakiet graficzny w Androidzie zawiera jeszcze jedną klasę związaną z animacją — a dokładniej z transformacją — klasę `Camera`. Można ją wykorzystać do symulowania głębi poprzez rzutowanie obrazu dwuwymiarowego, poruszającego się w przestrzeni trójwymiarowej po płaszczyźnie. Możemy na przykład wysłać nasz widok `ListView` o 10 pikseli w głąb ekranu po osi `z` i obrócić ją o 30 stopni wokół osi `y`. Na listingu 16.19 podajemy przykład modyfikowania macierzy za pomocą klasy `Camera`:

Listing 16.19. Zastosowanie klasy `Camera`

```

...
public class ViewAnimation extends Animation {
    float centerX, centerY;
    Camera camera = new Camera();
    public ViewAnimation1(float cx, float cy){

```

```
        centerX = cx;
        centerY = cy;
    }
    @Override
    public void initialize(int width, int height, int parentWidth, int parentHeight) {
        super.initialize(width, height, parentWidth, parentHeight);
        setDuration(2500);
        setFillAfter(true);
        setInterpolator(new LinearInterpolator());
    }
    @Override
    protected void applyTransformation(float interpolatedTime, Transformation t) {
        applyTransformationNew(interpolatedTime,t);
    }
    protected void applyTransformationNew(float interpolatedTime, Transformation t)
    {
        final Matrix matrix = t.getMatrix();
        camera.save();
        camera.translate(0.0f, 0.0f, (1300 - 1300.0f * interpolatedTime));
        camera.rotateY(360 * interpolatedTime);
        camera.getMatrix(matrix);

        matrix.preTranslate(-centerX, -centerY);
        matrix.postTranslate(centerX, centerY);
        camera.restore();
    }
}
```

Animacja widoku ListView przebiega tu w następujący sposób: najpierw jest on umieszczony w odległości 1300 pikseli od ekranu po osi *z*, a następnie wraca do płaszczyzny, w której ós *z* przyjmuje wartość 0. W międzyczasie zostaje on również obrócony od 0 do 360 stopni wokół osi *y*. Zobaczmy, w jaki sposób w kodzie jest zdefiniowane to zachowanie, opisane w poniższej metodzie:

```
camera.translate(0.0f, 0.0f, (1300 - 1300.0f * interpolatedTime));
```

Metoda ta powoduje, że obiekt *camera* przesuwa się w taki sposób, iż przy wartości 0 parametru *interpolatedTime* (początek animacji) wartość *z* będzie wynosiła 1300. Podczas trwania animacji wartość *z* będzie systematycznie malała aż do samego końca, gdy wartość parametru *interpolatedTime* wyniesie 1, a tym samym wartość parametru *z* będzie równa 0.

Metoda *camera.rotateY(360 * interpolatedTime)* wykorzystuje możliwość obracania bryły w trójwymiarze wokół wybranej osi przez obiekt *camera*. Na początku animacji jej wartość wynosi 0. Na końcu animacji przybierze wartość 360.

Metoda *camera.getMatrix(matrix)* pobiera operacje dotychczas wykonane na obiekcie *Camera* i narzuca je przekazanej macierzy transformacji. W tym momencie klasa *matrix* posiada wszystkie translacje potrzebne do uzyskania końcowego efektu, zapewnione przez klasę *Camera*. Teraz klasa *Camera* schodzi z widoku (niezamierzona gra słów), ponieważ w macierzy zostały zaimplementowane wszystkie niezbędne operacje. Wykonujemy teraz operacje *pre* i *post* w celu przesunięcia środka widoku i sprowadzenia go z powrotem. Na koniec przywracamy obiekt *Camera* do pierwotnego, uprzednio zachowanego stanu.

Po wstawieniu kodu do naszego przykładu zobaczymy kontrolkę *ListView* zbliżającą się ze środka widoku w stronę użytkownika, przy okazji wirującą, dokładnie tak jak zaplanowaliśmy.

Część naszej analizy wiążącej się z animacją widoku dotyczyła sposobu animowania dowolnego widoku poprzez rozszerzenie klasy `Animation` i zastosowanie jej wobec tego widoku. Poza modyfikowaniem macryc (bezpośrednio i za pomocą klasy `Camera`) klasa `Animation` umożliwia też wykrywanie poszczególnych etapów animacji. Właśnie tym się teraz zajmiemy.

Analiza interfejsu `AnimationListener`

Android wykorzystuje interfejs nasłuchujący `AnimationListener` do monitorowania zdarzeń animacji (listing 16.20). Możemy nasłuchiwać tych zdarzeń poprzez zaimplementowanie interfejsu `AnimationListener` i skonfigurowanie tej implementacji wobec klasy `Animation`.

Listing 16.20. Implementacja interfejsu `AnimationListener`

```
public class ViewAnimationListener
implements Animation.AnimationListener {

    private ViewAnimationListener(){}

    public void onAnimationStart(Animation animation)
    {
        Log.d("Przykładowa animacja", "onAnimationStart");
    }
    public void onAnimationEnd(Animation animation)
    {
        Log.d("Przykładowa animacja", "onAnimationEnd");
    }
    public void onAnimationRepeat(Animation animation)
    {
        Log.d("Przykładowa animacja", "onAnimationRepeat");
    }
}
```

Klasa `ViewAnimationListener` służy jedynie do tworzenia dzienników komunikatów. Możemy zaktualizować metodę `animateListView` w naszym przykładzie animacji widoku (listing 16.17), żeby dołączyć obiekt nasłuchujący animację:

```
private void animateListView()
{
    ListView lv = (ListView)this.findViewById(R.id.list_view_id);
    ViewAnimation animation = new ViewAnimation();
    animation.setAnimationListener(new ViewAnimationListener());
    lv.startAnimation(animation);
}
```

Kilka uwag na temat macierzy transformacji

Jak pokazaliśmy w tym rozdziale, macierze stanowią podstawę przekształcania widoków i przetwarzania animacji. Teraz omówimy w skrócie niektóre kluczowe metody klasy `Matrix`. Poniżej zostały wymienione podstawowe operacje na macierzach:

```
matrix.reset();
matrix.setScale();
matrix.setTranslate();
matrix.setRotate();
matrix.setSkew();
```

Pierwsza operacja przekształca macierz do postaci macierzy jednostkowej, która po zastosowaniu nie wprowadza zmian w widoku. Operacja `setScale` jest odpowiedzialna za zmianę rozmiaru, `setTranslate` powoduje przesunięcie pozycji obiektu imitujące ruch, a `setRotate` służy do zmiany orientacji. Operacja `setSkew` pozwala na wykrzywienie widoku.

Można powiązać ze sobą macierze lub je wspólnie powielać, aby utworzyć efekt złożony z wielu transformacji. Rozpatrzmy następujący przykład, w którym `m1`, `m2` oraz `m3` są macierzami jednostkowymi:

```
m1.setScale();  
m2.setTranlate()  
m3.concat(m1,m2)
```

Transformacja widoku przez macierz `m1` i następująca po niej transformacja widoku przez macierz `m2` są tożsame z transformacją tego samego widoku przez macierz `m3`. Zwróćmy uwagę, że metody typu `set` zastępują poprzednie transformacje, natomiast `m3.concat(m1,m2)` nie jest tym samym co `m3.concat(m2,m1)`.

Pokazaliśmy już sposób postępowania podczas stosowania metod `preTranslate` oraz `postTranslate` wobec zmiany macierzy transformacji. W rzeczywistości metody `pre` i `post` nie są przeznaczone wyłącznie dla operacji `translate`, lecz tego typu odmiany są dostępne dla każdego rodzaju metod transformacji typu `set`. Ostatecznie metoda `preTranslate`, taka jak `m1.preTranslate(m2)`, jest równoważna operacji:

```
m1.concat(m2,m1)
```

W analogiczny sposób metoda `m1.postTranslate(m2)` jest tożsama operacji:

```
m1.concat(m1,m2)
```

Po rozszerzeniu ekwiwalentem poniższego kodu:

```
matrix.setScale(interpolatedTime, interpolatedTime);  
matrix.preTranslate(-centerX, -centerY);  
matrix.postTranslate(centerX, centerY);
```

jest:

```
Matrix matrixPreTranslate = new Matrix();  
matrixPreTranslate.setTranslate(-centerX, -centerY);
```

```
Matrix matrixPostTranslate = new Matrix();  
matrixPostTranslate.setTranslate(ceTnerX, centerY);
```

```
matrix.concat(matrixPreTranslate,matrix);  
matrix.postTranslate(matrix,matrixPostTranslate);
```

Odnośniki

Poniżej prezentujemy przydatne odnośniki do materiałów, dzięki którym jeszcze lepiej zrozumimy koncepcje zawarte w tym rozdziale:

- <http://developer.android.com/reference/android/view/animation/package-summary.html> — znajdziemy tu różnorodne interfejsy związane z animacją, w tym również interpolatory.

- <http://developer.android.com/guide/topics/resources/animation-resource.html> — omówienie znaczników XML stosowanych w różnych odmianach animacji.
- <ftp://ftp.helion.pl/przyklady/and3ta.zip> — znajdziemy tu projekt do pobrania, przygotowany specjalnie do tego rozdziału. Jest to plik umieszczony w katalogu *ProAndroid3_R16_Animacje*.

Podsumowanie

W tym rozdziale zaprezentowaliśmy ciekawy sposób uatrakcyjnienia interfejsu użytkownika poprzez zastosowanie animacji. Omówiliśmy wszystkie podstawowe typy animacji obsługiwane w Androidzie: animację poklatkową, animację układu graficznego oraz animację widoku. Opiliśmy także dodatkowe pojęcia dotyczące animacji, między innymi interpolatory i macierze transformacji.

Skoro Czytelnik poznał już podstawy, proponujemy przejrzeć przykładowe interfejsy API, udostępnione w zestawie Android SDK, aby przeanalizować pliki XML definiujące różne typy animacji. Poruszymy jeszcze temat animacji w rozdziale 20., poświęconym rysowaniu i animowaniu za pomocą technologii OpenGL. Natomiast w rozdziale 29. możemy się zapoznać z ogólnym omówieniem animacji opartej na właściwościach, stosowanej wraz z fragmentami.

