

ERIK HELLMAN

# PLATFORMA ANDROID

NOWE WYZWANIA



Profesjonalne  
programowanie  
w systemie **Android!**

Tytuł oryginału: Android Programming: Pushing the Limits

Tłumaczenie: Łukasz Piwko

ISBN: 978-83-246-9525-6

© 2014 Erik Hellman

All Rights Reserved. Authorised translation from the English language edition published by John Wiley & Sons Limited. Responsibility for the accuracy of the translation rests solely with Helion S.A. and is not the responsibility of John Wiley & Sons Limited.

No part of this book may be reproduced in any form without the written permission of the original copyright holder, John Wiley & Sons Limited.

Translation copyright © 2014 by Helion S.A.

Wiley and the Wiley logo are trademarks or registered trademarks of John Wiley and Sons, Inc. and/or its affiliates in the United States and/or other countries, and may not be used without written permission. Android is a trademark of Google, Inc. All other trademarks are the property of their respective owners. John Wiley & Sons, Ltd. is not associated with any product or vendor mentioned in the book.

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 bierze jednak żadnej odpowiedzialności ani za ich wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych lub autorskich. Wydawnictwo HELION nie ponosi również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

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

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

Drogi Czytelniku!  
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres  
<http://helion.pl/user/opinie/plandr>  
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

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

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

# Spis treści

<b>O autorze</b> .....	<b>13</b>
<b>Wprowadzenie</b> .....	<b>15</b>
<b>Część I: Solidne podstawy</b> .....	<b>21</b>
<b>Rozdział 1. Konfigurowanie środowiska programistycznego</b> .....	<b>23</b>
<b>Systemy operacyjne do programowania Androida</b> .....	<b>23</b>
<b>Zaawansowane narzędzia SDK Androida</b> .....	<b>24</b>
Narzędzie adb .....	24
Testowanie przeciążeniowe interfejsu użytkownika przy użyciu Monkey .....	27
System kompilacji Gradle .....	28
Optymalizacja i zaciemnianie za pomocą ProGuard .....	30
<b>Projekty bibliotek Android i biblioteki zewnętrzne</b> .....	<b>31</b>
Skompilowane pliki JAR .....	31
Tworzenie projektu biblioteki .....	32
<b>Kontrola wersji i zarządzanie kodem źródłowym</b> .....	<b>33</b>
<b>Środowisko programistyczne</b> .....	<b>35</b>
Diagnozowanie aplikacji Android .....	35
Statyczna analiza kodu przy użyciu lint .....	37
Refaktoryzacja kodu .....	39
<b>Opcje programistyczne w urządzeniach z Androidem</b> .....	<b>41</b>
Ustawienia programistyczne .....	42
<b>Podsumowanie</b> .....	<b>44</b>
<b>Dodatkowe źródła informacji</b> .....	<b>44</b>
Książki .....	44
Strony internetowe .....	44
<b>Rozdział 2. Efektywne programowanie w Javie na Androidzie</b> .....	<b>45</b>
<b>Porównanie Javy Dalvik z Javą SE</b> .....	<b>45</b>
<b>Optymalizacja kodu Java dla Androida</b> .....	<b>47</b>
Bezpieczne pod względem typów wyliczenia w Androidzie .....	48
Udoskonalona pętla for w Androidzie .....	49
Kolejki, synchronizacja i blokady .....	50
<b>Zarządzanie pamięcią i alokacją</b> .....	<b>52</b>
Ograniczanie liczby alokacji obiektów .....	52

<b>Wielowątkowość w Androidzie</b> .....	<b>55</b>
Klasa Thread .....	56
Klasa AsyncTask .....	57
Klasa Handler .....	58
Wykonywanie operacji w regularnych odstępach czasu .....	60
Używanie klasy MainLooper w połączeniu z Handler .....	61
<b>Jak wybrać najlepsze rozwiązanie</b> .....	<b>62</b>
<b>Podsumowanie</b> .....	<b>63</b>
<b>Dodatkowe źródła informacji</b> .....	<b>63</b>
Dokumentacja .....	63
Książki .....	63
Strony internetowe .....	63
<b>Część II: Efektywne wykorzystanie składników</b> .....	<b>65</b>
<b>Rozdział 3. Składniki, manifesty i zasoby</b> .....	<b>67</b>
<b>Składniki Androida</b> .....	<b>67</b>
Składnik Activity .....	68
Składnik Service .....	68
Składnik BroadcastReceiver .....	68
Składnik ContentProvider .....	69
Składnik Application .....	69
Architektura aplikacji .....	71
<b>Manifest aplikacji Android</b> .....	<b>72</b>
Element manifest .....	72
Filtry i uprawnienia Google Play .....	73
Element application .....	75
Elementy i atrybuty składników .....	76
Filtrowanie intencji .....	77
<b>Zasoby i środki</b> .....	<b>78</b>
Zaawansowane zasoby łańcuchowe .....	79
Lokalizacja .....	80
Kwalifikatory zasobów .....	80
Środki .....	82
<b>Podsumowanie</b> .....	<b>83</b>
<b>Dodatkowe źródła informacji</b> .....	<b>83</b>
Dokumentacja .....	83
<b>Rozdział 4. Interakcja z użytkownikiem i projektowanie interfejsów</b> .....	<b>85</b>
<b>Historyjki użytkowników</b> .....	<b>85</b>
Użytkownicy i postaci .....	86
<b>Projektowanie interfejsu użytkownika Androida</b> .....	<b>87</b>
Nawigacja .....	87
Prototypowanie interfejsu użytkownika .....	88
Projektowanie interfejsu użytkownika w Android Studio .....	88
<b>Elementy interfejsu użytkownika Androida</b> .....	<b>89</b>
<b>Tekst w aplikacjach na Androida</b> .....	<b>89</b>
Krój pisma .....	89
Układ tekstu .....	90

<b>Wymiary i rozmiary</b> .....	<b>90</b>
Zalecane wymiary .....	91
Rozmiary ikon .....	91
Rozmiar tekstu .....	92
Kolory .....	92
Daltonizm .....	93
<b>Obrazy i ikony</b> .....	<b>93</b>
Perspektywa kanoniczna .....	94
Geony .....	94
Rozpoznawanie twarzy .....	95
<b>Walory użytkowe</b> .....	<b>95</b>
Wskazówki wizualne .....	95
<b>Nagradzanie użytkownika</b> .....	<b>96</b>
Grywalizacja .....	96
<b>Podsumowanie</b> .....	<b>97</b>
<b>Dodatkowe źródła informacji</b> .....	<b>98</b>
Książki .....	98
Strony internetowe .....	98
<b>Rozdział 5. Interfejs użytkownika dla zaawansowanych</b> .....	<b>99</b>
<b>Aktywności i fragmenty</b> .....	<b>99</b>
<b>Używanie wielu ekranów</b> .....	<b>100</b>
<b>Projektowanie własnych widoków</b> .....	<b>102</b>
Cykl życia widoku .....	103
Widżet o wyglądzie klawiatury fortepianowej .....	103
<b>Wielodotyk</b> .....	<b>108</b>
Współrzędne wskaźnika .....	110
Gest obrotu .....	111
<b>OpenGL ES</b> .....	<b>112</b>
Grafy sceny i silniki gier .....	113
<b>Podsumowanie</b> .....	<b>113</b>
<b>Dodatkowe źródła informacji</b> .....	<b>114</b>
Książki .....	114
Strony internetowe .....	114
<b>Rozdział 6. Usługi i zadania w tle</b> .....	<b>115</b>
<b>Jak i kiedy używać usług</b> .....	<b>115</b>
Typy usług .....	116
<b>Cykl życia usług</b> .....	<b>116</b>
Tworzenie i niszczenie usług .....	116
Uruchamianie usług .....	117
Wiązanie usług .....	118
Utrzymywanie się przy życiu .....	120
Zatrzymywanie usług .....	121
<b>Działanie w tle</b> .....	<b>123</b>
Klasa IntentService .....	123
Równoległe wykonywanie .....	124
<b>Komunikacja z usługami</b> .....	<b>126</b>
Asynchroniczne wysyłanie powiadomień przy użyciu intencji .....	126
Lokalne wiązanie usług .....	127

Podsumowanie .....	130
Dodatkowe źródła informacji .....	131
Blogi .....	131
<b>Rozdział 7. Komunikacja międzyprocesowa .....</b>	<b>133</b>
<b>Binder .....</b>	<b>134</b>
Adres Bindera .....	134
Transakcje Bindera .....	135
Klasa Parcel .....	136
Łącze ze śmiercią .....	138
<b>Projektowanie API .....</b>	<b>138</b>
AIDL .....	139
Wywołania zwrotne przy użyciu AIDL .....	142
Klasa Messenger .....	143
Opakowywanie API w biblioteki .....	146
Zabezpieczanie zdalnych API .....	148
<b>Podsumowanie .....</b>	<b>149</b>
<b>Dodatkowe źródła informacji .....</b>	<b>149</b>
Strony internetowe .....	149
<b>Rozdział 8. Odbieranie komunikatów i zmiany konfiguracji .....</b>	<b>151</b>
<b>Klasa BroadcastReceiver .....</b>	<b>152</b>
Lokalne odbiorniki powiadomień .....	153
Powiadomienia normalne i uporządkowane .....	154
Powiadomienia kleiste .....	155
Powiadomienia bezpośrednie .....	156
Włączanie i wyłączanie odbiorników .....	156
Intencje rozsyłane przez system .....	157
<b>Zmianie konfiguracji urządzenia .....</b>	<b>160</b>
<b>Podsumowanie .....</b>	<b>161</b>
<b>Dodatkowe źródła informacji .....</b>	<b>161</b>
Dokumentacja .....	161
<b>Rozdział 9. Przechowywanie i serializacja danych .....</b>	<b>163</b>
<b>Techniki utrwalania danych w Androidzie .....</b>	<b>163</b>
<b>Przechowywanie danych w plikach preferencji .....</b>	<b>164</b>
<b>Opcje użytkownika i ustawienia interfejsu użytkownika .....</b>	<b>166</b>
<b>Dostawcy treści o wysokiej wydajności .....</b>	<b>168</b>
Projektowanie bazy danych w Androidzie .....	168
Tworzenie i aktualizowanie baz danych .....	169
Implementacja metod zapytaniowych .....	171
Transakcje bazy danych .....	172
Zapisywanie danych binarnych w ContentProvider .....	174
<b>Serializacja danych do utrwalania .....</b>	<b>175</b>
JSON .....	175
Zaawansowana obsługa JSON przy użyciu Gson .....	177
Protocol Buffers Google .....	179
<b>Robienie kopii zapasowej danych aplikacji .....</b>	<b>181</b>
<b>Podsumowanie .....</b>	<b>183</b>

<b>Dodatkowe źródła informacji</b> .....	<b>183</b>
Dokumentacja .....	183
Strony internetowe .....	183
<b>Rozdział 10. Pisanie automatycznych testów</b> .....	<b>185</b>
<b>Zasady testowania w Androidzie</b> .....	<b>185</b>
Co testować .....	186
Podstawowe testy jednostkowe .....	187
Testowanie aktywności .....	188
Testowanie usług .....	190
Testowanie dostawców treści .....	191
Wykonywanie testów .....	193
<b>Ciągła integracja</b> .....	<b>194</b>
<b>Podsumowanie</b> .....	<b>195</b>
<b>Dodatkowe źródła informacji</b> .....	<b>195</b>
Porady dotyczące testowania .....	195
Książki .....	195
Strony internetowe .....	195
<b>Część III: Przekraczanie barier</b> .....	<b>197</b>
<b>Rozdział 11. Zaawansowane aplikacje obsługujące audio, wideo oraz kamerę</b> .....	<b>199</b>
<b>Zaawansowane aplikacje dźwiękowe</b> .....	<b>199</b>
Odtwarzanie dźwięków z krótkim opóźnieniem .....	200
OpenSL ES dla Androida .....	202
Czytanie tekstu na głos .....	205
Rozpoznawanie mowy .....	207
<b>Przetwarzanie filmów przy użyciu OpenGL ES 2.0</b> .....	<b>209</b>
<b>Przetwarzanie obrazu z kamery przy użyciu OpenGL ES 2.0</b> .....	<b>212</b>
<b>Kodowanie mediów</b> .....	<b>213</b>
Nagrywanie sceny OpenGL .....	214
<b>Podsumowanie</b> .....	<b>218</b>
<b>Dodatkowe źródła informacji</b> .....	<b>218</b>
Dokumentacja .....	218
<b>Rozdział 12. Bezpieczeństwo aplikacji na Androidzie</b> .....	<b>219</b>
<b>Pojęcia dotyczące bezpieczeństwa</b> .....	<b>219</b>
Klucze i podpisy .....	219
Uprawnienia w Androidzie .....	220
Ochrona danych użytkownika .....	221
Weryfikowanie wywołujących aplikacji .....	222
<b>Szyfrowanie danych u klienta</b> .....	<b>223</b>
API crypto .....	223
Generowanie klucza .....	223
Szyfrowanie danych .....	223
Deszyfrowanie danych .....	224
Praca z zaszyfrowanymi danymi .....	225
<b>Zarządzanie pękami kluczy</b> .....	<b>226</b>
<b>API zarządzania urządzeniem</b> .....	<b>229</b>
<b>Podsumowanie</b> .....	<b>232</b>

<b>Dodatkowe źródła informacji</b> .....	<b>232</b>
Książki .....	232
Dokumentacja .....	232
Strony internetowe .....	232
<b>Rozdział 13. Mapy, lokalizacja i API aktywności</b> .....	<b>233</b>
<b>Scalony menedżer lokalizacji</b> .....	<b>233</b>
<b>Integracja API map Google</b> .....	<b>234</b>
<b>Korzystanie z map Google</b> .....	<b>236</b>
Znaczniki na mapach .....	237
Rysowanie kótek .....	238
Rysowanie wielokątów .....	239
Przydatne narzędzia API lokalizacyjnego .....	240
Geokodowanie .....	240
<b>Klasa LocationClient</b> .....	<b>241</b>
<b>Geofencing</b> .....	<b>242</b>
<b>Rozpoznawanie aktywności</b> .....	<b>244</b>
<b>Podsumowanie</b> .....	<b>246</b>
<b>Dodatkowe źródła informacji</b> .....	<b>246</b>
Dokumentacja .....	246
Strony internetowe .....	246
<b>Rozdział 14. Kod macierzysty i JNI</b> .....	<b>247</b>
<b>Kilka uwag o architekturze CPU</b> .....	<b>247</b>
<b>Pisanie aplikacji na Androida w języku C</b> .....	<b>248</b>
Skrypty kompilacji NDK .....	248
Macierzyste aktywności .....	249
<b>Macierzysty interfejs Javy</b> .....	<b>250</b>
Wywoływanie funkcji macierzystych w Javie .....	250
Wywoływanie metod Javy z kodu macierzystego .....	252
<b>Macierzyste API Androida</b> .....	<b>255</b>
Biblioteka C .....	256
Macierzyste funkcje obsługi dziennika .....	256
OpenGL ES 2.0 w kodzie macierzystym .....	256
OpenSL ES w kodzie macierzystym .....	256
<b>Przenoszenie macierzystej biblioteki do Androida</b> .....	<b>257</b>
Kompilowanie macierzystej biblioteki .....	257
<b>Podsumowanie</b> .....	<b>262</b>
<b>Dodatkowe źródła informacji</b> .....	<b>263</b>
Portal dla programistów Androida .....	263
Strony internetowe firmy Oracle .....	263
<b>Rozdział 15. Ukryte API Androida</b> .....	<b>265</b>
<b>API ukryte i oficjalne</b> .....	<b>265</b>
<b>Wyszukiwanie ukrytych API</b> .....	<b>266</b>
<b>Bezpieczne wywoływanie ukrytych API</b> .....	<b>267</b>
Wydobywanie ukrytych API z urządzenia .....	268
Wywoływanie ukrytych API przy użyciu refleksji .....	270



