Software Developer's JOURNAL 02/2009

CD PYTHON DLA S60 | ANDROID SDK 1.0 | ECLIPSE GANYMEDE SR1 | NETBEANS 6.5

Software Developer's JOURNAL

new ideas & solutions for professional programmers

nr 02 (170) Luty 2009 cena 26,75 stawka VAT 0% INDEX: 35497X

PROGRAMOWANIE URZĄDZEŃ MOBILNYCH

WINDOWS MOBILE
ANDROID
SYMBIAN

ANDROID VS JAVA ME

GPS Z .NET COMPACT FRAMEWORK

BIBLIOTEKA MIESIĄCA

OpenCV
Komputerowa wizja w praktyce

JQUERY I ASP.NET

WPROWADZENIE DO BIBLIOTEKI JQUERY Z PRZYKŁADAMI W ASP.NET AJAX

ORACLE CEP – ZŁOŻONE PRZETWARZANIE ZDARZEŃ

COMPLEX EVENT PROCESSING/ EVENT STREAM PROCESSING

APLIKACJE DESKTOPOWE

TWORZENIE BOGATYCH INTERFEJSÓW UŻYTKOWNIKA W PRZEGLĄDARCE WWW

NA CD

- Python dla S60
- Android SDK 1.0
- Eclipse Ganymede SR1
- Netbeans 6.5

ISSN 1734-3917

Programowanie urządzeń mobilnych

DOSTĘPNY W DRUGIEJ POŁOWIE LUTEGO

Jak robić żeby się nie narobić
skorzystaj z rozwiązań **IBM Lotus**

IBM we współpracy z Software Developer's Journal przygotowuje numer specjalny SDJ poświęcony rozwiązaniom IBM Lotus

W numerze m.in.:
Nowoczesne studio projektowe IBM Lotus Notes Domino 8.5.
Przygotuj elektroniczny formularz dla swojego szefa.
Szybko zintegruj pocztę z bazą danych.
Jak stworzyć mashupy?
Przygotuj kurs e-learningowy w firmie.
Z Księżyca też można pracować w zespole.
Wywiad z super bloggerem.

DVD
z rozwiązaniami
IBM Lotus i kursami
e-learningowymi

Lotus. software

Software Developer's JOURNAL
new ideas & solutions for professional programmers

IBM

mPKI

MobileExperts

Innovative mobile solutions

Autoryzacja operacji bankowych

Szyfrowanie komunikacji w transakcjach giełdowych

Elektroniczne podpisywanie dokumentów i korespondencji

mPKI - SZEROKI ZAKRES ZASTOSOWAŃ – elektroniczne podpisywanie i uwierzytelnianie umów, e-maili, korespondencji z e-urzędami, autoryzowanie dostępu do chronionych baz danych, sieci VPN, szyfrowanie transakcji bankowych i giełdowych.

mPKI - WYGODA - swobodny dostęp do PKI w kraju i za granicą. Wystarczy posiadać telefon komórkowy, PDA, lub modem GSM wraz z kryptograficzną kartą SIM.

mPKI - BEZPIECZEŃSTWO - dostarczona przez operatora GSM i chroniona jego polityką bezpieczeństwa karta SIM sama (w procesie aktywacji) generuje parę kluczy (prywatny i publiczny) udostępniając na zewnątrz jedynie klucz publiczny. Klucz prywatny nigdy nie opuszcza karty SIM, a dostęp do niej jest autoryzowany kodem PIN i PUK.

Mobile Experts Sp. z o.o. to doświadczony zespół specjalistów od wielu lat związanych z branżą mobilnego IT (Gold Award w SIMagine 2007, Grand Prix w konkursie Gazety Bankowej na Najlepszy Projekt Informatyczny 2005). Firma tworzy innowacyjne rozwiązania i aplikacje mobilne i świadczy w oparciu o nie usługi w zakresie mobilnych kanałów dostępu.

Więcej informacji pod adresem www.mobileexperts.pl

SPIS TREŚCI

06 Aktualności
Rafał Kocisz

33 Opis CD
Duży zbiór materiałów do programowania urządzeń mobilnych.

BIBLIOTEKA MIESIĄCA

12 OpenCV: komputerowa wizja w praktyce
Rafał Kocisz
Czy komputer może widzieć podobnie jak człowiek? Czy komputerowa wizja to mit czy rzeczywistość? Na te, oraz inne pytania można znaleźć odpowiedz, czytając poniższy artykułu traktujący o wyjątkowej bibliotece wywodzącej się z laboratoriów firmy Intel.

JĘZYKI PROGRAMOWANIA

18 Oracle CEP – złożone przetwarzanie zdarzeń
Waldemar Kot
Complex Event Processing (CEP) to jedna z technik przetwarzania zdarzeń w aplikacjach zdarzeniowych, posługujących się paradygmatem *nasłuchuj-i-odpowiedz* (ang. *listen-and-respond*), czyli takich, w których logika biznesowa jest uruchamiana w wyniku pojawienia się określonego zdarzenia.

20 jQuery i ASP.NET – wprowadzenie do biblioteki z przykładami w ASP.NET AJAX
Łukasz Szewczak
Otwarta biblioteka JavaScript jQuery sprawia, że pisanie kodu po stronie klienta staje się przyjemniejsze, a dzięki połączeniu jQuery z technologią ASP .NET AJAX otrzymujemy potężny, efektywny szkielet aplikacyjny do pisania aplikacji ajaksowych. Łukasz Szewczak wprowadza nas do biblioteki z przykładami ASP.NET AJAX.

WARSZTATY

26 Aplikacje desktopowe – Ext GWT (GXT) jako środowisko do tworzenia bogatych interfejsów użytkownika w przeglądarce WWW
Rafał Pietrasik
Do niedawna w interfejsach opierających się na przeglądarce uzyskanie pełnej funkcjonalności, którą oferowały aplikacje oparte na grubym kliencie, było bardzo trudne, jeśli wręcz nie niemożliwe. Środowisko GWT w połączeniu z biblioteką Ext GWT zmieniają diametralnie ten stan rzeczy. W artykule autor pokazuje jak łatwo jest stworzyć taką aplikację.

34 Wykładnicze mapy cieni – charakterystyka i implementacja algorytmu miękkich cieni w grafice komputerowej czasu rzeczywistego
Paweł Rohleder
Mapy cieni są najczęściej wykorzystywaną techniką cieniowania w grach komputerowych. W niniejszym artykule postaram się przybliżyć technikę map cieni oraz zaimplementować algorytm miękkich cieni przy wykorzystaniu HLSL i biblioteki DirectX 9.0.

PROGRAMOWANIE URZĄDZEŃ MOBILNYCH

40 PKI na urządzeniach mobilnych – autoryzacja i uwierzytelnianie użytkowników
Leszek Siwik, Krzysztof Lewandowski, Adam Woś
W artykule autorzy przyglądają się implementacji PKI na urządzeniach mobilnych, na przykładzie wdrożenia w jednej z polskich służb mundurowych. Tłumaczą, co to LAP i LASS, dlaczego łatwo jest używać CSP oraz jakie niespodzianki czekają na piszących oprogramowanie na karty SIM

Miesięcznik **Software Developer's Journal** (12 numerów w roku) jest wydawany przez Software-Wydawnictwo Sp. z o.o.

Dyrektor wydawniczy: Anna Adamczyk

Junior Market Manager: Anna Adamczyk

Senior Product Manager: Katarzyna Juszczyńska

Redaktor naczelny: Łukasz Łopuszański *lukasz.lopuszanski@software.com.pl*

Kierownik produkcji: Marta Kurpiewska *marta.kurpiewska@software.com.pl*

Projekt okładki: Agnieszka Marchocka

Skład i łamanie: Grzegorz Laskowski *grzegorz.laskowski@software.com.pl*

Wyróżnieni betatesterzy: Ł. Lechert

Nakład: 6 000 egz.

Adres korespondencyjny:
Software-Wydawnictwo Sp. z o.o.,
ul. Bokserska 1, 02-682 Warszawa, Polska
tel. +48 22 427 36 91, fax +48 22 224 24 59
www.phpsolmag.org cooperation@software.com.pl

Dział reklamy: *adv@software.com.pl*
Prenumerata: Marzena Dmowska *pren@software.com.pl,*
tel. +48 22 427 36 79; +48 22 427 36 53

Dołączoną do magazynu płytę CD przetestowano programem AntiVirenKit firmy G DATA Software Sp. z o.o.

Redakcja dokłada wszelkich starań, by publikowane w piśmie i na towarzyszących mu nośnikach informacje i programy były poprawne, jednakże nie bierze odpowiedzialności za efekty wykorzystania ich; nie gwarantuje także poprawnego działania programów shareware, freeware i public domain.

Uszkodzone podczas wysyłki płyty wymienia redakcja.

44 Mobilny magazyn – łamigłówki architektoniczne
Rafał Deja
Autor przedstawia dylematy związane z doborem właściwej architektury systemu komputerowego do obsługi magazynu. Zadaniem jest przeniesienie wybranych procesów logistycznych na urządzenia mobilne przy równoczesnej integracji z tradycyjnym systemem zarządzania magazynem a szerzej z systemem klasy ERP.

46 Jak to się robi w Krakowie
Kamil Kowalski, Artur Chruściel
Dla każdego, kto nigdy nie miał do czynienia z programowaniem urządzeń mobilnych, specyfika tego zajęcia sprowadza się do kilku szeroko znanych haseł, takich jak ograniczona ilość pamięci, real-time, środowisko embedded itp. Wszystko to oczywiście prawda, jednak czasy się zmieniają i wiele z tych haseł straciło swoją ważność, inne zyskały na znaczeniu, a w dodatku pojawiły się całkiem nowe. Nowe – nie znaczy nieznane w świecie informatyki, lecz nigdy niebrane pod uwagę w kontekście systemów embedded.

50 Wąż w komórce
Marian Witkowski
Telefony komórkowe obecnie to nie tylko proste urządzenia do wykonywania połączeń głosowych czy korzystania z wiadomości tekstowych i multimedialnych. To zaawansowane mikrokomputery, pełniące często rolę elektronicznych terminarzy (PDA), wyposażone nierzadko we własny system operacyjny. Ewolucja, jaką przeszły umożliwiła rozszerzanie funkcjonalności aparatów poprzez instalację aplikacji wykonanych nie przez ich producentów.

56 GPS z .NET Compact Framework
Daniel Stoiński
Chip GPS coraz częściej znajduje się w niedrogich telefonach oraz urządzeniach PDA z zainstalowanym systemem Windows CE lub Mobile. Z dobrą baterią można takie urządzenie zabrać nawet na całodzienną wędrówkę i późniejsze pytania *gdzie byłeś?* skwitować gotowym zapisem trasy.

66 Android vs. Java ME
Albert Wachowicz
Platforma Sun dla urządzeń mobilnych istnieje już od dłuższego czasu. Podbiła rynek dzięki implementacji na wielu urządzeniach różnych producentów (obecnie istnieje około 2000 różnych typów modeli z Java ME). Główną ideą Sun było zapewnienie uniwersalności i jednolitości aplikacji wytworzonych na platformę Java ME.

FELIETON

76 Czy nowoczesna firma może funkcjonować bez najnowszej technologii?
Adam Tyla

Wszystkie znaki firmowe zawarte w piśmie są własności odpowiednich firm. Zostały użyte wyłącznie w celach informacyjnych.

Redakcja używa systemu automatycznego składu

Osoby zainteresowane współpracą prosimy o kontakt:
cooperation@software.com.pl

Druk: ArtDruk

Wysokość nakładu obejmuje również dodruki. Redakcja nie udziela pomocy technicznej w instalowaniu i użytkowaniu programów zamieszczonych na płycie CD-ROM dostarczonej razem z pismem.

Sprzedaż aktualnych lub archiwalnych numerów pisma po innej cenie niż wydrukowana na okładce – bez zgody wydawcy – jest działaniem na jego szkodę i skutkuje odpowiedzialnością sądową.

Windows 7: DirectX 10 obsługiwany przez CPU

Od dłuższego czasu krążą pogłoski, że Windows 7, w stosunku do swojego poprzednika, będzie miał mniejsze wymagania sprzętowe. Większość tego typu informacji pozostawała do tej pory jednak w sferze spekulacji, teraz pojawiły się pierwsze konkretne wiadomości. Ma tak się stać za sprawą implementacji w nowym systemie Microsoftu platformy WARP (Windows Advanced Resterization Platform). Umożliwi ona uruchamianie wersji 10.0 i 10.1 bibliotek DirectX poprzez odciążenie słabszych, często zintegrowanych układów graficznych przez procesor, który ma odpowiadać za wykonywanie operacji graficznych. Co ciekawe, możliwość taka ma być dostępna nie tylko w przypadku nowszych układów wspierających DX10, ale też wykorzystujących DirectX 9. Według testów Microsoft, Crysis uruchomiony na niskich detalach w konfiguracji z czterordzeniowym procesorem Intel osiągnął wynik 5.69 FPS w rozdzielczości 800 na 600 pikseli podczas, gdy kompatybilny układ IGP Intela osiąga zaledwie 5.17 FPS, co czyni go zbędnym w połączeniu z nowoczesnym procesorem. Minimalne wymagania WARP to procesor taktujący z częstotliwością 800 MHz oraz 512 MB pamięci RAM, nie potrzebuje z kolei obsługi instrukcji MMX, SSE czy SSE2. Microsoft najprawdopodobniej próbuje przekonać swoimi działaniami użytkowników starszego sprzętu do wyboru Windows 7, nie zaś Windows XP. Z drugiej strony, krok ten wydaje się swego rodzaju ruchem wstecz obserwując obecną tendencję na rynku.

http://www.theinquirer.net/

Linux gotowy na IPv6

Linux Foundation, organizacja nonprofit zajmująca się działaniami na rzecz szybkiego rozwoju systemu GNU/Linux poinformowała o pełnej zgodności tego systemu z certyfikacją DOD IPv6. W 2007 r. Biuro Departamentu Obrony (DISA) podjęło się stworzenia projektu zakładającego zapewnienie możliwości komunikacji pomiędzy urządzeniami za pomocą IPv6. Prace nad implementacją stosu w jądrze trwały już od kilku lat, ale wciąż nie spełniała wymagań DISA, między innymi w zakresie obsługi protokołów ICMP, IPSec, DHCP i MIB. W odpowiedzi na taki stan rzeczy, Fundacja utworzyła grupę dedykowaną problemowi składającą się głównie z przedstawicieli firm partnerskich. Jej zadaniem było wywieranie nacisku na twórcach jądra w celu dostosowania go do wymagań Biura. W efekcie ich prac, otrzymało ono wreszcie ubiegany certyfikat, dzięki czemu wszystkie dystrybucje wyposażone w najnowszą wersję jądra powinny spełniać wymogi. Latem, identyczny certyfikat został przyznany dla SUSE Linux Enterpise (SLES) oraz Red Hat Enterprise Linux 5.2. W gronie jego posiadaczy znaleźć można także systemy Windows Vista i Solaris 10.

http://www.gcn.com/

Google: czy Native Client zmieni oblicze Sieci?