Skorowidz

3GPP, 3rd Generation Partnership Project, 625

A

AAC, Advance Audio Coding, 626

AAPR, Android Asset Packaging Tool, 69

abstrakcja, 472

adapter

- ArrayAdapter, 203

- ManateeAdapter, 218

- ManateeAdapter, 218

- niestandardowy, 218

- SimpleCursorAdapter, 200

- standardowy, 218

ADB, Android Debug Bridge, 83

ADP, Android Developer Phone, 1009

ADT, Android Development Tools, 38, 51

AGC, Automatic Gain Control, 625

agregacja, 970, 1001

akcelerometr, 924

- mierzenie kątów, 930

- pomiar grawitacji, 927

- składowa ruchu, 929

- tryb krajobrazowy, 925

- tryb wyświetlania, 926

- współpraca z magnetometrami, 931

- współrzędne, 924

aktualizacja lokacji, 569

aktualizowanie danych, 138

aktualizowanie widoku, 742

aktywne foldery, 42

- AllContactsLiveFolderCreatorActivity.java, 725

- AndroidManifest.xml, 723

- BetterCursorWrapper.java, 732

- folder kontaktów, 721

- lista, 720

- MyContactsProvider.java, 726

- MyCursor.java, 731

- rejestrowanie identyfikatora URI, 730

- testowanie, 733

- tworzenie folderu, 722

aktywność, 39, 59, 428

- ACTION_DIAL, 159

- animacji widoku, 537

- DetailsActivity, 1053

- klienta nadawczego, 457

- konfiguratora widżetu, 739

- ListActivity, 209, 210

- LocalSearchEnabledActivity, 794, 795

- MainActivity, 375

- MultiViewTestHarness, 669

- NotesList, 72

- OpenGL20MultiViewTestHarness, 705

- paska działania, 1096

 - klasa bazowa, 1085

 - pliki projektu, 1081

- paska działania wyświetlającego zakładki, 1081

- paska zakładek, 1088

- PreferenceActivity, 304, 312

- przechowująca pasek działania, 1098, 1099

- RegularActivity, 778

- SearchActivity, 791, 792, 795

- TestHandlersDriverActivity, 446

- TestOpenGLMainDriver, 670

- ViewAnimationActivity, 537

- wyłączająca wyszukiwanie, 786

- wyszukiwania

 - kod źródłowy, 804

- wyszukiwania globalnego, 770

- wyświetlająca pasek zakładek, 1092

- wywołania wyszukiwania, 808

- aktywność
 - wywołująca klasę AsyncTask, 355
 - z paskiem zakładek, 1080
 - z rozwiniętą listą, 1098
- aktywny folder, live folder, 717
- alarm powtarzalny, 504
 - anulowanie, 506
 - kompilowanie kodu, 506
- alert odległościowy, 578
- aliasy kolumn w strukturach danych
 - kontaktu, 1000
- AMDDA, Android Market Developer Distribution Agreement, 1006
- AMR, Adaptive Multi-Rate, 626
- analiza baz danych, 121
- analiza wbudowanych dostawców treści, 120
- Android Debug Bridge, 121
- Android Developer Phone, 1009
- Android Market, 319, 1005, 1022
 - alternatywy dla serwisu, 1023
 - katalog Moje zamówienia, 1022
- Android SDK, 32, 56
- android.app, 44
- android.bluetooth, 44
- android.content, 44
- android.content.pm, 44
- android.content.res, 44
- android.database, 45
- android.database.sqlite, 45
- android.gesture, 45
- android.graphics, 45
- android.graphics.drawable, 45
- android.graphics.drawable.shapes, 45
- android.hardware, 45
- android.location, 45
- android.media, 45
- android.net, 45
- android.net.wifi, 45
- android.opengl, 46
- android.os, 46
- android.preference, 46
- android.provider, 46
- android.sax, 46
- android.speech, 46
- android.speech.tts, 46
- android.telephony, 46
- android.telephony.cdma, 46
- android.telephony.gsm, 46
- android.text, 46
- android.text.method, 47
- android.text.style, 47
- android.util, 47
- android.view, 47
- android.view.animation, 47
- android.view.inputmethod, 47
- android.webkit, 47
- android.widget, 47
- animacja, 517
 - klatek kluczowych, 524
 - poklatkowa, 518
 - dodawanie animacji do aktywności, 520
 - lista klatek, 521
 - planowanie, 518
 - środowisko testowe, 519, 522
 - tworzenie aktywności, 519
 - układ graficzny, 519
 - rotacyjna, 524, 531
 - skali, 524
 - definiowanie w pliku XML, 528
 - translacyjna, 524, 531
 - typu alfa, 524, 530
 - układu graficznego, 523
 - kod aktywności, 527
 - środowisko testowe, 525
 - tworzenie aktywności, 525
 - widok ListView, 525
 - w osi X oraz w wymiarze alfa, 1077
 - widoku, 533
 - aktywność, 534
 - dodawanie animacji, 536
 - kod aktywności, 535
 - kod źródłowy aktywności, 537
 - metody preTranslate i postTranslate, 538
 - układ graficzny, 534
 - widoku ListView, 540
- animatorem niestandardowy, 1076
- animowane przejścia, tweening, 42
- animowanie obiektów, 691
- animowanie trójkąta, 676
- animowanie widoku ListView, 528
- ANR, Application Not Responding, 351, 427
- API Google Maps, 52

aplikacja

- adb, 123
- Android Debug Bridge, 121
- Android Hierarchy, 57
- BDayWidget, 747
- Browser, 38
- Contacts, 38
- Downloads, 367
- Gestures Builder, 898, 901
- HelloAndroidApp, 65
- Home, 38
- Hierarchy Viewer, 242
- J2EE, 68
- keytool, 318, 319
- Kontakty, 958
- Lista czujników, 909
 - kod Java, 909
 - układ graficzny, 909
 - wyniki, 911
- MapViewDemo, 549, 889
- Monitor akcelerometru, 918
- Monitor czujnika oświetlenia, 911
- NfcDemo, 950
- Notepad, 69
 - analiza kodu, 71
 - tworzenie, 70
 - uruchomienie, 69
 - wczytywanie, 70
- obsługująca mapy, 548
- obsługująca multimedia, 606
- Package Browser, 409
- Phone, 38
- pocztowa, 593
- przedstawiająca mapę świata, 157
- przeglądarki stron, 157
- rejestrująca, 631
- SDK Manager, 54
- SipDemo, 599
- służąca do przeciągania obiektów, 876
- służąca do tłumaczeń, 397
 - główny plik aplikacji, 399
 - plik AIDL usługi tłumacza, 399
 - plik AndroidManifest.xml, 403
 - plik usługi tłumaczenia, 402
 - plik z ciągami znaków, 399
 - plik z funkcją tłumaczenia, 402
 - plik z tablicami, 399

- układ graficzny, 398
- StreetView, 890
- Terminal, 56
- TouchDemo1, 863
 - główna aktywność, 866
 - interfejs użytkownika, 864
 - kod Java, 865
 - komunikaty, 867
 - układ graficzny, 863
- ukazująca zastosowanie fragmentów, 1034
- umożliwiająca połączenie telefoniczne, 157
- Ustawienia, 954
- Ustawienia wyszukiwania, 775
- wykrywająca gesty, 902
- wyświetlająca klawiaturę, 157
- zawierająca szczegółowe mapy, 157

aplikacje Androida, 48

architektura REST, 119

argument

- @avdname, 122
- CHECK_VOICE_DATA_PASS, 845
- fill_parent, 182
- metody startSearch(), 839
- minSdkVersion, 182

ARM, Advanced RISC Machine, 39

artefakty usługi i klienta, 395

atrybut android

- gravity, 231
- layout_gravity, 231
- layout_span, 234
- padding, 235
- permission, 334
- prompt, 215
- readPermission, 334
- text, 181
- writePermission, 334

atrybut

- FadeOffset, 901
- id, 97
- quantity, 102
- queryAfterZeroResults, 822
- REFERER, 396
- ringtoneType, 309
- searchSuggestIntentData, 822
- showSilent, 309
- suggestActionMsgColumn, 837

atrybuty
 uprawnienia, 329
 węzła data, 160
AVD, Android Virtual Device, 51, 60, 65

B

badanie aktywności zawierającej pasek
 działania, 1098, 1102
badanie kontaktów zbiorczych, 980
 elementy menu, 986
 funkcje użytkowe, 980
 główna aktywność, 986
 klasa bazowa, 981
 kolumny kursora, 988
 pliki projektu, 980
 testowanie kontaktów, 982
badanie nieprzetworzonych kontaktów, 989
 opcje menu, 990
 pliki projektu, 989
 pola kursora, 993
 przeglądanie danych, 994
 testowanie danych kontaktu, 995
 testowanie kontaktów, 990
baza danych contacts.db, 133
baza danych SQLite, 119, 125
bezpieczeństwo
 certyfikat cyfrowy, 318
 definiowanie uprawnień, 334
 klucz prywatny, 318
 klucz publiczny, 318
 kontrolowanie dostępu do zasobów, 334
 magazyn kluczy, 318
 podpisywanie aplikacji, 318
 przekazywanie uprawnień, 333
biblioteka
 FreeType, 37, 38
 języka C, 37
 OpenGL, 37
 Skia, 38
 Surface Manager, 38
 WebKit, 37, 38
biblioteka gestów, 900
biblioteka OpenGL, 649
 animowanie trójkątów, 676
 dodawanie trójkątów, 675
 glClear, 658
 glColor, 659

 glDrawElements, 656
 glFrustum, 661
 gluLookAt, 660
 glVertexPointer, 654
 glViewport, 663
 koncepcja widzenia, 660
 kształty, 678
 objętość widzenia, 661
 OpenGLTestHarnessActivity, 667
 podstawy rysowania, 654
 RegularPolygon, 681
 rozmiar ekranu, 663
 rysowanie prostokąta, 679
 rzutowanie ortograficzne, 662
 rzutowanie perspektywiczne, 662
 tekstury, 678, 694
 trójwymiarowa scena, 659
 tworzenie trójkątów, 676
 wielobok foremny, 681
biblioteka OpenGL ES, 649
 proste kształty, 654
biblioteka OpenGL ES 2.0, 704
 aktywność sterująca, 706
 dostęp do jednostek cieniowania, 711
 funkcje, 706
 jednostki cieniujące, 708
 jednostki cieniujące wierzchołki, 708
 jednostki cieniujące fragmenty, 708
 renderowanie, 707
 trójkąt, 711
 źródła, 715
biblioteki Apache HTTP, 48
biblioteki Java Androida, 32
biblioteki multimediów, 37
blokada przechodzenia w stan zatrzymania,
 477, 486
błąd kompilacji we wtyczce ADT, 111
błędy w aplikacji, 81
BSD, Berkeley Software Distribution, 37
bufor koloru, 658
bufor nio, 655, 658, 666

C

CA, certificate authority, 318
CAD, Computer Aided Design, 650
cal, 235
Centrum Oprogramowania Linuksa, 53

- certyfikat
 - Android Debug, 321
 - cyfrowy, 318
 - debugowania, 320
 - okres ważności, 324
 - PKI, 412
 - podpisany samoistnie, 318
 - produktu, 1018
 - programistyczny, 320
 - testowy, 546
 - własny, 318
 - X.509, 318
 - CHOICE_MODE_MULTIPLE, 211
 - CHOICE_MODE_NONE, 211
 - CHOICE_MODE_SINGLE, 211
 - ciąg znaków, 115
 - ciężar, weight, 228
 - com.google.android.maps, 47
 - CRUB, Create, Read, Update, Delete, 77
 - cyfrowy podpis aplikacji, 322
 - cykl życia
 - aktywności, 446, 447
 - aplikacji, 78
 - dostawcy treści, 448
 - odbiorców komunikatów, 448
 - usługi, 448
 - czas życia procesu, 446
 - czas życia składnika, 446
 - częstotliwość próbkowania, 628
 - czujnik
 - aktualizacje odczytów, 915
 - interpretacja danych, 921
 - pobieranie zdarzeń, 911
 - pozostawianie włączonego ekranu, 920
 - rodzaje czujników, 908
 - rozwiązywanie problemów, 915
 - termometry, 922
 - trudne problemy, 914
 - wybór wartości odświeżania, 913
 - wykrywanie, 908
 - czujnik NCF, *Patrz* NFC, 907
 - czujniki
 - akcelerometry, *Patrz* akcelerometr, 924
 - ciśnienia, 923
 - grawitacji, 939
 - informacje o położeniu, 932
 - komunikacji bliskiego pola, 939
 - magnetometri, 930
 - orientacji w przestrzeni, 931
 - oświetlenia, 921
 - przyspieszenia liniowego, 939
 - wektora obrotu, 939
 - zbliżeniowe, 922
 - żyroskopij, 923
- ## D
- Dalvik VM, 33, 36
 - dane, 159, 167
 - dane dodatkowe, 161
 - dane nieprzetworzonego kontaktu, 994
 - dane typu Bundle, 568
 - DATABASE_MODE_2LINES, 801
 - DATABASE_MODE_QUERIES, 801
 - DDMS, Dalvik Debug Monitor Server, 82
 - Debug, 82
 - debugger debug_layout_activity.xml, 977
 - debugowanie, 293, 1090
 - debugowanie aplikacji, 82
 - definicja
 - nieregularnej tabeli, 232
 - tabeli danych kontaktów, 968
 - tabeli kontaktów zbiorczych, 969
 - tabeli nieprzetworzonych kontaktów, 966
 - definiowanie
 - działania w dostawcy widżetu, 1118
 - kontrolki GridView, 214
 - prostokąta o zaokrąglonych rogach, 109
 - wielokrotności, 102
 - zasobów typu Color, 105
 - zasobów typu color-drawable, 108
 - zasobów typu dimension, 106
 - zasobów typu string, 103
 - deklarowanie niestandardowego
 - uprawnienia, 328
 - deklinacja magnetyczna, 938
 - Developer Account, 1006
 - dip, 235
 - długie kliknięcie, 261
 - długoterminowa usługa, 473
 - długoterminowy odbiorca komunikatów, 476

dodawanie

- animacji do widoku, 536
- elementów do menu, 249
- funkcji dotyku, 889
- identyfikatorów do kontrolki, 181
- kontaktu, 998
- plików do magazynu multimediów, 644
- plików dźwiękowych do silnika TTS, 854
- pliku do dostawcy treści, 137
- trójkątów za pomocą indeksów, 675
- typów do obiektu, 161
- znaczników, 553

dokumentacja pakietu SDK, 412

dostawca

- BookProvider, 141
 - dodawanie książki, 150
 - usuwanie książki, 150
 - wyświetlanie listy książek, 151
 - zliczanie książek, 151

Contacts, 131

GPS_PROVIDER, 577

lokalizacji, 567

MediaStore, 131

NETWORK_PROVIDER, 577

PASSIVE_PROVIDER, 577

pasywny, 567

położenia, 568

propozycji, 798

- aktywność wyszukiwania, 803
- główna aktywność, 810
- manifest zawierający definicję, 802
- pliki implementacji, 799
- pole wyszukiwania lokalnego, 811
- propozycja lokalna, 812
- propozycje globalne, 813
- użytkowanie, 810
- włączanie, 812
- wyniki wyszukiwania lokalnego, 811
- zadania, 800

propozycji niestandardowy

- badanie metadanych, 821
- identyfikatory URI, 818
- implementacja, 813
- implementacja aktywności
 - wyszukiwania, 824
- klasa SearchActivity, 825
- korzystanie, 831

metadane wyszukiwania, 820

plik manifest, 830

przekazywanie kwerendy, 820

wyniki, 832

wywołanie SearchActivity, 827

zakończenie aktywności, 829

SearchRecentSuggestionsProvider, 806

sieciowy, 567

SimpleSuggestionProvider, 804

metadane wyszukiwania, 807

SuggestUrlProvider, 814

implementacja klasy, 815

kod źródłowy, 815

pliki projektu, 814

systemu GPS, 567

treści, 41, 59, 119

analiza wbudowanych dostawców, 120

architektura, 126

baza danych, 139

dodawanie pliku, 137

identyfikator URI, 128

implementacja, 139

odczyt danych, 130

rejestrowanie dostawcy, 149

widżetu, widget provider, 736

dostępność, 704

dp, 235

dynamiczne dane, 180

działanie, 159, 167

ACTION_DOWN, 862

ACTION_DRAG_ENDED, 1144

ACTION_DRAG_STARTED, 1144

ACTION_GET_CONTENT, 171

ACTION_MOVE, 862, 894

ACTION_NDEF_DISCOVERED, 942

ACTION_PICK, 169

ACTION_POINTER_DOWN, 894

ACTION_POINTER_UP, 894

ACTION_SEARCH, 828, 833

ACTION_TAG_DISCOVERED, 942

ACTION_TECH_DISCOVERED, 942

ACTION_UP, 862

ACTION_VIEW, 828, 833

CALL, 159

Intent.ACTION_CALL, 160

Intent.ACTION_DIAL, 160

VIEW, 825

E

Eclipse, 38, 51
 EditText, 184
 edycja kontaktu, 960
 ekran dotykowy, 861
 ekran preferencji, 297
 ekran startowy, 1110
 eksploracja kont, 972