<b>Przykłady ukrytych API</b> .....	<b>271</b>
Odbieranie i czytanie SMS-ów .....	271
Tethering Wi-Fi .....	272
Ukryte ustawienia .....	273
<b>Podsumowanie</b> .....	<b>274</b>
<b>Dodatkowe źródła informacji</b> .....	<b>274</b>
Strony internetowe .....	274
<b>Rozdział 16. Hakowanie platformy Android</b> .....	<b>275</b>
<b>Odblokowywanie urządzenia</b> .....	<b>276</b>
Wgrywanie obrazów fabrycznych .....	277
Odblokowywanie urządzeń innych niż Google Nexus .....	277
Oprogramowanie układowe od grup programistów .....	278
<b>Kod źródłowy Androida</b> .....	<b>278</b>
Konfiguracja środowiska kompilacji .....	278
Kompilowanie i wgrywanie oprogramowania układowego .....	279
<b>Pisanie aplikacji systemowych</b> .....	<b>280</b>
Certyfikaty platformy .....	281
Podpisywanie aplikacji .....	282
<b>Hakowanie platformy Android</b> .....	<b>283</b>
Konfiguracje środowiska programistycznego .....	283
Projekty Android .....	284
Jądro Linuksa w Androidzie .....	285
Dodawanie usługi systemowej .....	285
Przyspieszanie prac nad platformą .....	288
<b>Własny wkład w rozwój AOSP</b> .....	<b>289</b>
<b>Podsumowanie</b> .....	<b>289</b>
<b>Dodatkowe źródła informacji</b> .....	<b>291</b>
Strony internetowe .....	291
<b>Rozdział 17. Praca w sieci, usługi sieciowe oraz zdalne API</b> .....	<b>293</b>
<b>Korzystanie z sieci w Androidzie</b> .....	<b>293</b>
Klasa HttpURLConnection .....	294
Volley .....	297
OkHttp i SPDY .....	299
Gniazda sieciowe .....	300
<b>Integracja usług sieciowych</b> .....	<b>303</b>
Static Maps v2 Google .....	303
Używanie OAuth2 w API Foursquare .....	305
SDK Facebooka dla Androida .....	308
Znajdowanie usług sieciowych i API .....	312
<b>Korzystanie z sieci a zużycie energii</b> .....	<b>312</b>
Ogólne wskazówki .....	313
Energooszczędne odpytywanie sieci .....	314
Powiadamianie przez serwer .....	314
<b>Podsumowanie</b> .....	<b>317</b>
<b>Dodatkowe źródła informacji</b> .....	<b>317</b>
Dokumentacja .....	317
Strony internetowe .....	317

<b>Rozdział 18. Komunikacja ze zdalnymi urządzeniami</b> .....	<b>319</b>
Technologie łączności Androida .....	319
<b>USB w Androidzie</b> .....	<b>320</b>
<b>Bluetooth LE</b> .....	<b>322</b>
<b>Wi-Fi w Androidzie</b> .....	<b>325</b>
Wykrywanie usług sieciowych .....	325
Wi-Fi Direct .....	326
<b>Usługi sieciowe na urządzeniach</b> .....	<b>329</b>
Tworzenie usług typu RESTful przy użyciu Restlet .....	329
Serwer WebSocket .....	332
<b>Podsumowanie</b> .....	<b>334</b>
<b>Dodatkowe źródła informacji</b> .....	<b>335</b>
Strony internetowe .....	335
<b>Rozdział 19. Usługi Google Play</b> .....	<b>337</b>
<b>Uwierzytelnianie</b> .....	<b>337</b>
<b>Dane aplikacji z dysku Google</b> .....	<b>340</b>
<b>Zakończenia Google Cloud</b> .....	<b>343</b>
<b>Google Cloud Messaging</b> .....	<b>345</b>
Klient GCM .....	346
<b>Usługi Google Play dla gier</b> .....	<b>349</b>
Przesyłanie danych w powiadomieniach .....	351
Strategia przesyłania wiadomości .....	352
<b>Podsumowanie</b> .....	<b>353</b>
<b>Dodatkowe źródła informacji</b> .....	<b>354</b>
Strony internetowe .....	354
<b>Rozdział 20. Dystrybucja aplikacji w sklepie Google Play</b> .....	<b>355</b>
<b>Pobieranie opłat wewnątrz aplikacji</b> .....	<b>356</b>
Konsumowanie produktów .....	358
Subskrypcje wewnątrz aplikacji .....	358
<b>Reklamy w aplikacjach na Androida</b> .....	<b>359</b>
Kierowanie reklam .....	360
Kolory reklam .....	361
Reklamy pełnoekranowe .....	361
<b>Licencjonowanie aplikacji</b> .....	<b>362</b>
<b>Pliki rozszerzeń APK</b> .....	<b>363</b>
Tworzenie plików rozszerzeń .....	363
Pobieranie plików rozszerzeń .....	363
<b>Podsumowanie</b> .....	<b>365</b>
<b>Dodatkowe źródła informacji</b> .....	<b>365</b>
Strony internetowe .....	365
<b>Skorowidz</b> .....	<b>367</b>

## Rozdział 11.

# Zaawansowane aplikacje obsługujące audio, wideo oraz kamerę

Podczas gdy do pracy z dźwiękiem, filmami i kamerą wystarczą podstawowe API, w zaawansowanych aplikacjach często potrzebne są dodatkowe funkcje. Do zwykłego nagrywania i odtwarzania mediów wystarczą dostępne w Androidzie API Javy. Niestety, opóźnienia powodowane przez te wysokopoziomowe API mogą być czasami nie do przyjęcia. Poza tym można mieć trudności w czasie pracy z aplikacjami wykorzystującymi kamerę, takimi jak aplikacje rozszerzonej rzeczywistości i inne, w których potrzebne jest reagowanie programu na bieżąco.

W tym rozdziale dowiesz się, jak maksymalnie wykorzystać różne właściwości mediów w Androidzie. W systemie tym można używać zarówno wysokowydajnych, jak i niskopoziomowych API, takich jak OpenGL ES do grafiki czy OpenSL ES do dźwięku. Nauczysz się używać tych API w kombinacji i poznasz kilka przypadków użycia zaawansowanych funkcji dźwiękowych, wideo i kamery.

---

Niektóre przedstawione w tym rozdziale przykłady (m.in. ten odnoszący się do OpenSL ES) zawierają kod w języku C. Jeśli nie wiesz, jak się używa NDK Androida do budowy aplikacji przy użyciu rodzimego kodu C, możesz przed tym rozdziałem przeczytać rozdział 14.

---

## Zaawansowane aplikacje dźwiękowe

Pierwszymi aplikacjami dźwiękowymi w Androidzie były odtwarzacze muzyki. Można było włączyć strumieniowe odtwarzanie piosenek z internetu albo odtworzyć utwory posiadane w pamięci urządzenia. Jednak w miarę jak platforma Android się rozwijała, zaczęły być potrzebne bardziej zaawansowane API do obsługi dźwięku. Dlatego firma Google postanowiła dodać API umożliwiające strumieniowanie i nagrywanie dźwięku z małym opóźnieniem.

Obecnie dostępne w Androidzie API dźwiękowe obsługują zaawansowane funkcje, których można używać w aplikacjach. Teraz już nie jest trudne zaimplementowanie aplikacji VoIP (ang. *voice over internet protocol*), zbudowanie własnego strumieniowego klienta muzycznego czy zaimplementowanie nieopóźnionych odgłosów w grach. Ponadto dostępne są API umożliwiające korzystanie z funkcji czytania tekstu na głos i rozpoznawania mowy, dzięki którym użytkownicy mogą obsługiwać aplikacje za pomocą dźwięków zamiast wizualnego interfejsu użytkownika lub technologii dotykowych.

W tym podrozdziale objaśnię, jak używać tych funkcji, oraz przedstawię przykłady kodu, dzięki którym łatwiej Ci będzie rozpocząć pracę z nimi.

## Odtwarzanie dźwięków z krótkim opóźnieniem

System Android zawiera cztery API do odtwarzania dźwięków (a nawet pięć, jeśli doliczy się MIDI) i trzy do ich nagrywania. W kolejnych kilku sekcjach znajduje się przegląd tych API wraz z przykładami kodu ilustrującymi sposób użycia bardziej zaawansowanych funkcji.

### API do odtwarzania dźwięku

Domyślną klasą do obsługi odtwarzania dźwięku jest `MediaPlayer`. Klasa ta zawiera podstawowe funkcje dotyczące odtwarzania dźwięku odpowiednie do użycia w odtwarzaczach muzyki (lub filmów) i udostępnia wygodne API do odtwarzania dźwięków zarówno ze źródeł strumieniowych (np. radia internetowego), jak i z lokalnych plików. Z klasą `MediaPlayer` związana jest zaawansowana maszyna stanów, którą musi śledzić aplikacja. Przy użyciu tego API można dodać do aplikacji odtwarzacz muzyki lub filmów i nie martwić się o dodatkowe przetwarzanie ani opóźnienia.

Inną możliwością w kwestii odtwarzania dźwięków jest klasa `SoundPool`, która charakteryzuje się odtwarzaniem dźwięku z bardzo małym opóźnieniem, a więc jest odpowiednia do odtwarzania odgłosów i innych krótkich dźwięków. API tego używa się np. do włączania odgłosów w grach. Brak w nim jednak obsługi strumieniowania dźwięku, więc nie nadaje się do użycia w aplikacjach odtwarzających strumienie dźwiękowe, takich jak VoIP.

Trzecią możliwość odtwarzania dźwięku reprezentuje klasa `AudioTrack`, która umożliwia przekazanie do sprzętu pojedynczego strumienia dźwiękowego w postaci surowych buforów PCM. Rozwiązanie to zapewnia niskie opóźnienia nawet przy strumieniowaniu. Klasa `AudioTrack` jest tak efektywna, że może być z powodzeniem używana w VoIP i podobnych technologiach.

Poniżej znajduje się przykład użycia klasy `AudioTrack` w typowej aplikacji VoIP:

```
public class AudioTrackDemo {

    private final AudioTrack mAudioTrack;
    private final int mMinBufferSize;

    public AudioTrackDemo() {
        mMinBufferSize = AudioTrack.getMinBufferSize(16000,
            AudioFormat.CHANNEL_OUT_MONO,
            AudioFormat.ENCODING_PCM_16BIT);
        mAudioTrack = new AudioTrack(AudioManager.STREAM_VOICE_CALL,
            16000,
            AudioFormat.CHANNEL_OUT_MONO,
            AudioFormat.ENCODING_PCM_16BIT,
            mMinBufferSize * 2,
            AudioTrack.MODE_STREAM);
    }

    public void playPcmPacket(byte[] pcmData) {
        if(mAudioTrack != null
            && mAudioTrack.getState() == AudioTrack.STATE_INITIALIZED)
        {
            if(mAudioTrack.getPlaybackRate()
                != AudioTrack.PLAYSTATE_PLAYING) {
                mAudioTrack.play();
            }
        }
    }
}
```

```

    }
    mAudioTrack.write(pcmData, 0, pcmData.length);
}
}

public void stopPlayback() {
    if(mAudioTrack != null) {
        mAudioTrack.stop();
        mAudioTrack.release();
    }
}
}
}

```

Najpierw określiłem minimalny rozmiar bufora dla strumienia audio. Aby to zrobić, trzeba znać częstotliwość próbkowania, wiedzieć, czy sygnał jest mono-, czy stereofoniczny oraz czy będzie używane 8-, czy 16-bitowe kodowanie PCM. Częstotliwość próbkowania i rozmiar próbki są parametrami metody `AudioTrack.getMinBufferSize()`, która zwraca minimalny rozmiar bufora w bajtach dla danego egzemplarza klasy `AudioTrack`.

Egzemplarz klasy `AudioTrack` tworzy się przy użyciu parametrów dostosowanych do potrzeb programisty. Pierwszy parametr dostosowuje się do typu używanego sygnału dźwiękowego. W przypadku aplikacji VoIP należy użyć `STREAM_VOICE_CALL`, a do strumieniowania muzyki — `STREAM_MUSIC`.

Drugi, trzeci i czwarty parametr zmieniają się w zależności od sytuacji. Służą one odpowiednio do określania częstotliwości próbkowania, czy sygnał jest stereo, czy mono, oraz rozmiaru próbkowania. W aplikacji VoIP można by było zastosować 16 kHz przy 16 bitach mono, natomiast w odtwarzaczu muzyki lepsze byłoby ustawienie 44,1 kHz przy 16 bitach stereo. Wyższa częstotliwość próbkowania przy 16 bitach stereo wymaga zastosowania większego bufora oraz sprawniejszego przesyłania danych, ale pozwala uzyskać lepszą jakość. Wszystkie urządzenia z Androidem obsługują odtwarzanie sygnału PCM o częstotliwości próbkowania 8, 16 i 44,1 kHz, z rozmiarem próbki 8 i 16 bitów stereo.

Wartość parametru określającego rozmiar bufora `AudioTrack` powinna być wielokrotnością minimalnego rozmiaru bufora. Wybór konkretnej wartości zależy od potrzeb, a także od opóźnień sieciowych i innych czynników.

---

**Nigdy** nie dopuszczaj, aby bufor był pusty, bo spowoduje to powstanie zakłóceń podczas odtwarzania dźwięku.

---

Ostatni parametr konstruktora określa, czy dane akustyczne mają zostać wysłane raz (`MODE_STATIC`), czy jako ciągły strumień (`MODE_STREAM`). W pierwszym przypadku konieczne jest przestanie całości. Natomiast w przypadku strumieniowania można wysłać porcje danych PCM o dowolnym rozmiarze, co jest bardzo przydatne w aplikacjach odtwarzających strumienie muzyki i VoIP.

## API do rejestrowania dźwięku

Pierwsze API do rejestrowania dźwięku (a także wideo) to `MediaRecorder`. Jest podobne do `MediaPlayer`, bo również utrzymuje wewnętrzny stan, który należy śledzić w kodzie. Za pomocą klasy `MediaRecorder` można tylko zapisywać przechwycone dane w pliku, a więc nie nadaje się ona do użytku w aplikacjach strumieniowych.