Od kilku miesięcy blogosfera plotkowała o nowym systemie operacyjnym od Google, który miał być całkowicie autorskim projektem według jednych lub kolejną dystrybucją Linuksa według drugich. 8 grudnia 2008 roku gigant z Mountain View poinformował o projekcie, którego nikt z dyskutantów nie był w stanie sobie wyobrazić – otwartej technologii o nazwie Native Client. Czym jest Native Client? Brad Chen, szef zespołu pracującego nad projektem napisał o tym tak: „W Google zawsze staramy się zrobić z Sieci jak najlepszą platformę. Dlatego stworzyliśmy projekt Native Client – technologię, która udostępni deweloperom pełną moc obliczeniową procesora, pozwalając jednocześnie zachować neutralność względem przeglądarki i systemu operacyjnego". Native Client jest binarną wtyczką dla przeglądarek, która ma umożliwić gigantowi z Mountain View pełne panowanie w dziedzinie aplikacji webowych. Korporacja, która opanowała większość rynku wyszukiwarek i reklamy on-line nie była do tej pory w stanie zapewnić użytkownikom aplikacji, które pod względem szybkości działania i funkcjonalności mogłyby dorównać programom pisanym bezpośrednio dla systemów operacyjnych. Teraz, jeśli projekt Native Client dojrzeje, Google zachęci użytkowników do jego instalacji a programistów do pisania dla niego oprogramowania, Google będzie mogło zwyciężyć także i na tym froncie. Pod wieloma względami Native Client jest rozwiązaniem zbliżonym do Javy, Flasha czy Silverlighta – biblioteką uruchomieniową, która działając na szczycie systemu operacyjnego pozwala na uruchamianie neutralnych względem platformy aplikacji. Dzięki bezpośredniemu jednak dostępowi do warstwy sprzętowej systemu operacyjnego, pozwoli na uruchamianie wewnątrz przeglądarek nawet takich aplikacji jak gry 3D (jako jeden z przykładów wykorzystania technologii jest gra Quake, którą można uruchomić z poziomu przeglądarki – płynnie i z dźwiękiem). Obecnie Native Clienta można uruchomić na praktycznie dowolnym komputerze z procesorem x86, pracującym pod kontrolą systemu Windows, Mac OS X lub Linux. Wtyczka działa w przeglądarkach Google Chrome, Firefox, Safari i Opera. Twórcy technologii zapowiadają rychłe przeniesienie Native Clienta na platformy ARM i PowerPC. Więcej informacji oraz pliki instalacyjne do pobrania można znaleźć na stronie projektu w ramach serwisu Google Code (*http://code.google.com/p/nativeclient/?tbbrand=GZEZ&utm_campaign=en&utm_source=en-et-osrcblog&utm_medium=et*).

http://webhosting.pl/

NetBeans 6.5

Pół roku pracy zajęło firmie Sun Microsystems przygotowanie kolejnej wersji zintegrowanego środowiska programistycznego NetBeans. Edycja 6.5 nie jest zupełnie nowym wydaniem, ale poważną aktualizacją poprzedniego, czyli 6.1. Nowe IDE jest coraz silniej adresowane poza swoją klasyczną grupę docelową, czyli programistów Javy i oferuje coraz więcej funkcji także dla programistów aplikacji internetowych. PHP, Ruby, Groovy, SVG, Ajax, ulepszone edytory JavaScriptu i CSS-ów – to odpowiednie hasła do opisu NetBeans 6.5. Nie oznacza to, że nic się nie zmieniło w kwestii Javy. Ulepszono np. obsługę kilku frameworków (Spring, Hibernate, JSF, JSF CRUD Generator i JPA) i programiści mogą teraz budować usługi w oparciu o REST z wykorzystaniem tabel bazodanowych lub elementów JPA. W pakiecie znajdują się także narzędzia dla sunowskiej platformy RIA, czyli JavyFX, która prawdopodobnie w finalnej wersji zostanie opublikowana w grudniu 2008 roku. Osoby wykorzystujące PHP otrzymały do dyspozycji poprawiony edytor z kolorowaniem i uzupełnianiem kodu, możliwość debugowania za pomocą Xdebug, generator snippetów dla MySQL-a i obsługę metody przeciągnij i upuść w odniesieniu do usług SaaS umieszczanych w plikach PHP. Z kolei pod hasłem „JavaScript i CSS" oprócz poprawionych edytorów można też zapisać ulepszony menedżer bibliotek (domyślnie instalowane są Yahoo UI, Woodstock, jQuery, Dojo, Scriptaculous i Prototype). Z długą listą zmian w nowym NetBeans można zapoznać się pod adresem *http://wiki.netbeans.org/NewAndNoteWorthyNB65*. Środowisko w sześciu różnych odmianach można za darmo pobrać z witryny *http://www.netbeans.org/downloads/index.html*.

http://webhosting.pl

Kompilator C/C++ dla Flasha

Już wkrótce dla platformy Flash może pojawić się wiele aplikacji, które dotychczas działały pod kontrolą klasycznych systemów operacyjnych. Jest to możliwe dzięki kompilatorowi Alchemy, pozwalającego transformować kod napisany w C i C++ na ActionScript 3.0, który z kolei można uruchomić w ActionScript Virtual Machine (AVM2). Uzyskany w ten sposób kod bajtowy może być następnie łatwo uruchomiony przez FlashPlayera 10 lub w środowisku AIR 1.5. Według przeprowadzonych przez Adobe testów, użycie Alchemy prowadzi do jedynie niewielkiego spadku wydajności kodu. Wydajność osiągana dzięki takiemu przekompilowaniu jest znacząco wyższa od ActionScript 3.0 i od 2-10 razy niższa od natywnego kodu C/C++. Firma podkreśla, że aplikacje Flash nie powinny być jednak pisane ani w C, ani w C++. Właściwym zastosowaniem kompilatora ma być umożliwienie aplikacjom webowym korzystania z niezliczonych, wolnych bibliotek napisanych w tych wydajnych językach. Adobe wyjaśniło, że chodzi przede wszystkim o „transkodowanie audio i wideo, kryptografię i parsery XML". Pod adresem *labs.adobe.com/technologies/alchemy/videos/brandenhall/* można obejrzeć wypowiedź Brandena Halla, CTO firmy Automata Studios, który opowiada o swoich doświadczeniach z wykorzystaniem Alchemy do przeniesienia bibliotek Ogg Vorbis na Flasha. Kompilator dostępny jest na razie w wersji beta dla Linuksa i MacOS X-a. Aby uruchomić go pod Windows, należy zainstalować symulujące Uniksa środowisko CygWin. Nie wiadomo jeszcze, kiedy Alchemy opuści Adobe Labs i stanie się oficjalnym produktem firmy. By ułatwić zapoznanie się z kompilatorem, Adobe przygotowało przykładowe biblioteki, nakłania też programistów do dzielenia się własnym kodem. Warto też wspomnieć, iż w ramach demonstracji rozwiązania zaprezentowano grę Quake w wersji flashowej, skompilowanej przy użyciu wczesnej wersji Alchemy. Później przedstawiono również działający interpreter Pythona oraz emulator Nintentdo.

http://webhosting.pl

Windows 7: szybciej niż się spodziewasz?

Microsoft przekonuje, że Windows 7 nie stanowi tak wielkiego skoku technologicznego jaki wystąpił w przypadku Visty, ale mimo tego warto będzie uaktualnić system. Wielu użytkowników obawia się, że migracja do Windows 7 będzie równie problematyczna, jak to było w przypadku Windows Vista - zwłaszcza na początku, tuż po premierze systemu. W reakcji na zapewnienia Microsoftu, że Windows 7 to tylko wydanie pośrednie, już teraz pojawiają się głosy, że w takim razie siódemka nie wniesie niczego nowego. Bill Veghte, senior vice-president w dziale Windows Business podjął się trudnego zadania i przekonywał dzisiaj inwestorów giełdowych, że z Windows 7 problemów nie będzie, a nowe funkcje faktycznie będą skłaniały do uaktualnienia. Jest wiele nowości, które sprawiają, że Windows 7 nie można nazwać Service Packiem - mówił. Nowy pasek zadań, pełna obsługa ekranów dotykowych, Direct Access pozwalający firmom omijać konieczność tworzenia sieci VPN to tylko niektóre z wymienionych przez niego zalet nowego systemu Windows. Co ciekawe, równolegle pojawiły się nowe informacje na temat dostępności pierwszej wersji beta. Według Keitha Comba i jego bloga hostowanego przez Microsoft TechNet Blogs beta Windows 7 może pojawić się już w pierwszym kwartale 2009 roku. Z nieoficjalnych doniesień wynika, że wydanie to będzie bardzo obszerne - całkiem możliwe, że Microsoft zdecyduje się na w pełni otwarty program beta, podobnie jak to było z Windows Vista (choć tam nastąpiło to nieco później - dopiero w fazie Beta 2).

http://www.cnet.com/

Linux na iPhone

Systemy spod znaku pingwina nie cieszą się wysokim zainteresowaniem ze strony użytkowników domowych, nieco inaczej sytuacja przedstawia się w przypadku zastosowań serwerowych, a także urządzeń wbudowanych, takich jak konsole do gier, odtwarzacze DVD i MP3, czy... tostery. Stworzenia specjalnego systemu operacyjnego bazującego na komponentach ze świata Linuksa przeznaczonego do obsługi zaawansowanych telefonów komórkowych, podjął się dwa lata temu Google, który parę miesięcy później zadebiutował w telefonach z rodziny G1. Tymczasem grupie osób zrzeszonej w projekt o nazwie Linux on the iPhone, udało się przeportować stabilną gałąź 2.6 jądra pod sprzęt Apple, w tym urządzeń iPod Touch, iPhone i iPhone 3G. Ze względu na stosunkowo wczesną fazę rozwojową stabilne działanie pewnych elementów jest kwestionowane. W chwili obecnej gotowe są między innymi sterowniki Framebuffer, Serial (w tym poprzez sterownik USB) oraz przerwania, zegary i MMU (Memory Management Unit) - technika realizująca dostęp do pamięci fizycznej. Obsługa nośników pamięci Flash (NAND) w trybie tylko do odczytu dostępna jest poprzez openboot, nie została jednak jeszcze przeportowana. Nadal brakuje wsparcia dla sieci Wi-Fi, ekranów dotykowych, dźwięku, czy akcelerometru. Dla osób zainteresowanych autorzy przygotowali demonstrację wideo przedstawiającą działanie portu w praktyce wraz z instrukcją uruchomienia.

http://linuxoniphone.blogspot.com

Co nowego w Silverlight 3?

Stosunkowo niedawno ukazała się oficjalna, stabilna wersja platformy Silverlight 2.0, nowej technologii Microsoftu służącej do prezentacji zaawansowanej zawartości multimedialnej w Internecie. Tymczasem z różnych źródeł docierają pierwsze informacje o planowanych zmianach, które czekają nas w wydaniu tej technologii oznaczonej numerem 3.0. Jak czytamy na blogu Scotta Guthria zajmującego w Microsofcie stanowisko vice president działu .NET Developer Platform, nowy Silverlight 3.0 ma wprowadzać przede wszystkim szereg istotnych nowości w obsłudze multimediów, także pod kątem sprzętowym. Pojawi się długo oczekiwane wsparcie dla H.264, kodeka wykorzystywanego obecnie głównie do transmisji telewizji wysokiej rozdzielczości. Zaimplementowana zostanie obsługa grafiki 3D oraz jej sprzętowej akceleracji. Silverlight 3 przyniesie również zmiany wprowadzone z myślą o programistach, między innymi bogatszą obsługę bindowania danych oraz dodatkowe kontrolki. Również kolejne wersje Visual Studio i Visual Developer Express oferować będą możliwość interaktywnego projektowania rozwiązań opartych na Silverlight. Premiera Silverlight 3.0 planowana jest w roku 2009.

http://neowin.net/

Polska na czele w rankingach piractwa komputerowego

Polscy użytkownicy Internetu bardzo chętnie udostępniają pirackie pliki do ściągnięcia w sieciach P2P – wynika z „Raportu na temat gospodarki czarnorynkowej" przygotowanego przez firmę Symantec. Polska zajęła 6. miejsce w światowym rankingu ilości udostępnianych plików (5% wszystkich udostępnień), wyprzedzając takie kraje, jak: Francja, Szwecja, Holandia i Australia. Przed Polską znalazły się: USA (19%), Wielka Brytania (7%), Kanada (6%), Hiszpania (5%) oraz Brazylia (5%). Najczęściej udostępniane aplikacje to gry na komputery PC, które stanowią aż 49% wszystkich nielegalnych programów w sieci BitTorrent. Niestety duży wkład w upowszechnianiu pirackich gier mają polscy użytkownicy. Aż 8% wszystkich udostępnianych plików z grami PC pochodzi z Polski, co daje nam drugie miejsce na świecie! Pierwsze miejsce przypadło USA (14%), trzecie Brazylii (7%). Firma Symantec obserwowała sieć BitTorrent oraz transfer pirackimi plikami w okresie trzech miesięcy – od lipca do września 2008 roku.

http://www.symantec.pl/

Bill Gates unowocześni polskie biblioteki

Polski rząd wraz z Fundacją Billa Gatesa podjęli pierwsze działania na rzecz zmiany polskich bibliotek w prawdziwe centra kultury na miarę XXI wieku. Przeznaczyli na ten cel 200 milionów złotych - czytamy w serwisie Gazeta.pl. Celem Fundacji jest przekształcenie bibliotek, zwłaszcza w małych miasteczkach tak, aby pełniły funkcję centrum życia kulturalnego mieszkańców. Mają dostarczać i uczyć obsługi Internetu, a także stanowić miejsce spotkań organizacji. Całość ma przypominać w działaniu program budowy boisk piłkarskich oraz budowy dróg, gdzie część pieniędzy pochodzi od rządu, część od samorządów. Pewna ich część ma zostać wykorzystana też do podnoszenia standardów sanitarnych.

http://gazeta.pl/

Wyłączenie komputera SMS-em

Firma Lenovo udostępni oprogramowanie, za pomocą którego użytkownicy będą mogli wyłączyć laptopa wysyłając wiadomość SMS. Funkcja ma nosić nazwę Lenovo Constant Secure Remote Disable i ma być przydatna w przypadku kradzieży lub zagubienia laptopa. Z każdym ThinkPadem będzie można skojarzyć do 10 numerów telefonów, z których wysłanie SMS-a sprawi, że komputer stanie się bezużyteczny dla złodzieja. Laptop zostaje unieruchomiony przez funkcję zaszytą w BIOSie i jego ponowne uruchomienie będzie możliwe jedynie po podaniu hasła. Ponieważ wyłączenie następuje przez sieć komórkową, w laptopie musi być obecna odpowiednia karta, użytkownik musi płacić abonament operatorowi komórkowemu, zaś sam laptop musi znajdować się w zasięgu sieci.

http://www.eweek.com/

Finalna wersja Google Chrome już dostępna!

Trzymiesięczny okres testowy to zdaniem twórców przeglądarki Google Chrome wystarczająco dużo czasu na wyłapanie wszystkich błędów i dopracowanie aplikacji w takim stopniu, by mogła ona opuścić fazę testów. Tak też się stało - Google Chrome - wydane zostało oficjalnie w wersji 1.0. W tym momencie niemile zdziwieni będą użytkownicy systemów operacyjnych innych niż Windows: Google Chrome, pomimo statusu wersji finalnej, działa tylko pod systemem rodem z Redmond... Według słów Sundara Pichaia, wiceprezesa ds. zarządzania produktem, prace nad programem na systemy alternatywne cały czas postępują. Warto w tym miejscu przypomnieć, że przeglądarka Google Chrome ujrzała po raz pierwszy światło dzienne we wrześniu 2008 roku. W pierwszych dniach po wydaniu aplikacja cieszyła się olbrzymim zainteresowaniem wśród użytkowników. Z czasem jednak popularność ta bardzo szybko spadła, liczne błędy, luki itp. powodowały, że większość osób wracała do swoich dawnych przeglądarek. Firma Google z kolei stara się robić wszystko co możliwe, by zainteresować swoim produktem potencjalnych użytkowników. Jednym z takich zabiegów jest plan wprowadzenia w do Chrome obsługi tzw. rozszerzeń. Aktualnie w dużej mierze mają to być rozwiązania dobrze znane z konkurencyjnego oprogramowania, pozwalające użytkownikowi dostosować przeglądarkę do własnych potrzeb. Według udostępnionych przez Google informacji mechanizm obsługi rozszerzeń przypominał będzie podobny mechanizm zastosowany

Google Chrome

w konkurencyjnym Firefoksie. Twórcy Chrome na pierwszy plan stawiają prostotę instalacji i łatwość dostępu do udostępnionych wtyczek. Pomimo, że obecnie nie podano konkretnego terminu, w którym zaimplementowana zostanie nowa funkcjonalność to już teraz wiadomo jakie rozszerzenia pojawią się jako pierwsze. Wśród nich znaleźć się ma m.in. popularny AdBlock, umożliwiający blokowanie wyświetlania uciążliwych reklam na stronach internetowych, rozbudowany moduł historii i pobierania, mechanizm kontroli rodzicielskiej czy wtyczka Skype'a i RealPlayera. Takie posunięcie ze strony Google znacząco przyczynić się może do wzrostu popularności przeglądarki Chrome. Obsługa rozszerzeń cieszy się aktualnie olbrzymim zainteresowaniem wśród użytkowników np. Firefoksa i podobny sukces przynieść może również raczkującej na chwile obecną aplikacji giganta w branży wyszukiwarek.

http://www.techcrunch.com/

Python 3000 już jest!

4 grudnia 2008 roku, o godzinie 2:51 rano czasu polskiego, oficjalnie została wydana nowa wersja Pythona 3.0, znanego również jako Python 3000. Pod wieloma względami jest to wydanie wyjątkowe, a zarazem dla wielu kontrowersyjne. Wyjątkowe dlatego, że usuwa z języka wiele niedogodności przeszkadzających w programowaniu, kontrowersyjne dlatego, że aby to osiągnąć programiści Pythona musieli zerwać z wcześniejszą zgodnością. Pomimo ryzyka stałego rozszczepienia społeczeństwa, developerzy nie bali się wprowadzenia zmian, dzięki którym Python stanie się lepszym językiem. Oczywiście nie zapomniano o ogromnej rzeszy programistów i programów, które wykorzystują Pythona z serii 2.x, dwa miesiące wcześniej wydano Pythona 2.6, który wprowadza wiele zmian znanych z Pythona 3.0, ale równocześnie zachowuje kompatybilność wstecz. Taki zabieg ma na celu ułatwić migrację do nowej gałęzi. Developerzy Pythona spędzili również dużo czasu na tworzeniu programu 2to3, który automatycznie przekonwertuje nasz program do nowszej wersji, a co ważniejsze, działa on wyśmienicie. Okazja do wprowadzenia zmian, bez oglądania się na wsteczną zgodność, spowodowała napływ tak wielu różnorakich pomysłów, że musiał powstać specjalny dokument opisujący co na pewno nie zmieni się w nowym Pythonie: http://www.python.org/dev/peps/pep-3099/.