- funkcje testujące, 973
 - główna aktywność sterująca, 977
 - plik manifest, 978
 - plik menu, 973
 - spis plików, 978

 element Activity.managedQuery, 74
 element Dot, 876
 element nasłuchujący, 192
 element SimpleCursorAdapter, 74
 element TextView, 94
 elementy składowe Androida, 68

- AndroidManifest.xml, 68
- anim, 68
- assets, 68
- drawable, 68
- layout, 68
- menu, 68
- raw, 68
- res, 68
- src, 68
- values, 68
- xml, 68

 emulacja karty NFC, 949
 emulacja rozruchu urządzenia, 64
 emulator Androida, 38
 EULA, 1019
 EXTRA_EMAIL., 162
 EXTRA_SUBJECT, 162

F

filtr intencji, 160, 166, 943, 945
 flaga

- CONTEXT_IGNORE_SECURITY, 414
- CONTEXT_INCLUDE_CODE, 414
- CONTEXT_RESTRICTED oznacza, 414
- Menu.FLAG_APPEND_TO_GROUP, 266
- krawędzi, 869

nietrwałości, 485
 Service.START_REDELIVER, 485
 Service.START_STICKY, 485
 trwałości, 485
 folder

- android\AVD, 65, 66
- assets, 69
- drawable, 897
- raw, 69
- res, 69, 74
 - res/layout-land, 240
 - res/layout-port, 240
 - res/layout-square, 240
- układu graficznego res/layout, 240
- values, 74

 foldery wiadomości SMS, 592, 593
 format dźwięku 3GPP, 625
 format VCF, 963
 fragment, 40, 1026

- aplikacja ukazująca cykl życia, 1033
- cykl życia, 1028
- formy komunikowania, 1073
- komunikacja pomiędzy fragmentami, 1074
- metoda onAttach(), 1030
- metoda onCreate(), 1030
- metoda onCreateView(), 1031
- metoda onDestroyView(), 1033
- metoda onDetach(), 1033
- metoda onInflate(), 1030
- metoda onPause(), 1032
- metoda onResume(), 1032
- metoda onStart(), 1032
- metoda onStop(), 1032
- metoda setRetainInstance(), 1033
- metoda zwrotna onActivityCreated(), 1032
- przechowujący dialog, 1055
- przejścia i animacje, 1044
- stosowanie, 1027
- stosowanie odniesień, 1046
- struktura, 1027
- transakcja fragmentu, 1042
- trwałość, 1054
- tworzenie hierarchii widoków, 1031
- wycofanie okna dialogowego, 1059
- wyświetlanie okna dialogowego, 1054
- wyświetlanie nowej aktywności, 1051

fragment wyświetlający okna dialogowe, 1060
główna aktywność, 1061
główny układ graficzny, 1072
interfejs użytkownika, 1061
kod Java, 1064
pliki projektu, 1060
układ graficzny, 1064

funkcja

debugowania, 84
getACursor(), 982
getDistinctPendingIntent(), 510
getEventsFromAnXMLFile, 111
getExternalStorageDirectory(), 607
getExtras, 161
Install New Software..., 56
java.util.AttributeSet, 110
listContacts(), 988
ProGuard, 1018
przeciągania, 876, 1131
główna aktywność, 1136
interfejs aplikacji, 1134
lista plików, 1133
podstawowe informacje, 1131
przykładowa aplikacja, 1133
testowanie aplikacji, 1145
tworzenie układu graficznego, 1133
układ graficzny, 1134
putExtras, 161
setData(), 435
StrictMode, 85, 89
StrictMode w trybie debugowania, 86
testThread(), 437
TTS, 843
type-to-search, 797
wielodotykowości, 879, 887

funkcje

daty, 760
kalendarza, 494
użytkowe, 980

G

geokodowanie w Androidzie, 559, 563
geokodowanie w oddzielnym wątku, 563
geolokalizacja, 559

gest ściskania, 891
kod Java, 892, 896
ScaleGestureDetector, 896
układ graficzny, 896
gest zaznaczenia, 899
gesty, 891
aplikacja, 902
biblioteka, 900
kod Java aplikacji, 902
magazynie, 900
rejestrowanie, 905
struktura klas, 900
układ graficzny aplikacji, 902
gesty niestandardowe, 898
gesty właściwe, 900, 904
glClear
zerowanie koloru, 658
glDrawElements
koncepcja pasa, 657
koncepcja wachlarza, 657
kształt, 657
kwadrat, 657
linia, 657
pas linii, 657
pętle linii, 657
punkt, 657
trójkąty, 657
glFrustum
bliski punkt, 662
daleki punkt, 662
objętość widzenia, 662
ostrosłup widzenia, 662
promień, 662
gluLookAt
orientacja aparatu, 661
punkt oczny, 660
punkt spoglądania, 661
punkt widoku, 661
wektor góry, 661
współrzędne świata, 660
glVertexPointer
bryła okalająca, 655
interfejs API, 656
objętość okalająca, 655
współrzędne świata, 655

glViewport
 wziernik, 663
 głębia w obrazie dwuwymiarowym, 539
 Google Checkout, 1022
 Google Maps, 546
 Google Nexus One, 1009
 Google Nexus S. Android Developer Phone, 1009
 GPS, Global System Positioning, 545
 GPU, Graphical Processing Unit, 649
 grafika trójwymiarowa, 652
 grawitacja, gravity, 228
 grupa opcji, 193
 grupy widoków, 39
 GSM, Global System for Mobile
 Communication, 38

H

handheld, 32
 Hashimi Sayed Y., 23
 hasła storepass i keypass, 321
 hiperbola, 532
 historia Androida, 34

I

IANA, Internet Assigned Numbers
 Authority, 129
 IDE, Integrated Development Environment, 38
 identyfikator
 Contacts.People.CONTENT_URI, 131
 predefiniowany, 526
 public static, 74
 treści nieprzetworzonego kontaktu, 997
 układu graficznego, 527
 URI, 76, 128, 130, 332
 klasa UriMatcher, 147
 rozpoznawanie kolekcji, 140
 rozpoznawanie URI, 147
 wprowadzanie klauzuli WHERE, 134
 wstawianie rekordów, 136
 URI propozycji, 815
 URI wyszukiwania, 815
 URI wyszukiwania kontaktów, 989
 zasobów, 97, 114
 t1_1_en_port, 115
 t1_enport, 114
 t2, 115

testport_port, 114
 teststring_all, 114, 116
 zasobów R.menu.moje_menu, 269
 IDialogProtocol, 288
 IETF, Internet Engineering Task Force, 597
 ikony akustyczne, 855
 implementacja
 aktywności wyszukiwania, 824
 bazowych klas aktywności, 1082
 długoterminowej usługi, 483, 486
 dostawców treści, 139
 dostawcy treści BookProvider, 141
 dostawcy widżetu, 751
 interfejsu AIDL, 379
 interfejsu AnimationListener, 541
 interfejsu Parcelable, 386
 klasy AlarmManagerService, 514
 klasy Renderer, 664
 klasy ReportStatusHandler, 439
 klasy WorkerThreadRunnable, 438
 kształtu RegularPolygon, 682, 683
 metody delete, 147
 metody getType(), 819
 metody insert, 147
 metody query, 146
 metody update, 147
 modeli widżetów, 753
 niestandardowego dostawcy propozycji, 814
 obiektu nasłuchującego zdarzeń, 1087
 oświetlonego zielonego pokoju, 478
 usługi lokalnej, 373
 plik AndroidManifest.xml, 374
 plik main.xml, 373
 plik MainActivity.java, 373
 usługi StockQuoteService2, 388
 wątku roboczego, 438
 instalacja narzędzi ADT, 57
 instalowanie aktualizacji, 324
 instancja paska działania, 1089
 instancja widżetu, 738
 instrukcja awk, 123
 instrukcja create, 125
 instrukcja find, 123
 instrukcja grep, 123
 instrukcja insert, 137

- intencja, intent, 40, 59, 155, 156
 - atrybuty dodatkowe, 161
 - dane, 159, 167
 - działanie, 159, 167
 - jawna nazwa klasy, 159
 - kategorie intencji, 168
 - mapa typu klucz – wartość, 159
 - MediaStore.ACTION_IMAGE_CAPTURE, 644
 - niejawna, 159
 - oczekująca, pending intent, 172
 - PendingIntent, 495
 - przydzielanie do ich składników, 166
 - putExtras, 161
 - schemat danych, 167
 - ścieżka danych, 168
 - typ danych, 167
 - uprawnienia do danych, 168
 - VIEW, 160
 - interakcja aktywności wyszukiwania lokalnego, 792
 - interakcja aktywności z przyciskiem wyszukiwania, 777
 - interfejs
 - ActionBar, 1079
 - AJAX Language, 396
 - API, 43, 653
 - API Google Maps, 396
 - API multimediów, 619
 - aplikacji obsługującej multimedia, 606
 - AttributeSet, 1030
 - ContactsContract, 967
 - createPackageContext(), 413
 - DDMS, 571
 - DialogInterface, 276
 - DialogInterface.OnClickListener, 1072
 - EGL, 651
 - glDrawElements, 654
 - GLSurfaceView.Renderer, 663
 - Google AJAX Language API, 396
 - Google Directions, 569
 - graficzny
 - pojemnik, kontener, 176
 - układ graficzny, 176
 - widok, widżet, kontrolka, 176
 - IDialogFinishedCallback, 291
 - IDialogProtocol, 286, 288
 - JDBC, 127
 - kontaktów, 972
 - modelu widżetu, 753
 - nasłuchujący AnimationListener, 541
 - obiektu nasłuchującego, 1063
 - OnCheckedChangeListener, 193
 - OnDialogCompleteListener, 1063, 1067
 - onItemClickListener, 208
 - onLoadCompleteListener, 616
 - pomiędzy OpenGL ES a Androidem, 663
 - Projection, 888
 - rejestratora wideo, 632
 - Renderer, 664
 - ResolveInfo, 164
 - SensorEventListener, 913
 - Shape, 682
 - Tłumacz Google, 395, 397
 - UI, 175, *Patrz* interfejs użytkownika
 - UI Androida, 39
 - UI Emulator Control, 591
 - UI kontrolek DatePicker i TimePicker, 198
 - UI odtwarzacza plików wideo, 620
 - UI widoku RingtonePreference, 308
 - użytkownika, 39
 - konstruowanie interfejsu za pomocą kodu oraz języka XML, 180
 - programowanie za pomocą kodu, 177
 - projektowanie interfejsu, 176
 - tworzenie interfejsu w pliku XML, 179
 - interpolator, 531, 532
 - interpolator liniowy, linear interpolator, 1075
 - interpolator accelerate_interpolator, 532
 - IPC, Inter-Process Communication, 37
- ## J
- J2EE, Java 2 Platform Enterprise Edition, 58
 - Java Development Kit, 52
 - Java Standard Edition, 32
 - jawna nazwa klasy, 159
 - jawne przywołanie aktywności, 447
 - JCP, Java Community Process, 651
 - JDK, Java Development Kit, 52, 53, 56
 - JDK, Java SE Development Kit, 51
 - język AIDL, 368

język HTML, 91
 język XML, 69
 język XUL, 39
 JIT, Just-In-Time Compiler, 36
 JRE, Java Runtime Environment, 52
 JSON, JavaScript Object Notation, 343
 JVM, Java Virtual Machine, 32