Jeśli rejestrowane dźwięki mają być przesyłane strumieniowo, można użyć API `AudioRecord`, które jest bardzo podobne do opisanego wyżej API `AudioTrack`.

Poniżej znajduje się przykład użycia klasy `AudioRecord` do nagrywania 16-bitowego sygnału monofonicznego o częstotliwości próbkowania 16 kHz:

```
public class AudioRecordDemo {

    private final AudioRecord mAudioRecord;
    private final int mMinBufferSize;
    private boolean mDoRecord = false;
    public AudioRecordDemo() {
        mMinBufferSize = AudioTrack.getMinBufferSize(16000, AudioFormat.CHANNEL_OUT_MONO,
            ↳AudioFormat.ENCODING_PCM_16BIT);
        mAudioRecord = new AudioRecord(
            MediaRecorder.AudioSource.VOICE_COMMUNICATION, 16000,
            ↳AudioFormat.CHANNEL_IN_MONO, AudioFormat.ENCODING_PCM_16BIT,
            ↳mMinBufferSize * 2);
    }

    public void writeAudioToStream(OutputStream stream) {
        mDoRecord = true;
        mAudioRecord.startRecording();
        byte[] buffer = new byte[mMinBufferSize * 2];
        while(mDoRecord) {
            int bytesWritten = mAudioRecord.read(buffer, 0, buffer.length);
            try {
                stream.write(buffer, 0, bytesWritten);
            } catch (IOException e) {
                // pominięte dla uproszczenia...
                mDoRecord = false;
            }
        }
        mAudioRecord.stop();
        mAudioRecord.release();
    }

    public void stopRecording() {
        mDoRecord = false;
    }
}
```

Jako że kod ten jest bardzo podobny do wcześniej pokazanego z użyciem klasy `AudioTrack`, można go wykorzystać w celu użycia kombinacji tych dwóch klas w aplikacjach VoIP itp.

## OpenSL ES dla Androida

Wcześniej napisałem, że istnieją cztery API do odtwarzania dźwięku i trzy do jego nagrywania. Do tej pory opisałem trzy z pierwszej grupy i dwa z drugiej. Natomiast w tej sekcji znajduje się opis ostatniego z tych API, o nazwie `OpenSL ES`, które służy zarówno do odtwarzania, jak i rejestrowania dźwięku. `OpenSL ES` to standard opracowywany przez `Khronos Group` (tę samą grupę, która zajmuje się standardem `OpenGL`).

API `OpenSL ES` zawiera niskopoziomowe narzędzia do pracy bezpośrednio ze sprzętem, które zapewniają bardzo małe opóźnienia w odtwarzaniu i nagrywaniu dźwięku. Podczas gdy pozostałe opisywane API mają wygodne interfejsy w `Javie`, `OpenSL ES` aktualnie dostępne jest wyłącznie w języku `C` poprzez `NDK` Androida. W tej sekcji opisuję tylko rodzimą część API `OpenSL ES`. Jeśli chcesz dowiedzieć się, jak przy użyciu `NDK` Androida pisać kod `JNI` do współpracy z kodem w `C`, przeczytaj rozdział 14.

---

Zakładam, że czytelnik zna podstawy programowania w języku `C`.

---

Pierwsza część przykładu użycia biblioteki OpenSL ES zawiera dyrektywy dołączające potrzebne pliki oraz deklaracje globalnych obiektów, które będą używane w dalszej części programu:

```
#include <pthread.h>

// potrzebne do obsługi OpenSL ES
#include <SLES/OpenSLES.h>
#include <SLES/OpenSLES_Android.h>

static pthread_cond_t s_cond;
static pthread_mutex_t s_mutex;

static SLObjectItf engineObject = NULL;
static SLEngineItf engineEngine;
static SLObjectItf outputMixObject = NULL;
static SLObjectItf bqPlayerObject = NULL;
static SLPlayItf bqPlayerPlay;
static SLAndroidSimpleBufferQueueItf bqPlayerBufferQueue;
```

Poniżej znajduje się funkcja zwrotna bqPlayerCallback(), która jest wywoływana przez OpenSL ES przy odtwarzaniu próbki:

```
static void waitForPlayerCallback()
{
    pthread_mutex_lock(&s_mutex);
    pthread_cond_wait(&s_cond, &s_mutex);
    pthread_mutex_unlock(&s_mutex);
}

SLresult enqueueNextSample(short* sample, int size, short waitForCallback)
{
    if(waitForCallback)
    {
        waitForPlayerCallback();
    }
    return (*bqPlayerBufferQueue)->Enqueue(bqPlayerBufferQueue,
                                           nextBuffer,
                                           nextSize);
}

void bqPlayerCallback(SLAndroidSimpleBufferQueueItf bq, void *context)
{
    pthread_cond_signal(&s_cond);
}
```

W tym przypadku przekazujemy bufor wyjściowy z innego wątku poprzez funkcję enqueueNextSample(). Do synchronizacji wątku, w którym następuje wywołanie zwrotne, z wątkiem wywołującym funkcję enqueue użyliśmy pthread\_mutex. Wywołanie funkcji waitForPlayerCallback() blokuje wątek do czasu wywołania funkcji pthread\_cond\_signal() w wywołaniu zwrotnym (podobnie działa blokowanie metodą Object.wait() w Javie).

W poniższym kodzie następuje inicjacja silnika OpenSL ES:

```
SLresult initOpenSLES()
{
    // Służy do sprawdzania wyniku każdej operacji...
    SLresult result;

    int speakers;
```

```

int channels = 2;

// Najpierw tworzymy muteks, który będzie potrzebny do odtwarzania później.
pthread_cond_init(&s_cond, NULL);
pthread_mutex_init(&s_mutex, NULL);

// utworzenie i realizacja silnika
result = slCreateEngine(&engineObject, 0, NULL, 0, NULL, NULL);
if(result != SL_RESULT_SUCCESS) return result;
result = (*engineObject)->Realize(engineObject, SL_BOOLEAN_FALSE);
if(result != SL_RESULT_SUCCESS) return result;
result = (*engineObject)->GetInterface(engineObject, SL_IID_ENGINE, &engineEngine);
if(result != SL_RESULT_SUCCESS) return result;

// utworzenie i realizacja miksera wyjściowego
const SLInterfaceID outputIds[1] = {SL_IID_VOLUME};
const SLboolean outputReq[1] = {SL_BOOLEAN_FALSE};
result = (*engineEngine)->CreateOutputMix(engineEngine, &outputMixObject, 1, outputIds,
↳outputReq);
if(result != SL_RESULT_SUCCESS) return result;
result = (*outputMixObject)->Realize(outputMixObject, SL_BOOLEAN_FALSE);
if(result != SL_RESULT_SUCCESS) return result;

// utworzenie bufora wyjściowego i ujęcia
SLDataLocator_AndroidSimpleBufferQueue bufferQueue =
↳{SL_DATALOCATOR_ANDROIDSIMPLEBUFFERQUEUE, 2};
speakers = SL_SPEAKER_FRONT_LEFT | SL_SPEAKER_FRONT_RIGHT;
SLDataFormat_PCM formatPcm = {SL_DATAFORMAT_PCM, channels, SL_SAMPLINGRATE_44_1,
↳SL_PCMSAMPLEFORMAT_FIXED_16, SL_PCMSAMPLEFORMAT_FIXED_16, speakers,
↳SL_BYTEORDER_LITTLEENDIAN};
SLDataSource audioSource = {&bufferQueue, &formatPcm};
SLDataLocator_OutputMix dataLocOutputMix = {SL_DATALOCATOR_OUTPUTMIX, outputMixObject};
SLDataSink audioSink = {dataLocOutputMix, NULL};

// utworzenie i realizacja obiektu odtwarzacza
const SLInterfaceID playerIds[] = {SL_IID_ANDROIDSIMPLEBUFFERQUEUE};
const SLboolean playerReq[] = {SL_BOOLEAN_TRUE};
result = (*engineEngine)->CreateAudioPlayer(engineEngine, &bqPlayerObject, &audioSource,
↳&audioSink, 1, playerIds, playerReq);
if(result != SL_RESULT_SUCCESS) return result;
result = (*bqPlayerObject)->Realize(bqPlayerObject, SL_BOOLEAN_FALSE);
if(result != SL_RESULT_SUCCESS) return result;
result = (*bqPlayerObject)->GetInterface(bqPlayerObject, SL_IID_PLAY, &bqPlayerPlay);
if(result != SL_RESULT_SUCCESS) return result;

// pobranie obiektu kolejki bufora odtwarzacza
result = (*bqPlayerObject)->GetInterface(bqPlayerObject, SL_IID_ANDROIDSIMPLEBUFFERQUEUE,
↳&bqPlayerBufferQueue);
if(result != SL_RESULT_SUCCESS) return result;

// rejestracja funkcji zwrotnej
result = (*bqPlayerBufferQueue)->RegisterCallback(bqPlayerBufferQueue, bqPlayerCallback,
↳NULL);
if(result != SL_RESULT_SUCCESS) return result;

return SL_RESULT_SUCCESS;
}

```

Wszystkie inicjacje w tym kodzie są wykonywane wg jednej podstawowej zasady: najpierw **tworzy** się obiekt (np. silnik), a następnie się go **realizuje**. Potem pobiera się **interfejs** do sterowania tym obiektem.



W powyższym przykładzie najpierw zostały utworzone obiekty silnika i odtwarzacza. Ponadto powstał obiekt bufora na próbki dźwiękowe przeznaczone do odtworzenia. Odtwarzacz jest przystosowany do odtwarzania 16-bitowego stereofonicznego (dwa kanały) dźwięku o częstotliwości 44,1 kHz. Na końcu znajduje się funkcja zwrotna dla odtwarzacza próbek, która zostanie wywołana, gdy odtwarzacz będzie gotowy do odbioru nowej próbki do odtworzenia.

Zwróć też uwagę, że na początku tej funkcji tworzone są obiekty związane z muteksem użytym w funkcji zwrotnej i w funkcji dodającej nowe próbki.

Ostatnia część tego przykładu zawiera dwie funkcje. Jedna służy do kontrolowania stanu obiektu odtwarzacza (odtworzenie i wstrzymanie), a druga do jego zamykania i porządkowania zasobów.

Poniżej znajduje się przykład użycia biblioteki OpenSL ES do odtwarzania dźwięków z bardzo małym opóźnieniem. API to jest niezastąpione, gdy wymagane są ekstremalnie krótkie opóźnienia oraz gdy większość kodu jest już gotowa i napisana w C. Do nagrywania używa się tego samego API w bardzo podobny sposób jak do odtwarzania, z tym że należy posługiwać się innymi interfejsami.

```
SLresult pausePlayback()
{
    return (*bqPlayerPlay)->SetPlayState(bqPlayerPlay, SL_PLAYSTATE_PAUSED);
}

SLresult startPlayback()
{
    return (*bqPlayerPlay)->SetPlayState(bqPlayerPlay, SL_PLAYSTATE_PLAYING);
}

void shutdownOpenSLES()
{
    if (bqPlayerObject != NULL) {
        (*bqPlayerObject)->Destroy(bqPlayerObject);
        bqPlayerObject = NULL;
        bqPlayerPlay = NULL;
        bqPlayerBufferQueue = NULL;
    }

    if (outputMixObject != NULL) {
        (*outputMixObject)->Destroy(outputMixObject);
        outputMixObject = NULL;
    }

    if (engineObject != NULL) {
        (*engineObject)->Destroy(engineObject);
        engineObject = NULL;
        engineEngine = NULL;
    }

    pthread_cond_destroy(&s_cond);
    pthread_mutex_destroy(&s_mutex);
}
```

## Czytanie tekstu na głos

Elementy wizualne pozwalają najszybciej dostarczyć informacje do użytkownika, wymagają one jednak jego pełnej uwagi. Jeśli użytkownik nie może patrzeć na urządzenie, trzeba znaleźć inny sposób, aby się z nim porozumiał. W systemie Android dostępne jest API do odczytywania tekstu na głos (TTS), za

pomocą którego można zastosować w aplikacji głosowe powiadomienia, niewymagające od użytkownika spoglądania na ekran.

Poniżej znajduje się przykład użycia API TTS:

```
public class TextToSpeechDemo implements TextToSpeech.OnInitListener {
    private final TextToSpeech mTextToSpeech;
    // do kolejkowania wiadomości zanim nastąpi inicjacja silnika TTS..
    private final ConcurrentLinkedQueue<String> mBufferedMessages;
    private Context mContext;
    private boolean mIsReady;

    public TextToSpeechDemo(Context context) {
        mContext = context;
        mBufferedMessages = new ConcurrentLinkedQueue<String>();
        mTextToSpeech = new TextToSpeech(mContext, this);
    }

    @Override
    public void onInit(int status) {
        if (status == TextToSpeech.SUCCESS) {
            mTextToSpeech.setLanguage(Locale.ENGLISH);
            synchronized (this) {
                mIsReady = true;
                for (String bufferedMessage : mBufferedMessages) {
                    speakText(bufferedMessage);
                }
                mBufferedMessages.clear();
            }
        }
    }

    public void release() {
        synchronized (this) {
            mTextToSpeech.shutdown();
            mIsReady = false;
        }
    }

    public void notifyNewMessages(int messageCount) {
        String message = mContext.getResources().getQuantityString(R.plurals.msg_count,
            ↪messageCount, messageCount);
        synchronized (this) {
            if (mIsReady) {
                speakText(message);
            } else {
                mBufferedMessages.add(message);
            }
        }
    }

    private void speakText(String message) {
        HashMap<String, String> params = new HashMap<String, String>();
        params.put(TextToSpeech.Engine.KEY_PARAM_STREAM, "STREAM_NOTIFICATION");
        mTextToSpeech.speak(message, TextToSpeech.QUEUE_ADD, params);
        mTextToSpeech.playSilence(100, TextToSpeech.QUEUE_ADD, params);
    }
}
```

Dzięki temu, że silnik TTS jest inicjowany asynchronicznie, można kolejkować wiadomości zanim wykona się operację odczytu tekstu na głos.

Do silnika TTS można wystać kilka parametrów. W powyższym przykładzie pokazałem, jak wybrać strumień dźwiękowy dla wiadomości głosowej. W tym przypadku użyto tego samego strumienia co dla dźwięków powiadomień.

Jeśli chcesz „wygłosić” kilka wiadomości, to po każdej z nich powinna następować krótka przerwa. Umożliwi to użytkownikowi rozróżnienie poszczególnych komunikatów.

## Rozpoznawanie mowy

Oprócz syntezy mowy Android umożliwia także jej rozpoznawanie. Ta funkcja jest nieco bardziej skomplikowana i jest dostępna w mniejszej liczbie języków niż API TTS, ale i tak umożliwia zastosowanie alternatywnego sposobu komunikacji z programem, gdy funkcje dotykowe są ograniczone.