http://linuxnews.pl

JavaFX 1.0 wydany

Sun Microsystems już dawno pogodziło się z klęską w wojnie o panowanie nad desktopami w dziedzinie aplikacji typu RIA – taka jest przynajmniej obiegowa opinia. W ostatnich latach amerykańska firma koncentrowała się na rynku urządzeń mobilnych, starając się wypracować jak najlepszą pozycję dla JavyFX. Już podczas konferencji JavaOne 2007 Sun koncentrował się na smartfonach i usługach sieciowych. Korporacyjne produkty oraz serwery trafiły na boczny tor. Podczas imprezy zaprezentowano nowy język skryptowy o nazwie JavaFX Script. Miał on zapoczątkować rozwój nowej platformy dla urządzeń mobilnych o nazwie JavaFX Mobile. Mimo że Sun Microsystems koncentrował się na komórkach, to jednocześnie zapowiedziała wersje swojego oprogramowania dla desktopów, a nawet cyfrowych telewizorów. JavaFX Mobile miała pojawić się pod koniec 2007 roku – a pierwsze urządzenia planowano udostępnić klientom w pierwszej połowie 2008 roku. W końcu jednak amerykańska korporacja wycofała się z tej inicjatywy. Zamiast tego skoncentrowała się na desktopowej wersji swojej platformy. Na początku grudnia 2008 roku Sun Microsystems ogłosił wydanie pakietu JavaFX Desktop 1.0, nazywanego po prostu JavaFX 1.0. Bazująca na tym rozwiązaniu edycja mobilna ma pojawić się dopiero wiosną 2009 roku. Zunifikowane środowisko pozwoli na łatwe przenoszenie aplikacji pomiędzy platformami, a jego rozprowadzanie w binarnej formie zapobiegnie „bałkanizacji" języka, tak aby producenci nie mogli popsuć kompatybilności między wersjami, tworząc własne, przerobione wersje JavyFX. Eric Schwartz, CEO Sun Microsystems powiedział: „udostępniliśmy binarne pakiety producentom telefonów, tak aby możliwe stało się zunifikowanie implementacji mobilnej Javy. Oceniamy że rynek ten jest wart miliardy dolarów". Firma zamierza pobierać tantiemy od każdego sprzedanego telefonu z JavaFX – a że marka Javy jest dobrze rozpoznawalna, Sun liczy, że przyczyni się to do znacznej poprawy sytuacji finansowej firmy. Schwartz podkreślił, że producenci powinni zainteresować się tą platformą przede wszystkim dlatego, że pochodzi ona od niezależnego dostawcy technologii, a nie potencjalnego rywala (jak jest to w wypadku Microsoftu czy Nokii). JavaFX 1.0 zawiera wtyczki do NetBeans 6.5, Eclipse, a także do Adobe Creative Suite 3 i 4. Jednocześnie Sun poinformował, że współpracuje obecnie z jakąś zewnętrzną firmą (jej nazwy nie podano). Celem jest wydanie narzędzia do szybkiego tworzenia aplikacji (RAD). Ma ono uprościć proces przygotowywania programów w JavaFX Scripcie. JavaFX ma znacznie ułatwić pisanie aplikacji multimedialnych. Napisanie odtwarzacza multimedialnego w tradycyjnej Javie wymagało ok 100 linii kodu, w JavaFX Script można to zrobić w ramach 20-30 linii. Generalnie, dzięki JavaFX Sun chce osiągnąć to, co nie udało mu się ze zwykłą Javą. Miała być ona przenośna, ale różnice pomiędzy poszczególnymi wersjami utrudniały to. Ponadto inaczej się pisało aplikacje desktopowe, inaczej mobilne a jeszcze inaczej aplety. Dzięki JavaFX ma się to zmienić. Czy tak się rzeczywiście stanie? Czas pokaże.

http://webhosting.pl

Moonlight 1.0 beta 1

Wydano pierwszą betę Moonlight 1.0, implementacji technologii Silverlight na platformę Linux. Prace nad Moonlight rozpoczęły się w ubiegłym roku podczas konferencji MIX07. Wtedy Miguel de Icaza, kierujący projektem Mono, oraz jego grupa, stworzyli działający prototyp w ciągu 20 dni. Kiedy Silverlight 1.0 został wydany we wrześniu, Microsoft ogłosił plany współpracy z pracującymi w Novellu programistami Moonlight w celu upewnienia sie, że Silverlight będzie dostępny także na Linuksie. Korporacja dostarczyła wtedy zestawy testów oraz zgodziła się na udostępnienie binarnych kodeków audio i wideo dla plugina linuksowego. Moonlight 1.0 jest napisany w C++ i jest udostępniony jako wtyczka dla Firefoksa. Posiada własny silnik JavaScript. Jest w całości otwarty i podlega licencji GNU LGPLv2. Nie zawiera kodeków własnościowych. W przypadku konieczności ich użycia pojawia się możliwość ich ściągnięcia. Użytkownicy mogą też skompilować z Moonlight z biblioteką ffmpeg zamiast microsoftowych kodeków. Moonlight jest dostępny obecnie na platformy x86 i x86-64. W przyszłości obsługiwanych platform może być więcej. Obecnie Moonlight nie jest jeszcze kompatybilny z Silverlight 2.0. Jednak już w marcu ma zostać wydana alfa wersji wspierającej Silverlight 2.0. Wersja wspierająca najnowszy Silverlight ma używać Mono do wykonywania kodu napisanego w C# oraz innych językach działających na bazie Dynamic Language Runtime, które Microsoft udostępnia jako open source.

http://arstechnica.com/

Google Chrome z każdym nowym komputerem?

Twórcy Google Chrome snują plany i zastanawiają się, co zrobić by produkt ten mógł konkurować z najpopularniejszym obecnie Internet Explorerem. Co ciekawe, koncern z Mountain View planuje w tym celu wykorzystać praktyki, które stosuje sam Microsoft i które przyczyniają się do tak olbrzymiej popularności jego przeglądarki. Jak to się stało, że Internet Explorer jest numerem jeden na rynku przeglądarek internetowych? Otóż zdaniem Sundara Pichai, wiceprezesa do spraw zarządzania produktami w firmie Google, przyczynia się do tego fakt, iż przeglądarka w postaci aplikacji gotowej do użycia trafia do użytkowników nowych komputerów, na których preinstalowany jest system Windows. Dlatego też, zgodnie z ujawnionymi przez Pichaia wywiadzie dla dziennika The Times planami koncernu, wykorzysta on do walki z Microsoftem na rynku przeglądarek internetowych jego własną broń i w momencie oficjalnego wydania Google Chrome rozpocznie jego dystrybucję, rozpowszechniając aplikację z nowymi komputerami. Według zapowiedzi ma to mieć miejsce prawdopodobnie już na początku przyszłego roku. Wtedy również mają być dostępne wersje przeglądarki dla systemów Linux i Mac OS, dzięki czemu będzie ona mogła pracować na 99% komputerów na całym świecie.

http://www.readwriteweb.com/

Microsoft naciska na Google w sprawie retencji danych

Microsoft poinformował Komisję Europejską, że jest w stanie ograniczyć czas przechowywania danych dotyczących wyszukiwań użytkowników korzystających z Windows Live Search do 6 miesięcy. Właśnie pół roku to okres proponowany przez administrację Unii Europejskiej. Po tym czasie dane, według Komisji, powinny być anonimizowane - czyli przetwarzane w taki sposób, by nie zawierały adresów IP użytkowników. Microsoft jednak wstrzymuje się z wdrożeniem nowych zasad do momentu, gdy... na podobne zmiany zgodzi się jego największy konkurent na rynku wyszukiwania, Google. W przeciwnym wypadku Microsoft sam osłabiłby swoje możliwości względem rywala. Warto przypomnieć, iż Google we wrześniu 2008 roku pod naciskiem Unii Europejskiej zmieniło swoją politykę w sprawie czasu retencji danych użytkowników i skróciło okres przechowywania adresów IP do 9 miesięcy. Zalecenie Unii Europejskiej w sprawie czasu przechowywania danych użytkowników pojawiło się w kwietniu 2008 roku. Od tamtej pory żadna firma nie zgłosiła chęci, by się do niego zastosować. Microsoft jest pierwszą, ale póki co deklaruje jedynie gotowość - czy faktycznie czas ten zostanie skrócony okaże się dopiero wtedy, gdy Google odpowie Microsoftowi i zgodzi się na podobny krok.

http://vnunet.com/

Microsoft i Yahoo! tracą, Google zyskuje

Jak wynika z badań przeprowadzonych przez specjalizującą się w badaniu Internetu firmę Nielsen Online, ilość zapytań do wyszukiwarek internetowych zmalała wśród internatów z USA o mniej niż dwa punkty procentowe w skali roku. Jednocześnie widać wzrost zainteresowania usługami Google. Zdaniem korporacji, liczba wyszukiwań w całych Stanach Zjednoczonych zmniejszyła się do 7,78 mld. Ilość zapytań kierowanych do wyszukiwarki giganta z Mountain View wzrosła o 8,1 punktu procentowego do 4,76 mld zapytań miesięcznie. Pozwoliło to firmie uplasować się z ponad 60-procentowym udziałem na rynku wyszukiwarek. Straty odnotowały Yahoo! oraz Microsoft. Usługi pierwszego z nich straciły bowiem dwanaście punktów procentowych w ciągu ostatniego roku do 1,31 mld, z kolei Microsoftu o kolejne siedem - do 89 mln.

http://www.cnet.com/

Archiwa czasopism w Google Book Search

Google Book Search to wyszukiwarka, która umożliwia przeszukiwanie całego tekstu książek w celu znalezienia pozycji interesujących użytkownika oraz uzyskanie informacji, gdzie można je kupić lub wypożyczyć. Projekt ruszył w polskiej wersji językowej pod koniec ubiegłego roku, a teraz poszerzono zakres dostępnych opcji o możliwość wyszukiwania także archiwów czasopism. Zaktualizowana wersja usługi pozwala dotrzeć do starych, często nieaktualnych czasopism, które udostępnione zostały w Internecie. Z dostępnych zarchiwizowanych wersji znaleźć możemy nie tylko starsze pozycje w postaci takich tytułów jak New York Magazine, Popular Mechanics, czy Ebony, ale także nowe publikacje. Materiały tego typu są oznaczane w wynikach wyszukiwania słowem Magazyn, a korporacja czerpie zyski na usłudze poprzez wyświetlane obok artykułów boksów reklamowych, dzieląc się zyskami z ich wydawcami.

http://searchengineland.com/

ODF Toolkit

Podczas konferencji OpenOffice.org 2008 w Pekinie firmy Sun i IBM ogłosiły powstanie ODF Toolkit Union. W jej ramach tworzony będzie ODF Toolkit, zestaw różnych narzędzi dla formatu ODF. Będą to zarówno małe narzędzia ułatwiające używanie ODF w procesie tworzenia oprogramowania jak i duże biblioteki dla Javy i .NET, które mogą być używane w innych projektach. Toolkit nie będzie tworzony od zera, początkowy kod dostarczy firma Sun. Zawiera on API do odczytu, zapisu i manipulowania dokumentami ODF. Jest też ODF Validator służący do walidacji dokumentów oraz sprawdzania innych kryteriów. Gdy ODF Toolkit będzie gotowy, umożliwi m.in. automatyczne tworzenie raportów na podstawie zapytań do baz danych, indeksowanie dokumentów do wyszukiwania, konwersję do innych formatów oraz renderowanie dokumentów. ODF Toolkit będzie dostępny na liberalnej licencji Apache.

http://dobreprogramy.pl

Europeana: europejskie dziedzictwo w Internecie

Pod koniec listopada 2008 roku ruszył nowy, spektakularny projekt: Europeana, czyli europejska multimedialna biblioteka cyfrowa. Na stronie internetowej europeana.eu Internauci z całego świata mają dostęp do ponad dwóch milionów książek, map, nagrań, fotografii, dokumentów archiwalnych, obrazów i filmów z bibliotek narodowych i instytucji kulturalnych z dwudziestu siedmiu państw członkowskich UE. Europeana ma otworzyć nową drogę do korzystania z europejskiego dziedzictwa: każdy, kto interesuje się literaturą, sztuką, nauką, polityką, historią, architekturą, muzyką lub filmem, będzie miał swobodny i szybki dostęp do największych zbiorów i arcydzieł europejskich w ramach jednej wirtualnej biblioteki, dzięki portalowi internetowemu dostępnemu we wszystkich językach Unii Europejskiej. Ale jest to dopiero początek. Viviane Reding, komisarz UE ds. społeczeństwa informacyjnego i mediów powiedziała, że celem jest, aby w roku 2010 Europeana pomieściła co najmniej 10 milionów obiektów pochodzących z wszystkich krajów Unii. Europeana została zainicjowana przez Komisję w 2005 r. i opracowana szczegółowo w ramach ścisłej współpracy z bibliotekami narodowymi i innymi instytucjami kulturalnymi państw członkowskich, jak również przy szerokim poparciu ze strony Parlamentu Europejskiego. Jest ona zarządzana przez Fundację Europejskiej Biblioteki Cyfrowej, która łączy najważniejsze europejskie stowarzyszenia bibliotek, archiwów, muzeów, archiwów audiowizualnych i instytucji kulturalnych. Europeana ma swoją siedzibę w niderlandzkiej bibliotece narodowej, Koninklijke Bibliotheek. Materiały dla Europeany pochodzą z ponad tysiąca organizacji kulturalnych z całej Europy. Europejskie muzea, w tym paryski Luwr i amsterdamskie Rijksmuseum, dostarczyły cyfrowe zapisy obrazów i obiektów ze swoich kolekcji. Archiwa państwowe udostępniły ważne doku

Windows Mobile 6.5 w drugiej połowie 2009 roku

Podczas kwartalnej konferencji Motoroli, dyrektor wykonawczy Sanjay Jha poinformował, że firma ta planuje wyposażyć swoje telefony w wersję 6.5 systemu Windows Mobile w drugiej połowie roku 2009. Również Steve Ballmer podczas konferencji w Melbourne powiedział, że Windows Mobile 6.5 ma się pojawić mniej więcej w tym czasie. Później jednak rzecznik Microsoftu powiedział, że słowa Ballmera nie należy traktować jako oficjalną mapę drogową, a wszelkie wcześniejsze zapowiedzi należy traktować ogólnie. Być może Windows Mobile 6.5 zostanie wydany razem z Windows 7. Motorola zapowiada też, że razem z telefonami z WM 6.5 wyda urządzenia korzystające z platformy Android opracowanej przez Google. Natomiast Windows Mobile 7 pojawi się najwcześniej w 2010 roku. Spekuluje się, że Windows Mobile 7 będzie bardziej nastawiony na muzykę. Konkurowałby więc m.in. z obecnie sprzedawanymi przez Microsoft odtwarzaczami Zune. Ballmer mówiąc o rynku telefonów komórkowych powiedział też, że obecnie rocznie sprzedaje się 1,2 miliarda tych urządzeń, z czego 100 milionów to telefony typu smartphone. Na 20 milionach z nich działał Windows Mobile, co było lepszym wynikiem sprzedażowym w porównaniu do konkurencyjnych rozwiązań takich firm jak Apple czy BlackBerry.

http://betanews.com/

Google łata „najgłupszy błąd świata"

Google załatało błąd w platformie Android, dostępnej w telefonach G1, który nie bez przyczyny, uznano za jeden z najgłupszych w historii programowania. Problem polegał na tym, że słowa wpisywane na telefonie, np. podczas pisania SMSa były traktowane jako polecenia i wykonywane przez powłokę z uprawnieniami roota. Jeśli np. użytkownik w treści SMSa użył słowa "reboot", wówczas następowało ponowne uruchomienie telefonu. Google naprawiło błąd w aktualizacji RC30.

http://www.cnet.com/

10 tysięcy aplikacji w Apple App Store

Firma Apple poinformowała, że w jej sklepie z oprogramowaniem App Store użytkownicy mogą pobrać już 10 000 aplikacji. Apple App Store został otwarty w lipcu 2008 roku. Znajdowało się w nim wtedy tylko 500 programów; we wrześniu 2008 roku było ich już 3000. Wystarczyło niecałe pół roku aby początkowa liczba wzrosła dwudziestokrotnie. W tym czasie użytkownicy pobrali oprogramowanie 300 milionów razy. Nie wiadomo jednak ile z tych ściągnięć dotyczyło aplikacji płatnych, a ile darmowych. Wiadomo, że 11 października 2008 roku, gdy w sklepie były 4093 programy, wśród nich znajdowało się 974 darmowych i 3119 komercyjnych. Sklep, jaki stworzyła firma Apple, stał się jednym z jej większych sukcesów. Przede wszystkim umożliwił użytkownikom iPhone i iPod Touch wygodne ściąganie aplikacji z jednego miejsca.

http://www.apple.com/

Thumbtack do zarządzania informacjami

Microsoft Live Labs, ten sam zespół, który rozwija między innymi aplikację Photosynth ma teraz nową propozycję dla tych, którzy szukają łatwej i efektywnej metody przechowywania i analizowania różnego typu informacji. Thumbtack, bo tak nazywa się nowa usługa Live Labs, to miejsce w Internecie dostępne po zalogowaniu przy pomocy Windows Live ID, które umożliwia przechowywanie różnego rodzaju informacji - na przykład wycinków stron internetowych, obrazów, własnych notatek, fragmentów dokumentów. Wszystkie te materiały można przypisywać do kolekcji, które następnie mogą być udostępniane innym. Ideę działania usługi szybko zrozumieją zwłaszcza ci, którzy korzystają z Office OneNote - tam wstawianie notatek i organizacja w zakładki jest bardzo podobna, ale Thumbtack ma tę przewagę, że jest dostępny on-line z każdego komputera. Na stronie domowej usługi dostępny jest film ilustrujący działanie usługi oraz przykładowe kolekcje. Thumbtack może wypróbować każdy - podobnie jak każde inne narzędzie udostępniane przez Microsoft Live Labs, również i to nie ma statusu finalnej wersji i cały czas jest unowocześniane, może również nie działać w pełni prawidłowo. Doświadczenia z innymi aplikacjami Live Labs wskazują jednak na to, że może to być całkiem ciekawa i przydatna usługa.