K

kamera
 zmiana ustawień, 672
 karta SD, 601, 602
 foldery, 605
 źródło plików audio, 612
 katalog
 /res, 203
 /res/layout/, 95
 /res/menu, 269
 /res/values, 1143
 /res/xml, 297
 /tools, 242
 assets, 111
 DCIM, 604
 drawable-port, 240
 drawable-square, 240
 frameworks/base, 49
 HOME, 53
 layout, 69
 layout-en, 114
 Moje zamówienia, 1022
 par klucz – wartość, 136
 rawable-land, 240
 tools, 55, 122
 tools/android, 54
 katalogi alternatywnych zasobów, 113
 katalogi na karcie SD
 DIRECTORY_ALARMS, 605
 DIRECTORY_DCIM, 605
 DIRECTORY_DOWNLOADS, 605
 DIRECTORY_MOVIES, 605
 DIRECTORY_MUSIC, 605
 DIRECTORY_NOTIFICATIONS, 605
 DIRECTORY_PICTURES, 605
 DIRECTORY_PODCASTS, 605
 DIRECTORY_RINGTONES, 605

katalogi zasobów, 112
 kategoria LAUNCHER, 304, 1053
 kategorie aktywności
 CATEGORY_ALTERNATIVE, 165
 CATEGORY_BROWSABLE, 165
 CATEGORY_DEFAULT, 165
 CATEGORY_EMBED, 165
 CATEGORY_GADGET, 165
 CATEGORY_HOME, 165
 CATEGORY_LAUNCHER, 165
 CATEGORY_PREFERENCE, 165
 CATEGORY_SELECTED_ALTERNATIVE, 165
 CATEGORY_TAB, 165
 CATEGORY_TEST, 165
 kategorie intencji, 163, 168
 klasa
 AbstractRenderer, 664
 abstrakcyjna, 472
 AccountsFunctionTester, 974
 ActionBar, 1079, 1080, 1084
 Activity, 214, 262
 AdapterView, 200, 201
 AlarmManager, 748
 AlarmManagerService, 514
 AlertBuilder, 1060
 AlertDialogFragment, 1071
 AllContactsLiveFolderCreatorActivity, 725
 ALongRunningReceiver, 475
 android.app.AlertDialog.Builder, 274
 android.content.ContentProvider, 139
 android.content.ContentResolver, 136
 android.content.ContentValues, 136
 android.graphics.Matrix, 537
 android.location.Geocoder, 559
 android.media.MediaPlayer, 601
 android.os.Bundle, 161
 android.os.Debug, 83
 android.preference.ListPreference
 atrybuty, 298
 konfigurowanie projektu, 298
 android.preference.PreferenceActivity, 297
 android.util.Log, 81
 android.view.LayoutInflater, 278
 android.view.Menu, 247
 android.view.SubMenu, 247

klasa

- android.view.ViewGroup, 175
- android.widget.ImageButton, 188
- android.widget.ListAdapter, 213
- android.widget.RadioButton, 193
- AndroidHttpClient, 349, 350
- AnimatedSimpleTriangleRenderer, 677
- Animation, 541
- AnimationDrawable, 521
- app_name, 93
- Application, 81
- AppWidgetProvider, 742, 1107
- ArrayAdapter, 202, 203, 208
- AssetManager, 112
- AsyncPlayer, 606, 617
- AsyncTask, 351, 354, 357
- AudioFormat, 629
- AudioRecord, 626
- AudioTrack, 618
- BaseActionBarActivity, 1097, 1101
- BaseAdapter, 218
- BaseTester, 974
- BDayWidgetModel, 752
- BetterCursorWrapper, 732
- BitmapFactory, 196
- BookProviderMetaData, 139
- BookTableMetaData, 140
- Builder, 85
- CamcorderProfile, 639
- Camera, 42, 539
- CameraProfile, 639
- ClientCustPermMainActivity, 331
- ComponentName, 162
- ContactsContract.AggregationExceptions, 1001
- ContentProvider, 141
- ContentProviderOperation, 1003
- ContentResolver, 137
- ContentValues, 136, 137, 138
- Context, 453
- CustomHttpClient, 347
- DatabaseHelper, 77
- DebugActivity, 975, 1083
- DeferWorkHandler, 436
- DetailsFragment, 1037, 1050, 1052
- DialogFragment, 1055, 1056, 1060
- Dot, 1143
- DownloadImageTask, 358
- DownloadManager, 362
- Drawable, 521
- ES20AbstractRenderer, 711
- Fragment, 1025, 1027
- FragmentManager, 1045
- FragmentTransaction, 1042, 1044, 1076
- GenericManagedAlertDialog, 291
- GenericPromptDialog, 292
- Geocoder, 560, 565
- GeomagneticField, 938
- GeoPoint, 559
- GestureDetector, 895
- GestureOverlayView, 904
- GLSurfaceView, 664, 667, 704
- GridViewActivity, 214
- GS20SimpleTriangleRenderer, 714
- HelpDialogFragment, 1068
 - kod Java, 1069
 - układ graficzny, 1069
- HttpActivity, 358
- HttpClient, 338
- HttpGet, 338
- HttpURLConnection, 349
- ImageView, 196
- Inflater, 221
- Intent, 40, 162
- IntentService, 469, 492
 - kod źródłowy, 470
 - rozszerzenie na odbiorcę komunikatów, 472
- ItemizedOverlay, 554, 557
- JetPlayer, 606, 617
- LayoutInflater, 278, 1031
- LightedGreenRoom, 476
- LinearLayout, 228
- ListActivity, 593
- ListFragment, 1047
- ListPreference, 296
- LocationManager, 567, 580, 583
- MainActivity, 867, 1035, 1061
- ManagedActivityDialog, 287, 288
- ManagedDialogsActivity, 287, 290
- ManateeAdapter, 221
- MapActivity, 548, 550

- MapController, 551
- MapView, 548, 888
- Matrix, 539
- MediaPlayer, 601, 610, 618
- MediaRecorder, 622
- MediaScanner, 647
- MediaStore, 640, 646
- MotionEvent, 861, 869, 873, 880
- MyContactsProvider, 726
- MyCursor, 731
- MyFragment, 1029
- MyLocationOverlay, 574
 - dostosowywanie, 577
 - zastosowanie, 574
- MySMSMonitor, 590
- NotePadProvider, 76
- NotesList, 73
- ObjectAnimator, 1075, 1139
- OnGestureListener, 895
- Overlay, 557
- PaintDrawable, 108
 - podstawowa
 - Activity, 427
 - BroadcastReceiver, 427
 - ContentProvider, 427
 - Service, 427
- PolygonRenderer, 691
- PrivActivity, 327
- PromptDialogFragment, 1064, 1067
- PromptListener, 279
- RadioGroup, 194
- RadioGroup.OnCheckedChangeListener, 193
- RegularPolygon, 681
 - animowanie obiektów, 691
 - calcArrays, 689
 - Constructor, 689
 - getAngleArrays, 689
 - getIndexBuffer, 689
 - getVertexBuffer, 689
 - getXMultiplierArray, 689
 - getYMultiplierArray, 689
 - renderowanie kwadratu, 689
 - rysowanie koła, 693
- RemoteViews, 741
- RemoteViewsFactory, 1114
- Renderer, 664
- ReportStatusHandler, 439
- ScaleGestureDetector, 896
- SearchActivity, 825, 828
- SearchRecentSuggestionsProvider, 813, 840
- SendAlarmOnceTester, 499
- SensorManager, 908, 930, 936
- Shakespeare, 1042
- SimpleCursorAdapter, 200, 201, 202, 208
- SimpleRectangleRenderer, 690
- SimpleSuggestionProvider, 799
 - kod źródłowy, 800
 - tryby bazodanowe, 801
- SimpleTriangleRenderer, 665
- SimpleTriangleRenderer2, 675, 676
- SipAudioCall, 599
- SipSession, 599
- SmsManager, 588
- SoundPool, 612, 617
 - maksymalna liczba próbek, 616
 - odtworzenie dźwięku, 613
 - parametr SRC_QUALITY, 616
 - strumień audio, 616
- Spannable, 224
- Spinner, 215
- SpinnerAdapter, 1095
- SQLiteQueryBuilder, 146, 149
- static final ints, 93
- StrictModeWrapper, 88
- System.out.println, 81
- TextToSpeech, 841
- TexturedSquareRenderer, 698
- ThreadGroup, 372
- TitlesFragment, 1047
- Toast, 293
 - debugowanie, 293
- UriMatcher, 146, 147
- Utils, 455
- VelocityTracker, 874
- ViewAnimation, 536
- ViewGroup, 1043
- widget.RadioGroup, 193
- WorkerThreadRunnable, 438
- XmlPullParser, 110
- klasy
 - aktywności sterującej, 975
 - android.view.View, 175

klasy

bazowe aktywności

ActionBar, 1084

AndroidManifest.xml, 1093

showAsAction, 1092

SpinnerAdapter, 1095

służące do obsługi widoków zdalnych, 1108

sterownika

DeferWorkHandler.java, 441

ReportStatusHandler.java, 441

Utils.java, 441

WorkerThreadRunnable.java, 441

związane z menu, 248

klatka kluczowa, 517

klauzula select, 141

klauzula WHERE, 134

klient usługi IStockQuoteService, 382

kliknięcie, 187

klucz

API AJAX, 397

API MAP, 1018

interfejsu API mapy, 546

map-api, 321

prywatny, 318, 411

publiczny, 318, 411

KML, Keyhole Markup Language, 572

kod niestandardowy, 169

kod odbiorcy, 454

kod odbiorcy komunikatów, 465

kod usługi zdalnego widoku, 1128

kod źródłowy Androida, 48

kod źródłowy Git, 48

kody przycisków działania, 835

kolumny kursora encji kontaktu, 997

kolumny kursora propozycji, 823

definiowanie, 824

Komatineni Satya, 23

komentarze w kodzie., 140

kompilacja, 501, 503, 506, 508, 511, 513

kompilator JIT, 36

kompilowanie jednostek cieniujących, 709

kompilowanie kodu, 449

kompilowanie zasobów, 98

komunikat

ANR, 432, 450

testowy, 456

wysyłanie komunikatu, 454

konfiguracja

alarmu, 493

kanałów, 629

strumieni audio, 855

uruchomieniowa, 63

urządzenia pionowa, *portrait*, 240urządzenia pozioma, *landscape*, 240urządzenia tryb kwadratowy, *square*, 240

konfigurator widżetów, 739

konfigurowanie

alertu odległościowego, 579

ciężaru, 230

klasy RemoteViewsFactory, 1114

konstruktora alertów, 278

menu za pomocą kodu, 255

obiektów nasłuchujących, 278

odbiorcy dla alarmu, 495

paska działania, 1096

powtarzalnego alarmu, 503

procedury obsługi kliknięcia, 188

usługi RemoteViewsService, 1112

widoków obiektów, 1140

wirtualnego urządzenia AVD, 64

zasad ThreadPolicy, 85

zasad VmPolicy funkcji, 86

zdarzeń onClick, 1117

źródła danych, 611

konsola programisty, 1009

konstruktor alertów, 278

konstruktor klasy RemoteViewsFactory, 1114

kontakty

agregacja, 1001

analiza, 964

dodawanie kontaktu, 998

aktywność sterująca, 999

edytowanie niestandardowych danych, 961

edytowanie szczegółów, 960

eksportowanie, 962

interfejs, 972

nieprzetworzone, 965, *Patrz także* badanie

nieprzetworzonych kontaktów

odczytywanie kontaktów zbiorczych, 971

pobieranie bazy kontaktów, 965

standardowe typy danych, 964

synchronizacja, 1002

tabela danych, 967

testowanie danych

aktywność sterująca, 996

- testowanie nieprzetworzonych kontaktów
 - aktywność sterująca, 992
- typy danych, 964
- umieszczanie zdjęcia, 962
- widok `contact_entities_view`, 971
- widok `view_contacts`, 971
- wyświetlanie, 958
- wyświetlanie szczegółów, 959
- zbiorcze, *Patrz także* badanie kontaktów zbiorczych
- kontekst wyszukiwania, 838
- kontener, `container`, 176
- konto
 - dodawanie konta Google, 956
 - lista kont, 958
 - logowanie, 957
 - odczytywanie zawartości, 958
 - tworzenie konta Google, 956
 - ustawienia kont i synchronizacji, 955
 - wstawianie kontaktów, 957
 - zarejestrowane na urządzeniu, 957
- konto programisty, `Developer Account`, 1006
- kontrakt klasy `RemoteViewsFactory`, 1114
- kontroler układu graficznego, 529
- kontrolka, `control`, 176
 - `AdapterView`, 204
 - `AnalogClock`, 199
 - `Button`, 187
 - `CheckBox`, 190
 - `Chronometer`, 223
 - `com.google.android.maps.MapView`, 200
 - `DatePicker`, 197
 - `DigitalClock`, 199
 - `EditText`, 233
 - `Gallery`, 217
 - `GridControl`, 213
 - `GridView`, 213, 221
 - definiowanie kontrolki w pliku XML, 214
 - `ImageButton`, 188
 - `ImageView`, 195
 - `ListView`, 205
 - dodawanie elementów, 205
 - dodawanie innych kontroltek, 208
 - przyjmowanie danych, 207
 - reakcja na kliknięcie, 206
 - wyświetlanie wartości, 205
 - listy, 205
 - `MapView`, 200, 549
 - `ProgressBar`, 223
 - `RadioButton`, 192
 - `RatingBar`, 223
 - `ScrollBar`, 223
 - `Spinner`, 215
 - `TableRow`, 233
 - `TimePicker`, 197
 - `ToggleButton`, 190
- kontrolki
 - Androida, 182
 - Androida 2.2, 467
 - `AutoCompleteTextView`, 185
 - daty i czasu, 197
 - `AnalogClock`, 199
 - `DatePicker`, 197
 - `DigitalClock`, 199
 - `TimePicker`, 197
 - `ImageView`, 196
 - przycisków, 187
 - `Button`, 187
 - `CheckBox`, 190
 - `ImageButton`, 188
 - `RadioButton`, 192
 - `ToggleButton`, 190
 - kontrolki tekstu, 183
 - `AutoCompleteTextView`, 185
 - `EditText`, 184
 - `MultiAutoCompleteTextView`, 186
 - `TextView`, 183
 - `TextView`, 181
 - widoku, 741
- koprocetor graficzny, 649
- kreator New Android Project, 598
- kryteria dopasowania, 166
- kSOAP2, 343
- kursor propozycji, 822
- kursor systemu Android, 133
- kwalfikatory konfiguracji, 113
- kwalfikatory zasobów, 241
 - Gęstość pikseli na ekranie, 241
 - Język i region, 241
 - Klawiatura, 241
 - Orientacja ekranu, 241
 - Rodzaj tekstowych danych wejściowych, 241
 - Rozmiary ekranu, 241

kwalifikatory zasobów
Sterowanie przy braku klawiatury dotykowej, 241
Szersze/wyższe ekrany, 241
Typ ekranu dotykowego, 241
Wersja środowiska SDK, 241
kwerenda wyszukiwania, 833