**Uwaga:** aby używać funkcji rozpoznawania mowy, należy zadeklarować użytkownika z uprawnieniem `android.permission.RECORD_AUDIO`.

Poniżej znajduje się kod, na początku którego tworzony jest obiekt `SpeechRecognizer`, po czym zostaje ustawiona procedura nasłuchująca dla wywołania zwrotnego. Gdy zostaje wywołana procedura nasłuchująca kliknięć `doSpeechRecognition()`, następuje inicjacja rozpoznawania mowy w języku określonym parametrem `i` z flagą oznaczającą, że w czasie przetwarzania chcemy otrzymywać częściowe wyniki.

```
public class SpeechRecognitionDemo extends Activity {

    private SpeechRecognizer mSpeechRecognizer;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.speech_recognition_demo);
        mSpeechRecognizer = SpeechRecognizer.createSpeechRecognizer(this);
        mSpeechRecognizer.setRecognitionListener(new MyRecognitionListener());
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        mSpeechRecognizer.destroy();
    }

    public void doSpeechRecognition(View view) {
        view.setEnabled(false);
        Intent recognitionIntent = new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);
        recognitionIntent.putExtra(RecognizerIntent.EXTRA_PARTIAL_RESULTS, true);
        recognitionIntent.putExtra(RecognizerIntent.EXTRA_LANGUAGE, "en-US");

        mSpeechRecognizer.startListening(recognitionIntent);
    }

    private class MyRecognitionListener implements RecognitionListener {
        @Override
        public void onReadyForSpeech(Bundle bundle) {

```

```

@Override
public void onBeginningOfSpeech() {
    ((TextView) findViewById(R.id.speech_result)).setText("");
}

@Override
public void onRmsChanged(float rmsdB) {
    //nieużywana
}

@Override
public void onBufferReceived(byte[] bytes) {
}

@Override
public void onEndOfSpeech() {
    findViewById(R.id.do_speech_recognition_btn).setEnabled(true);
}

@Override
public void onError(int i) {
    // Coś się nie udało...
    findViewById(R.id.do_speech_recognition_btn).setEnabled(true);
}

@Override
public void onResults(Bundle bundle) {
    ArrayList<String> partialResults =
        ↳bundle.getStringArrayList(SpeechRecognizer.RESULTS_RECOGNITION);
    if (partialResults != null && partialResults.size() > 0) {
        String bestResult = partialResults.get(0);
        ((TextView) findViewById(R.id.speech_result)).setText(bestResult + ".");
    }
}

@Override
public void onPartialResults(Bundle bundle) {
    ArrayList<String> partialResults =
        ↳bundle.getStringArrayList(SpeechRecognizer.RESULTS_RECOGNITION);
    if (partialResults != null && partialResults.size() > 0) {
        String bestResult = partialResults.get(0);
        ((TextView) findViewById(R.id.speech_result)).setText(bestResult);
    }
}

@Override
public void onEvent(int i, Bundle bundle) {
    //nieużywana...
}
}
}

```

Procedura nasłuchująca otrzymuje wywołanie każdej metody po kolei. W tym przykładzie wyświetlamy częściowy wynik rozpoznawania (najlepsze dopasowanie) w metodzie `onPartialResults()` i kontynuujemy to aż do otrzymania ostatecznego wyniku w `onResult()`.

Bardziej zaawansowana aplikacja mogłaby interpretować słowa i nasłuchiwać konkretnych poleceń. Mogłaby rozpoznawać mowę, aż użytkownik kazałby jej przestać. Mogłoby to być przydatne w programie do dyktowania, w którym użytkownik mógłby powiedzieć „stop”, aby przerwać na chwilę dyktowanie zdań.

## Przetwarzanie filmów przy użyciu OpenGL ES 2.0

System Android obsługuje akcelerację przetwarzania grafiki przy użyciu bibliotek OpenGL ES 2.0 i 3.0. Mimo że biblioteka OpenGL jest już standardowo używana do renderowania i składania interfejsu użytkownika w Androidzie, API to jest bardziej znane z tego, że służy do tworzenia silników dwu- i trójwymiarowych gier. Skoro jednak OpenGL ES 2.0 i nowsze wersje tej biblioteki wymagają do przetwarzania dedykowanego GPU, można jej używać do przetwarzania na bieżąco danych wideo lub z kamery. W tym podrozdziale pokazujemy, jak dodawać efekty za pomocą biblioteki OpenGL ES 2.0. Zaczniemy od przykładu z wideo.

---

Zakładam, że czytelnik zna podstawy OpenGL ES 2.0.

---

W nowszych wersjach Androida (poczynając od API 11) można używać rozszerzenia o nazwie **tekstura strumieniowa** (ang. *streaming texture*) do przesyłania ciągłych strumieni obrazów, np. wideo, do kontekstu OpenGL ES 2.0 jako tekstury. Z pewnością przykład przedstawiający film odtwarzany na obracającym się sześcianie byłby bardzo ciekawy, ale w praktyce opisywanego API częściej używa się do dodawania efektów wizualnych do filmów. W poniższym kodzie uwzględniłem tylko te części, które dotyczą używania tekstury strumieniowej. Pełne przykłady użycia biblioteki OpenGL ES 2.0 można znaleźć w źródłach wymienionych w sekcji „Dodatkowe źródła informacji” oraz w katalogu z przykładami w SDK Androida.

Poniżej znajduje się początek implementacji `Renderer`, która będzie używana przez `GLSurfaceView`.

```
public class MyVideoRenderer implements GLSurfaceView.Renderer,
↳SurfaceTexture.OnFrameAvailableListener {
    private static int GL_TEXTURE_EXTERNAL_OES = 0x8D65;
    private MediaPlayer mMediaPlayer;
    private float[] mSTMatrix = new float[16];
    private int muSTMatrixHandle;
```

Użyte w tym przypadku rozszerzenie GL ma nazwę `GL_TEXTURE_EXTERNAL_OES` i nie jest zdefiniowane w SDK Androida. Ale jest to tylko stała, którą definiuje się w kodzie, tak jak pokazano. `Renderer` nasz zawiera też odwołanie do obiektu `MediaPlayer`, który będzie używany do odtwarzania wideo. Na końcu znajdują się definicje macierzy przekształcenia i odpowiedniego uchwytu shadera. Macierz zostanie później wykorzystana do odwzorowania współrzędnych tekstury otrzymanych z wideo lub kamery w shaderze fragmentów.

W widocznej poniżej metodzie `onSurfaceCreated()` tworzona jest tekstura, która będzie używana na wejściu.

```
public void onSurfaceCreated(GL10 glUnused, EGLConfig config) {
    ...fragment opuszczony dla uproszczenia...

    muSTMatrixHandle = GLES20.glGetUniformLocation(mProgram, "uSTMatrix");
    checkGlError("glGetUniformLocation uSTMatrix");
    if (muSTMatrixHandle == -1) {
        throw new RuntimeException("Nie można pobrać uSTMatrix");
    }

    // tworzenie tekstury
    int[] textures = new int[1];
    GLES20.glGenTextures(1, textures, 0);
    mTextureID = textures[0];
    GLES20.glBindTexture(GL_TEXTURE_EXTERNAL_OES, mTextureID);
```

```

    GLES20.glTexParameterf(GL_TEXTURE_EXTERNAL_OES,
                          GLES20.GL_TEXTURE_MIN_FILTER,
                          GLES20.GL_NEAREST);
    GLES20.glTexParameterf(GL_TEXTURE_EXTERNAL_OES,
                          GLES20.GL_TEXTURE_MAG_FILTER,
                          GLES20.GL_LINEAR);

    // definicja obiektu SurfaceTexture i przypisanie go do MediaPlayer
    mSurface = new SurfaceTexture(mTextureID);
    mSurface.setOnFrameAvailableListener(this);
    Surface surface = new Surface(mSurface);
    mMediaPlayer.setSurface(surface);
    surface.release();

    synchronized (this) {
        updateSurface = false;
    }

    mMediaPlayer.start();
}

```

Zamiast `GLES20.GL_TEXTURE2D` użyłem wcześniej zdefiniowanej stałej. Następnie utworzyłem obiekt klasy `SurfaceTexture` i dodałem go do obiektu klasy `Surface`, który to obiekt jest używany jako powierzchnia renderingu przez `MediaPlayer`. Dzięki temu każda klatka filmu będzie renderowana w teksturze strumieniowej, a nie na ekranie. Ponadto dodałem wywołanie zwrotne aktualizujące zmienną `updateSurface`, gdy pojawia się nowa klatka. Na końcu włączyłem odtwarzanie filmu w `MediaPlayer`.

W widocznej poniżej metodzie `onDrawFrame()` najpierw sprawdzam, czy dostępna jest nowa klatka. Jeśli tak, następuje aktualizacja `SurfaceTexture` i macierz przekształcenia odpowiednia dla tej klatki zostaje zapisana w `mSTMatrix`. Zostanie ona użyta do przekształcenia współrzędnych tekstuury na odpowiednią lokalizację próbkowania w klatce. Zwróć uwagę, że także wiązanie tekstuury zostało zmienione na stałą `GL_TEXTURE_EXTERNAL_OES`.

```

public void onDrawFrame(GL10 glUnused) {
    synchronized (this) {
        if (updateSurface) {
            mSurface.updateTexImage();
            mSurface.getTransformMatrix(mSTMatrix);
            updateSurface = false;
        }
    }

    ...fragment opuszczony dla uproszczenia...

    GLES20.glActiveTexture(GLES20.GL_TEXTURE0);
    GLES20.glBindTexture(GL_TEXTURE_EXTERNAL_OES, mTextureID);

    GLES20.glUniformMatrix4fv(muMVPMatrixHandle, 1, false, muMVPMatrix, 0);
    GLES20.glUniformMatrix4fv(muSTMatrixHandle, 1, false, mSTMatrix, 0);

    GLES20.glDrawArrays(GLES20.GL_TRIANGLE_STRIP, 0, 4);
    checkGLError("glDrawArrays");
    GLES20.glFinish();
}

```

Poniżej znajduje się kod źródłowy użytego w tym przykładzie shadera wierzchołków. W shaderze tym przy użyciu zmiennej `uniform uSTMatrix` w połączeniu ze zwykłą macierzą przekształcenia i pozycją wierzchołka obliczana jest ostateczna wartość `vTextureCoord`.

```

uniform mat4 uMVPMatrix;
uniform mat4 uSTMatrix;
attribute vec4 aPosition;
attribute vec4 aTextureCoord;
varying vec2 vTextureCoord;
void main() {
    gl_Position = uMVPMatrix * aPosition;
    vTextureCoord = (uSTMatrix * aTextureCoord).xy;
}

```

Shader fragmentów rozpoczyna się od zaznaczenia, że do jego działania potrzebne jest rozszerzenie GL\_OES\_EGL\_image\_external. Następnie sTexture zmienia typ na samplerExternalOES.

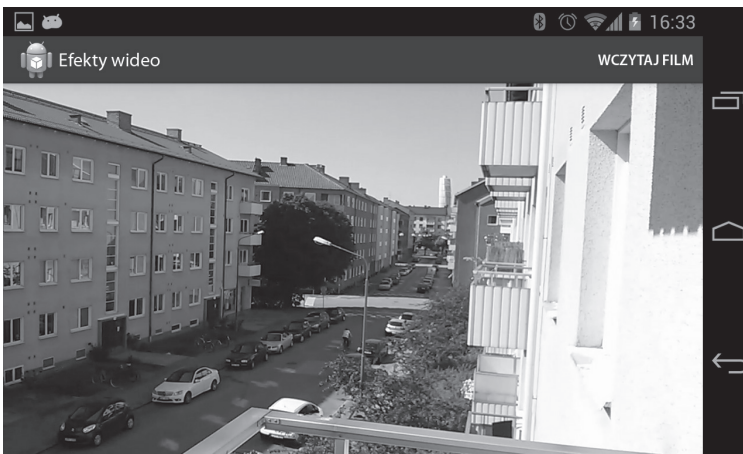
```

#extension GL_OES_EGL_image_external : require precision mediump float;
varying vec2 vTextureCoord;
uniform samplerExternalOES sTexture;

void main() {
    gl_FragColor = texture2D(sTexture, vTextureCoord);
}

```

Powoduje to, że obraz wideo jest renderowany na ekranie w pierwotnym stanie, tzn. bez żadnych zmian, jak widać na rysunku 11.1.



Rysunek 11.1. Renderowanie wideo przy użyciu biblioteki OpenGL ES 2.0 i klasy SurfaceTexture

Teraz wystarczy tylko zmienić coś w shaderze fragmentów, aby uzyskać różne ciekawe efekty wizualne.

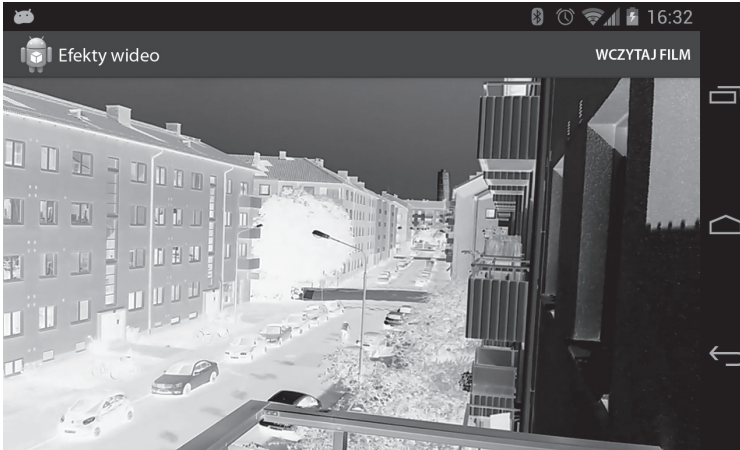
Poniższy kod stosuje na bieżąco do filmu efekt **negatywu** (rysunek 11.2):

```

#extension GL_OES_EGL_image_external : require precision mediump float;
varying vec2 vTextureCoord;
uniform samplerExternalOES sTexture;
uniform float uResS;
uniform float uResT;

void main() {
    vec2 onePixel = vec2(1.0 / uResS, 1.0 / uResT);
    float T = 1.0;
    vec2 st = vTextureCoord.st;
}

```



Rysunek 11.2. Renderowanie wideo przy użyciu biblioteki OpenGL ES 2.0 i filtru dającego efekt negatywu

```

vec3 irgb = texture2D(sTexture, st).rgb;
vec3 neg = vec3(1., 1., 1.)-irgb;
gl_FragColor = vec4(mix(irgb, neg, T), 1.);
}

```

Ten shader pobiera dwa dodatkowe parametry określające rozmiar tekstury. Parametry te są przydatne także w innych rodzajach filtrów. Więcej informacji na ten temat można znaleźć w „Dodatkowych źródłach informacji” wymienionych na końcu tego rozdziału.