http://www.microsoft.com/poland

Komputer z HPC Server 2008 w dziesiątce najszybszych

Maszyna pracująca pod kontrolą systemu operacyjnego Microsoft Windows HPC Server 2008 znalazła się na 10 miejscu listy Top500 zawierającej 500 najszybszych superkomputerów na świecie. Jeszcze rok temu najszybszy komputer działający pod kontrolą systemu Microsoftu znajdował się na 116 miejscu. Superkomputer firmy Dawning Information Industry osiągnął prędkość 180,6 teraflopów przy wydajności wynoszącej 77,5%. Strategia Microsoftu to ułatwienie korzystania z superkomputerów poprzez zmniejszanie kosztów i złożoności, a także dostarczanie zbioru aplikacji, narzędzi zarządzających, narzędzi programistycznych oraz budowanie społeczności niezależnych dostawców oprogramowania. Korporacja chce, aby komputery o dużej mocy obliczeniowej mogły być używane także w codziennych zastosowaniach. Przykładem może być zwykły komputer desktopowy z zainstalowanym Excelem, który wywołuje wykonywanie złożonych obliczeń na klastrze i zwraca dane wyjściowe użytkownikowi. Użytkownik ten nie musi nawet wiedzieć, że przy wykonywanych przez niego operacjach wykorzystywany jest superkomputer. Microsoft obecnie kontroluje niecałe 5% rynku superkomputerów. 74% należy do Linuksa a 21% do różnej maści Uniksów. W celu popularyzacji swojego systemu Microsoft dostarcza akademicką wersję Windows HPC Server 2008 w cenie 15 dolarów za jeden węzeł w klastrze. Komercyjna wersja kosztuje 450 za instalację na jednym węźle klastra.

http://www.techit.pl/

Microsoft Robotics Developer Studio 2008

Podczas RoboDevelopment Conference & Expo w Santa Clara Microsoft ogłosił udostępnienie finalnej wersji Robotics Developer Studio 2008, platformy umożliwiającej tworzenie oprogramowania dla robotów, zarówno prawdziwych jak i symulowanych. W tym wydaniu skupiono się na zwiększeniu wydajności. Ulepszono również narzędzia VPL (Visual Programming Language) oraz VSE (Visual Simulation Environment), zwiększono elastyczność wdrażania przez obsługę Microsoft Visual Studio 2005 jak i Visual Studio 2008, poprawiono uruchamianie VPL i VSE na 64-bitowych Windows oraz dodano obsługę własnych mechanizmów przesyłania wiadomości. Microsoft RDS zawiera prosty model programowy wspierający tworzenie asynchronicznych aplikacji, narzędzia do tworzenia i symulacji aplikacji przez niezaawansowanych użytkowników, a także samouczki i przykładowe kody źródłowe. RDS 2008 jest już trzecią wersją pakietu programistycznego dla robotów. Poprzednie wersje doczekały się 250000 pobrań, a ponad 60 firm tworzących zarówno sprzęt jak i oprogramowanie wspiera lub używa Microsoft Robotics Developer Studio w swoich produktach. Zainteresowane osoby mogą za darmo pobrać wersję Express lub zakupić wersję Standard.

http://www.eweek.com/

Windows obchodzi 23 urodziny

20 listopada 1985, nieco ponad 23 lata temu Microsoft wydał pierwszy system operacyjny z rodziny Windows, oznaczony jako 1.0. Pomimo, że była to tak naprawdę nakładka na popularnego wówczas DOS'a to wniosła ona wiele nowatorskich rozwiązań w świecie oprogramowania komputerowego. W stosunku do poprzednich, tekstowych systemów Microsoftu 16-bitowy Windows 1.0 zawierał wiele nowych rozwiązań, który czyniły go użyteczniejszym i wygodniejszym w codziennej pracy. Pojawiła się w nim obsługa myszki oraz wątków, które umożliwiały pracę z kilkoma aplikacjami jednocześnie. Interfejs graficzny opierał się o paletę 256 kolorów, zaimplementowano w nim dobrze znane dzisiaj okna, które mogły być minimalizowane, maksymalizowane i wyświetlane na ekranie w tzw. układzie taflowym. Po raz pierwszy w ręce użytkowników trafiło także coś na wzór Panelu Sterowania i paska zadań. Co ciekawe, koncepcja okien pojawiła się w Microsofcie już wiosną 1983 roku, ale dopiero dwa lata później na rynku pojawił się Windows 1.0, który bazował na wspomnianym rozwiązaniu. Pracowało nad nim 55 programistów, którym proces tworzenia tego systemu zajął około roku.

http://neowin.net/

OpenCV

komputerowa wizja w praktyce

Czy komputer może widzieć podobnie jak człowiek? Czy komputerowa wizja to mit czy rzeczywistość? Na te, oraz inne pytania można znaleźć odpowiedz, czytając poniższy artykułu traktujący o wyjątkowej bibliotece wywodzącej się z laboratoriów firmy Intel.

Dowiesz się:

- Jakie możliwości oferuje biblioteka OpenCV;
- Jak przy pomocy tej biblioteki wykonywać podstawowe operacje na obrazach.

Powinieneś wiedzieć:

- Podstawy programowania w języku C/C++.

Poziom trudności

OpenCV to otwarta biblioteka stworzona w językach C oraz C++, oferująca cały szereg udogodnień związanych z szeroko pojętą dziedziną wizji komputerowej – marketingowy slogan promujący Open CV głosi, iż biblioteka ta pozwala tworzyć *oprogramowanie, które widzi*. Czytelników niezaznajomionych z tematyką zapraszam do ramki zatytułowanej *Wizja komputerowa w pigułce*). OpenCV została zaprojektowana przede wszystkim z uwzględnieniem efektywności – dlatego też kluczowe komponenty biblioteki zakodowano w czystym C; autorzy uwzględnili również możliwość wykorzystania potencjału maszyn wielordzeniowych. Podstawowym celem stworzenia OpenCV było dostarczenie prostej, a jednocześnie potężnej infrastruktury, na bazie której programiści mogliby relatywnie szybko budować zaawansowane rozwiązania z dziedziny wizji komputerowej. W rezultacie OpenCV oferuje ponad 500 funkcji, które można zastosować przy budowaniu rozwiązań z zakresu inspekcji produkcji masowej, wizualizacji medycznej, bezpieczeństwa, kalibracji kamer, stereowizji czy robotyki. OpenCV oferuje również łakomy kąsek dla praktyków z zakresu dziedziny uczenia maszynowego w postaci dedykowanej, zagnieżdżonej biblioteki: MLL (ang. *Machine Learning Library*), która oferuj przede wszystkim funkcjonalność z zakresu statystycznego dopasowywania wzorców (ang. *statistical pattern recognition*) oraz klasteryzacji (ang. *clustering*).

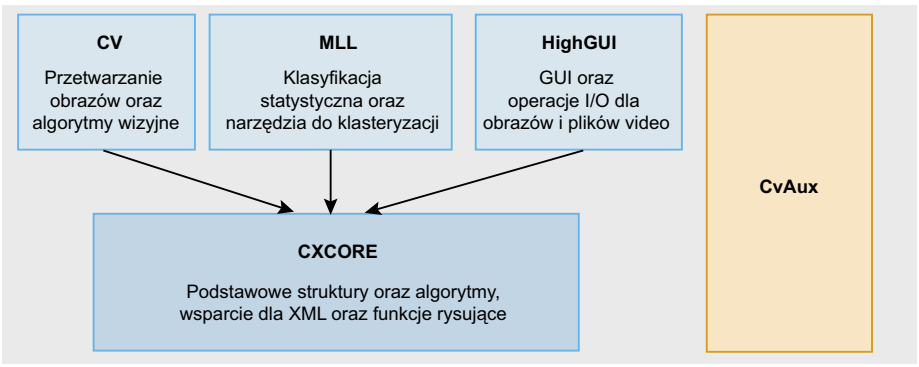
Plan i cel artykułu

OpenCV stanowi jedną z tych bibliotek, o których powinno się raczej pisać książki, a nie artykuły – aczkolwiek, autor mimo wszystko podejmie próbę przedstawienia tego użytecznego rozwiązania. Z góry należy zaznaczyć, iż celem niniejszego artykułu nie jest opisanie pełnej funkcjonalności biblioteki, a raczej przekrojowe przedstawienie jej możliwości i zachęcenie Czytelnika do dalszych eksperymentów.

Struktura artykułu jest następująca: na początku przedstawiony będzie krótki rys historyczny opisujący genezę OpenCV. W dalszej części Czytelnicy będą mieli okazję zapoznać się ze strukturą biblioteki. Kolejne podpunkty opisywać będą bardziej szczegółowo podstawowe moduły OpenCV; tutaj zaprezentowane będą również proste przykłady zastosowania biblioteki w postaci krótkich programów napisanych w języku C++. Na końcu zapoznamy się przekrojowo z zaawansowanymi możliwościami biblioteki. Oprócz tego Czytelnicy znajdą w tekście szereg ramek zawierających dodatkowe, ciekawe informacje związane pośrednio lub bezpośrednio z OpenCV oraz rozdział podsumowujący.

Rys historyczny

Korzenie biblioteki OpenCV wiążą się z działem rozwojowym firmy Intel (Intel Research). W dziale tym prowadzono wiele badań dotyczących aplikacji wymagających intensywnych obliczeń – między innymi związanych z grafiką komputerową. Pracownicy Intel Research odwiedzali w tym czasie wiele ośrodków naukowych na całym świecie i w trakcie tych wizyt poczynili wiele interesujących obserwacji. Jedną z nich było spostrzeżenie, iż wiele grup uniwersyteckich buduje biblioteki wspierające badania z dziedziny wizji komputerowej. W większości przypadków były to nie do końca usystematyzowane rozwiązania typu *ad-hoc*, przekazywane z rąk do rąk po to, aby za każdym razem nie trzeba było implementować podstawowych udogodnień od początku. W kontekście tego spostrzeżenia, pracownicy Intel Research wpadli na pomysł stworzenia OpenCV – otwartej i usystematyzowanej biblioteki oferującej bazową infrastrukturę do przeprowadzania badań związanych z tematyką wizji komputerowej. Stworzenie pierwszej implementacji zlecono rosyjskiemu zespołowi zajmującemu się tworzeniem rozwiązań bibliotecznych Intela. W tej grupie właśnie, pod przewodnictwem Vadim Pisarevskiego, powstała pierwsza wersja OpenCV. Twórcom biblioteki przyświecały trzy cele: wysoka wydajność, czytelność kodu oraz otwartość – włącznie z możliwością wykorzystania rozwiązania w komercyjnych produktach – bez konieczności ich publikowania. Wszystko to działo się w 1999 roku. Do 2007 roku trzeba było czekać na oficjalną, finalną wersję bi-



Rysunek 1. *Struktura biblioteki OpenCV*

blioteki oznaczoną numerem 1.0. Na dzień dzisiejszy szacuje się, że biblioteka została pobrana ponad 2 miliony razy, zaś liczba jej użytkowników stale rośnie. OpenCV jest nadal intensywnie rozwijana – jednakże już poza Intelem; projekt ma aktualnie charakter społecznościowy. I co najważniejsze – biblioteka tak nadal pozostaje rozwiązaniem otwartym i całkowicie darmowym.

Struktura biblioteki

Patrząc przekrojowo, OpenCV składa się z pięciu podstawowych komponentów. Strukturę biblioteki przedstawiono na Rysunku 1.

Jądro biblioteki stanowi komponent CXCORE – tutaj zaimplementowany jest podstawowy szkielet przetwarzania oparty na podstawowych strukturach danych i algorytmach z dziedziny wizji komputerowej, a także na pewnych udogodnieniach podstawowych (wsparcie dla XML i funkcje rysujące). Kolejne trzy komponenty to wyspecjalizowane rozszerzenia biblioteki. CV oferuje zaawansowane algorytmy z dziedziny przetwarzania obrazu oraz wizji, MML – mechanizmy wsparcia dla uczenia maszynowego, HighGUI – elementy graficznego interfejsu użytkownika oraz operacje wejścia/wyjścia dla obrazów i plików video. Ostatni komponent OpenCV – CvAux, istnieje niejako w oderwaniu od całości biblioteki i stanowi pole doświadczalne tudzież inkubator pomysłów. Można tu znaleźć między innymi implementację algorytmów rozpoznawania twarzy, gestów, śledzenia ruchów ust i oczu, czy klasy użytkowe do kalibracji kamer. Komponenty te jednak są słabo udokumentowane i należy traktować je z pewną dozą ostrożności, dopóki nie zostaną przeniesione do jednego z oficjalnych komponentów biblioteki.

Wyświetlanie obrazu

W niniejszym punkcie rozważymy pierwszy, prosty program napisany przy pomocy OpenCV – wczytanie obrazu. Program ten, razem z jego następnikiem stanowić będą bazę do prezentacji kolejnych – bardziej zaawansowanych przykładów. Idea działania programu jest bardzo prosta – będzie on wczytywał i wyświetlał obraz o ścieżce przekazanej jako argument linii poleceń. Po wciśnięciu dowolnego klawisza program zakończy swoje działanie. Kod programu przedstawiony jest na Listingu 1.

Przeanalizujmy po kolei poszczególne fragmenty tego programu. Już na samym początku w oczy rzuca się niestandardowy nagłówek `<highgui.h>`. Nagłówek ten stanowi część pakietu OpenCV. Uważnie Czytelnicy pamiętają zapewne, że to właśnie moduł OpenGUI oferuje elementy graficznego interfejsu użytkownika oraz operacje wejścia/wyjścia dla obrazów i plików video. Tak się składa, że właśnie takiej funkcjonalności potrzebuje nasz pierwszy program – dlatego w pierwszym przykładowym programie umieściłem taki, a nie inny nagłówek. Przed funkcją `main()` zdefiniowana jest globalna stała opisująca nazwę okna (`g_windowName`). W ciele `main()`, korzystając z mechanizmu asercji sprawdzamy dla porządku dostępność argumentu po czym przechodzimy do realizacji naszego zadania (oczywiście w produkcyjnym kodzie warto by pokusić się o nieco bardziej wyrafinowany mechanizm obsługi niepoprawnej składni linii poleceń). W linii:

```
IplImage* image = cvLoadImage( argv[ 1 ] );
```

wczytujemy obraz. Funkcja `cvLoadImage()`, przyjmująca jako argument ścieżkę docelowego pliku, potrafi rozpoznać z jakim formatem ma do czynienia. Formaty wspierane przez wspomnianą funkcję to: BMP, DIB, JPEG, JPE, PNG, PBM,

Listing 1. *Wyświetlanie obrazu przy pomocy OpenCV*

```
#include <highgui.h>
#include <cassert>
const char* g_windowName = "ImageView";
int main( int argc, char** argv )
{
    assert( argc == 2 );
    IplImage* image = cvLoadImage(
                        argv[ 1 ] );
 cvNamedWindow( g_windowName, CV_WINDOW_
                        AUTOSIZE );
    cvShowImage( g_windowName, image );
    cvWaitKey( 0 );
    cvReleaseImage( &image );
    cvDestroyWindow( g_windowName );
}
```

Szybki start

W niniejszej ramce przedstawiony zostanie proces przygotowania do pracy z biblioteką OpenCV na platformie Windows przy wykorzystaniu środowiska Microsoft Visual C++ 2005. W tym celu należy wykonać następujące kroki:

- pobrać pakiet instalacyjny biblioteki pod adresem *http://sourceforge.net/project/showfiles.php?group_id=22870&package_id=16937*,
- zainstalować pobrany pakiet w wybranym katalogu (w dalszej części tej instrukcji będziemy odnosić się do tego katalogu jako CV_HOME),
- uruchomić środowisko Microsoft Visual C++ 2005 i stworzyć nowy, pusty projekt typu *Win32 console application*,
- we właściwościach projektu (*C/C++ –> General –> Additional Include Directories*) ustawić katalogi nagłówkowe biblioteki OpenCV: *CV_HOME\cv\include; CV_HOME\cxcore\include; CV_HOME\ml\include; CV_HOME\otherlibs\highgui*,
- we właściwościach projektu (*Linker –> General –> Additional Library Directories*) ustawić katalog zawierający statyczne biblioteki OpenCV: *CV_HOME\OpenCV\lib*,
- we właściwościach projektu (*Linker –> Input –> Additional Dependencies*) ustawić statyczne biblioteki OpenCV: *highgui.lib cxcore.lib ml.lib cv.lib*,
- dodać do projektu plik źródłowy i wpisać do niego zawartość Listingu 1,
- zbudować projekt,
- skopiować dynamiczne biblioteki OpenCV z katalogu *CV_HOME\bin* do miejsca z którego będą widoczne dla aplikacji (np. *C:/windows/system32*),
- uruchomić zbudowany program podając mu jako parametr linii poleceń ścieżkę do wybranego obrazu.

Wynikiem powyższego postępowania powinien być wyświetlony obraz – podobnie jak widać to na Rysunku 1 (oczywiście zawartość obrazu będzie inna – w zależności od tego jaką podano ścieżkę).

Licencja i użytkownicy

Licencja otwartej biblioteki OpenCV skonstruowana jest w taki sposób, że dany podmiot może darmowo wykorzystywać dowolne jej części w komercyjnych rozwiązaniach nie będąc przy tym obligowanym do udostępniania swojego produktu na zasadach wolnego oprogramowania. Jednakże autorzy biblioteki mają nadzieję, że osoby bądź firmy korzystające i rozszerzające OpenCV będą się dzielić rezultatami swoich prac ze społecznością. W dużej części dzięki stosowaniu takiego liberalnego podejścia OpenCV zyskała wielką rzeszę użytkowników – wliczając w to takie duże korporacja jak IBM, Microsoft , Intel, SONY, Siemens, Google i inne. Wiele ośrodków akademickich używa oraz bierze aktywny udział w rozwoju OpenCV (są to między innymi takie placówki jak Stanford, MIT, CMU, Cambridge, czy INRIA).