L

layout, *Patrz* układ graficzny
lista
aktywnych folderów, 720
animowanych klatek, 521
baz danych znajduje się w katalogu, 123
kodów przycisków działania, 835
kolumn, 823
kompletacyjna, 273
pakietów, 408
preferencji, 297, 302
propozycji, 768
technologii, 946
układów graficznych, widżetów i widoków, 1106
widżetów, 1130
widżetów ekranu startowego, 738
ListActivity
odczytywanie danych, 212
listingi, 449, 489
lo, 56
localhost, 56
lokalizacja certyfikatu testowego, 547
lupa, 243
LVL, License Verification Library, 1017

M

M3G, 652
macierz jednostkowa, 542
macierz transformacji, 539, 541
MacLean Dave, 23
magazyn gestów, 900
magazyn kluczy, 318, 319, 320
magazyn MediaStore, 645
magazyn multimediiów, 644
ManagedActivityDialog
klasa DialogRegistry, 289

mapa typu klucz – wartość, 159
mapy, 546
aplikacja, 548
nakładanie własnej warstwy, 553
obsługa za pomocą dotyku, 888
usługa Google Maps, 546
mapy projekcji, 149
MD5, 546
mechanizm przechowywania i dostępu, 120
Pliki, 120
Preferencje, 120
Sieć, 120
SQLite, 120
mechanizm refleksji, 87
mechanizm type-to-search, 797
menedżer
FrameLayout, 237
LinearLayout, 228
RelativeLayout, 235
TableLayout, 231
menedżer alarmów, 493, 449
aktywność do testowania ustawień, 500
alarm powtarzalny, 503
czas uruchomienia alarmu, 494
jednorazowe wysłanie alarmu, 498
konfiguracja, 493
konfigurowanie odbiorcy, 495
pierwszeństwo intencji, 512
praca z wieloma alarmami, 508
projekt, 497
testowanie scenariuszy, 501
trwałość, 515
twierdzenia, 515
układ graficzny, 502
ustawiania, 496
uzyskanie dostępu, 494
wersje alternatywne, 503
menedżer LinearLayout, 607
menedżer powiadomień, 463
menedżer telefonii, 594
menedżer układu graficznego, 227
FrameLayout, 228
LinearLayout, 228
RelativeLayout, 228
TableLayout, 228

- menu, 247
 - aktywności, widoki i menu kontekstowe, 262
 - aktywność, 254
 - aktywności SearchInvokerActivity, 790
 - alternatywne, 264
 - dotatkowe znaczniki, 271
 - dotawanie elementów, 255, 256
 - dotawanie podmenu, 260
 - drugorzędne, 256
 - dynamiczne, 268
 - grupy, 250
 - konfiguracja, 255
 - Konta i synchronizacja, 954
 - kontekstowe, 261, 263
 - modyfikowanie AndroidManifest.xml, 258
 - odpowiedź na kliknięcie, 251, 257
 - opcji, 249
 - rejestrwanie widoku TextView, 263
 - rozszerzone, 259
 - standardowe, 255
 - systemowe, 261
 - środowisko testowe, 253
 - tworzone za pomocą kodu Java, 268
 - układ graficzny, 254
 - w postaci ikon, 259
 - wybór elementów, 251
 - wykorzystanie intencji, 252
 - zapełnianie menu, 266
 - zdefiniowane w plikach XML, 268
 - zmiana danych, 268
- metadane bazy danych, 139
- metadane dostawcy widżetów, 1128
- metadane wyszukiwania, 793, 807
- metoda
 - activity.onCreateContextMenu(), 262
 - addEarcon(), 855
 - addPreferencesFromResource(), 297
 - addSubMenu(), 261
 - animate(), 523
 - ArrayAdapter.createFromResource(), 217
 - callService(), 393
 - cancel(), 515
 - captureImage(), 644
 - commit(), 314
 - context.getSharedPreferences(), 757
 - createFromResource(), 203
 - DefaultHttpClient(), 347
 - detectAll(), 87
 - detectDiskReads(), 87
 - dismiss(), 1058, 1068
 - Display.getOrientation(), 926
 - Display.getRotation(), 926
 - distanceTo(), 569
 - doInBackground(), 353, 354, 356
 - doSpeak(), 845
 - doUpdate(), 937
 - draw(), 877
 - enableDefaults(), 87
 - enqueue(), 364
 - fabrykująca, 1029
 - findLocation(), 565
 - findPreference(), 312
 - fromRawResource(), 903
 - GestureLibraries.fromFile(), 903
 - GET, 340
 - getAccuracy(), 569
 - getAction(), 867, 886
 - getActionIndex(), 887
 - getActionMasked(), 887
 - getCheckedItemIds(), 213
 - getCheckedItemPositions(), 211
 - getCheckedRadioButtonId(), 195
 - getCount(), 150, 221
 - getDefaultEngine(), 856
 - getEdgeFlags(), 869
 - getExternalStoragePublicDirectory(), 607
 - getHttpClient(), 347, 348
 - getHttpContent(), 349
 - getInterpolation(), 532
 - getIntrinsicHeight(), 557
 - getIntrinsicWidth(), 557
 - getItemId(), 251
 - getLastNonConfigurationInstance(), 357
 - getMinBufferSize(), 629
 - getOrientation(), 931, 937
 - getPathSegments(), 146
 - getPointerCount(), 880
 - getPrefsToSave(), 754
 - getRotationMatrix(), 931
 - getSharedPreferences(), 303, 314
 - getString(), 303
 - getTag(), 1063

metoda

- getText(), 208
- getType(), 146, 819
- getViewTypeCount(), 221
- glDrawElements, 657
- glVertexPointer, 654
- handleMessage, 436
- hasAccuracy(), 569
- HTTP GET, 405
- HTTP POST, 405
- initCamera(), 634
- initRecorder(), 637
- insert(), 78, 204
- invalidate(), 1144
- isCancelled(), 356
- isChecked(), 192
- isEnabled(), 222
- isLocationDisplayed(), 552, 575
- isRouteDisplayed(), 552
- LayoutInflater(), 278
- LayoutInflater.from(), 278
- ListView, 74
- Log.d, 111
- makeText(), 294
- MenuBuilder.addIntentOptions, 267
- mIndexBuffer, 658
- MotionEvent.getAction(), 887
- moveToFirst(), 133
- newInstance(), 350
- notifyDataSetChanged(), 204
- nstartDrag(), 1143
- obtain(), 873
- obtainMessage(), 435
- onAccuracyChanged(), 913, 914
- onActivityCreated(), 1032
- onActivityResult(), 568
- onCheckedChanged(), 195
- onClick(), 192, 873
- onCreate(), 73, 79, 88, 206, 805, 903
- onCreateContextMenu(), 264
- onCreateDialog(), 284
- onCreateOptionsMenu, 249, 265
- onCreateView(), 1031, 1056, 1068
- onDestroy(), 80, 446
- onEnabled(), 753
- onInflate(), 1030
- onInit(), 856
- onItemClick(), 208
- onListItemClick(), 75
- onMeasure(), 1144
- onNewIntent(), 803, 805, 811
 - testowanie, 806
- onOptionsItemSelected, 251, 252, 270
- onPause, 446
- onPostExecute(), 353
- onPreExecute(), 353, 361
- onPrepareDialog(), 284
- onPrepareOptionsMenu, 268
- onProgressUpdate(), 353
- onProviderDisabled(), 573
- onReceive(), 591, 753, 1120
- onRestart(), 80
- onResume(), 79, 80
- onRetainNonConfigurationInstance(), 357
- onSaveInstanceState(), 1047
- onSearchRequested(), 789, 838
- onSensorChanged(), 913, 919, 929
- onStart(), 79
- onStartCommand, 485
- onStop(), 80, 446
- onTouchEvent(), 862, 863, 877, 893
- onUpdate(), 743, 744, 1111
- penaltyDeath(), 85
- PendingIntent.getActivity(), 172, 173
- permitDiskReads(), 87
- playSilence(), 856, 858
- populate(), 557
- postInvalidate(), 576
- postTranslate, 538
- PreferenceManager.getDefaultShared
 - ↳ Preferences(this), 303
- preTranslate, 538, 542
- publishProgress(), 353
- putFragment(), 1047
- queueSound(), 616
- recognize(), 905
- recycle(), 873
- registerListener(), 913
- requestLocationUpdates(), 571, 573
- respondToMenuItem(), 433
- rotate, 539
- scanFile(), 646

- scheduleDistinctIntents(), 511
- scheduleSameIntentMultipleTimes(), 510
- sendAlarmOnce(), 499
- sendBroadcast(), 173, 453, 454
- sendDataMessage(), 588
- sendMessage(), 435
- sendMessageDelayed(), 435
- sendMultipartTextMessage(), 589
- sendSmsMessage(), 587
- setAdapter(), 214
- setBounds(), 557
- setChecked(), 193
- setContentView(), 211, 214
- setData(), 435
- setDataSource(), 611, 612, 621
- setEdgeFlags(), 869
- setEngineByPackageName(), 856
- setEntries(), 313
- setGroupVisible, 250
- setImageResource(), 189
- setLatestEventInfo(), 466
- setListAdapter(), 214
- setMeasureAllChildren(), 239
- setOnCheckedChangeListener(), 193
- setOneShot(), 522
- setOnTouchListener(), 888
- setOnUtteranceCompletedListener(), 846
- setOptionText(), 303
- setPendingIntentTemplate(), 1119
- setRepeating(), 505
- setRetainInstance(), 1033
- setTargetFragment(), 1074
- showAllRawContacts(), 993
- sleep(), 432, 439
- sort(), 204
- speak(), 846
- startActivity(), 169, 1074
- startActivity(intent), 173
- startSearch(), 789, 839
 - appSearchData, 789
 - globalSearch, 789
 - initialQuery, 789
 - selectInitialQuery, 789
- startService(), 173, 372, 429, 469
- stop(), 441
- stopSelf, 485
- surfaceCreated(), 635
- testAccounts(), 975
- toUri(), 1118
- translate, 539
- tv.getText(), 225
- uiCallback.sendEmptyMessage(0), 565
- Utils.logThreadSignature(), 436, 439
- VelocityTracker.obtain(), 874
- zoomToSpan(), 558
- zwrotna getCount(), 1115
- zwrotna getItemId(), 1116
- zwrotna getLoadingView(), 1116
- zwrotna getViewAt(), 1115
- zwrotna getViewTypeCount(), 1116
- zwrotna hasStableIds(), 1117
- zwrotna onAttach(), 1030
- zwrotna onCreate(), 1030, 1115
- zwrotna onCreateView(), 1031
- zwrotna onDataSetChanged(), 1117
- zwrotna onDestroy(), 1033, 1115
- zwrotna onDestroyView(), 1033
- zwrotna onDetach(), 1033
- zwrotna onDrag(), 1144
- zwrotna onInflate(), 1030
- zwrotna onPause(), 1032
- zwrotna onResume(), 1032
- zwrotna onStop(), 1032
- metody cyklu życia aktywności, 79
- metody główne
 - delete, 139
 - getType, 139
 - insert, 139
 - query, 139
 - update, 139
- metody pobierające, 1046
- metody uzyskiwania aktualizacji położenia, 574
- MFC, Microsoft Foundation Classes, 39
- mikrostopnie, 562
- mikrotesła, μT , 930
- milimetr, 235
- MIME, Multipurpose Internet Mail Extensions, 127
- moduł HttpClient, 338, 343, 405
- monitorowanie zdarzeń animacji, 541
- motyw, 227
- MultiAutoCompleteTextView, 186