## Przetwarzanie obrazu z kamery przy użyciu OpenGL ES 2.0

Przy użyciu biblioteki OpenGL ES 2.0 można przetwarzać obraz z kamery w taki sam sposób jak dane wideo. Można podłączyć strumień klatek podglądu do tego samego rodzaju tekstury strumieniowej i przetwarzać te klatki na bieżąco. W ten sposób tworzy się aplikacje rzeczywistości rozszerzonej (ang. *augmented reality* — AR), stosuje się na bieżąco filtry oraz wykonuje skomplikowane obliczenia na obrazach, których wykonanie przy użyciu CPU zajęłoby znacznie więcej czasu.

```

mSurface = new SurfaceTexture(mTextureID);
mSurface.setOnFrameAvailableListener(this);

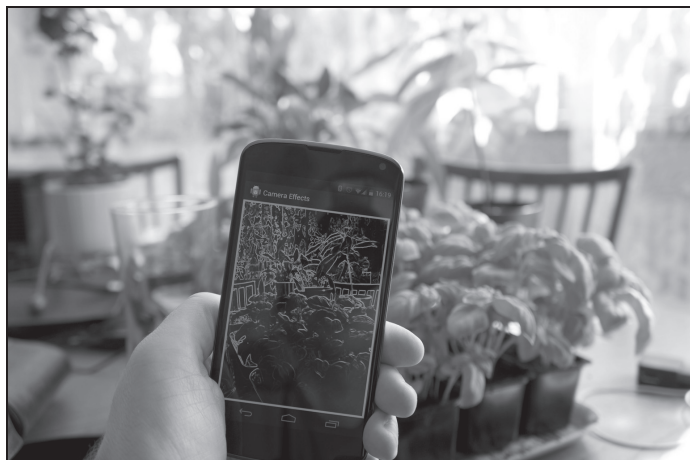
try {
    mCamera.setPreviewTexture(mSurface);
    mCamera.startPreview();
} catch (IOException e) {
    e.printStackTrace();
}

```

Jedyna różnica w stosunku do poprzedniego przykładu znajduje się w metodzie `onDrawFrame()`, w której nie jest używany obiekt klasy `MediaPlayer`, ale bezpośrednio przekazuje się `SurfaceTexture` do `Camera` jako teksturę podglądu.

Użyty w tym przykładzie shader fragmentów wykrywa krawędzie klatek podglądu. Wynik tego działania jest pokazany na rysunku 11.3.





Rysunek 11.3. Efekt wykrywania krawędzi w klatkach podglądu z kamery w telefonie Nexus 4

Poniżej znajduje się kod źródłowy opisywanego shadera.

```
#extension GL_OES_EGL_image_external : require precision mediump float;
varying vec2 vTextureCoord;
uniform samplerExternalOES sTexture;
uniform float uResS;
uniform float uResT;

void main() {
    vec3 irgb = texture2D(sTexture, vTextureCoord).rgb;
    float ResS = uResS;
    float ResT = uResT;
    vec2 stp0 = vec2(1./ResS, 0.);
    vec2 stOp = vec2(0., 1./ResT);
    vec2 stpp = vec2(1./ResS, 1./ResT);
    vec2 stpm = vec2(1./ResS, -1./ResT);
    const vec3 W = vec3(0.2125, 0.7154, 0.0721);
    float i00 = dot(texture2D(sTexture, vTextureCoord).rgb, W);
    float im1m1 = dot(texture2D(sTexture, vTextureCoord-stpp).rgb, W);
    float ip1p1 = dot(texture2D(sTexture, vTextureCoord+stpp).rgb, W);
    float im1p1 = dot(texture2D(sTexture, vTextureCoord-stpm).rgb, W);
    float ip1m1 = dot(texture2D(sTexture, vTextureCoord+stpm).rgb, W);
    float im10 = dot(texture2D(sTexture, vTextureCoord-stp0).rgb, W);
    float ip10 = dot(texture2D(sTexture, vTextureCoord+stp0).rgb, W);
    float i0m1 = dot(texture2D(sTexture, vTextureCoord-stOp).rgb, W);
    float i0p1 = dot(texture2D(sTexture, vTextureCoord+stOp).rgb, W);
    float h = -1.*im1p1 - 2.*i0p1 - 1.*ip1p1 + 1.*im1m1 + 2.*i0m1 + 1.*ip1m1;
    float v = -1.*im1m1 - 2.*im10 - 1.*im1p1 + 1.*ip1m1 + 2.*ip10 + 1.*ip1p1;
    float mag = length(vec2(h, v));
    vec3 target = vec3(mag, mag, mag);
    gl_FragColor = vec4(mix(irgb, target, 1.0),1.);
}
```

## Kodowanie mediów

Android 4.3, a dokładniej API 18, zawiera kilka udoskonaleń w API mediów. Dwie ważne klasy tego API to MediaCodec i MediaMuxer. Pierwsza z nich, wprowadzona we wcześniejszej, bardziej ograniczonej wersji

Androida 4.2 (API 16), umożliwia dostęp do niskopoziomowych funkcji kodowania mediów. W API 18 klasa `MediaCodec` dodatkowo obsługuje kodowanie z `Surface`, co oznacza, że można przy jej użyciu rejestrować sceny OpenGL ES 2.0 do strumienia wideo.

Klasa `MediaMuxer` służy do multipleksowania surowych strumieni medialnych do plików przeznaczonych do odtwarzania. Przy użyciu tych dwóch klas można np. dodać do gry możliwość nagrywania rozgrywki albo, w połączeniu z poprzednimi przykładami przetwarzania wideo i obrazu z kamery, zapisać wynik w pliku MP4.

## Nagrywanie sceny OpenGL

Poniżej znajduje się prosty przykład użycia klas `MediaCodec` i `MediaMuxer` do nagrywania scen OpenGL ES 2.0 do pliku MP4. Metoda ta powinna działać z każdym rodzajem treści OpenGL ES 2.0. W kodzie tym tworzone są koder i multiplekser. Ponadto został w nim użyty obiekt typu `MediaFormat` do określenia parametrów kodera:

```
private void prepareEncoder() {
    mBufferInfo = new MediaCodec.BufferInfo();
    MediaFormat format = MediaFormat.createVideoFormat("video/avc", VIDEO_WIDTH,
        ↪VIDEO_HEIGHT);
    format.setInteger(MediaFormat.KEY_COLOR_FORMAT,
        ↪MediaCodecInfo.CodecCapabilities.COLOR_FormatSurface);
    format.setInteger(MediaFormat.KEY_BIT_RATE, 6000000); // 6 Mbps
    format.setInteger(MediaFormat.KEY_FRAME_RATE, 30);
    format.setInteger(MediaFormat.KEY_I_FRAME_INTERVAL, 10);

    // utworzenie kodera MediaCodec i konfiguracja go przy użyciu naszego formatu
    mEncoder = MediaCodec.createEncoderByType(MIME_TYPE);
    mEncoder.configure(format, null, null, MediaCodec.CONFIGURE_FLAG_ENCODE);

    mMuxer = new MediaMuxer(mOutputFile.getAbsolutePath(),
        ↪MediaMuxer.OutputFormat.MUXER_OUTPUT_MPEG_4);
    mSurface = mEncoder.createInputSurface();
}
```

W powyższym przykładzie na wyjściu użyto formatu H.264, z szybkością transmisji bitów 6 Mbps, częstotliwością klatek 30 na sekundę oraz 10 klatkami między każdą parą klatek typu i-frame. Ponadto został utworzony obiekt `Surface`, który będzie przekazywany na wejściu do kodera.

Po utworzeniu kodera i multipleksera kolejnym krokiem jest utworzenie kontekstu EGL, który zostanie użyty do nagrywania:

```
private static final int EGL_RECORDABLE_ANDROID = 0x3142;

private void recorderEglSetup() {
    mEGLDisplay = EGL14.eglGetDisplay(EGL14.EGL_DEFAULT_DISPLAY);
    if (mEGLDisplay == EGL14.EGL_NO_DISPLAY) {
        throw new RuntimeException("Nieudany dostęp do ekranu EGL!");
    }

    int[] version = new int[2];
    if (!EGL14.eglInitialize(mEGLDisplay, version, 0, version, 1)) {
        mEGLDisplay = null;
        throw new RuntimeException("Błąd inicjacji EGL!");
    }
}
```

```

int[] attribList = {
    EGL14.EGL_RED_SIZE, 8,
    EGL14.EGL_GREEN_SIZE, 8,
    EGL14.EGL_BLUE_SIZE, 8,
    EGL14.EGL_RENDERABLE_TYPE, EGL14.EGL_OPENGL_ES2_BIT,
    EGL_RECORDABLE_ANDROID, 1,
    EGL14.EGL_NONE
};
EGLConfig[] configs = new EGLConfig[1];
int[] numConfigs = new int[1];
if (!EGL14.eglChooseConfig(mEGLDisplay, attribList, 0, configs, 0, configs.length,
    numConfigs, 0)) {
    throw new RuntimeException("Błąd konfiguracji EGL!");
}

int[] glAttribs = {
    EGL14.EGL_CONTEXT_CLIENT_VERSION, 2,
    EGL14.EGL_NONE
};
mEGLContext = EGL14.eglCreateContext(mEGLDisplay, configs[0],
↳EGL14.eglGetCurrentContext(), glAttribs, 0);
int[] surfaceAttribs = {
    EGL14.EGL_NONE
};
mEGLSurface = EGL14.eglCreateWindowSurface(mEGLDisplay, configs[0], mSurface,
↳surfaceAttribs, 0);
}

public void releaseRecorder() {
    mEncoder.stop();
    mEncoder.release();
    mEncoder = null;

    mMuxer.stop();
    mMuxer.release();
    mMuxer = null;

    EGL14.eglDestroySurface(mEGLDisplay, mEGLSurface);
    EGL14.eglDestroyContext(mEGLDisplay, mEGLContext);
    EGL14.eglReleaseThread();
    EGL14.eglTerminate(mEGLDisplay);

    mSurface.release();
    mSurface = null;

    mEGLDisplay = null;
    mEGLContext = null;
    mEGLSurface = null;
}

```

Dwie najważniejsze części tego kodu zostały oznaczone pogrubieniem. Zdefiniowana przed metodą stała informuje Androida, że używany jest kontekst z możliwością rejestracji. Zmienna `mSurface` jest taka sama jak utworzona w metodzie `prepareEncoder()`.

Metoda `release()` musi zostać wywołana po zakończeniu nagrywania, aby uporządkować i zwolnić zajmowane przez aplikację zasoby. Zauważ, że zwolnienie omawianego kontekstu EGL nie powoduje zwolnienia tego samego kontekstu używanego do renderowania na ekranie.

Jako że nagranie sceny wymaga dwóch przebiegów renderowania (raz na fizycznym wyświetlaczu i raz w obiekcie Surface używanym do kodowania), potrzebne są dwie metody do zapisywania i przywracania stanu renderingu OpenGL. Poniżej znajduje się kod źródłowy tych metod.

```
private void storeRenderState() {
    System.arraycopy(mProjMatrix, 0, mSavedProjMatrix, 0, mProjMatrix.length);
    mSavedEglDisplay = EGL14.eglGetCurrentDisplay();
    mSavedEglDrawSurface = EGL14.eglGetCurrentSurface(EGL14.EGL_DRAW);
    mSavedEglReadSurface = EGL14.eglGetCurrentSurface(EGL14.EGL_READ);
    mSavedEglContext = EGL14.eglGetCurrentContext();
}

private void restoreRenderState() {
    if (!EGL14.eglMakeCurrent(mSavedEglDisplay,
        mSavedEglDrawSurface,
        mSavedEglReadSurface,
        mSavedEglContext)) {
        throw new RuntimeException("Nieudane wywołanie eglMakeCurrent!");
    }
    System.arraycopy(mSavedProjMatrix, 0, mProjMatrix, 0, mProjMatrix.length);
}
```

Następnie potrzebna będzie też metoda do przesyłania zakodowanego strumienia wideo do multipleksera służącego do zapisywania danych w pliku MP4:

```
private void drainEncoder(boolean endOfStream) {
    if (endOfStream) {
        mEncoder.signalEndOfInputStream();
    }

    ByteBuffer[] encoderOutputBuffers = mEncoder.getOutputBuffers();

    while (true) {
        int encoderStatus = mEncoder.dequeueOutputBuffer(mBufferInfo, 0);
        if (encoderStatus == MediaCodec.INFO_TRY_AGAIN_LATER) {
            break;
        } else if (encoderStatus == MediaCodec.INFO_OUTPUT_BUFFERS_CHANGED) {
            encoderOutputBuffers = mEncoder.getOutputBuffers();
        } else if (encoderStatus == MediaCodec.INFO_OUTPUT_FORMAT_CHANGED) {
            MediaFormat newFormat = mEncoder.getOutputFormat();
            mTrackIndex = mMuxer.addTrack(newFormat);
            mMuxer.start();
            mMuxerStarted = true;
        } else {
            ByteBuffer encodedData = encoderOutputBuffers[encoderStatus];

            if ((mBufferInfo.flags &
                MediaCodec.BUFFER_FLAG_CODEC_CONFIG) != 0) {
                mBufferInfo.size = 0;
            }

            if (mBufferInfo.size != 0) {
                encodedData.position(mBufferInfo.offset);
                encodedData.limit(mBufferInfo.offset + mBufferInfo.size);

                mMuxer.writeSampleData(mTrackIndex, encodedData, mBufferInfo);
            }

            mEncoder.releaseOutputBuffer(encoderStatus, false);
        }
    }
}
```

```

        if ((mBufferInfo.flags
            & MediaCodec.BUFFER_FLAG_END_OF_STREAM) != 0) {
            break;
        }
    }
}

```

Użyta w tym kodzie pętla `while` działa, dopóki na wyjściu kodera dostępne są dane.

Poniżej znajduje się typowy przykład implementacji metody `onSurfaceChanged()`, służącej do obsługi sceny OpenGL ES, w której dodatkowo ustawiane są koder i kontekst EGL do nagrywania sceny:

```

public void onSurfaceChanged(GL10 gl10, int width, int height) {
    GLES20.glViewport(0, 0, width, height);
    float ratio = (float) width / height;
    Matrix.frustumM(mProjMatrix, 0, -ratio, ratio, -1, 1, 3, 7);

    prepareEncoder(mContext);
    if (mEncoder != null) {
        storeRenderState();
        recorderEglSetup();
        mEncoder.start();
        if (!EGL14.eglMakeCurrent(mEGLDisplay, mEGLSurface,
                                mEGLSurface, mEGLContext)) {
            throw new RuntimeException("Błąd w metodzie eglMakeCurrent");
        }
        restoreRenderState();

        mFrameCount = 0;
    }
}

```

Przedstawiona poniżej metoda `onDrawFrame()` najpierw rysuje scenę na głównym wyświetlaczu, a następnie sprawdza, czy koder jest poprawny. Później zmienia kontekst EGL, konfiguruje obszar widoku dla drugiego renderowania i rysuje scenę w tym obszarze.

```

public void onDrawFrame(GL10 gl10) {
    drawFrame();

    if (mEncoder != null) {
        storeRenderState();
        if (!EGL14.eglMakeCurrent(mEGLDisplay, mEGLSurface,
                                mEGLSurface, mEGLContext)) {
            throw new RuntimeException("Nieudane wywołanie eglMakeCurrent");
        }
        Matrix.orthoM(mProjMatrix, 0, 0, mWidth, 0, mHeight, -1, 1);
        GLES20.glViewport(mViewportXoff, mViewportYoff,
                        mViewportWidth, mViewportHeight);

        drawFrame();
        drainEncoder(false);
        long when = System.nanoTime();
        EGLExt.eglPresentationTimeANDROID(mEGLDisplay, mEGLSurface, when);
        EGL14.eglSwapBuffers(mEGLDisplay, mEGLSurface);
        restoreRenderState();
    }
}

```

Po zakończeniu drugiego rysowania koder jest pusty.