OpenCV: przenośność

OpenCV od samego początku była projektowana jako biblioteka przenośna. W tym celu pierwotna wersja biblioteki była kompilowana równolegle narzędziami firmy Borland, Microsoft oraz Intel. Zaowocował to powstawaniem wysoce przenośnego kodu C/C++. W rezultacie OpenCV udało się uruchomić platformach Windows, Linux, MacOS, BSD, Solaris i innych. Na dzień dzisiejszy najmocniejszym wsparciem ze strony OpenCV szczycić się może platforma Windows oparta na 32-bitowej architekturze Intela. Zaraz po niej można wymienić system Linux. Wersja dla MacOS jest nieco opóźniona w stosunku do swoich konkurentów, gdyż jej rozwój nabrał tempa dopiero od momentu gdy Apple zdecydował się na korzystanie z procesorów Intela. Pisząc o przenośności biblioteki warto wspomnieć również o alternatywnych interfejsach programistycznych: w języku Python oraz Ruby. Istnieje również możliwość zintegrowania OpenCV z pakietem Mathlab.

PGM, PPM, SR, RAS, oraz TIFF. `cvLoadImage()` zwraca wskaźnik do struktury `IplImage` reprezentującej obraz, który zachowujemy sobie na przyszły użytek. W kolejnej linii:

```
cvNamedWindow( g_windowName, CV_WINDOW_
                        AUTOSIZE );
```

program tworzy tak zwane *nazwane okno*, które posłuży nam do tego aby wyświetlić wczytany obraz. Warto zauważyć, że użyta tu funkcja `cvNamedWindow()`, nie zwraca żadnego wskaźnika. Wynika to z faktu, że komponent HughGUI zapamiętuje nazwę przekazaną do tej funkcji (w naszym przypadku: `g_windowName`) dzięki czemu będziemy mogli ją wykorzystać przy wywoływaniu kolejnych operacji związanych z obsługą okien. Kilka słów komentarza należy drugiemu argumentowi przekazanemu do funkcji `cvNamedWindow()`. W ramach tego argumentu można przekazać zestaw flag konfiguracyjnych. Użyta w naszym przypadku flaga `CV_WINDOW_AUTOSIZE`, zgodnie ze swoją nazwą pozwala aby okno automatycznie dopasowało się do wielkości wyświetlanego obrazka. Kolejnej linia w naszym programie realizuje właśnie za to zadnie:

```
cvShowImage( g_windowName, image );
```

Funkcja `cvShowImage()` jako pierwszy parametr przyjmuje nazwę okna; drugi parametr to wskaźnik do obrazka, który chcemy wyświetlić. I na tym w zasadzie kończy się praca naszego programu – pozostała część kodu odpowiada za sprzątanie. Najpierw program czeka na wciśnięcie klawisza, później zwalnia obraz i niszczy okno. Voila! Rezultat działania programu można obejrzeć sobie na Rysunku 2. Po analizie tego krótkiego programu można wyrobić sobie pierwsze wrażenie odnośnie biblioteki; to co mocno rzuca się w oczy to bardzo wysoki poziom abstrakcji oferowanych udogodnień. Pokrywa się to z filozofią biblioteki, którą można by podsumować stwierdzeniem: proste rzeczy powinny być proste. W kolejnym przykładzie pójdziemy o krok dalej i wyświetlimy sekwencję wideo.

Wyświetlanie sekwencji wideo

Odgrywanie sekwencji wideo przy pomocy OpenCV jest niemalże tak samo proste jak wyświetlanie pojedynczego obrazu. Czytelników, którzy chcieliby dowiedzieć się jak można to zrobić, zapraszam do analizy zawartości Listingu nr 2. Z niektórymi elementami programu poznaliśmy się w poprzednim przykładzie. Nowością jest struktura `CvCapture`; służy ona do reprezentacji strumienia wideo. W naszym przypadku tworzymy strumień na podstawie pliku:

```
CvCapture* capture = cvCreateFileCapture(
                        argv[ 1 ] );
```

Szkielet programu bazuje na nieskończonej pętli, w której pobieramy kolejne klatki ze strumienia:

```
IplImage* frame = cvQueryFrame( capture );
```

po czym wyświetlamy je w zadanym oknie:

```
cvShowImage( g_windowName, frame );
```

W pętli sprawdzamy również czy sekwencja się nie zakończyła, tudzież czy użytkownik nie przerwał działania programu wciskając klawisz [*ESC*]. To ostatnie sprawdzamy co 33 milisekundy, przekazując do funkcji cvWaitKey() odpowiednią wartość. Funkcja ta działa w ten sposób, że po jej wywołaniu oczekuje przez określoną liczbę milisekund na zdarzenie klawiatury i jeśli takowe się pojawi to zwraca ona kod wciśniętego klawisza; w innym przypadku z funkcji zwracana jest wartość *-1* i sterowanie wraca ponownie do głównej pętli. Spostrzegawczy Czytelnik zauważy zapewne, że w poprzednim przykładzie wspomniana tu funkcja wywołana została z parametrem *0*. W takim przypadku (oraz wtedy gdy przekazany parametr jest ujemy), funkcja czeka bezterminowo na reakcję użytkownika blokując program.

Analizując program wyświetlający sekwencję wideo można poczynić kolejne ważne spostrzeżenie odnośnie OpenCV: widać w tym przypadku wyraźnie, że biblioteka jest zaprojektowana na potrzeby zastosowań związanych z przetwarzaniem obrazów. Objawia się to chociażby faktem, iż przy odtwarzaniu sekwencji wideo programista w wygodny sposób może dostać się do poszczególnych klatek sekwencji.

Mam nadzieję, że przedstawione powyżej, proste przykłady pozwolą Czytelnikom poczuć klimat biblioteki OpenCV i stanowić będą przyjazne wprowadzenie w nasze dalsze rozważania. W kolejnych podpunktach zaprezentuję możliwości OpenCV w kontekście jej najbardziej popularnych zastosowań.

HighGUI oraz udogodnienia podstawowe

To z czym Czytelnik zapoznał się w dwóch pierwszych przykładach, to jedynie wycinek możliwości komponentu HighGUI. Moduł ten oferuje zestaw

Rysunek 2. *Rezultat działania programu opisanego na Listingu 1*

Listing 2. *Odgrywanie sekwencji wideo przy pomocy OpenCV*

```
#include "highgui.h"
const char* g_windowName = "PlayVideo";
const char* g_keyCode_ESC = 27;
const int   g_delay = 33;
int main( int argc, char** argv )
{
    cvNamedWindow( g_windowName, CV_
                        WINDOW_AUTOSIZE
                        );
    CvCapture* capture =
      cvCreateFileCapture( argv[ 1 ] );
    while ( true )
    {
        IplImage* frame = cvQueryFrame(
                        capture );
        if ( !frame )
        {
            break;
        }
        cvShowImage( g_windowName,
                        frame );
        char c = cvWaitKey( g_delay );
        if( c == g_keyCode_ESC )
        {
            break;
        }
    }
    cvReleaseCapture( &capture );
    cvDestroyWindow( g_windowName );
}
```

funkcji, które odpowiadają za interakcję z system operacyjnym, z systemem plików oraz ze sprzętem (w tym ostatnim przypadku chodzi przed wszystkim o wszelkiego rodzaju kamery). Jak było wcześniej pokazane, HighGUI pozwala otwierać okna, czytać i zapisywać różnej maści formaty graficzne (zarówno obrazy jak i sekwencje wideo) i wyświetlać ich zawartość na ekranie. Dodatkowo oferuje wsparcie dla obsługi podstawowych kontrolerów (mysz, klawiatura) oraz użyteczne kontrolki (np. suwak). Patrząc z punktu widzenia programisty korzystającego z HighGUI, w ramach tego komponent można wydzielić trzy niezależne części: obsługa sprzętu, interakcja z systemem plików oraz graficzny interfejs użytkownika.

Mechanizmy obsługi sprzętu skupiają się przede wszystkim na obsłudze kamer. Jest to bardzo użyteczne narzędzie, jako że obsługa natywna kamery jest w przypadku większości systemów operacyjnych trudnym i uciążliwym zadaniem. HighGUI pozwala w łatwy sposób pobierać dane z kamery ukrywając jednocześnie zbędne szczegóły implementacji.

Część odpowiedzialna za interakcję z systemem plików oferuje głównie funkcjonalność związaną z czytaniem i zapisywaniem wszelkiej maści obrazów. Bardzo ciekawy aspekt biblioteki związany jest z faktem, iż autorzy zunifikowali mechanizm odczytu danych ze strumienia wideo oraz z kamery, dzięki czemu na bazie OpenCV można w łatwy sposób pisać programy niezależne od źródła danych. Bardzo miłe dla użytkownika jest to, że nie trzeba się martwić o formaty danych; funkcje czytające potrafią same rozpoznać z jakim formatem mają do czynienia.

Ostatni, trzecia cześć OpenGUI to zestaw graficznych kontrolek oraz mechanizmy obsługi interakcji z użytkownikiem. Możliwości biblioteki w tym zakresie są dość ograniczone (np. OpenCV nie oferuje bezpośredniego wsparcia dla przycisków), aczkolwiek dostępne mechanizmy są zoptymalizowane pod kątem zastosowań specyficznych dla wizji komputerowej i praktyka pokazuje, że za ich pomocą można zaprogramować całe spektrum użytecznych interfejsów użytkownia – co ważne, w sposób niezależny od platformy.

Oprócz HighGUI, w ramach OpenCV otrzymujemy cały szereg dodatkowych mechanizmów wspierających, które można by zebrać razem i umieścić pod wspólnym szyldem: udogodnienia podstawowe. W ramach tychże udogodnień otrzymujemy podstawowe typy danych (`CvPoint`, `CvPoint2D32f`, `CvPoint3D32f`, `CvSize`, `CvRect`, `CvScalar`), a także struktury danych do reprezentacji macierzy i obrazów oraz szereg operacji pozwalających manipulować tymi strukturami. Przykłady wykorzystania niektórych ze wspomnianych tutaj udogodnień pokazane będą w kolejnych punktach.

Przejdziemy teraz do opisu udogodnień oferowanych przez OpenCV a odnoszących się bezpośrednio do tematu wizji komputerowej.

Przetwarzanie obrazów

Znając zgrubnie wszystkie podstawowe udogodnienia OpenCV, możemy przejść do opisu bardziej wysokopoziomowych operacji, związanych z wizją komputerową. W niniejszym podpunkcie zajmiemy się tematyką przetwarzania obrazów. Zagadnienie to pełni bardzo istotną rolę w dziedzinie wizji komputerowej – można powiedzieć, że jest to jedno z podstawowych narzędzi. Obrazy przetwarza się zazwyczaj w celu wyekstrahowania pewnych szczególnych cech, potrzebnych do przeprowadzania zadanej analizy. Jak referencyjny przykład tego rodzaju zastosowania zbadamy w jaki sposób można przy pomocy OpenCV wykonać operację wygładzania obrazu (ang. *smoothing*). Na początek spójrzmy na Listing 3.

Lwia część zawartości Listingu 3 wygląda znajomo. Nowym elementami są wywołania funkcji `cvCreateImage()` oraz `cvSmooth()`. Pierwsza

Rysunek 3. *Rezultat działania programu opisanego na Listingu 3*

Rysunek 4. *Rezultat działania programu Rorozoom: obraz oryginalny*

Rysunek 5. *Rezultat działania programu Rorozoom: obraz zmodyfikowany; parametry [angle=55.0; zoom=0.7]*

Listing 3. *Wygładzanie obrazu przy pomocy OpenCV*

```
#include <cv.h>
#include <highgui.h>
#include <cassert>
const char* g_inWindowName = "In";
const char* g_outWindowName = "Out";
int main( int argc, char** argv )
{
    assert( argc == 2 );
    IplImage* in = cvLoadImage( argv[
                        1 ] );
    cvNamedWindow( g_inWindowName, CV_
                        WINDOW_AUTOSIZE
                        );
    cvNamedWindow( g_outWindowName, CV_
                        WINDOW_AUTOSIZE
                        );
    cvShowImage( g_inWindowName, in );
    IplImage* out = cvCreateImage(
        cvGetSize( in ),
        IPL_DEPTH_8U,
        3
    );
    cvSmooth( in, out, CV_GAUSSIAN, 11,
                        11 );
    cvShowImage( g_outWindowName, out
                        );
    cvReleaseImage( &out );
    cvWaitKey( 0 );
    cvDestroyWindow( g_inWindowName );
    cvDestroyWindow( g_outWindowName );
}
```

z wymienionych funkcji służy do tworzenia nowej instancji obrazu w pamięci – w naszym przypadku wykorzystamy go przy wywołaniu funkcji `cvSmooth()`, która zapisze w nim wyniki swojego działania. Wywołanie `cvSmooth()` jest esencją tego przykładu – pod interfejsem tej funkcji kryje się implementacja różnej maści algorytmów wygładzania. W naszym przypadku użyliśmy metody Gaussa, z wykorzystaniem maski o rozmiarze 11x11 pikseli. Rezultat działania omawianego programu pokazany jest na Rysunku 3: w pierwszym oknie (oznaczonym jako *In*) wyświetlony jest obraz oryginalny, na drugim (*Out*) – obraz wynikowy.

Pokazany tutaj przykład jest jedną z wielu możliwych wariacji wykorzystania funkcji `cvSmooth()`. Czytelników zainteresowanych szczegółami odsyłam do dokumentacji biblioteki. Oprócz operacji wygładzania OpenCV oferuje szereg użytecznych funkcji odpowiedzialnych za różnego rodzaju przetwarzanie obrazów. Każdy z tych algorytmów wart jest oddzielnego opisu – zarówno w kontekście implementacji jak i znaczenia praktycznego; niestety, tematyka ta wykracza poza ramy niniejszego artykułu. W tym miejscu ograniczymy się jedynie do wymieniania nazw wspomnianych algorytmów i przedstawiania kilku przykładów ich działania:

- transformacja morfologiczna: dylatacja
- transformacja morfologiczna: erozja
- transformacje *Top Hat* oraz *Black Hat*
- wypełnianie (ang. *flood fill*)
- zmiana wielkości
- piramidy obrazów
- próg (ang. *threshold*)
- próg adaptacyjny (ang. *adaptive threshold*)

Transformacja obrazów

W poprzednim punkcie mówiliśmy o dość ogólnych procesach przetwarzania obrazów. W tym miejscu skupimy się na procesach transformacji, gdzie na wyjściu może powstać obraz wizualnie całkiem niepodobny do zadanego wejścia. Dobrym przykładem może być w tym przypadku transformata Fouriera. Inne rodzaje transformacje mogą z kolei całkowicie modyfikować geometrię wejściowego obrazu – tak właśnie dzieje się w przypadku transformacji afinicznej. Tę właśnie transformację zbadamy dokładniej w ramach kolejnego przykładu. Transformacje afiniczne pozwalają zmodyfikować wejściowy obraz na kilka sposobów: poprzez rozciąganie, ściskanie, deformowanie i obracanie. W tym miejscu rozważymy prosty program służący do obracania oraz rozciąganie/ściskania wejściowego obrazu (stąd nazwa programu: *Rotozoom*). Program oprócz ścieżki do pliku z docelowym obrazem przyjmuje również dwa dodatkowe parametry: kat obrotu oraz współczynnik skalowania. Na początek, zapraszam Czytelników do zapoznania się z Listingiem 4, na którym przedstawiony jest kod źródłowy omawianego programu.

Program działa następująco. Na początek odczytuje i konwertuje parametry przekazane z linii poleceń (wykorzystałem do tego rzutowanie leksykalne dostępne w bibliotece Boost) oraz czyta i wyświetla wejściowy obraz. W kolejnym kroku przygotowuje sobie macierz transformacji. Warto zauważyć wykorzystanie standardowej struktury `CvMat` oraz funkcji `cvCreateMat()` odpowiedzialnej za alokację macierzy. Macież transformacji tworzona jest na podstawie punktu określającego środek obrotu (zmienna `center`), kąta obrotu oraz współczynnika zoomu. Mając gotową macierz, program tworzy klon obrazu wejściowego, czyści go i przeprowadza transformację afiniczną. Do tej wykonania tej ostatniej operacji służy funkcja `cvWarpAffine()`, przyjmująca jako parametry wejściowy i wyjściowy obraz oraz macierz transformacji. Wynik działania programu przedstawiony jest w nowym oknie. Jeśli użytkownik podał dodatkowy parametr, to przed zakończeniem działania program zapisuje plik z przetransformowanym obrazem. Na Rysunkach 4, 5 i 6 przedstawione są odpowiednio: oryginalny obraz, wynikowy obraz dla parametrów [angle=55.0; zoom=0.7] oraz wynikowy obraz dla parametrów [angle=80.5; zoom=1.6].

OpenCV oferuje oczywiście cały szereg możliwości dostrojenia mechanizmu transformacji afinicznej (zainteresowanych Czytelników ponownie odsyłam do dokumentacji biblioteki). Inne transformacje obrazów wspierane przez OpenCV to:

- Splot (ang. *convolution*),
- Operator Sobel'a,
- Operator Laplace'a,
- Wykrywanie krawędzi metodą Canny,
- Transformacje Hough'a (liniowa i kołowa),
- Mapowanie pikseli,
- Transformacje układów współrzędnych (kartezjańskie oraz polarne),
- Dyskretna Transformata Fouriera (DFT),
- Dyskretna Transformata Kosinusowa (DCT).

Zagadnienia zaawansowane

Opisane wyżej udogodnienia stanowią podstawowe bloki budulcowe przy tworzeniu aplikacji wizji komputerowej. Na ich bazie stworzono i udostępniono w ramach OpenCV szereg bardziej zaawansowanych mechanizmów. Niestety, analiza tychże mechanizmów wymaga sporej wiedzy z dziedziny analizy obrazów i wykracza poza ramy niniejszego tekstu. W tym miejscu pozwolę sobie wymienić i pokrótce opisać bardziej zaawansowane udogodnienia oferowane w ramach OpenCV.