N

nagrywanie i odtwarzanie dźwięku, 622

nakładanie tekstury, 683

nakładka ItemizedOverlay, 558

narzędzia

AAPR, 69

ADT, 38, 56

DDMS, 1019

Developer Tools, 57

do usuwania błędów, 81

wątkowania, 430

narzędzie

Abstract Window Toolkit, 32

adb, 323

AVD Manager, 83

edytora manifestu, 327

Export Unsigned Application Package, 321

Hierarchy Viewer, 175, 242

ekran urządzeń, 243

tryb Pixel Perfect, 244

układ graficzny, 242

jarsigner, 318

keytool, 321, 547

LogCat, 81, 82, 85, 867

wpisy, 870

Swing, 32

widżet Toast, 571

zipalign, 322

nazwa składnika, component name, 266

NDK, Native Development Kit, 940

NFC, Near Field Communication, 35, 939

aktywacja, 940

emulacja karty, 949

odbieranie terminali, 942

odczytywanie terminali, 946

P2P, 949

testowanie technologii, 950

trasowanie terminali, 941

tryby działania, 940

wybór filtru intencji, 943

numer portu 5554, 588

O

obiekt

addressContainer, 178

ApplicationInfo, 86

AudioRecord, 629

AudioRecord., 628

Builder, 87

ClipData, 1143

Criteria, 568

criteriaIntent, 266

cursor, 130, 772

DatePicker, 273

DragEvent, 1132

ACTION_DRAG_ENDED, 1132

ACTION_DRAG_ENTERED, 1132

ACTION_DRAG_EXITED, 1132

ACTION_DRAG_LOCATION, 1132

ACTION_DRAG_STARTED, 1132

ACTION_DROP, 1132

Drawable, 109

falseLayoutBottom, 873

FileDescriptor, 611

flight_sort_options_values, 301

Geocoder, 891

GridView, 214

HttpClient, 349

HttpGet, 349

HttpParams, 349

HttpPost, 349

includeInGlobalSearch, 812

IntentService, 485

klasy Spinner, 216

Location, 568

ManateeAdapter, 221

map, 141

MediaController, 621

Menu, 261

Message, 435

MotionEvent, 862, 873, 877, 880, 891

MotionView, 862

nasłuchujący, 251, 897, 1087

nasłuchujący OnInitListener, 845

NdefMessage, 949

NdefRecord, 948

nio, 666

OnCheckedChangeListener, 192

Parcelable, 391

PendingIntent, 515

PreferenceCategory, 311

RadioButton, 195

RemoteViews, 466

SensorEvent, 913

- SensorListener, 915
- SharedPreferences, 314
- SipProfile, 599
- Spinner, 204, 216, 1095
- SubMenu, 260
- TextToSpeech, 845
- TimePicker, 273
- Toast, 294
- trueBtnTop, 867
- VelocityTracker, 875
- ViewHolder, 221
- wakelock, 478
- XmlPullParser, 110
- XmlResourceParser, 110
- obiekty nasłuchujące, 278
- obsługa map, 888
- obsługa wyjątków, 343
- obszar dropTarget, 1146
- ochrona zasobów i funkcji urządzenia, 325
- odbieranie terminali NFC, 942
- odbiorca BroadcastReceiver, 580, 858, 919
- odbiorca komunikatów, broadcast receiver, 453, 462, 467, 579
 - alert odległościowy, 579
 - definicja odbiorcy, 460
 - długoterminowy, 467, 474
 - opóźnienia czasowe, 461
 - powiadomienia, 463
 - powielanie, 460
- odbiorca pozaprocesowy, 462
- odbiorca przebywający we własnym procesie, 462
- odbiorca TestReceiver, 495
- odbiorca treści, 461
- odczytywanie danych w ListActivity, 210
- odległość pomiędzy dwoma obiektami, 569
- odpowiedź na wybór elementów, 251
- odpowiedź na zdarzenia onClick, 1120
- odpowiedź na zdarzenie onDrag w strefie upuszczania, 1137
- odstęp, 235
- odtworzenie cisy, 856
- odtworzenie ikony akustycznej, 856
- odtworzenie multimediów, 606
 - AsyncPlayer, 617
 - AudioTrack, 618
 - interfejs API multimediów, 619
 - JetPlayer, 617
 - kod aplikacji, 607
 - MediaPlayer, 618
 - setDataSource, 611
 - SoundPool, 612
 - układ graficzny, 607
- ograniczenia klasy AnimationDrawable, 522
- OHA, 48
- okna alertów, 274
 - projektowanie, 274
- okna dialogowe
 - alertów, 273
 - asynchroniczne, 273
 - informujące, 273
 - listy kompletacyjne, 273
 - modalne, 281
 - obiekt DatePicker, 273
 - obiekt TimePicker, 273
- okna niezarządzane, 283
- okna zarządzane
 - DialogRegistry, 289
 - GenericManagedAlertDialog, 291
 - GenericPromptDialog, 292
 - IDialogProtocol, 288
 - ManagedActivityDialog, 288
 - ManagedDialogsActivity, 289, 290
 - protokół, 283
 - struktura, 286
 - upraszczanie protokołu, 285
- pojedynczego wyboru, 273
- synchroniczne, 273
- wielokrotnego wyboru, 273
- zachęty, 276
 - kod, 280
 - obiekt nasłuchujący, 279
 - plik XML układu graficznego, 277
 - projektowanie, 276
 - przeprojektowanie okna, 282
 - tworzenie i wyświetlenie, 279
- okno
 - Devices, 242
 - Emulator Control, 576
 - File Explorer, 604
 - Hierarchy Viewer, 243
 - kreatora New Android Project, 61

okno

- Launch Options, 84
- LogCat, 82, 582, 629, 885, 988
- Package explorer, 1018
- terminalu, 52

opcja

- Add External JARs, 404
- Add note, 73
- Android SDK and AVD Manager, 323
- Android Tools, 321
- Build Path/Configure Build Path., 404
- Create project from existing sample, 899
- Debug As/Android Application, 82
- Debugowanie USB, 82
- Export Signed Application Package, 323
- Launch from snapshot, 84
- QUEUE_ADD, 845
- QUEUE_FLUSH, 845
- Upload Application, 1018
- Wipe user data, 84

Open Graphics Library, 650

OpenCORE PacketVideo, 37

OpenGL ES, 39

OpenGL ES 2.0, *Patrz* biblioteka OpenGL ES 2.0

OpenJDK, 52

operacja

- setRotate, 542
- setScale, 542
- setSkew, 542
- setTranslate, 542

operacje na macierzach, 541

oprogramowanie integracyjne, middleware, 337

optymalizacja aplikacji, 322

Oracle/Sun JDK, 53

organizowanie preferencji w kategorii, 310

orientacja w przestrzeni, 936

oś głębi, 660

oświetlony zielony pokój, 478

- implementacja, 478

P

P2P, peer-to-peer, 949

pakiet

- .apk, 610
- android.app, 1027
- android.location, 559

android.media, 601

android.nfc, 948

android.opengl.GLES20., 704

android.provider, 120

android.providers.Contacts, 132

android.view.animation, 525

com.svox.pico, 856

java.nio, 666

map, 545

nio, 652

OpenGL ES, 38

R.java, 92

pakiety, 407

dokumentacja SDK, 412

lista, 408

nazwa, 408

podpisywanie, 409

specyfikacja, 407

usuwanie, 409

pakiety java.*, 47

para kluczy, 318

parametr

childLayout, 202

from, 202

Intent, 169

odświeżania czujnika, 913

requestCode, 169

SRC_QUALITY, 616

to, 202

uri, 137

parser JSON, 342

parser SOAP, 342

parser XML, 342

pary typu MIME, 129

pasek działania, 1079, 1080

interakcja z menu, 1091

obszar menu, 1081

obszar paska narzędzi, 1081

obszar przycisku ekranu startowego, 1080

obszar tytułu, 1080

obszar zakładek, 1081

tryb wyświetlania listy, 1094, 1096

tryby nawigacji, 1089

pasek zakładek

badanie aktywności, 1093

pasująca aktywność, 266

- PDU, Protocol Description Unit, 591
 perspektywa, 82
 perspektywa DDMS, 82, 603
 perspektywa Debug, 82
 pętla komunikatów, 282
 Phillips Dylan, 25
 Pico, 43
 pierwsza aplikacja, 60
 piksel, 235
 piksele niezależne od gęstości, 235
 piksele niezależne od skali, 235
 PKI, Public Key Infrastructure, 411
 planowanie bazy danych, 139
 plik
 - .adt, 546
 - .apk, 318, 407, 601
 - .dex, 36
 - .jar, 36
 - .profile, 53
 - _has_set_default_values.xml, 304
 - AbstractRenderer.java, 672
 - AccountsFunctionTester.java, 974
 - AIDL usługi tłumacza, 399
 - aktywności sterującej, 442
 - AlarmIntentPrimacyTester.java, 513
 - alpha.xml, 530
 - android.bat, 66
 - android.jar, 48
 - AndroidManifest.xml, 58, 60, 68, 72, 213, 258, 304, 323, 374, 419, 422, 442, 445, 456, 459, 462
 - animacji, 529
 - arrays.xml, 301
 - attrs.xml, 1143
 - BackgroundService.java, 369
 - BaseTester.java, 498, 974
 - box1.xml, 1111
 - CancelRepeatingAlarmTester.java, 508
 - com.androidbook.services.stockquote
 - ↳service, 377
 - commons-lang.tar, 404
 - commons-lang.zip, 404
 - ContactData.java, 994
 - ContactDataFunctionTester.java, 994
 - contacts.db, 123, 125
 - CupcakeMaps.ini, 66
 - debug.store, 320
 - debug_layout_activity.xml, 977
 - DebugActivity.java, 975
 - details.xml, 1041
 - DownloadImageTask.java, 351
 - DropBox, 85
 - DropZone.java, 1137
 - dropzone.xml, 1135
 - dźwiękowy z tekstem, 849
 - eclipse.exe, 53
 - gestures, 902
 - GPX, 572
 - HttpActivity.java, 347, 355, 357
 - HttpGetDemo.java, 338
 - IReportBack.java, 497, 973, 1082
 - JAR, 407
 - KML, 572
 - KMZ, 572
 - layout/lib_main.xml, 418
 - List_Layout.xml, 529
 - main.xml, 241, 302, 331, 354, 373, 421, 442, 456, 672
 - main_layout.xml, 114
 - main_menu.xml, 422, 442, 456, 669, 672
 - MainActivity.java, 363, 373, 867
 - mainmenu.xml, 302
 - manifest, 445
 - manifest klienta, 332
 - menu, 445
 - menu/lib_main_menu.xml, 418
 - MP3, 606
 - MultiViewTestHarnessActivity.java, 672
 - myappraw.apk, 321
 - MyLocationDemoActivity.java, 574
 - NoSearchActivity, 786
 - NotesList.java, 88
 - outfile.apk, 323
 - Palette.java, 1136
 - palette.xml, 1135
 - Person.aidl, 388
 - planets.xml, 216
 - ProAndroid3_R13_ProceduryObsługi.zip, 441
 - proguard.cfg, 1018
 - prompt_layout.xml, 277
 - R.java, 74, 97, 112, 115, 424

plik

- RawContact.java, 990
- release.keystore, 319
- scale.xml, 524
- sdcard.img, 602
- SDK Manager, 54
- searchable.xml, 793
- SimpleSuggestionProvider.java, 800
- SimpleTriangleRenderer.java, 672
- StandaloneReceiver.java, 462
- strings.xml, 92, 181, 215, 302, 829, 1145
- TestAppActivity.java, 420
- TestBCRActivity.java, 456
- TestContactsDriverActivity.java, 977
- TestHandlersDriverActivity.java, 442
- TestLibActivity.java, 417
- TestListWidgetProvider.java, 1122
- TestOpenGLMainDriverActivity.java, 672
- TestReceiver.java, 456
- TestRemoteViewsFactory.java, 1126
- TestRemoteViewsService.java, 1128
- TranslateService.java, 402
- układu graficznego, 94, 444, 458
- układu graficznego main.xml, 94
- Utils.java, 455, 456, 462, 980
- web.xml, 68
- wewnętrzny, 137
- XML, 98, 109, 268
- XML animacji rotacyjnej, 531
- XML preferencji, 302
- XML zawierający definicje menu, 269
- zasobów menu, 458
- zawierający informacje o widżecie, 1129
- zdalnego układu graficznego, 1109
- ZIP, 449, 489

pliki

- aplikacji służącej do tłumaczeń, 397
- do testowania usług
- implementacji dostawcy propozycji, 799
- nieskompresowane, 98
- parcelowane, 388
- programu wyświetlającego listę kont, 972
- projektu menedżera alarmów, 497
- projektu TestBCR, 489
- projektu testowego, 456
- projektu z odbiorcą komunikatów, 490

- układów graficznych, 114
- wideo, 619
- widżetu urodzinowego, 746
- podmenu, 260
- podpis cyfrowy, 410
- podpisywanie aplikacji, 318
- podpisywanie plików, 411
- podpisywanie pliku .apk, 321
- podpisywanie pliku .jar, 318
- podręcznik referencyjny środowiska
 - OpenGL ES, 653
- pojemnik, 176
- pojemnik ListView, 201, 221
- pole NFC, 939
- pole QSB, 771, 793
- polecenia powłoki, 124
- polecenie
 - #ls /system/bin, 123
 - adb, 83
 - adb devices, 121, 122
 - adb help, 122
 - android list avd, 122
 - find, 123
 - ipconfig, 56
 - ls, 123
 - ls -l, 123
 - ls /data/data, 123
 - sqlite> .tables, 125
 - sqlite>.exit, 125
- połączenia równorzędne (P2P) NFC, 949
- położenie, 568, 932
- położenie bieżące, 576
- położenie geograficzne, 559
 - aktualizacja danych, 569
 - LocationManager, 571
 - MyLocationOverlay, 574
- pomiar grawitacji, 927
- pomoc techniczna, 1012
- POP, Post Office Protocol, 954
- port#, 83
- powiadomienia, 464
 - kod odbiorcy komunikatów, 465
- powłoka ash, 123
- powłoka dostawcy widżetu, 743
- predefiniowanie identyfikatora, 97