Poniższa metoda ilustruje sposób sygnalizowania, aby zatrzymać koder i zwolnić wszystkie jego zasoby:

```
public void stopRecording() {  
    drainEncoder(true);  
    releaseRecorder();  
}
```

Metodę tę powinno się wywoływać podczas opuszczania aplikacji przez użytkownika albo w momencie, gdy ma nastąpić zatrzymanie nagrywania.

## Podsumowanie

W nowszych wersjach Androida znacznie wzbogacono asortyment narzędzi do zaawansowanej obróbki mediów. Przedstawione w tym rozdziale przykłady ilustrują tylko niektóre z tych zaawansowanych API. Można zbudować kompletną aplikację z wybranego pojedynczego przykładu albo połączyć kilka przykładów, aby uzyskać bardziej złożony program.

Wprowadzie przedstawione w tym rozdziale przykłady nie są kompletnymi aplikacjami, ale doświadczony programista powinien sobie z nimi poradzić.

## Dodatkowe źródła informacji

### Dokumentacja

Dokumentacja API OpenGL ES na stronach Khronos Group: [www.khronos.org/opensles](http://www.khronos.org/opensles).

Specyfikacja rozszerzenia OpenGL GL\_OES\_EGL\_image\_external:

[http://www.khronos.org/registry/gles/extensions/OES/OES\\_EGL\\_image\\_external.txt](http://www.khronos.org/registry/gles/extensions/OES/OES_EGL_image_external.txt).

Wpis Yu Lu LittleCheeseCake na temat shaderów OpenGL ES 2.0:

<http://littlecheesecake.me/blog/13804700/opengles-shader>.

Zastosowania MediaCodec i MediaMuxer w Androidzie: <http://bigflake.com/mediacodec/>.

# Skorowidz

3G, 159

## A

ABI, 247

Accessory Development Kit,

*Patrz:* ADK

Activity, 67, 76, 99, 166, 293  
implementacja, 69

ADK, 319

adnotacja, 47

adres IP, 325, 326

adres MAC, 328

adres sieciowy, 328

agile development, *Patrz:*  
programowanie zwinne

AIDL, 119, 136, 139

wywołanie zwrotne,

*Patrz:* wywołanie zwrotne

akcelerometr, 151

akcja, 77

ACTION\_POWER\_CONNECTED,  
152

ACTION\_POWER\_  
↳DISCONNECTED, 152

Intent android.provider.

↳Telephony.SMS\_RECEIVED,  
271

powiadomieniowa, 152

aktywność, *Patrz:* Activity

algorytm

AES, 223

szyfrujący, 219, 223

analiza statyczna, 27, 35, 37

Android 2.3, 326

Android 3.0, 68, 85, 293

Android 4.1, 326

Android 4.2, 41, 214

Android 4.3, 214

Android Froyo, 46

Android Interface Definition

Language, *Patrz:* AIDL

Android Open Accessory Protocol, 319

Android Open Source Project, 278,  
299, *Patrz:* AOSP

Android Studio, 23, 24, 35, 36, 76,  
88, 89, 100, 281, 283

animacja parametry skalowania, 43

AOSP, 275, 283, 289

kod źródłowy, *Patrz:* kod

źródłowy AOSP

Apache Harmony, 47

API

billingowe, 356, 358

Bouncy Castle Crypto, 223

dla innych aplikacji, 138

dźwiękowe, 199, 256

nagrywanie, 200, 201, 202

odtworzenie, 200, 202

Fragments, 68

Game Cloud Save, 340

Google Play, 337

graficzne, 112

Java Reflection, 270

Location, 233

lokalizacyjne, 303, 340

macierzyste, 249, 255

map Google, 234

Monkeyrunner, 233

Monkeyrunner, *Patrz:*

Monkeyrunner

obsługi wątków, 254

odczytywania tekstu na głos,

*Patrz:* TTS

oficjalne, 265

opakowywanie w biblioteki, 146

OpenGL ES, *Patrz:* OpenGL ES

pęku kluczy, 226

poziom

lista, 74

obsługiwany, 74

optymalny, 74

Presentation, 100

Restlet, 332

SoundPool, 82

SQLite, 69

Static Maps, 303

szyfrowania i deszyfrowania, 223

ukryte, 265, 266, 282

frameworks, 266

odbieranie SMS, 271

w klasie publicznej, 266

wydobywanie z urządzenia,

268, 269

wywoływanie, 267, 270

ukrywanie, 265

USB, 320

wersja, 147

zarządzania urządzeniem, 229

zdalne

zabezpieczenie, 148

API 16, 214

API 18, 214

aplikacja, *Patrz też:* usługa

architektura, 71

katalog danych, 221

keytool, 219, 226

klucz, *Patrz:* klucz

licencja, 356

lista, 27

lokalizacyjna, 233, 234, 242,  
244

funkcje, 240

native-activity, 250

pobieranie opłat, 356

rzeczywistości rozszerzonej, 212

sprzedawanie, 355

sprzedaż treści, 356

subskrypcja, 358

systemowa, 280, 281

uprawnienia, 220, 222

uruchamianie automatyczne, 157

wieloe ekranowa, 43

wydajność, 43

Application, 67, 69, 75

application binary interface, *Patrz:* ABI

Application Exerciser Monkey,

*Patrz:* Monkey

AR, *Patrz:* aplikacja rzeczywistości

rozszerzonej

asercja na wyniku, 186

assets, *Patrz:* środki

audio, *Patrz:* dźwięk  
 augmented reality, *Patrz:* aplikacja  
 rzeczywistości rozszerzonej

## B

baza danych  
 aktualizacja, 169, 171  
 normalizacja, 168  
 projektowanie, 168  
 SQLite, *Patrz:* SQLite  
 transakcja, *Patrz:* transakcja  
 tworzenie, 169, 171  
 bezpieczeństwo, 72, 76, 148, 153,  
 219, 222  
 poziom ochrony, *Patrz:* ochrona  
 biblioteka, 28, 31  
 akcesoriów USB, 320  
 android\_native\_app\_glue, 250  
 API zdalnego, 146  
 Bionic, *Patrz:* Bionic  
 C library, 256  
 Download Library, 364  
 EGL, 256  
 Gson, *Patrz:* Gson  
 HTTP Volley, *Patrz:* Volley  
 kodeka audio, 257  
 libc, 256  
 macierzysta, 278  
 kompilowanie, 257, 258  
 moduł JNI, 259  
 moduł statyczny, 258  
 OkHttp, *Patrz:* OkHttp  
 OpenGL ES, *Patrz:* OpenGL ES  
 OpenSL ES, 256, *Patrz:*  
 OpenSL ES  
 Opus, 257  
 projekt, 32  
 pthread, 254, 256  
 Restlet, *Patrz:* Restlet  
 rozszerzeń, 363  
 Scribe, *Patrz:* Scribe  
 TooTallNate, *Patrz:* TooTallNate  
 wcielanie do aplikacji, 257  
 WebSocket, *Patrz:* biblioteka  
 TooTallNate  
 wyszukiwanie, 32  
 Binder, 133, 134  
 adres, 134  
 ServiceManager, *Patrz:*  
 ServiceManager  
 transakcja, *Patrz:* transakcja  
 Binder IPC, 133  
 Bionic, 256  
 Bloch Joshua, 53  
 blok  
 kodu wywoływalny, 51  
 synchronizowany, 50

blokada, 50  
 Bluetooth, 319, 325  
 Bluetooth LE, 319, 322  
 implementacja, 322  
 Bluetooth Smart, 319, 322  
 bootloader, 276, 277  
 Bornstein Dan, 46  
 BroadcastReceiver, 67, 68, 69, 71,  
 76, 116, 126, 145, 151, 152,  
 155, 326  
 administracja urządzeniem, 230  
 bufor  
 obiektów, 53  
 odpowiedzi, 294, 300, 313

## C

CA, 226  
 certyfikat, 226, 227, 228  
 Children's Online Privacy Protection  
 Act, *Patrz:* COPPA  
 Cloud Messaging Google, 315  
 component, *Patrz:* składnik  
 ConnectivityManager, 159  
 ContentProvider, 67, 69, 71, 76,  
 163, 168, 222  
 testowanie, 191  
 ukryty, 271  
 COPPA, 360  
 CyanogenMod, 278  
 czcionka, 89  
 bezszeryfowa, 89  
 dekoracyjna, 89  
 miara x-height, 92  
 Roboto, 89, 92  
 szeryfowa, 89  
 czujnik światła, 151

## D

Dalvik, 30, 46, 47, 75, 133, 268,  
 278  
 dane  
 aplikacji, *Patrz:* aplikacja  
 katalog danych  
 binarne, 164, 174, 179  
 deszyfrowanie, 224, 225  
 dotykowe, 43  
 format, 314  
 kodowanie strukturalne, 179  
 kopia zapasowa, 181, 182  
 optymalizacja przesyłania, 313,  
 314  
 podpis, 228  
 przechowywanie, *Patrz:* dane  
 utrwalanie

przesyłanie  
 na serwer sieciowy, 296  
 przez internet, 175  
 w powiadomieniach, 351,  
 352  
 reprezentacja, *Patrz:* serializacja  
 szyfrowanie, 225  
 klucz, 223, *Patrz też:* klucz  
 sól, 223  
 u klienta, 223  
 wektor inicjacji, 223  
 typ, *Patrz:* typ danych  
 utrwalanie, 163  
 wyciek, 295  
 dead reckoning, *Patrz:* nawigacja  
 obliczeniowa  
 debugowanie, 25, 42, 276  
 dekompilacja, 30  
 demon adb, *Patrz:* narzędzie adb  
 density-independent pixel, *Patrz:*  
 piksel niezależny od gęstości  
 deserializacja, 32, 175, 177, 180,  
 225  
 protobuf, 180  
 direct broadcast, *Patrz:*  
 powiadomienie bezpośrednio  
 doSpeechRecognition, 207  
 dostawca treści, *Patrz:*  
 ContentProvider  
 Dropbox, 163  
 dynamic translation, *Patrz:*  
 translacja dynamiczna  
 Dysk Google, 337, 340  
 dziennik, 25, 256  
 dźwięk, 199, 256  
 częstotliwość próbkowania, 201  
 kodowanie PCM, 201  
 nagrywanie, 200, 201, 202  
 odtwarzanie, 200, 202  
 strumieniowy, 200, 201, 319

## E

Eclipse, 35, 283  
 EDGE, 151  
 ekran  
 blokady, 158  
 dotykowy, 74  
 symulacja, 43  
 LCD, 92  
 orientacja, 116, 160  
 rozmiar, 74  
 wielodotkowy, 74  
 wielokrotny, 100, 102  
 Expansion Library, *Patrz:* biblioteka  
 rozszerzeń



## F

Facebook, 95, 308, 309, 312  
 filtr intencji, *Patrz:* intencja filtr  
 folder, *Patrz:* katalog  
 font, *Patrz:* czcionka  
 Foursquare, 96, 305  
 fragment, 71, 99  
 FreeXperia Project, 278  
 funkcja  
   diagnostyczna, 35, 36  
   macierzysta, 250

## G

gamification, *Patrz:* grywalizacja  
 GATT, 319  
 GCM, 345, 346, 347  
 geofencing, 241, 242  
 geokodowanie, 240  
 geon, 94  
 Gerrit, 34  
 gesty, 43, 108, 111  
 Gingerbread, *Patrz:* Android 2.3  
 Git, 33  
 GitHub, 34  
 gitolite, 34  
 Gmail, 220, 280  
 gniazdo sieciowe, 300, 329, 332  
   powiadomienia, 316  
 Google Cloud, 337  
 Google Cloud Endpoints, 345  
 Google Cloud Messaging, 313,  
   *Patrz:* GCM  
 Google Cloud Platform, 343  
 Google Drive, 163, *Patrz:* Dysk  
   Google  
 Google Nexus, 275, *Patrz też:*  
   urządzenie Nexus  
 Google Play, 32, 73, 220, 234,  
   281, 355  
   klucz, 219  
   konsola, 355  
   pobieranie opłat, 355  
   rozmiar pliku, 356  
   usługa licencjonowania, 356  
 Google Translator Toolkit, 80  
 Google Wallet Merchant, 355, 356  
 Google+, 280  
 GoogleMaps, 280  
 Gosling James, 45  
 gospodarz, *Patrz:* host  
 Gradle, 28, 186, 193, 219, 281,  
   299, 337  
   migracja, 30  
   podręcznik, 29  
 graf sceny, 113  
 grafika, 199

API, 308  
 grywalizacja, 96  
 Gson, 177, 225

## H

H klasa ttpClient, 294  
 Hackborn Dianne, 133  
 historyjka użytkownika, *Patrz:*  
   użytkownik historyjka  
 Honeycomb, *Patrz:* Android 3.0  
 host, 226, 320  
 HTC, 277

## I

IDE, 219  
 IDL, 139  
 IntelliJ IDEA, 35  
 IntelliJ IDEA CE, 283  
 intencja, 77, 151  
   filtr, 77, 117, 156  
   jawna, 77  
   niejawna, 77  
   powiadomienia, 26  
   rozsyłana przez system, *Patrz:*  
     zdarzenie systemowe  
   rozsyłanie, 151  
   wybieranie, 134  
   zewnętrzna, 26  
 Intent, *Patrz:* intencja  
 interface definition language,  
   *Patrz:* IDL  
 interfejs  
   App Engine, 343  
   binarnym aplikacji, *Patrz:* ABI  
   Cloud Datastore, 343  
   Compute Engine, 343  
   dostosowujący się do rozmiaru  
     urządzenia, 68  
   element główny/szczegóły, 100  
   URLConnection, 299  
   IBinder, 134, 135  
   IBinder.DeathRecipient, 138  
   macierzysty Javy, *Patrz:* JNI  
   MyLocalService.Callback, 129  
   Parcelable, 136, 137, 175  
   RoomStatusUpdateListener,  
     351  
   RoomUpdateListener, 351  
 użytkownika, 68, 85, 99  
   definiowanie ustawień, 71  
   Holo, 85  
   kolor, 92, 93  
   nawigacja, 87  
   projektowanie, 87, 88, 92,  
     93, 94, 95  
   prototyp, 88

rozmiar elementu, 90, 91  
 test, 186, 188  
 ustawienia programu, 166,  
   *Patrz też:* ustawienia  
   użyteczność, 95  
 widżet, *Patrz:* widżet  
   wskazówki wizualne, 95  
 wielodotkowy, 43  
 zdalny, 143  
 inter-proces communication, *Patrz:*  
   komunikacja międzyprocesowa  
 Ippel Dennis, 113  
 Ivy, 28