- Histogramy i dopasowanie: *histogram* jest jednym z kluczowych pojęć w teorii analizy obrazów. Pozwala przedstawić pewne statystyczne cechy obrazu, na przykład dystrybucję kolorów. Histogramy wykorzystuje się również bardzo intensywnie w aplikacjach wizji komputerowej; jednym z najbardziej podstawowych ich zastosowań jest wykrywanie zmian na scenie na bazie porównywania cech statystycznych kolejnych klatek obrazu. OpenCV oferuje struktury danych do reprezentacji histogramów, operacje wspomagające ich tworzenie oraz implementacje różnorakich metod ich porównywania i dopasowywania.

Wizja komputerowa w pigułce

Pisząc w największym możliwym uproszczeniu, wizja komputerowa (ang. *computer vision*) to dziedzina wiedzy opisująca w jaki sposób można transformować strumień danych opisujących obraz (np. przychodzący z kamery) w inną reprezentację, bądź – w decyzję. Wspomniane transformacje wykonywane są po to aby osiągnąć określony cel, zaś strumień danych może zawierać dodatkowe informacje kontekstowe (np. informacja o położeniu kamery, lub o strukturze obserwowanej sceny). Typową aplikacją wizji komputerowej może być program monitorujący pewne środowisko (np. magazyn bądź fabrykę) i aktywujący system alarmowy w razie wykrycia intruza. Człowiek będący stworzeniem, którego percepcja opiera się przede wszystkim na postrzeganiu wizualnym, nie zdaje sobie często sprawy ze złożoności tego procesu. Ludzki mózg dzieli sygnał wizyjny na wiele strumieni i przetwarza go w skomplikowany, w dużej części nadal niejasny dla nauki sposób. Z punktu widzenia komputera, obraz to po prostu tablica wartości odczytana z zewnętrznego urządzenia, na dodatek dość prymitywnego w porównaniu z ludzkim okiem. To co dla istoty ludzkiej stanowi w kontekście wizualnego postrzegania rzeczywistości naturalne, rutynowe czynnością - dla maszyny jest zazwyczaj nie lada wyzwaniem. Dlatego też istniejące systemy wizji komputerowej są najczęściej automatami, których mocną stroną jest raczej szybkość i dokładność przetwarzania danych niż finezyjność stosowanych algorytmów – chociażby w porównaniu do złożoności procesów kontrolowanych przez ludzki mózg. Jak wiadomo, trudne wyzwania są najbardziej ekscytujące – stąd niewątpliwie bierze się tak duże zainteresowanie tą dziedziną, szczególnie w środowiskach naukowych. Przemysł również docenia rolę patrzących maszyn – np. wiele istniejących hal produkcyjnych monitorowane jest za pośrednictwem systemów wizji komputerowej. Rozwiązania pokroju biblioteki OpenCV stanowią nieocenioną wartość dodaną dla ludzi i firm parających się tematyką wizji komputerowej, stanowiąc wspólną, solidną podstawę do budowania kolejnych, bardziej złożonych rozwiązań.

Rysunek 6. *Rezultat działania programu Rorozoom: obraz zmodyfikowany; parametry [angle=80.5; zoom=1.6]*

- Określanie konturów: algorytmy wykrywania krawędzi pozwalają wykrywać piksele rozgraniczające segmenty obrazu; niestety nie dostarczają one informacji o strukturze tych segmentów. Kolejnym logicznym krokiem jest złożenie wykrytych krawędzi w kontur. OpenCV oferuje funkcje do wyszukiwania i budowania konturów, a także do ich późniejszej analizy i przetwarzania.
- Części obrazów i segmentacja: OpenCV pozwala izolować wybrane obiekty sceny i wycinać je z wejściowego obrazu; stosowane w tym przypadku metody bazują przede wszystkim na opisanych wcześniej algorytmach przetwarzania obrazu.
- Śledzenie i ruch: w rzeczywistym świecie rzadko kiedy mamy do czynienia ze statycznymi obrazami; najczęściej zachodzi potrzeba obserwacji i analizy sekwencji ruchów określonych obiektów; OpenCV oferuje w tym zakresie cały arsenał implementacji użytecznych metod.
- Modele kamer i kalibracja: wizja komputerowa zaczyna się od wykrywania natężenia światła – jest to głównym zadaniem wszelkiej maści kamer. Niestety, interakcja z tymi urządzeniami nie należy do najprostszych; dzięki OpenCV zadanie to jest o wiele łatwiejsze.
- Projekcja i wizja 3D: zestaw metod do ekstrakcji informacji o 3-wymiarowej geometrii obiektów na bazie zestawu 2-wymiarowych obrazów (między innymi: stereowizja).
- Uczenie maszynowe: mechanizmy wspomagające automatyczne przekształcania danych w informacje – bazujące przede wszystkim na statystycznym dopasowywaniu wzorców (ang. *statistical pattern recognition*) oraz klasteryzacji (ang. *clustering*).; oferowane w OpenCV w postaci dedykowanej, zagnieżdżonej biblioteki: MLL (ang. *Machine Learning Library*).

Podsumowanie

Przedstawiony wyżej, pobieżny przegląd możliwości OpenCV miał na celu zapoznanie Czytelnika ze strukturą i filozofią biblioteki, oraz przekonanie o wysokiej praktyczności prezentowanego rozwiązania. Otwartość, czytelność, efektywność i starannie dobrany poziom abstrakcji – to niewątpliwie czynniki przemawiające za wykorzystywaniem OpenCV, zarówno w zastosowaniach komercyjnych, edukacyjnych jak i hobbystycznych. Ze względu na obszerność prezentowanego zagadnienia, wiele z prezentowanych tematów potraktowane zostało dość powierzchownie. W tej sytuacji autorowi pozostaje odesłać żądnych kolejnych porcji wiedzy Czytelników do zewnętrznych źródeł (patrz Ramka *Materiały*) i życzyć wielu udanych eksperymentów z OpenCV. Nich moc komputerowej wizji będzie z Wami!

PS. Na sam koniec chciałbym podziękować mojemu przyjacielowi, Dawidowi de Rosier, za użyczenie pięknych zdjęć z jego podróży, na użytek powyższego artykułu.

RAFAŁ KOCISZ

Rafał Kocisz pracuje na stanowisku Dyrektora Technicznego w firmie Gamelion, wchodzącej w skład Grupy BLStream. Rafał specjalizuje się w technologiach związanych z produkcją oprogramowania na platformy mobilne, ze szczególnym naciskiem na tworzenie gier. Grupa BLStream powstała, by efektywniej wykorzystywać potencjał dwóch szybko rozwijających się producentów oprogramowania – BLStream i Gamelion. Firmy wchodzące w skład grupy specjalizują się w wytwarzaniu oprogramowania dla klientów korporacyjnych, w rozwiązaniach mobilnych oraz produkcji i testowaniu gier. Kontakt z autorem: rafal.kocisz@game-lion.com

W Sieci

- *http://sourceforge.net/projects/opencvlibrary* : strona projektu na SourceForge.net
- *http://opencv.willowgarage.com/wiki/* : ofcjalne wiki projektu
- *http://tech.groups.yahoo.com/group/OpenCV/* : grupa dyskusyjna projektu

Materiały

Czytelnicy zainteresowani tematem wizji komputowej oraz OpenCV powinni zainteresować się książką Learning OpenCV: Computer Vision with the OpenCV Library, autorstwa panów Garego Bradskiego oraz Adriana Kaehlera. Pozycja ta polecana jest na wiki OpenCV jako oficjalna referencja biblioteki. Oprócz szczegółowej prezentacji możliwości OpenCV, Czytelnik wspomnianej książki będzie miał okazję zapoznać się z wieloma teoretycznymi zagadnieniami z zakresu wizji komputerowej, zilustrowanymi działającymi programami zbudowanymi na bazie OpenCV. Niniejszy artykuł powstał właśnie pod natchnieniem tej pozycji.

Listing 4. *Kod źródłowy przykładowego programu Rotozoom*

```
#include <cv.h>
#include <highgui.h>
#include <cassert>
#include <boost/lexical_cast.hpp>
const char* g_inWindowName = "Rotozoom (in)";
const char* g_outWindowName = "Rotozoom (out)";
int main( int argc, char** argv )
{
    assert( argc == 4 || argc == 5 );
    double angle = boost::lexical_cast< double >( argv[ 2 ] );
    double scale = boost::lexical_cast< double >( argv[ 3 ] );
    IplImage* src = cvLoadImage( argv[ 1 ], 1 );
    IplImage* dst = 0;
    assert( src != 0 );
    cvNamedWindow( g_inWindowName, 1 );
    cvShowImage( g_inWindowName, src );
    CvMat* rotationMatrix = cvCreateMat( 2, 3, CV_32FC1 );
    CvPoint2D32f center = cvPoint2D32f(
        src->width / 2,
        src->height / 2
    );
    cv2DRotationMatrix( center, angle, scale, rotationMatrix );
    dst = cvCloneImage( src );
    dst->origin = src->origin;
    cvZero( dst );
    cvWarpAffine( src, dst, rotationMatrix );
    cvNamedWindow( g_outWindowName, 1 );
    cvShowImage( g_outWindowName, dst );
    cvWaitKey();
    if ( argc == 5 )
    {
        cvSaveImage( argv[ 4 ], dst );
    }
    cvReleaseImage( &src );
    cvReleaseImage( &dst );
    cvDestroyWindow( g_inWindowName );
    cvDestroyWindow( g_outWindowName );
    cvReleaseMat( &rotationMatrix );
    return 0;
}
```

Oracle CEP – złożone przetwarzanie zdarzeń

Complex Event Processing / Event Stream Processing

Complex Event Processing (CEP) to jedna z technik przetwarzania zdarzeń w aplikacjach zdarzeniowych, posługujących się paradygmatem "nasłuchuj-i-odpowiedz" (ang. listen-and-respond), czyli takich, w których logika biznesowa jest uruchamiana w wyniku pojawienia się określonego zdarzenia.

Dowiesz się:
- Na czym polega specyfika Complex Event Processing;
- Jak buduje się aplikacje CEP w Java;
- Czym jest zdarzeniowy serwer aplikacyjny taki jak Oracle CEP.

Powinieneś wiedzieć:
- Trochę o SQL;
- Czym jest serwer aplikacyjny;
- Czym jest Spring Framework.

Poziom trudności

Zdarzenie jest zapisem zajścia określonej zmiany. Zdarzenia (ang. *events*) są najczęściej *transportowane* do aplikacji zdarzeniowych w postaci komunikatów (ang. *messages*). Typowymi źródłami zdarzeń są sensory (np. czujniki), czy generatory, ale mogą to być także aplikacje biznesowe (np. wykonanie transakcji, bądź procesu biznesowego, co powoduje powstanie jednego lub – co częstsze – większej liczby zdarzeń).

Na czym polega CEP?

Specyfika CEP jako techniki przetwarzania zdarzeń polega na tym, że CEP skupia się nie tyle na przetwarzaniu pojedynczych zdarzeń, co raczej na przetwarzaniu grup zdarzeń, poszukiwaniu korelacji pomiędzy zdarzeniami w grupie, szczególnie korelacji mających charakter czasowy (strumieni zdarzeń), przyczynowo-skutkowy (czy w innym wymiarze, np. geograficznym). Zwykle, przetwarzanie tych grup zdarzeń w CEP odbywa się w czasie rzeczywistym, tzn. czas reakcji na pojawienie się określonej korelacji wśród płynących do aplikacji zdarzeń jest skończony. Korelacja jest zdaniem logicznym typu *kiedy ...., to ....* Przykłady korelacji:

- kiedy pojawi się komunikat typu A, a po nim – w przeciągu maksymalnie 10 minut – pojawi się komunikat typu B, to wykonaj operację X;
- kiedy pojawi się komunikat typu *Transakcja Giełdowa*, którego wartość atrybutu *Cena* odbiega od średniej wartości tego atrybutu w ciągu ostatnich 30 minut, to wykonaj operację *Kupuj*;
- kiedy w ciągu 15 minut po pojawieniu się komunikatu typu A, pojawi się komunikat typu B, posiadający tę samą wartość atrybutu id co wartość atrybutu id w A oraz NIE pojawi się komunikat typu C (także mający ten sam id), to wykonaj operację X.

Korelacje mogą także obejmować wiele źródeł zdarzeń (np. wiele strumieni transakcji giełdowych).

CEP w ofercie Oracle

Technologią wspierającą budowę aplikacji zdarzeniowych, korzystających ze złożonego i strumieniowego przetwarzania zdarzeń (CEP/ESP) w Java jest Oracle CEP Server (dawniej BEA WebLogic Event Server). Jest to dedykowany przetwarzaniu zdarzeń serwer aplikacyjny Java, oferujący model programistyczny oparty o Spring, Spring-DM i OSGi, wraz z zaawansowanym, wysokowydajnym procesorem zdarzeń, którego zadaniem jest monitorowanie w sposób ciągły strumieni zdarzeń przychodzących do aplikacji i poszukiwaniu w tych strumieniach zdefiniowanych wcześniej korelacji. Do opisu korelacji (reguł) wykorzystywany jest oparty o SQL język EPL (*Event Processing Language*), który wzbogaca znane z SQL mechanizmy o konstrukcje typowe dla złożonego przetwarzania zdarzeń – takie, jak płynące (ang. *sliding window*) lub skaczące (ang. *batching window*) okna (czasowe lub obejmujące określoną liczbę zdarzeń), wyszukiwanie wzorców (przyczyna-skutek), trendów, a nawet identyfikację brakujących elementów (zdarzeń) we wzorcu.

W EPL zamiast nazw kolumn (po klauzuli `SELECT`) znajdują się nazwy atrybutów zdarzeń, a zamiast nazw tabel (po `FROM`) znajdują się nazwy strumieni zdarzeń (choć mogą także znajdować się nazwy tabel lub buforów cache, co może być przydatne przy łączeniu dynamicznych, szybkozmiennych danych w zdarzeniach z bardziej statycznymi danymi w bazach danych). Podobnie w EPL wykorzystywane są takie konstrukcje SQL jak `INSERT INTO` (czyli tworzenie nowego strumienia zdarzeń), `WHERE`, `INNER/OUTER/FULL/LEFT/RIGHT JOIN`, `ORDER BY`, `GROUP BY`, `HAVING`, itp.

CEP w przykładach

W poniższym przykładzie reguła EPL dokonuje filtrowania zdarzeń w strumieniu `MarketTransactions` (np. modelującym transakcje kupna/sprzedaży walut na giełdzie) i zwraca (*przepuszcza* dalej, tj. do kolejnego elementu w sieci EPN – *Event Processing Network*, o której w dalszej części artykułu) tylko te zdarzenia, których wartość atrybutu `price` dla danej waluty wynosi mniej niż 75% średniej ruchomej wartości tego atrybutu z ostatnich 10 sekund. Reguła tego typu pozwala wykrywać w czasie rzeczywistym nagłe spadki ceny określonych walut. (Listing 1.)

Widoczne jest tutaj wprowadzenie koncepcji płynącego okna czasowego (`RETAIN 10 SECONDS`), przechowującego tyle zdarzeń ze strumienia `MarketTransactions`, ile pojawiło się w ciągu ostatnich 10 sekund. Wykonanie reguły (`SELECT`, `GROUP BY` i `HAVING`) jest zawężone do tego okna czasowego. Reguła jest wykonywana w sposób ciągły (ang. *continuous query*, *streaming query*).

Inny przykład EPL dotyczy wykrywania najwyżej wycenionych akcji (walorów): (Listing 2.)

Tutaj, ze strumienia `MarketTrade` filtrujemy najpierw transakcje o określonej wielkości (`blockSize > 10`) – widać przykład użycia podzapytań (zagnieżdżonych `SELECT`). Dla każdego waloru (`symbol`), przechowywanych jest 100 takich transakcji o najwyższej cenie (`RETAIN 100 EVENTS WITH LARGEST price PARTITION BY symbol`). Czyli dla 1000 symboli będzie przechowywanych 100.000 takich transakcji. Wynik zapytania jest dodatkowo posortowany (`ORDER BY`) i pogrupowany (`GROUP BY`) wg symbolu waloru, z odrzuceniem tych walorów, których średnia cena jest mniejsza od 100 (`HAVING`).

W kolejnym przykładzie tworzony jest nowy strumień zdarzeń (`INSERT INTO MessagesStream`), w którym będą pojawiać się zdarzenia o określonych atrybutach (`SELECT 'terminal 1 is offline'`), gdy w dowolnym oknie czasowym o długości 65 sekund (`WITHIN 65 SECONDS`), w strumieniu `Status` NIE POJAWI się zdarzenie, którego atrybut term.id jest równy `T1` (`MATCHING NOT`). Dodatkowo jest kontrolowana *częstotliwość* pojawiania się zdarzeń w tym nowym strumieniu (`MessagesStream`) – tak, aby tylko raz na pięć minut takie zdarzenie było wysyłane (`OUTPUT FIRST EVERY 5 MINUTES`): (Listing 3.)

Wynik wykonania takiej reguły ma najczęściej postać nowego strumienia zdarzeń, które *wzbudzają* wykonywanie określonej logiki biznesowej (np. kodu Java bądź wywołania zewnętrznych usług – notabene w tym drugim przypadku, często mówi się o ED-SOA, czyli *Event-driven Service Oriented Architecture*, tj. architekturze zorientowanej na usługi, w której żądanie wykonania usług jest określone przez pojawienie się zdarzeń lub – jak w przypadku CEP – określonej korelacji zdarzeń w czasie).