- preferencje, 295
 - ekran preferencji, 297
 - kategorie, 311
 - klasa ListPreference, 296
 - lista preferencji, 297
 - magazyn danych, 306
 - programowe zarządzanie, 312
 - przechowywanie stanu aktywności, 313
 - widok CheckBoxPreference, 305
 - widok EditTextPreference, 307
 - widok RingtonePreference, 308
 - zagnieżdżenie elementów
 - PreferenceScreen, 310
 - zapisywanie stanu, 313
 - preferencje aplikacji, 295
 - preferencje pola wyboru, 305
 - preferencje RingtonePreference, 309
 - prefiks vnd, 129
 - procedura DeferWorkHandler, 433
 - procedura obsługi, handler, 431, 432
 - klasy sterownika, 441
 - menu, 445
 - proces w Androidzie
 - Activity, 427
 - BroadcastReceiver, 427
 - ContentProvider, 427
 - Service, 427
 - procesy, 407
 - program SQLite Explorer, 965
 - program testujący menedżer alarmów, 502
 - projekt bibliotek, 414, 420, 425
 - identyfikatory współdzielonych zasobów, 424
 - kod aktywności, 420
 - manifest, 419, 422
 - menu, 418, 422
 - twierdzenia, 414, 417
 - układ graficzny, 418, 421
 - projekt Provider, 48
 - protokół
 - odbiorcy komunikatów, 468
 - SIP, inicjalizacji sesji, 597
 - SOAP, 127
 - SSL, 37
 - zarządzanych okien dialogowych, 283, 287
 - przeciąganie obiektów, 876
 - kod Java, 876
 - układ graficzny, 876
 - przekroczenie limitu czasu, 344, 348
 - przekroczenie limitu czasu gniazda, 343
 - przekroczenie limitu czasu połączenia, 343
 - przesłonięcie kontrolki ListView, 209
 - przetwarzanie tekstu na mowę, 841
 - pętla przekazywania wyrażień, 847
 - pliki dźwiękowe, 848
 - prędkość mowy, 842
 - silnik Pico, 842
 - śledzenie wyrażień, 846
 - układ graficzny, 848
 - ustawienia silnika, 842
 - usunięcie z kolejki tekstu, 845
 - wyrażenie, 846
 - zapisywanie pliku dźwiękowego, 850
 - przycisk
 - Generate API Key, 547
 - Inspect Screenshot, 243
 - Install Selected, 55
 - Load View Hierarchy, 242
 - przełączania, 190
 - Publish, 1021
 - Screen Capture, 1019
 - wyszukiwania, 769, 777, 778
 - przyciski działania
 - definicja, 836
 - keycode, 836
 - kolumny, 837
 - queryActionMsg, 837
 - suggestActionMsg, 837
 - suggestActionMsgColumn, 837
 - punkt, 235
- ## Q
- QEMU, 39
 - QSB, Quick Search Box, 769
- ## R
- raporty o błędach aplikacji, 1011
 - refleksja, 87
 - reguły przydzielania intencji, 166
 - rejestracja upoważnienia, 126
 - rejestrator dźwięku, 642
 - rejestrator wideo, 632
 - aktywność, 632
 - AndroidManifest.xml, 639

- rejestrator wideo
 - kod obsługujący wstrzymywanie, 633
 - kod przetwarzania, 636
 - metody zwrotne, 638
- rejestrowanie aktualizacji lokacji, 569
- rejestrowanie aktywności, 156
- rejestrowanie dostawcy, 150
- rejestrowanie multimediów, 621
 - analiza procesu rejestracji, 630
 - AudioRecord, 626
 - CAMCORDER, 625
 - MediaRecorder, 622
 - nieskompresowane dane audio, 630
 - rejestrowanie rozmowy, 625
 - VOICE_RECOGNITION, 625
 - za pomocą intencji, 641
- rejestrowanie odbiorcy komunikatów, 456
- rejestrowanie widoku dla menu kontekstowego, 263
- rejestrzy dziennika LogCat, 872
- rekord, 136
- renderowanie, 707
- renderowanie kwadratu, 689
- REST, REpresentational State Transfer, 119
- RESTful, 41
- RFID, Radio Frequency Identification, 939
- RISC, Reduced Instruction Set Computer, 39
- rodzaje adapterów, 204
 - ArrayAdapter<T>, 204
 - CursorAdapter, 204
 - ResourceCursorAdapter, 204
 - SimpleAdapter, 204
 - SimpleCursorAdapter, 204
- rodzaje menu, 259
- rodzaje zasobów, 99
 - ciągi znaków, 99
 - kolorowe obiekty rysowane, 100
 - kolory, 99
 - obrazy, 100
 - tablice ciągów znaków, 99
 - wielokrotności, 99
 - własne pliki XML, 100
 - własne, nieskompresowane pliki dodatkowe, 100
 - własne, nieskompresowane zasoby, 100
 - wymiary, 99

- rozszerzanie klasy ContentProvider, 141
- RPC, Remote Procedure Call, 368
- RTP, Real-time Transport Protocol, 598
- RTSP, Real-time Streaming Protocol, 598
- rysowanie wielokąta, 693
- rysowanie wielu figur geometrycznych, 699
- rzutowanie obrazu trójwymiarowego, 659
 - glFrustum, 659
 - gluLookAt, 659
 - glViewport, 659

S

- schemat danych, 167
- SD, Secure Digital, 601
- SDK, Software Development Kit, 32
- SDP, Session Description Protocol, 598
- segment ścieżki, 135
- sekwencja przeciągania, 1140
- serwis Google Maps, 553
- sieć
 - 3G, 38
 - Bluetooth, 38
 - EDGE, 38
 - WiFi, 38, 364
- silnik Pico, 842
- silnik przetwarzania tekstu na mowę, 43
- silnik renderujący prostokąt, 679
- silnik SquareRenderer, 690
- silnik TTS, 845, 850, 859
 - dostępność języka, 857
 - funkcje zaawansowane, 854
 - metody językowe, 857
 - odtworzenie ciszy, 856
 - odtworzenie ikony akustycznej, 856
- SIP, Session Initiation Protocol, 585
- skala mapy, 552
- skalowanie, 528
- sklep, *Patrz* Android Market
- składnia odniesienia do zasobu, 95
- skrót MD5 certyfikatu testowego, 547
- skrót dla elementu menu, 272
- skrzynka odbiorcza, 592
- SMS, Short Messaging Service, 585
 - foldery, 593
 - monitorowanie wiadomości, 589
 - skrzynka odbiorcza, 592

- wiadomości przychodzące, 589
 - wysyłanie wiadomości, 586
 - SOAP, Simple Object Access Protocol, 119
 - sp, 235
 - specyfikacja JSR 239, 652
 - specyfikacja pakietu, 407
 - spis dostępnych kontaktów, 979
 - spis kwalifikatorów konfiguracji, 113
 - sprzedaż aplikacji, 1012
 - lokalizacja aplikacji, 1014
 - obsługa różnych rozmiarów ekranu, 1012
 - ponowne kierowanie do sklepu, 1016
 - przygotowanie ikony aplikacji, 1015
 - przygotowanie pliku .apk do wysłania, 1018
 - przygotowanie pliku
 - AndroidManifest.xml, 1013
 - testowanie działania, 1012
 - usługa licencyjna, 1017
 - ustalanie ceny, 1016
 - SSL, Secure Sockets Layer, 37
 - stała
 - CATEGORY_SYSTEM., 261
 - FILL_PARENT, 182
 - intent.ACTION_SEARCH, 805
 - MATCH_PARENT, 182
 - Menu.CATEGORY_ALTERNATIVE, 266
 - Menu.CATEGORY_SECONDARY, 248
 - Menu.CATEGORY_SYSTEM, 248
 - Notes.CONTENT_URI, 73
 - stałe trybu agregacji, 970
 - stan uaktywnienia przycisku, 189
 - stan wciśnięty przycisku, 189
 - stan zatrzymania, 476, 477
 - stany aktywności, 80
 - stany wątku, 441
 - stos Java, 666
 - stos drugoplanowy, 1059
 - stos programowy, 337
 - stos programowy Android SDK, 33, 37
 - StrictMode, 84
 - strona startowa Androida, 719
 - strona startowa z polem QSB, 769
 - struktura klas gestów, 900
 - struktura preferencji, 296
 - struktura składników, 428
 - strumień audio, 855
 - styl EditText, 226
 - styl EditText.Danger, 226
 - style, 224
 - dla fragmentów tekstu, 225
 - nadrzędne, 226
 - umieszczane dynamicznie, 225
 - umieszczane w widoku, 226
 - wykorzystywane w wielu widokach, 225
 - Sun JDK, 52
 - superklasa aktywności, 172
 - symbol #, 123
 - synonim, 141
 - system GPS, 82
 - system operacyjny
 - iPhone OS, 33
 - Linux, 52
 - Mac OS X, 52
 - Mobile Linux, 33
 - Moblin, 33
 - Symbian OS, 33
 - Windows 7, 52
 - Windows Mobile, 33
 - Windows Vista, 52
 - Windows XP, 52
 - szablon intencji oczekującej, 1119
- ## Ś
- ścieżka danych, 168
 - środowisko
 - Android SDK, 52
 - chronionej pamięci, 78
 - Dalvik VM, 36
 - Eclipse, 51
 - Eclipse IDE for Java Developers, 53
 - IDE, 48
 - IDE Eclipse, 51
 - J2EE, 58, 338
 - Java ME, 652
 - JRE, 52
 - JVM, 32
 - OpenGL ES 2.0, 39
 - programowania, 51
 - projektowe, 89
 - testowe biblioteki OpenGL, 667
 - testowe do sprawdzania menu, 254

T

tabela contact, 1001
 tabela wyszukiwania, 968
 tabela zawierająca wyjątki agregacji, 1001
 tablica ciągów znaków, 101
 tablica dodanych elementów menu, 267
 tablica flight_sort_options, 301
 tagi do symulowania podmenu, 271
 technologia ARM, 39
 technologia M3G, 652
 technologia Ndef, 946
 technologia NFC, 939
 teksturowane koła, 703
 teksturowany kwadrat, 699
 teksturowany wielobok, 700
 tekstury, 694

- glActiveTexture, 697
- glBindTexture, 697
- glGenTextures, 697
- glTexCoordPointer, 697
- glTexEnv, 697
- glTexParameter, 697
- GLUtils.texImage2D, 697
- proces obsługi, 695
- rysowanie, 698
- znormalizowane współrzędne, 694

 terminal

- ACTION_NDEF_DISCOVERED, 942
- ACTION_TAG_DISCOVERED, 942
- ACTION_TECH_DISCOVERED, 942

 testowanie

- animacji typu alfa, 530
- danych kontaktu, 995
- długoterminowych usług, 488
- dostawcy BookProvider, 150
- kontaktów zbiorczych, 982
- nieprzetworzonych kontaktów, 990
- odbiorców komunikatów, 489
- pierwszeństwa intencji, 512
- procedur obsługi, 442
- technologii NFC, 950
- ustawień alarmów, 500
- widżetu wyświetlającego listę, 1130

 TextView, 183
 ThreadPolicy, 85
 transformacja widoku, 542
 trasowanie terminali NFC, 941

tryb

agregacji, 970
 debugowania, 82
 debugowania USB, 82
 MODE_PRIVATE, 314
 MODE_WORLD_READABLE, 314
 MODE_WORLD_WRITEABLE, 314
 portretowy, 1052
 rozwijalnego menu, 1104
 ruchu ulicznego, 552
 usuwania błędów, 82
 widoku ulic, 552
 wyszukiwania, 771
 TTS, Text To Speech, 841
 tworzenie

- aktywnego folderu, 722
- aplikacji Notepad, 70
- cyfrowego podpisu, 411
- dostawcy widżetów, 1122
- fragmentu wyświetlającego okna dialogowe, 1055
- instancji widżetu, 742
- intencji oczekującej, 495
- intencji oczekującej na komunikat, 1118
- intencji oczekujących, 511
- interfejsu użytkownika w pliku XML, 179
- interfejsu użytkownika za pomocą kodu, 177
- kategorii preferencji, 311
- klasy fabrykującej, 1126
- klasy SoundPool, 616
- komunikatu, 435
- konfiguracji uruchomieniowej, 63
- konta Google, 956
- listy pakietów, 408
- menu, 249
- menu za pomocą plików XML, 268
- niestandardowych adapterów, 218
- niestandardowych animacji, 1075
- obiektu nasłuchującego listy nawigacji, 1095
- odbiorcy komunikatów, 455
- odniesień do kontrolki, 182
- odpowiedzi dla elementów menu, 270
- odpowiedzi dla menu kontekstowego, 264
- okna alertu, 275
- okna dialogowego zachęty, 277
- powiadomień, 465
- pól wyboru, 190

- projektu, 449
- projektu bibliotek, 417
- prostego odbiorcy, 454
- tożsamości w sklepie, 1006
- trójkątów, 676
- urządzenia AVD, 67, 602
- wystąpienia fragmentu, 1029
- typy agregacji, 1001
- typy danych, 167
- typy MIME, 77, 128, 130, 265
- typy treści, 129

- application, 129
- audio, 129
- example, 129
- image, 129
- message, 129
- model, 129
- multipart, 129
- text, 129
- video, 129

U

- Ubuntu, 52

- układ graficzny, layout, 94, 95, 113, 176, 227
 - animacja, 523
 - dostosowanie do urządzenia, 239
 - optymalizacja, 242
 - tworzenie interfejsu użytkownika, 240
 - usuwanie błędów, 242

- układ graficzny

- aktywności LocalSearchEnabledActivity, 794
- aktywności SearchActivity, 792
- aktywności SearchInvokerActivity, 789
- aktywności TestAlarmsDriverActivity, 502
- aktywności TestOpenGLMainDriver, 671
- aktywności wyszukiwania, 828
- aplikacji MapViewDemo, 549
- aplikacji odtwarzającej multimedia, 607
- aplikacji rejestrującej, 631
- aplikacji tłumaczącej, 398
- dla aplikacji TouchDemo1, 863
- do animacji poklatkowej, 519
- do wywoływania klasy AsyncTask, 354
- FrameLayout
 - widok ImageView, 239
- klasy SearchActivity, 828