## J

Java, 32, 45, 78  
   budowa stosowa, 46  
   obiekt, *Patrz:* POJO  
 Java EE, 45  
 Java ME, 45  
 Java Native Interface, *Patrz:* JNI  
 Java SE, 45  
 Java SE 5.0, 49  
 JavaScript Object Notation, *Patrz:*  
   JSON  
 Javie SE 5.0, 48  
 Jazz Hands, 74  
 jądro, 285  
   licencja, 285  
   sterownik, 134  
 jednostka  
   dp, *Patrz:* piksel niezależny  
     od gęstości  
   in, 91  
   mm, 91  
   pt, 91  
   px, 91  
   sp, 90  
 Jenkins CI, 194  
 język  
   C, 199, 247, 248  
   C++, 248  
   definicji interfejsu, *Patrz:* IDL  
   Groovy, 28  
   Java, *Patrz:* Java  
   JSON, *Patrz:* JSON  
 JNI, 252, 253, 254  
 JSON, 175, 178

## K

kamera, 209, 212  
 katalog  
   aidl, 29  
   assets, 29  
   jni, 29, 248, 258  
   libopus, 258, 259

- katalog
    - libs, 31
    - main, 29
    - opus\_jni, 259
    - res, 29
    - rs, 29
    - vendor, 284
  - klasa
    - Activity, 68, 99
    - ActivityUnitTestCase, 188
    - AndroidTestCase, 187
    - Assert, 186
    - AssetManager, 82
    - AsyncTask, 57, 63, 68
    - AudioRecord, 201
    - AudioTrack, 200, 201
    - BackupAgentHelper, 182
    - Binder, 134
    - BroadcastReceiver, *Patrz:*
      - BroadcastReceiver
    - ChatClient, 303
    - Constants, 165
    - ContentProvider, 138, 174
    - ContentValues, 175
    - CountryDetector, 266
    - Cursor, 175
    - DeviceAdminReceiver, 230
    - DisplayManager, 101
    - DownloaderService, 364
    - Fragment, 99
    - GameClient, 352
    - Geocoder, 240
    - GestureDetector, 111
    - GoogleAuthUtil, 339
    - GoogleMap, 240
    - Handler, 49, 58, 59, 61, 63, 68, 123
    - HttpsURLConnection, 229
    - HttpURLConnection, 294, 296, 297, 301, 305, 314
    - IntentService, 123, 124, 126
    - JSONArray, 177
    - JSONObject, 177
    - JsonReader, 177
    - JsonWriter, 177
    - LocalBroadcastManager, 153
    - LocationClient, 241
    - Looper, 55, 58, 61
    - MediaCodec, 214
    - MediaMuxer, 214
    - MediaPlayer, 200
    - MediaRecorder, 201
    - Message, 55, 60
    - MessageQueue, 55
    - Messenger, 136, 139, 143
    - MotionEvent, 55, 108, 110
    - MyDatabaseHelper, 169, 171
    - MyWebSocketServer, 332
    - NativeSorting, 251
    - NsdManager, 326
    - OkHttpClient, 300
    - PackageManager, 156
    - Parcel, 55, 136
    - Parcelable, 146
    - PreferenceActivity, 71, 166
    - PreferenceFragment, 166
    - Presentation, 102
    - ReentrantReadWriteLock, 52
    - ScaleGestureDetector, 111
    - SecureRandom, 224
    - Service, 123, 138
    - Settings, 273
    - SharedPreferences, 163, 164
    - singletonowa, 69
    - słownikowa współbieżna, 51
    - Socket, 301
    - SoundPool, 200
    - SQLiteOpenHelper, 221
    - Thread, 56, 63
    - UiLifecycleHelper, 310
    - ukryta, 266
    - UsbDevice, 321
    - WebSocketClient, 303
    - WifiManager, 266, 272
    - wyliczeniowa, 49
  - klient
    - HTTP, 299
    - WebSocket, 301
  - klucz, 219, 223
    - diagnostyczny, 219
    - generowanie, 220, 223
    - konwersja, 281
    - pęk, 226
    - prywatny, 229
    - publiczny, 228
    - schowek, 219, 220, 226, 281
    - SecretKey, 223
    - testowy, 281
    - wygenerowany ręcznie, 219
  - kod
    - bajtowy, 45
    - dex, 30, 31
    - dostawcy, 284
    - działający po stronie klienta, 47
    - macierzysty, 247
      - OpenGL ES, 256
      - OpenSL ES, 256
    - optymalizacja, 46, 47
    - pokrycie testami, *Patrz:* test
      - pokrycie kodu
    - współużytkowanie, 32
    - XML, 78
    - wywotyalny, 51
    - zaciemnienie, 30, 31
    - źródłowy, 249, 283
      - AOSP, 275
      - przeszukiwanie, 266
  - kolejka, 50, 51
  - kolekcja, 50
  - kompilacja, 28, 248, 279, 288
    - na żądanie, 46
    - skrypt, 248
  - kompilator, 45
    - Java SE, 45
    - JIT, 46, 48
  - komunikacja
    - asynchroniczna, 300, 329, 332
    - dwukierunkowa, 136
    - jednokierunkowa, 136
    - międzyprocesowa, 133, 134
    - RIL, 133
    - synchroniczna, 329
    - TCP, 293
    - UDP, 293
    - USB, *Patrz:* USB
  - komunikat
    - SET\_PROGRESS, 62
    - SYNC\_DATA, 62
  - komunikator, 116
    - JSONArray, 176
  - konsumowanie produktów, 358
  - kopia zapasowa, 75
    - agent, 75
  - Krug Steve, 95
- ## L
- LastPass, 223
  - link to death, *Patrz:* łącze
    - ze śmiercią
  - Linux, 23, 26
  - local binder, *Patrz:* wiązanie lokalne
  - lokalizacja, 42, 80
  - LTE, 159
- ## Ł
- ładownica, 152, *Patrz też:* zasilanie
    - łącze ze śmiercią, 134, 138
- ## M
- makieta obiektów, *Patrz:* obiekt
    - makieta
  - manifest, 67, 72, 152, 220
    - Activity, 76
    - android:backupAgent, 75
    - android:installLocation, 73
    - android:largeHeap, 75
    - android:minSdkVersion, 74
    - android:process, 75
    - android:sharedUserId, 73
    - android:sharedUserLabel, 73
    - android:targetSdkVersion, 74

- android:theme, 75
  - android:versionCode, 73
  - android:versionName, 73
  - application, 75
  - BroadcastReceiver, 76
  - ContentProvider, 76
  - package, 72
  - Service, 76
  - składnik, 76, 77
  - supports-screens, 74
  - uses-features, 73
  - uses-sdk, 74
  - API, 236
  - mapa, 234, 236
    - statyczna, 303
    - współrzędne, 237, 240
  - master/detail flow, *Patrz:* interfejs
  - element główny/szczegóły
  - maszyna
    - interpretująca, 46
    - stanów, 200
    - wirtualna, 45
      - Dalvik, *Patrz:* Dalvik,
      - Patrz:* Dalvik
      - Javy, 250
      - rejestrowa, 46
      - stosowa, 46
  - Maven, 28, 31, 32, 299
  - MCC, 81, 82
  - mechanizm
    - dbus, 133
    - deserializacji, *Patrz:* deserializacja
    - deserializacja
    - serializacji, *Patrz:* serializacja
  - menedżer
    - aplikacji, 26
    - lokalizacji, 233, 241
    - pakietów, 26, 27
  - metoda
    - abortBroadcast, 154, 315
    - AccountPicker.newChoose
      - ↳ AccountIntent, 338
    - Activity.onConfigurationChanged, 160
    - addCallback, 142, 143
    - apply, 165
    - Arrays.sort, 252
    - bezpieczna wątkowo, 51
    - Binder.onTransac, 135
    - Binder.onTransact, 136
    - bindService, 120
    - Builder.build, 181
    - buildTransfer, 321
    - bulkInsert, 172
    - Collections.sort, 252
    - commit, 165
    - ContentProvider.insert, 172
    - ContentProvider.openFile, 174
    - ContentProvider.openFileHelper, 174
    - ContentProvider.query, 171
    - ContentResolver.openOutputStream, 175
    - ContentResolver.query, 171
    - Context.bindService, 117, 118, 121, 126
    - Context.getApplication, 69, 71
    - Context.getMainLooper, 55
    - Context.getService, 159
    - Context.getSharedPreference, 164
    - Context.getSystemService, 133
    - Context.openFileInput, 221
    - Context.openFileOutput, 221
    - Context.registerReceiver, 69
    - Context.sendBroadcast, 153, 155
    - Context.sendStickyBroadcast, 155, 156
    - Context.startActivity, 69
    - Context.startService, 69, 117, 118, 121, 126, 152
    - Context.unbindService, 121
    - createFromParcel, 137
    - encryptClearText, 224
    - fabryczna statyczna, 53
    - GoogleAuthUtil.getToken, 340
    - GoogleCloudMessaging.register, 347
    - IBinder.linkToDeath, 138
    - IBinder.transact, 135, 136
    - Javy, 252, 253, 255
    - KeyChain.choosePrivateKeyAlias, 227
    - linkToDeath, 138, 142
    - macierzysta, 251
    - NsdManager.registerService, 325
    - NsdManager.resolveService, 326
    - OAuthService.getAccessToken, 306
    - onActivityResult, 340
    - onBind, 119, 141, 143
    - onClose, 303
    - onConfigurationChanged, 161
    - onCreate, 116
    - onDestroy, 116, 117, 160
    - onDraw, 105
    - onHandleIntent, 123
    - onLayout, 105, 106
    - onMeasure, 106
    - onOpen, 303
    - onPause, 116, 119, 158, 160
    - onReceive, 68, 155
    - onResume, 55, 119, 158
    - onServiceConnected, 120, 130, 138
    - onServiceDisconnected, 120, 130
    - onSignInFailed, 350
    - onSignInSucceeded, 350
    - onStartCommand, 55, 117, 120
    - onStop, 160
    - onSurfaceChanged, 217
    - onX, 55
    - Parcel.readStringList, 137
    - PreferenceManager.getDefault
      - ↳ Preferences, 182
    - PreferenceManager.getDefault
      - ↳ SharedPreferences, 164
    - processSms, 315
    - query, 171
    - readStreamToString, 296
    - removeCallback, 143
    - runOnUiThread, 61
    - Service.onBind, 134
    - Service.startForeground, 120
    - Service.stopForeground, 121
    - setInexactRepeating, 314
    - setResultCode, 154
    - setResultData, 154
    - simpleRaceGameLoop, 353
    - SQLiteDatabase.setTransaction
      - ↳ Successful, 173
    - SQLiteException.endTransaction, 173
    - startForeground, 121
    - TaskInfoEndpoint.insertTaskInfo, 348
    - testowa, 189
    - updateGameState, 351
    - URLEncoder.encode, 305
    - View.onAttachedToWindow, 103
    - View.onDetachedFromWindow, 103
    - View.onDraw, 103
    - View.onLayout, 103
    - View.onMeasure, 103
    - zapytaniowa, 171
    - zwrotna, 99, 116, 303
      - implementacja, 60
      - zwrotnych, 151
  - Miracast, 101
  - MNC, 81, 82
  - mock objects, *Patrz:* obiekt makiet
  - model klient-serwer, 134
  - Monkey, 27, 28
  - Monkeyrunner, 28
  - Motorola, 278
  - muzyka, 116, 199
- ## N
- nagłówek HTTP, 300
  - narzędzie
    - Autoconf, 257
    - Automake, 257
    - bmgr, 182
    - dx, 46

## narzędzie

- ExeutorService, 62
- generowania autonomicznego
  - oprzyrządowania, 257
- jawah, 251
- keytool, 226
- lint, 37, 39, 72, 80
- Loader, 62
- nastuchiwanie, 155, 156
  - kliknąć, 165
- NativeActivity, 249
- nawigacja obliczeniowa, 353

## O

OAuth, 305, 308, 338

## obiekt

- AdRequest, 360
  - Binder, 222
  - BroadcastIntent, 69, 152
  - BroadcastReceiver, *Patrz też:*
    - BroadcastReceiver
  - Cipher, 223
  - ContentProvider, *Patrz:*
    - dostawca treści
  - Creator, 136
  - GoogleMap, 237, 238, 239
  - Handler, 143
  - Intent, 152
  - Java, *Patrz:* POJO
  - LatLng, 237
  - LocalBroadcastManager, 153
  - makieta, 186
  - Marker, 237
  - nasłuchujący, 68
  - Parcel, 135, 136
  - Presentation, 101
  - protobuf, 180
  - Service, *Patrz:* usługa
  - SharedPreferences, 164, 165
  - SpeechRecognizer, 207
- obraz fabryczny, 277
- ochrona, 280
  - dangerous, 220, 221
  - normal, 220
  - poziom, 280
  - signature, 220, 265, 281
  - signatureOrSystem, 220
  - system, 220, 265
- odbiornik, 153, 154, *Patrz też:*
  - BroadcastReceiver
    - implementacja, 154
    - nasłuchujący, 156, 157
    - włączanie, 156
    - wyłączanie, 156
- odtworczacz, 116, 121, 199
  - AudioTrack, 116, 200
  - MediaPlayer, 116, 200
  - SoundPool, 200

OkHttp, 299

- Opcje programisty, 41, 42
- OpenBinder, 133, 134
- OpenGL ES, 112, 113, 199, 209, 212
  - kod macierzysty, 256
  - nagrywanie, 214
- OpenGL ES, 199, 202
  - inicjacja, 203
  - kod macierzysty, 256
- operacja
  - atomowa, 50
  - PING\_SERVER, 60
  - sieciowa, 115
  - SYNC\_DATA, 60
- oprogramowanie
  - sterownik, 278
    - binarny, 284
  - układowe, 275, 279, 280
    - klucz testowy, 281
    - sterownik, 278
    - wymiana, 276
- OS X, 23
- oszczędzanie energii, 312, 313, 314

## P

- paczka, 136, *Patrz też:* klasa Parcel
- pakiet
  - android.hardware.usb, 320
  - android.net.nsd, 325
  - android.test, 186
  - com.android.internal, 265, 266
  - java.lang.reflect, 270
  - java.util.concurrent, 50, 51
  - java.util.concurrent.locks, 51
  - javax.crypto, 223
  - junit, 186
- pamięć
  - alokowanie obiektów, 52, 53
  - wyciek, 52, 129
  - zarządzanie automatyczne, 52
- parcel, *Patrz:* paczka
- partycja systemowa, 220
- persistence, *Patrz:* dane utrwalanie
- person, *Patrz:* postać
- perspektywa kanoniczna, 94
- pętla
  - for, 49
  - renderująca, 103
- piksel niezależny od gęstości, 90, 91
- pingowanie, 60
- PKI, 226
- Plain Old Java Object, *Patrz:* POJO
- plik
  - AIDL, 139, 140, 146, 286
  - android.jar, 265
  - Android.mk, 248, 284