Oracle CEP w zbliżeniu

Na aplikację w Oracle CEP Server składa się z szereg komponentów przetwarzających zdarzenia (adapterów, procesorów zdarzeń, kodu Java, obiektów Spring, usług OSGi, itd), połączonych w *sieć przetwarzania zdarzeń* (ang. *EPN – Event Processing Network*). Sieć jest tworzona (składana) w sposób deklaratywny, z wykorzystaniem takich technik programistycznych jak Dependency Injection i Aspect Oriented Programming. W łatwiejszym skorzystaniu z tych technik pomaga również wykorzystanie bardzo popularnej platformy aplikacyjnej – Spring Framework, wraz z jej rozszerzeniami w stronę innej, zdobywającej coraz większą popularność technologii komponentowej – OSGi (Oracle CEP Server jest jednym z pierwszych serwerów aplikacyjnych, które na produkcyjną skalę wykorzystują technologię OSGi). Wykorzystana jest tu rozwijana przez Oracle i SpringSource technologia Spring-DM (*Spring Dynamic Modules for OSGi Service Platforms*).

Publicznie dostępna w postaci HTML i PDF dokumentacja produktu (*http://www.oracle.com/technology/documentation/bea.html*) zawiera bardziej dokładny opis języka EPL i dalsze jego przykłady, a także znacznie dokładniejsze informacje nt. modelu programistycznego, sposobu budowania aplikacji zdarzeniowych i ich komponentów, administracji, klastrowania, itp.

Oracle CEP Server wspiera budowę wysokowydajnych aplikacji zdarzeniowych, tzn. jest specjalnie zaprojektowany pod kątem przetwarzania dużej liczby zdarzeń (rzędu milionów zdarzeń na sekundę i więcej), z zachowaniem reżimu bliskiemu czasowi rzeczywistemu (np. wymaganie, aby średni czas przetwarzania przez serwer aplikacyjny 99.99% zdarzeń był poniżej 5 milisekund). Także liczba równocześnie wykonywanych zapytań może być wysoka (rzędu tysięcy lub dziesiątków tysięcy wykonywanych w sposób ciągły reguł). Zagadnienie wydajności jest dosyć dokładnie omówione w publicznie dostępnym dokumencie, opisującym jeden z benchmarków produktu Oracle CEP Server (dokument dostępny pod adresem: *http://www.oracle.com/technology/products/event-driven-architecture/complex-event-processing.html*). Warto przy tym zaznaczyć, iż możliwość przetwarzania miliona zdarzeń na sekundę w czasie rzeczywistym (soft real-time), osiągnięto na przeciętnym dzisiaj sprzęcie klasy Intel x86 (4 CPU quad core, Intel Xeon).

Z racji tego, iż Oracle CEP Server jest serwerem aplikacyjnym Java, to zawiera także mechanizmy klastrowe, pozwalające budować skalowalne i wysokodostępne rozwiązania. Wykorzystywana jest tutaj wiodąca w tej klasie technologia gridowa (*in-memory data grid*) – Oracle Coherence. Pozwala ona tworzyć automatycznie zarządzane, przechowywane w pamięci bufory i ich rozproszenie na wiele (nawet tysiące) maszyn. Dzięki redundancji (replikacji danych) eliminowana jest możliwość utraty danych w wyniku awarii pojedynczej maszyny (a nawet większej ich liczby), a jednocześnie zachowana jest bardzo wysoka wydajność i krótki czas opóźnień (ang. *latency*) w dostępie do buforów. Dostępne są także zaawansowane mechanizmy przetwarzania (np. wykonywanie zapytań i logiki biznesowej w sposób masowo-równoległy).

Najnowsza wersja produktu przynosi także wiele innych przydatnych mechanizmów – np. możliwość nagrywania (ang. *record*), a następnie odtwarzania (ang. *playback*) zdarzeń przychodzących/wychodzących z określonych komponentów aplikacji. W zakresie możliwości integracyjnych – obok samodzielnie tworzonych w kodzie Java adapterów – możliwe jest także wykorzystanie (poprzez konfigurację, tj. bez potrzeby kodowania) technologii JMS, czy HTTP Publish-Subscribe (poprzez protokół Bayeux, znany m.in. z rozwiązań klasy `Comet`, czy `reverse-AJAX`/`push-AJAX`).

Oracle CEP w Internecie

W pełni funkcjonalna wersja ewaluacyjna Oracle CEP Server jest dostępna na stronach internetowych Oracle (*http://www.oracle.com/technology/software/products/cep/index.html*), na których także można znaleźć więcej informacji o samej technologii, typowych scenariuszach jej użycia i pozycjonowaniu w ramach szerszej infrastruktury SOA i EDA.

Praktyczne ćwiczenia, obejmujące m.in. demonstrację instalacji produktu, konfigurację środowiska deweloperskiego (z wykorzystaniem Eclipse), budowę prostych aplikacji zdarzeniowych oraz wskazówki dotyczące poznawania pozostałych możliwości technologii Oracle CEP Server, autor niniejszego artykułu zamieścił na swoim blogu: *http://jdn.pl/blog/88*.

WALDEMAR KOT

Waldemar Kot pracuje w Oracle Polska jako Principal Sales Consultant, Eastern Europe.
Kontakt z autorem: waldemar.kot@oracle.com

Listing 1. *Reguła EPL – przykład prostego filtrowania zdarzeń z użyciem okna czasowego*

```
SELECT currency, AVG(price) AS avgPrice, price AS MarketPrice
FROM MarketTransactions
RETAIN 10 SECONDS
GROUP BY currency
HAVING price < AVG(price) * 0.75
```

Listing 2. *Reguła EPL – przykład obliczania wartości maksymalnej w czasie*

```
SELECT symbol, AVG(price)
FROM (SELECT * FROM MarketTrade WHERE blockSize > 10)
RETAIN 100 EVENTS WITH LARGEST price PARTITION BY symbol GROUP BY symbol
HAVING AVG(price) >= 100
ORDER BY symbol
```

Listing 3. *Reguła EPL – przykład wykrywania brakujących zdarzeń we wzorcu*

```
INSERT INTO Messages
SELECT 'terminal 1 is offline'
MATCHING NOT Status(term.id = 'T1') WITHIN 65 SECONDS
OUTPUT FIRST EVERY 5 MINUTES
```

jQuery i ASP.NET

Wprowadzenie do biblioteki jQuery z przykładami w ASP.NET AJAX

Otwarta biblioteka JavaScript jQuery sprawia, że pisanie kodu po stronie klienta staje się przyjemniejsze a dzięki połączeniu jQuery z technologią ASP.NET AJAX otrzymujemy potężny, efektywny szkielet aplikacyjny do pisania aplikacji ajaksowych.

Dowiesz się:
- czym jest biblioteka jQuery,
- jakie korzyści daje integracja jQuery z ASP.NET.

Powinieneś wiedzieć:
- wiedzieć jak tworzyć strony w ASP.NET,
- posiadać podstawową wiedzę na temat biblioteki Microsoft ASP.NET AJAX,
- posiadać znajomość składni CSS,
- posiadać podstawową znajomość JavaScript.

Poziom trudności

Biblioteka jQuery została napisana przez Johna Resiga. Umożliwia ona przede wszystkim eleganckie i efektywne manipulowanie elementami HTML przy minimalnym nakładzie pracy. Programistów ASP.NET zainteresuje dodatkowo fakt, że Microsoft ogłosił pełne wsparcie dla frameworka jQuery. Połączenie jQuery z ASP.NET ułatwia praca każdemu programiście aplikacji webowych, ponieważ umożliwia bardzo szybkie wzbogacenie pisanej aplikacji o elementy interaktywne np. dynamiczną zmianę kolorów, przesunięcie elementu czy jego wygaszenie. W niniejszym artykule zostaną omówione podstawy oraz sposoby wykorzystania jQuery wraz z technologią ASP.NET AJAX.

Wprowadzenie do jQuery

Dzięki użyciu jQuery można diametralnie zmienić nastawienie do programowania po stronie klienta. Nauczenie się tej biblioteki może przynieść istotne korzyści poprzez zwiększenie naszej produktywności oraz podniesienie pewności siebie przy pisaniu skomplikowanych frontonów w Javascript.

Biblioteka jQuery dostarcza nam kilka kluczowych funkcjonalności, takich jak:

- Selektory elementów HTML.
- Operacje na zbiorach.
- Obsługę zdarzeń.
- Animacje.
- Łatwą rozszerzalność poprzez zastosowanie wtyczek.

Co jest również bardzo ważne, ta bogata funkcjonalność jQuery dostarczona jest w małej bibliotece o wadze około 19 KB (wersja skompresowana).

Aby można było rozpocząć pracę z biblioteką jQuery, należy dodać referencję do niej na swojej stronie.

Ostatnią wersję jQuery możemy pobrać ze strony *www.jQuery.com*. Bibliotekę dodajemy do strony poprzez znacznik `<script src="scripts/jquery.js" type="text/javascript">`.

Innym sposobem dodania referencji do biblioteki może być użycie kontrolki *ScriptManager* z biblioteki ASP.NET AJAX.

```
<asp:ScriptManager id="sm1" runat="server">
    <Scripts>
      <asp:ScriptReference Path="~/Scripts/
                    jquery-1.2.6.js" />
    </Scripts>
</asp:ScriptManager>
```

W celu pokazania omawianych poniżej właściwości biblioteki jQuery, stworzymy prostą aplikację w ASP.NET, widoczną na Listingu 1. Przedstawiona strona wyświetla listę filmów w postaci prostej tabeli, bez formatowania – jak widać na Rysunku 1. W kolejnych sekcjach dodawać będziemy kod jQuery do naszej przykładowej strony, aby tabela filmów wyglądała efektowniej oraz umożliwiała prostą interakcję z użytkownikiem.

Selektory elementów HTML

Selektory jQuery pozwalają wskazać elementy strony HTML, na których następnie możemy wywołać wbudowane metody biblioteki. jQuery używa składni CSS 3.0 (plus kilka rozszerzeń), aby wskazać jeden, bądź kilka, elementów w dokumencie. Dzięki użyciu składni CSS, jQuery jest bardzo prosty do nauczenia się i używania – programista aplikacji internetowych jest obeznany z tą składnia selektorów dzięki stylizowaniu stron HTML.

Element strony HTML możemy wskazać m.in. poprzez atrybut *id*

```
$(„#gdMovies”).css(„width”,”120px”),
```

poprzez klasę CSS

```
$(„.gridAlternate”).css(„background”,”yell
                    ow”),
```

albo możemy również wskazać elementy z kilku klas CSS

```
$(„.gridrow, .gridAlternate”).attr(„tag”,”
                    row”).
```

Do wskazywania elementów możemy używać również filtrów. Następujący przykład wybie-

ra wszystkie elementy w dokumencie będące przyciskami i zmienia im kolor tła:

```
$(„input:button").css(„background","yellow").
```

Jeśli składnia CSS nie jest Ci obca, to zrozumienie powyższych przykładów nie powinno stanowić żadnego problemu. Takiej samej składni możesz użyć w arkuszu stylów w celu znalezienia elementów do stylizacji (wyjątkiem jest filtr `:button`, ponieważ jest on specyficzny dla jQuery).

Jak widać, idea twórcy jQuery jest prosta: używaj to, co już dobrze znasz albo przynajmniej używaj standardowych mechanizmów zamiast nowych rozwiązań.

W Tabeli 1 zamieszczony został opis kilku najczęściej używanych selektorów w jQuery.

Wróćmy do naszego przykładu i zacznijmy od dodania skryptu z Listingu 2 na dole strony. Dzięki temu prostemu skryptowi otrzymaliśmy efekt pokolorowania parzystych wierszy tabeli co widać na Rysunku 2. Przeanalizujmy co sprawiło że ta prosta komenda jQuery zadziałała.

Jeżeli wrócimy do kodu z Listingu 2, to przede wszystkim zauważymy symbol *$*, który jest aliasem obiektu jQuery. W wyniku czego zapis `$(document)` można napisać `jQuery(document)`. Obiekt jQuery pobiera selektor jako parametr. Przeważnie selektory są reprezentowane przez łańcuchy znaków, ale możemy również przekazać elementy DOM albo inny obiekt jQuery.

Przykładowo zapis `$(document)` przekazuje element `document`, wynikiem czego jest zbiór jednoelementowy zawierający obiekt `document`.

Zwróćmy uwagę na użycie metody `$(document).ready()`. `.ready()` jest metodą obsługującą zdarzenie, informującą kiedy dokument jest gotowy do użycia oraz o fakcie, że wszystkie skrypty zostały załadowane. Metoda obsługująca zdarzenie `.ready()` może zostać umieszczona w dowolnym miejscu na stronie, nawet parę razy. Procedura obsługi została zaimplementowana przy użyciu anonimowej funkcji `$(document).ready(function(){...})`. Jest to bardzo częsta praktyka w jQuery. Jako parametr dla zdarzenia `ready` możemy również przekazać wskaźnik do funkcji, np.: `$(document).ready(onLoaded)`, gdzie `onLoaded` jest funkcją zdefiniowaną w kodzie.

Wracając do przykładu kolorowych wierszy, przeanalizujmy część kodu, który wykonuje akcję zamiany koloru. Cała magia tej akcji leży w użytym selektorze `$("#gdMovies tbody tr:even")`, który możemy przeczytać następująco: pobierz element `gdMovies` oraz wszystkie jego dzieci będące elementami `tbody` i znajdź wszystkie elementy `tr` będące poniżej oraz zwróć odfiltrowany `(tr:even)` zbiór, który zawiera tylko elementy parzyste. Na końcu użyta została metoda CSS, która dla każdego elementu ze zwróconego zbioru przypisuje styl definiujący kolor tła.

Listing 1. *Prosta strona ASP.NET używana w przykładach*

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">

    <title>Table</title>
 <script src="Scripts/jquery-1.2.6.js" type="text/javascript"></script>

</head>
<body>
    <form id="form1" runat="server">
    <div>

     <asp:GridView ID="gdMovies" runat="server" AutoGenerateColumns="False"
           DataKeyNames="Id" DataSourceID="srcMovies">
           <HeaderStyle CssClass="gridheader"/>
           <Columns>
               <asp:BoundField DataField="Id" HeaderText="Id" ReadOnly="True"
                   SortExpression="Id" />
               <asp:BoundField DataField="Title" HeaderText="Tytuł"
                     SortExpression="Title" />
               <asp:BoundField DataField="Director" HeaderText="Reżyser"
                   SortExpression="Director" />
               <asp:BoundField DataField="DateReleased" HeaderText="Data produkcji"
                   SortExpression="DateReleased" />
           </Columns>
       </asp:GridView>
            ...
    </div>
    </form>
</body>
</html>
```

Listing 2. *Przykład uzyskania efektu pokolorowania parzystych wierszy tabeli*

```
 <script type="text/javascript">
    $(document).ready(function() {

        $("#gdMovies tbody tr:even")
                   .css("background","yellow");
    }
    );
</script>
```

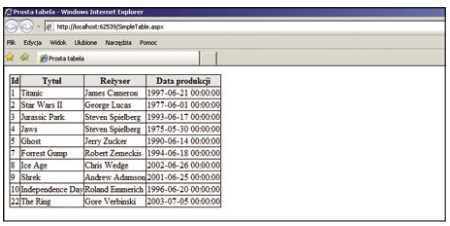
Operacje na zbiorach

Wynikiem komendy `$("#gdMovies tbody tr:even")` jest obiekt jQuery, który jest tablicą zawierającą każdy dopasowany element. Możemy odwołać się do elementów DOM w odszukanym zbiorze również bezpośrednio.