- LinearLayout, 178, 228

- ciężar, 228
- grawitacja, 228

- RelativeLayout

- interfejs użytkownika, 237

- TableLayout

- kontrolka EditText, 234
- nieregularna tabela, 233

- trueLayoutTop, 871

- usługi IStockQuoteService, 384

- usługi StockQuoteService2, 389

- widżetu, 749

- zawierający widok debugowania, 1090

- umowa EULA, 1019

- upoważnienie, 127

- uprawnienia, 325

- atrybuty, 329

- definiowanie uprawnień, 334

- dla funkcji i zasobów, 326

- identyfikatorów URI, 332

- do danych, 168

- niestandardowe, 326, 330

- przekazywanie uprawnień, 333

- w pliku AndroidManifest.xml, 325

- uprawnienie

- android.permission.ACCESS_COARSE_
 - ↳LOCATION, 567

- android.permission.ACCESS_FINE_
 - ↳LOCATION, 567, 580

- android.permission.INTERNET, 600, 610

- android.permission.READ_CONTACTS, 980

- android.permission.READ_PHONE_
 - ↳STATE, 596

- android.permission.RECORD_AUDIO, 626

- android.permission.USE_SIP, 600

- android.permission.WRITE_EXTERNAL_
 - ↳STORAGE, 853

- uruchamianie aktywności, 162

- uruchamianie emulatora, 83

- Urząd Przydzielania Numerów

- Internetowych, 129

- urządzenia AVD, 60, 63, 65, 122

- urządzenia typu handheld, 282

- urządzenie Android Developer Phone, 1005

usługa, 59

- długoterminowa, 486, 488
- Google Maps, 553
- Google Maps JavaScript API, 569
- HTTP, 43, 337
- IStockQuoteService, 377, 380, 381, 382, 384
- LocationManager, 566, 567, 569, 571
- lokalna, 367, 369
- nietrwała, 484
- notowań giełdowych, 377
- RemoteViewsService, 1112, 1113
- RESTful, 48
- ServiceManager, 566
- StockQuoteService2, 390
- Test60SecBCRService, 474
- trwała, 485
- WakefulIntentService, 472
- zdalna, 367
- zorientowana na położenie, 43

usługi

- definiowanie interfejsu, 376
 - przekazywanie plików parcelowanych, 388
 - przekazywanie typów danych, 385
 - uruchamianie i zatrzymywanie, 478
 - utworzenie i zamknięcie, 478
- usługi obsługujące język AIDL, 368, 376
- usługi w Androidzie, 368
- ustanawianie menedżera alarmów, 496
- ustawianie obrazu, 196
- usuwanie błędów, 82
- usuwanie danych, 138

V

VoIP, Voice over Internet Protocol, 597

W

walidator licencji Android Market, Market

License Validator, 52

waluta klienta, Buyer's Currency, 1016

wartość startOffset, 529

wątek

- narzędzia, 429
- stan Dead, 441
- stan New thread, 441
- stan Not runnable, 441
- stan Runnable, 441

wątek główny

- aktywności, 428
 - dostawcy treści, 429
 - informacje o stanie, 439
 - odbiorcy komunikatów, 429
 - opóźnianie operacji, 432
 - przetrzywanie, 432
 - usługi, 429
 - wątek pojedynczy, 429
- wątek roboczy, 436
- testowanie, 442
- węzły, 427
- wiadomości e-mail, 593
- widoczność menu, 272
- widok, view, 39, 58, 176
- CheckBoxPreference, 305
 - contact_entities_view, 971
 - EditTextPreference, 307
 - encji kontaktów, 971
 - GridView, 200
 - ImageView, 221, 520
 - Konta i synchronizacja, 954
 - ListPreference, 313
 - ListView, 200, 201, 202, 525
 - MapView, 554
 - MapView wraz ze znacznikami, 557
 - niestandardowy - kółko, 1140
 - RemoteViews, 735
 - RingtonePreference, 308
 - definiowanie preferencji, 309
 - interfejs UI, 308
 - TextView, 95, 202, 263
 - view_contacts, 971
 - zdalny, 1106
- widżet, widget, 176, 736
- cykl życia, 740
 - definiowanie, 740
 - definiowanie dostawcy, 747
 - definiowanie rozmiaru, 748
 - definiowanie w pliku manifest, 740
 - definiowanie w pliku XML, 741
 - implementacja abstrakcyjna modelu, 754
 - implementacja aktywności
 - konfiguratora, 761
 - implementacja dostawcy, 751
 - implementacja modeli, 753
 - implementacja modelu stanów, 758

- interfejs modelu, 753
- kształt tła, 750
- metody zdarzeń zwrotnych, 745
- odinstalowanie pakietów, 745
- ograniczenia i rozszerzenia, 764
- tworzenie instancji, 742
- układ graficzny, 749
- układ graficzny formularza, 763
- Urodziny, 757
- usunięcie instancji widżetu, 745
- współdzielone preferencje, 755
- widżet ekranu startowego
 - AndroidManifest.xml, 1129
 - metadane dostawcy widżetów, 1128
 - pliki projektu, 1121
 - układ graficzny widżetu, 1128
- widżet RadioButton, 192
- widżet urodzinowy, 746
- widżety ekranu startowego, 42, 736, 1105
 - lista, 738
 - tworzenie instancji widżetu, 737
- wieloczęściowa metoda POST, multipart
 - POST, 341
- wielodotykowość, 879
 - kod Java, 880
 - układ graficzny, 880
 - wyniki narzędzia LogCat, 883
 - zastosowanie, 882
- wielokrotności, 101
- wielowątkowy moduł HttpClient, 346
- wirtualna maszyna, 33
- własne pliki zasobów XML, 109
- właściwość
 - colspan, 234
 - onClick, 192
 - orientation, 228
- włączanie widoków, 603
- wskaźnik do danych, 159
- współdzielenie danych, 412
- współdzielone identyfikatory użytkownika, 412
- współrzędne tekstury, 694
- wtyczka ADT, 82, 546
- wtyczka Galaxy Tab, 52
- wyjątek IllegalArgumentException, 629
- wyjątki protokołowe, 343
- wyjątki transportowe, 343
- wykonywanie zdjęć, 643
- wymiary, 235
 - Całe, 235
 - Milimetry, 235
 - Piksele, 235
 - Piksele niezależne od gęstości, 235
 - Piksele niezależne od skali, 235
 - Punkty, 235
- wysokość pojemnika ListView, 209
- wysyłanie aplikacji, 1018
- wysyłanie komunikatu, 454
- wysyłanie powiadomienia, 464
- wyszukiwanie
 - aktywność wyszukiwania, 773
 - aplikacja odpowiedzialna za ustawienia, 775
 - dostawcy propozycji, 772
 - dostęp do aktywności testowych, 785
 - globalne, 768
 - interakcja aktywności z przyciskiem, 780, 782
 - jawne wywoływanie, 787
 - kod źródłowy aktywności, 778
 - kursor propozycji, 772
 - menu aktywności, 784
 - metadane, 793
 - pliki aktywności, 778
 - pliki projektu, 777
 - pliki układu graficznego, 778
 - pole wyszukiwania, 769
 - propozycje wyszukiwania, 771, 772
 - propozycje zerowe, 771
 - SearchInvokerActivity, 789
 - tryb propozycji zerowych, 771
 - typy aktywności, 777
 - układ graficzny aktywności, 781
 - ustawienia, 775, 777
 - wywoływanie aplikacji, 774
- wyszukiwanie globalne, 769
 - dostawcy propozycji, 774
- wyszukiwanie lokalne, 769, 790, 811
 - aktywność, 795
 - pole wyszukiwania, 796
 - wyniki wyszukiwania, 796
- wyszukiwanie w Androidzie, 768
- wywołanie dostawcy widżetu, 1117
- wywołanie obiektu Cursor, 132
- wywoływanie ekranu startowego, 166
- wywoływanie usługi, 391
- wzorce identyfikatorów URI, 148

X

XUL, XML User Interface Language, 39

Z

zabezpieczenia, 317

 certyfikat cyfrowy, 318

 stosowanie uprawnień, 325

zabezpieczenia na granicach procesu, 324

zabezpieczenia środowiska wykonawczego, 324

zakładka

 Permissions, 328

 Virtual devices, 83

 Window/Android SDK and AVD

 Manager, 58

zasady postępowania w Android Market, 1006

zasoby, 40, 91

 a zmiany konfiguracji, 112

 alternatywne, 113

 Androida, 98

 childLayout, 203

 domyślne, 113

 identyfikatory zasobów, 114

 nieskompresowane, 111

 obrazów w języku XML, 107

 plurals, 101

 R.drawable.frame_animation, 523

 typu Color, 104

 typu color w kodzie Java, 105

 typu color-drawable, 108

 typu color-drawable w kodzie Java, 108

 typu color-drawable w kodzie XML, 108

 typu dimension, 105

 typu dimension w kodzie Java, 106

 typu dimension w kodzie XML, 106

 typu drawable, 196

 typu image, 106

 typu image w środowisku Java, 107

 typu layout, 94

 typu string, 92, 95, 103

 typu string w języku XML, 104

 typu string w kodzie Java, 103

 wielokrotność, 102

 zastępowanie metody funkcją, 607

 zaufany wydawca certyfikatów, certificate authority, CA, 318

 zdalne wywołanie procedury, 368

 zdalny układ graficzny, 1109

 wczytywanie, 1111

 zdarzenia dotyku, 888

 zdarzenie ACTION_MOVE, 885

 zintegrowane przeszukiwanie Androida, 42

 zmienna

 ignoreLastFinger, 894

 PATH, 55

 systemowa PATH, 321

 środowiskowa JAVA_HOME, 53

 środowiskowa PATH, 55

 znacznik

 <activity>, 227

 <application>, 227

 <big>, 224

 <monospace>, 224

 <small>, 224

 <strike>, 224

 <sub>, 224

 <sup>, 224

 <uses-permissions>, 334

 accelerateInterpolator, 532

 group, 268

 ikony menu, 271

 kategorii grupy, 271

 menu, 268

 showAsAction, 1092

 uses-permission, 625

 włączania menu, 272

 wyłączania menu, 272

 zaznaczania, 271

Ż

 żądanie metody GET, 340

 żądanie metody POST, 340

 żądanie typu MIME, 129

PROGRAM PARTNERSKI

GRUPY WYDAWNICZEJ HELION



- 1. ZAREJESTRUJ SIĘ**
- 2. PREZENTUJ KSIĄŻKI**
- 3. ZBIERAJ PROWIZJĘ**

Zmień swoją stronę WWW
w działający bankomat!

Dowiedz się więcej i dołącz już dzisiaj!

<http://program-partnerski.helion.pl>

GRUPA WYDAWNICZA

 **Helion SA**

System operacyjny Android podbił rynek smartfonów, a teraz z dnia na dzień rośnie jego popularność wśród użytkowników tabletów. Sympatię zdobył sobie dzięki niezwykle przyjaznemu interfejsowi, szerokim możliwościom dostosowania do własnych potrzeb oraz niewyobrażalnej liczbie dostępnych aplikacji. W Android Market opublikowano ich już ponad 500 tysięcy! Wśród nich każdy znajdzie coś dla siebie niezależnie od tego, czy ma to być gra, czy unikalna aplikacja użytkowa.

Oparty na powszechnie znanym języku Java, posiadający obszerną, bogatą w przykłady dokumentację wprost zachęca do przygotowania ciekawej aplikacji, na której można zarobić konkretne pieniądze. Dzięki tej rewelacyjnej książce poświęconej Androidowi w wersji 3 w mig opanujesz jego tajniki. Na samym początku poznasz historię Androida oraz dowiesz się, jak przygotować środowisko pracy. Następnie zaznajomisz się ze strukturą aplikacji, sposobem korzystania z zasobów oraz dostawców treści. W dalszych rozdziałach nauczysz się budować estetyczny, funkcjonalny i atrakcyjny interfejs użytkownika, zapamiętywać preferencje użytkowników oraz korzystać z usług HTTP. Książka ta jest kompletnym i unikalnym kompendium wiedzy na temat Androida. Powinna znaleźć się na półce każdego dewelopera tworzącego oprogramowanie dla platformy Android!

- Przygotowanie środowiska pracy
- Pisanie aplikacji opartych na środowisku Java
- Projektowanie i budowanie interfejsu użytkownika
- Wysyłanie i odbieranie komunikatów
- Tworzenie animacji dwuwymiarowej
- Korzystanie z usług geolokalizacyjnych
- Przetwarzanie tekstu na mowę
- Publikacja aplikacji w Android Market

Zacznij tworzyć oprogramowanie dla najbardziej obiecującej platformy dla urządzeń mobilnych!

helion.pl
księgarnia
internetowa

Nr katalogowy: 7704



Księgarnia internetowa
<http://helion.pl>



Zamówienia telefoniczne:
0 801 339900



0 601 339900

Apress®



Helion

Sprawdź najnowsze promocje:
● <http://helion.pl/promocje>
Książki najchętniej czytane:
● <http://helion.pl/bestsellery>
Zamów informacje o nowościach:
● <http://helion.pl/nowości>

Helion SA
ul. Kościuszki 1c, 44-100 Gliwice
tel.: 32 230 98 63
e-mail: helion@helion.pl
<http://helion.pl>

sięgnij po **WIĘCEJ**



KOD KORZYŚCI

ISBN 978-83-246-3586-3



9 788324 635863

Cena: 149,00 zł

Informatyka w najlepszym wydaniu