- AndroidManifest.xml, 29, 72, 220
- APK, 78, 356
  - miejsce instalacji, 72
  - rozszerzenie, 356, 363
- Application.mk, 249
- binarny, 279
- build.gradle, 28, 30, 186
- com\_appt\_jnidemo\_
  - ↳NativeSorting.h, 251, 252
- DEX, 268
- JAR, 31, 140, 146
- JPG, 174
- libopus.so, 258, 259
- main.c, 250
- Makefile, 258
- MP4, 214
- nagłówkowy, 251
- odczyt, 221
- preferencji, 163, 164, 165
- stdio.h, 256
- stdlib.h, 256
- strings.xml, 78
- tworzenie, 221
- XML, 164, 166
- z kluczami, *Patrz:* klucz schowek
- ZIP, 363

POJO, 177

## polecenie

- adb, 24, 25, 26, 42
    - port, *Patrz:* port 5555
  - adb shell am, 26
  - add, 33
  - am, 26, 27
  - commit, 33
  - logcat, 25
  - lunch, 279
  - pm, 26, 27
  - Refactoring, 39
  - startservice, 26
- połączenie
  - EDGE, *Patrz:* EDGE
  - gniazdowe, 299
  - Wi-Fi, *Patrz:* Wi-Fi
- postać, 86
- powiadomienie, 126
  - android.intent.action.SCREEN\_
    - ↳OFF, 158
  - android.intent.action.SCREEN\_
    - ↳ON, 158
  - bezpośrednie, 156
  - gniazdo sieciowe, 316
  - intencja, 26
  - Intent.ACTION\_BATTERY\_
    - ↳CHANGED, 155
  - Intent.ACTION\_BOOT\_
    - ↳COMPLETED, 157
  - Intent.ACTION\_DOCK EVENT, 155

Intent.ACTION\_MY\_PACKAGE\_↪REPLACED, 157  
kleiste, 155  
lokalne, 153  
niezawodne, 352  
normalne, 154  
przesyłanie danych, 351, 352  
przez gniazdo sieciowe, 316  
przez serwer, 314  
przez SMS, 315  
uporządkowane, 154, 155  
powłoka, 280  
  flash-all.sh, 277  
  Linuksa, 26  
procedura obsługi błędów, 48  
proces  
  w tle, 44  
  zdalny, 134  
  zygoty, *Patrz:* zygote  
procesor zużycie, 43  
program rozruchowy, *Patrz:* bootloader  
programowanie zwinne, 85  
ProGuard, 30, 31  
protectionLevel, *Patrz:* ochrona  
  poziom  
protobuf, 179, 181  
  deserializacja, 180  
  serializacja, 179  
Protocol Buffers, *Patrz:* protobuf  
Protocol Buffers Google, 301, 314  
protokół  
  Android Open Accessory  
  Protocol, *Patrz:* Android Open  
  Accessory Protocol  
  DNS-SD, 325  
  grupowy UDP, 329  
  HTTP, 229, 293, 299, 329  
  mDNS, 325, 326  
  SPDY, 299  
  WebSocket, 301  
przypadek testowy, 186  
PublicKey, *Patrz:* klucz publiczny  
punkt wstrzymania, 25, 35, 36

## R

Radio Interface Layer, *Patrz:*  
  komunikacja RIL  
Rajawali, 113  
Rajlich Nathan, 301  
refaktoryzacja, 35, 39, 41, 186  
referencja  
  JNI, 253, 254  
  wyciek, 69  
refleksja, 270  
reklama, 356, 359  
  COPPA, 360  
  kierowana, 360

  kolorystyka, 361  
  odpłatne usuwanie, 359  
  pełnoekranowa, 361  
rendering, 113  
RenderScript, 29  
repozytorium Git, *Patrz:* Git  
resource zasób, 78  
Restlet, 329, 332  
Ritchie Dennis, 248  
rozpoznawanie mowy, 207  
rozpoznawanie twarzy, 95

## S

Scribe, 305, 308  
SDK, 23, 24, 36, 248, 265  
  AdMob Ads, 356  
  aktualizacja, 24  
  Facebooka, 309  
  Javy, 251  
semafor, 50  
serializacja, 32, 163, 175, 177,  
  178, 179, 225  
  protobuf, 179  
  wywołań metod między  
  procesami, 119  
Service, *Patrz:* usługa, *Patrz:* usługa  
ServiceManager, 135  
serwer  
  gitolite, *Patrz:* gitolite  
  integracji i kompilacji, 185, 186  
  Jenkins CI, 194  
  WebSocket, 301, *Patrz:*  
  WebSocket  
sieć, 293  
  3G, 159  
  LTE, 159  
  mobilna, 159  
  odpytywanie, 314  
  społecznościowa, 95  
  Wi-Fi, *Patrz:* Wi-Fi  
silnik 3D, *Patrz:* graf sceny  
składnik, 67, 76  
skrypt  
  Gradle, 28  
  IDL, 29  
  Monkey, 28  
słowo kluczowe  
  native, 250  
  oneway, 142  
  synchronized, 51, 52  
SMS, 265, 315  
  odbieranie, 271  
Sony Mobile, 277  
Sony Mobile Xperia, 278  
SQLite, 69, 163, 168, 172  
stała, 40, 49  
  całkowitoliczbowa, 48

standalone toolchain, *Patrz:*  
  narzędzie generowania  
  autonomicznego oprzyrządowania  
starter, 157  
sticky broadcast, *Patrz:*  
  powiadomienie kleiste  
streaming texture, *Patrz:* tekstura  
strumieniowa  
struktura  
  ArrayList, 50  
  HandlerThread, 58, 62  
  LinkedList, 50  
strumień  
  InputStream, 296  
  wejściowy, 176  
  wideo, *Patrz:* wideo strumień  
  wyjściowy, 181  
subskrypcja, 358  
system  
  automatycznego zarządzania  
  pamięcią, 45  
  ciągłej integracji, 194  
  kontroli dostępu, 34  
  kontroli wersji, 33, 34  
  Git, *Patrz:* Git  
  rozproszony, 33  
  operacyjny, 23  
  BeOS, 133  
  Windows, *Patrz:* Windows  
  testowy JUnit, 186  
  uprawnień, 219  
  usuwania nieużytków, 45, 52, 53  
szyfrator, 223

## Ś

środkii, 78, 82

## T

tabela, 168  
tablica, 50, 53  
  sortowanie, 251, 252, 253  
TDD, 185  
tekst, 89  
  czytelność, 89, 90, 92  
  kolor, 92, 93  
  rozmiar, 92  
  układ, 90  
tekstura strumieniowa, 209  
TelephonyManager, 159  
test  
  automatyczny, 27  
  ContentProvider, *Patrz:*  
  ContentProvider testowanie  
  funkcjonalny, 185  
  instrumentacyjny, 185  
  integracyjny, 185

## test

interfejsu użytkownika, *Patrz:*  
interfejs użytkownika test  
jednostkowy, 37, 185, 186, 187  
JUnit, 186  
komplet, 186  
pojedynczy, *Patrz:* przypadek  
testowy  
pokrycie kodu, 185  
uruchamianie, 193  
usług, *Patrz:* usługa testowanie

test case, *Patrz:* przypadek testowy  
test suite, *Patrz:* test komplet  
test-driven development, *Patrz:* TDD  
testowanie, 27, 185, *Patrz też:* test  
regresyjne, 28

tethering, *Patrz:* Wi-Fi tethering  
token, 338, 339  
TooTallNate, 301  
transakcja, 135, 172, 173  
translacja dynamiczna, 46  
tryb

diagnostyczny, 26  
samolotowy, 273  
ścisty, 43  
TCP/IP, 26

TTS, 206, 207

Twitter, 95

## typ

boolean, 164  
danych, 77  
float, 164  
int, 164  
kierunkowy, 139  
in, 139  
inout, 139  
out, 139  
long, 164  
MIME, 77  
String, 164

## U

Uid, *Patrz:* użytkownik identyfikator

Unity3D, 113

uprawnienia, *Patrz też:* ochrona

URI, 77

urządzenie, 229

GATT, *Patrz:* GATT

Nexus, 278

program rozruchowy, *Patrz:*  
bootloaderz systemem Android, 275, 276,  
277, 285

sterownik, 278

zdalne, 319, 322

zmiana konfiguracji, 160

USB, 319, 320, 325

filtr, 320

tryb hosta, 320

usługa, 60, 67, 68, 73, 76, 115,

*Patrz też:* aplikacjaadb, *Patrz:* narzędzie adb

AdMob, 359

billingowa, 356

cykl życia, 115, 116

Google Cloud Messaging, 337

Google Play, 337

Google Play Games, 349

internetowa Google, 337

komunikacja, 126

konwersji plików

multimedialnych, 124

licencjonowania w Google Play,  
356

na pierwszym planie, 120

niszczenie, 116

NsdManager, 325

odpinanie, 99, 120

pasywna, 288

przechowywania danych w

chmurze, *Patrz:* Dysk Google

RESTful, 329, 332

sieciowa, 293, 329

integracja, 303, 305, 308,  
312

wykrywanie, 325, 326

wyszukiwanie, 312

## stała

START\_NOT\_STICKY, 117

START\_REDELIVER\_INTENT,  
117

START\_STICKY, 117

systemowa, 116

dodawanie, 285, 286, 288

testowanie, 190

tworzenie, 116

typ, 116

uruchamianie, 117

wiązanie, *Patrz:* wiązaniewysyłania powiadomień przez  
serwer, *Patrz:* GCM

zamykanie, 138

zatrzymywanie, 99, 121

ustawienia ukryte, 273

## użytkownik

historyjka, 85

identyfikator, 72, 73, 219, 222

podpis, 219

uwierzytelnienie, 305

zaangażowanie, 96

## V

voice over internet protocol, *Patrz:*

VoIP

VoIP, 199, 200

Volley, 297, 299

## W

wątek, 56, 62, 254

główny, 55, 115, 123, 293

blokowanie, 165

interfejsu użytkownika, *Patrz:*

wątek główny

obsługa, 254

tła, 115, 124

WebSocket, 332, *Patrz:* gniazdo

sieciowe

wiązanie, 120

lokalne, 118, 127, 128

wideo, 209, 212

nagrywanie, 201, 214

strumień, 214

widok, 89

cykl życia, 103

projektowanie, 102, 103

widżet, 89, 102, 112

wielodotyk, 108

implementacja, 108

rysowanie palcem, 108

wielowątkowość, 55, 56, 57, 58, 63

wiersz poleceń Git, 33

Wi-Fi, 25, 26, 151, 152, 159, 313,  
319, 325

Direct, 319, 320, 325, 326

tryb współbieżny, 326

infrastrukturalne, 319

mobilny punkt dostępowy, 272

tethering, 270, 272, 282, 319

WifiManager, 159

WindowManager, 134

Windows, 23

wtyczka, 155, 156, 220

android, 28, 33

android-library, 33

instalacja, 138

Wuala, 223

## wyjątek

ClassNotFoundException, 269

nieobsłużony, 27

NoSuchMethodException, 269

OutOfMemory, 52

UserRecoverableAuthException,  
339wyliczenie bezpieczne pod  
względem typów, 48

wywołanie  
 sieciowe, 294, 295, 313  
 błędy, 296  
 zwrotne, 68, 142, 147  
 wywoływacz zdalny, 134  
 wzorzec konsument-producent, 50

**X**

XDA Developers, 278

**Y**

YouTube, 220

**Z**

zadanie  
 Change Method Signature, 41  
 Extract Method, 41  
 refaktoryzacji, *Patrz:*  
 refaktoryzacja

zależność  
 dla kompilacji, 28  
 magazyn zdalny, 31  
 projektu, 28  
 zapytanie, 171, 172  
 optymalizacja, 172  
 zasilanie, 152, 312, 320, *Patrz też:*  
 ładowarka  
 zasobnik aplikacji na ekranie  
 głównym, *Patrz:* starter  
 zasób, 78  
 definiowanie, 78  
 domyślny, 78, 80, 82  
 filtrowanie, 78  
 kwalifikator, 78, 80  
 logiczny, 81  
 łańcuchowy, 79, 80  
 typ, 78  
 zdarzenie  
 ConnectivityManager.CONNECTI  
 VITY\_ACTION, 159  
 MotionEvent, 108

odbiornik, *Patrz:*  
 BroadcastReceiver  
 rozsyłanie, 151  
 systemowe, 69, 157  
 zmienna, 50  
 zygote, 47

**Ż**

żądanie  
 HTTP  
 GET, 295, 296, 300  
 POST, 296  
 równoległe, 299





# 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**

# PLATFORMA ANDROID

## NOWE WYZWANIA

**Android to lider wśród systemów operacyjnych** dla telefonów i tabletów. Spotkasz go w większości tego typu urządzeń. Ciągły rozwój Androida sprawił, że jego obecne wersje zapewniają niesamowite możliwości, a przy tym są przyjazne dla programistów. Jeżeli poznałeś już ten system i marzysz o stworzeniu bardziej zaawansowanej aplikacji, to trafieś na doskonałą książkę.

### W trakcie jej lektury wzbogacisz swoją wiedzę o cenne informacje.

Dowiesz się, jak skonfigurować środowisko programistyczne oraz efektywnie wykorzystać język Java do tworzenia aplikacji. Po tym wstępie zaczniesz poznawać bardziej zaawansowane elementy platformy. Nauczysz się korzystać z zasobów, projektować interfejs użytkownika, tworzyć usługi i zadania pracujące w tle oraz odbierać komunikaty i dokonywać zmian w konfiguracji. Ponadto Twoją uwagę powinny przykuć rozdziały poświęcone testom automatycznym, geolokalizacji oraz bezpieczeństwu tworzonych aplikacji. Na koniec został gwóźdź programu — hakowanie platformy Android oraz wykorzystanie ukrytego API. Brzmi intrygująco? I tak właśnie jest!

### Dzięki tej książce:

- błyskawicznie zaznajomisz się z podstawami platformy Android
- wykorzystasz potencjał języka Java oraz składników Androida
- zapanujesz nad zasobami
- zbudujesz zaawansowany interfejs użytkownika
- poznasz ukryte API platformy
- opublikujesz aplikację w Google Play

### Twoja przepustka do zaawansowanego świata Androida!

**helion.pl**  
księgarnia  
internetowa

Nr katalogowy: 23913



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



Zamówienia telefoniczne:  
**0 801 339900**



**0 601 339900**



**Helion**

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

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

sięgnij po WIĘCEJ



KOD KORZYŚCI

ISBN 978-83-246-9525-6



9 788324 695256

Cena: 67,00 zł

Informatyka w najlepszym wydaniu