```
var res = $("#gdMovies tbody tr:even");
var len= = res.length; // liczba
   dopasowań. 0 = brak dopasowań
var el = res[0] // pierwszy element
```

jQuery zawsze zwraca rezultat w postaci obiektu, nawet jeśli selektor nie znalazł elementów, obiekt jest zwracany, dzięki czemu możemy sprawdzić właściwość `.length`, aby określić czy zostały zwrócone jakieś elementy.

Jeżeli jednak selektor znalazł elementy i zwrócił listę, można łatwo ją przetworzyć używając jednej z ponad 100 funkcji z jQuery API. Każda taka funkcja zostanie zastosowana dla każdego elementu ze znalezionej listy, bez potrzeby ręcznego przechodzenia przez ten zbiór elementów.



Rysunek 1. *Oryginalna wersja strony testowej po uruchomieniu*

jQuery posiada bardzo przydatną funkcjonalność jaką jest łączenie wywołań wielu funkcji w ramach jednej komendy. Przykład takiego wywołania znajduje się na Listingu 3.

Nie wszystkie metody mogą być łączone w łańcuchy wywołań, dotyczy to metod zwracających prostą wartość, a nie listę elementów. Tymi funkcjami są m.in.: `val()`, `text()`, `html()` – których rezultatem jest łańcuch znaków oraz metody `width()` i `height()` – zwracające liczby.

Obsługa zdarzeń

Obsługa zdarzeń w jQuery jest bardzo prosta i co najważniejsze spójna pomiędzy różnymi przeglądarkami. jQuery dostarcza dwie główne funkcje `bind()` oraz `unbind()`, które służą odpowiednio do przyłączenia i odłączenia procedury obsługi zdarzenia do elementów na liście elementów znalezionych przez selektor.

Dodatkowo zdarzenia klawiatury i myszy mają specjalne dedykowane funkcje takie jak `click()`, `mousedown()`, `change()` albo `keydown()`.

jQuery wprowadza bardzo prosty model dla obsługi zdarzeń:

- wskaźnik `this` zawsze odpowiada elementowi wywołującemu zdarzenie,
- obiekt zdarzenia jest zawsze przekazywany jako parametr,
- obiekt zdarzenia jest jednolity pomiędzy rożnymi przeglądarkami.

Wracając do naszego przykładu z tabelą, dodamy do naszej tabeli następujące zachowanie, aby po najechaniu myszą na wiersz zmienił się jego kolor tła oraz, aby po kliknięciu na danym wierszu wyświetliła się zawartość trzeciej kolumny.

Rozwiązanie tego zadania prezentuje Listing 4. Aby osiągnąć ten efekt, dodajmy dwie procedury obsługujące zdarzenia po najechaniu myszą na wiersz – zdarzenie `hover()` oraz zdarzenie kliknięcia myszą – zdarzenie `click()`.

Funkcja `hover()` pobiera jako argumenty dwie metody zwrotne. Pierwsza funkcja obsługuje zdarzenie najechania myszą, druga funkcja natomiast obsługuje zdarzenie usunięcia kursora myszy znad obiektu.

Procedura obsługi zdarzenia zawsze jest wywoływana w kontekście elementu powodującego zdarzenie – w naszym przypadku jest to element `tr`, do którego w kodzie mamy dostęp poprzez następujący zapis `$(this)`.

Animacje

jQuery udostępnia kilka bardzo prostych, ale efektownych metod pozwalających animować obiekty, są to:

- `$(...).hide()` – chowanie elementu,
- `$(...).show()` – pokazywanie elementu,
- `$(...).slideDown()` – rozwinięcie elementu z góry do dołu,
- `$(...).slideUp()` – rozwinięcie elementu z dołu do góry,
- `$(...).fadeIn()` – rozjaśnienie elementu,
- `$(...).fadeOut()` – ściemnienie elementu.

Każda z tych metod może przyjąć dwa parametry, pierwszy określa prędkość wykonywania animacji, można tu podać jedną z przedefiniowanych stałych (`'slow'`, `'normal'`,`'fast'`) albo podać czas w milisekundach. Drugi parametr określa funkcję zwrotną, która zostanie wykonana po zakończeniu animacji.

Taka prosta możliwość animacji obiektów dodatkowo ułatwia pisanie aplikacji z użyciem technologii AJAX, w których odświeżany jest tylko fragment strony. Dzięki takiemu prostemu zastosowaniu animacji możemy w łatwy sposób pokazać, że zaszła zmiana na elemencie odświeżanym.

Łatwa rozbudowa poprzez wtyczki

Wtyczki są jednym z głównych powodów, dlaczego jQuery stał się tak popularny. Na stronie *plugins.jquery.com*, możemy znaleźć bogatą kolekcję wtyczek dla jQuery podzielonych na różne kategorie, m.in. tabela, AJAX, menu, animacje, itp.

Użycie wtyczek jest bardzo proste, wystarczy pobrać wtyczkę z odpowiedniej strony, dodać do niej referencję, i już możemy wywołać ją na naszym obiekcie jQuery tak jak każdą inną wbudowaną metodę. Dla przykładu użyjemy wtyczki *tablesorter*, która dodaje do tablicy możliwość sortowania. Po pobraniu wtyczki z jej strony domowej (*http://tablesorter.com/docs/*), umieszczamy referencję do niej na naszej stronie, i już jesteśmy gotowi do jej użycia. Teraz wystarczy wykonać następującą komendę `$("#gdMovies").tablesorter()`. I to wszystko! Teraz po uruchomieniu strony i kliknięciu w nagłówek w tabeli dane zostaną posortowane po stronie klienta.

Tabela 1. *Lista najczęściej używanych selektorów w jQuery*

Selektor	Przykład	Opis
`element`	`$(„td")`	Wskaż znacznik HTML
`#id`	`$(„#divMessage")`	Wskaż element poprzez id
`.klasacss`	`$(„.gridrow")`	Wskaż elementy z klasą CSS
`selektor, selektor`	`(„input:button, input:text")`	Selektory rozdzielone przecinkiem mogą zostać połączone w pojedynczy selektor
`przodek potomek`	`(„#divMessage a")`	Znajduje elementy zagnieżdżone
`rodzic>dziecko>dziecko`	`$(„p>b")`	Znajduje wszystkie bezpośrednie dzieci elementu będącego rodzicem
`poprzedni ~ następny`	`$(„#row:nth-child(2)~td")`	Zaznacz wszystkie elementy występujące po elemencie *poprzedni* na tym samym poziomie
`brat + brat`	`$(„#tdCol+td")`	Zaznacz następny element
`:filtr`	`$(„input:button")`	Stosuje filtr do zapytania. Przykłady: `:not` – odfiltrowuje elementy pasujące do tego selektora, `:button` – wskazuje wszystkie elementy input typu button, `:checked` – wskazuje wszystkie elementy input typu checkbox, `:first` – pobiera pierwszy znaleziony element, `:last` - pobiera ostatni element, `:nth-child(1)` – wskazuje element, który jest n-tym dzieckiem rodzica, `:parent` – zaznacz wszystkich rodziców.
`[@atrybut]`	`$(„div[class=gridrow]")`	Wskazuje atrybut elementu. Dostępne operatory: `=` – równa się, `^=` – rozpoczyna się od, `$=` – kończy się na, `*=` – zawiera.

Microsoft i jQuery

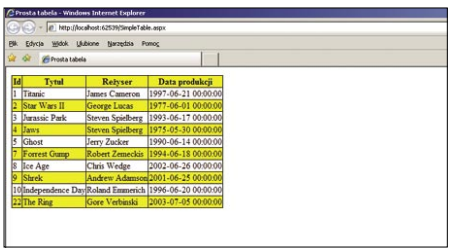
Biblioteka jQuery stała się bardzo popularna, co sprawiło, że firma Microsoft zainteresowała się nią. Wynikiem tego zainteresowania jest decyzja o pełnym wsparciu dla tego narzędzia, dzięki czemu będzie można traktować jQuery jako integralną część istniejących rozwiązań ASP.NET. Już w chwili obecnej biblioteka jQuery jest dostarczana wraz z ASP.NET MVC, a także dostępny jest intellisense dla jQuery (Rysunek 3 i Rysunek 4).

Aby włączyć w Visual Studio 2008 obsługę intellisense dla jQuery, należy pobrać poprawkę hotfix dla Visual Studio 2008 SP1 (*VS 2008 SP1 Hotfix for –vsdoc.js Intellisense* (KB958502)) oraz pobrać plik z dokumentacją jQuery – vsdoc.js. Następnie umieścić plik z dokumentacją w tym samym folderze, gdzie mamy bibliotekę jQuery, ważne, aby nazwy plików się zgadzały, jeżeli posiadamy plik jquery.js, to plik z dokumentacją musi się nazywać jquery-vsdoc.js. Taka operacja wystarczy, aby Visual Studio 2008 automatycznie znalazł plik dokumentacji i włączył intellisense dla jQuery.

Galeria zdjęć

Żeby zobaczyć korzyści z połączenia jQuery i ASP.NET AJAX, stworzymy prostą aplikację – galerię zdjęć, której zadaniem będzie wyświetlenie zdjęć należących do wskazanej kategorii. Informacja na temat zdjęć należących do danej kategorii pochodzi z prostej usługi WCF. Zwrócony zbiór danych zostanie przedstawiony po stronie klienta wykorzystując nową funkcjonalność biblioteki ASP.NET AJAX, która nazywa się Client Templates, czyli szablony po stronie klienta. Pełny kod aplikacji dostępny jest na płycie CD dołączonej do czasopisma. Listing 5 przedstawia stronę ASP.NET, która wyświetla zdjęcia. Aplikacja używa prostej usługi web, która zwraca z bazy danych kolekcję obiektów `Photo`. Każdy taki obiekt zawiera właściwość `Url` – która reprezentuje adres, gdzie jest zlokalizowany obrazek oraz właściwość `Title`, czyli nazwę obrazka.

Przeanalizujmy kod po stronie klienta. Pierwszą funkcją jest funkcja `pageLoad`, która jest wywoływana po załadowaniu strony. W metodzie tej wykonujemy dwie rzeczy. To, co jest najważniejsza w tej metodzie, znajdu-



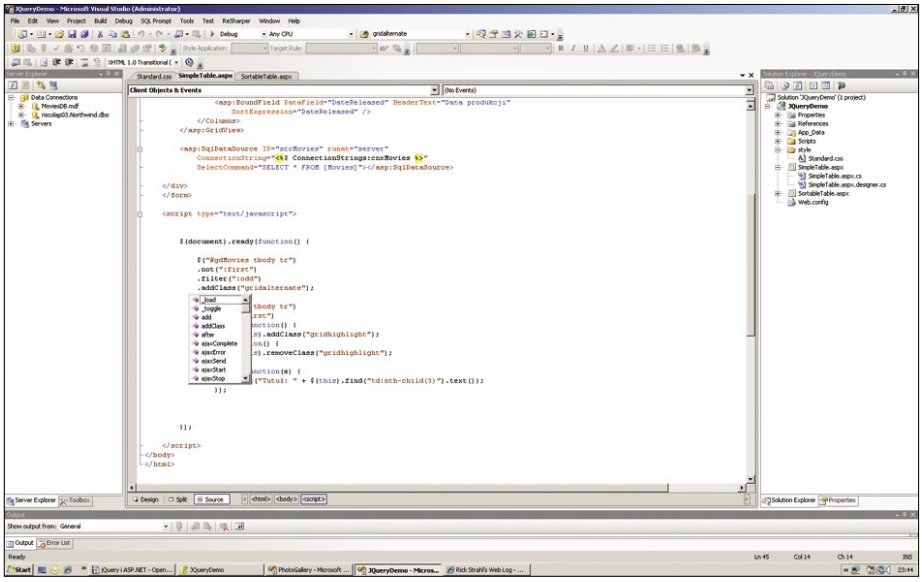
Rysunek 2. *Stworzenie efektu kolorowych naprzemiennych wierszy przy użyciu selektora z Listingu 2*

Listing 3. *Przykład wywołania wielu funkcji w ramach jednej komendy*

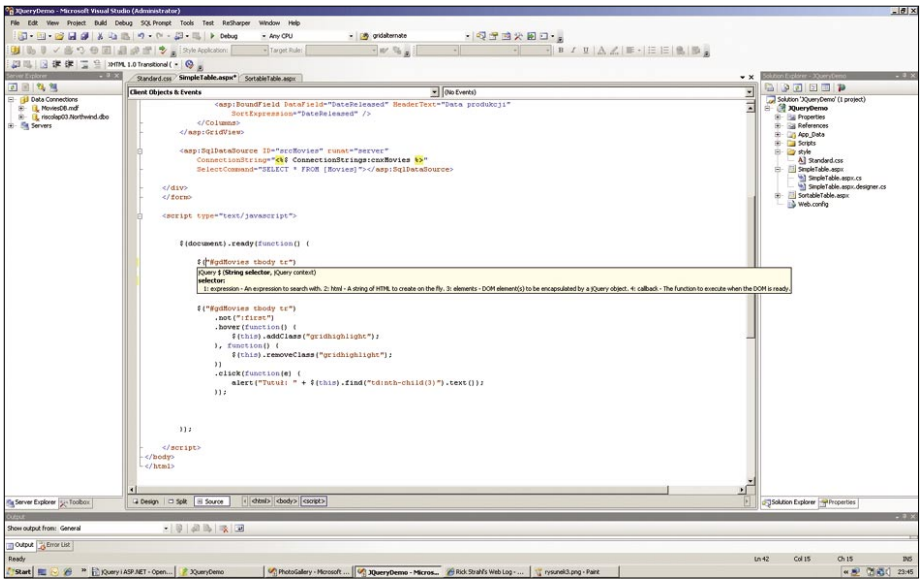
```
$("#gdMovies tbody tr")
                    .not(":first")
                    .filter(":odd")


                    .css("background", "red")
```

Listing 4. *Przykład obsługi zdarzeń*

```
$("#gdMovies tbody tr")
                .not(":first") //odfiltruj pierwszy wiersz
                .hover(function() {

                    $(this).addClass("gridhighlight");
                }, function() {

                    $(this).removeClass("gridhighlight");
                })
                .click(function(e) {
                    alert("Tutuł: " + $(this).find("td:nth-child(3)").text());
                });
```



Rysunek 3. *Intellisense dla jQuery*



Rysunek 4. *Intellisense dla jQuery*

je się w drugim wierszu, gdzie używając instrukcji ASP.NET AJAX, tworzymy obiekt DataView z przypisanym do niego szablonem HTML.

```
photoView = $create(Sys.UI.DataView, {},
  {}, {}, $get("photoContainer"));
```

Kontrolka `DataView` jest nową, dostępną w ASP.NET AJAX kontrolką po stronie klienta, która używa szablonu do dynamicznego wygenerowania interfejsu użytkownika w powiązanym z kontrolką kontenerze. W naszym przykładzie wiążemy kontrolkę `DataView` z kontenerem `photoContainer`, który prezentuje się w następujący sposób: (Listing 6)

W środku tego kontenera znajduje się szablon, który definiuje sposób, w jaki zostaną wyświetlone nasze zdjęcia.

Kolejną metodą w kodzie jest `select Category`, jest to metoda obsługująca zda-

Listing 5. *Strona ASP.NET do przykładu galeria zdjęć*

```
<html xmlns="http://www.w3.org/1999/xhtml" >

<head runat="server">
   <link href="PhotoGallery.css" rel="Stylesheet" type="text/
                    css" />

    <script type="text/javascript">

        var photoView;

        function pageLoad()
        {
            // pobieramy wszystkie linki z listy  o
                    identyfikatorze categories i przypisujemy
                    im procedurę obsługi zdarzenia
                    kliknięcia linku

            $("#categories a").click(selectCategory);
            photoView = $create(Sys.UI.DataView, {}, {}, {},
                    $get("photoContainer"));
        }

        function selectCategory(e)
        {

            //Zatrzymaj domyśle działanie elementu
            e.preventDefault();

            $("#categories a.selectedCategory").removeClass("s
                    electedCategory");

            $(this).addClass("selectedCategory");

            //Wywołaj usługę web
            PhotoServices.PhotoService.GetPhotos($(this).attr(
                    "categoryid"),getPhotosComplete);
        }

        function getPhotosComplete(photoData)
        {
            photoView.set_data(photoData);
            $("#photoContainer").litebox();
        }

    </script>
</head>
<body>

    <form id="form1" runat="server">
    <div>

        <asp:ScriptManager ID="sm1" runat="server">
            <Scripts>
                <asp:ScriptReference Path="~/Scripts/jquery-
                        1.2.6.js"/>

                <asp:ScriptReference Path="~/Scripts/
                        jquerylitebox.js"/>

                <asp:ScriptReference Path="~/Scripts/Microsoft
                        AjaxTemplates.debug.js" />

            </Scripts>
            <Services>
                <asp:ServiceReference Path="~/
                        PhotoService.svc"/>

            </Services>
        </asp:ScriptManager>

        <%-- Lista kategorii zdjęć --%>
        <asp:ListView ID="ListView1" runat="server" DataSource
                    ID="srcCategories">
            <LayoutTemplate>
                <ul id="categories">
                    <asp:PlaceHolder id="itemPlaceHolder"
                        runat="server"/>
                </ul>
            </LayoutTemplate>
            <ItemTemplate>
                <li> //atrybut categoryid służy do przechowania
                        identyfikatora kategorii zdjęć
                    <a href="Default.aspx" categoryid='<%#
                        Eval("Id") %>'><%# Eval("Name") %></a>
                </li>
            </ItemTemplate>
        </asp:ListView>

        …

        <%-- Kontener na zdjęcia --%>
        <div id="photoContainer" class="sys-template">
            <div>

                <a sys:href="{{Url}}"><img sys:src="{{Url}}"
                        alt="{{Title}}"/></a>
                <br />

                Nazwa: {{Title}}
            </div>
        </div>

    </div>
    </form>
</body>
</html>
```

rzenie kliknięcia linku z nazwą kategorii, w której wywołujemy usługę web, używając obiektu proxy `GetPhotos`, Metodzie `GetPhotos` przekazujemy identyfikator kategorii zdjęć, do wyświetlenia używając składni `$(this).attr(„categoryid”)`, jako drugi parametr definiujemy metodę zwrotną `getPhotosComplete`, która zostanie wywołana po zwróceniu wyniku działania usługi. W metodzie `getPhotosComplete` przypisujemy zwrócony rezultat do kontrolki `DataView`, która jednocześnie wyświetla listę zdjęć zgodnie ze zdefiniowanym szablonem.

Ostatnią komendą jest `$("#photoContainer").litebox()`, która uruchamia wtyczkę litebox, pozwalając przeglądać w efektowny sposób naszą galerię. Dzięki integraji jQuery i nowych funkcjonalności ASP.NET AJAX pisanie aplikacji ajaxowych jest prostsze niż kiedykolwiek wcześniej.

W Sieci

- Oficjalna strona biblioteki jQuery – *Http://www.jquery.com*,
- Strona z wtyczkami dla jQuery – *http://plugins.jquery.com*,
- Strona z dokumentacją ASP.NET AJAX 4.0 –*http://quickstarts.asp.net/previews/ajax/templates/default.aspx*.

Listing 6. *Szablon HTML przypisany do obiektu DataView, używany do dynamicznego generowania interfejsu użytkownika*

```
<div id="photoContainer" class="sys-template">
          <div>
              <a sys:href="{{Url}}"><img sys:src="{{Url}}" alt="{{Title}}" /></a>
              <br /> Nazwa: {{Title}}

          </div>
</div>
```

Podsumowanie

Wielu programistów nie lubi pisać kodu w JavaScript. Proces tworzenia kodu po stronie klienta z użyciem JavaScript wymaga często dużych nakładów pracy, zwłaszcza w bardziej zaawansowanych aplikacjach. Możemy ten proces usprawnić i przyspieszyć używając biblioteki jQuery. Dzięki połączeniu jQuery i nowych funkcjonalności biblioteki ASP.NET AJAX możemy szybko stworzyć prawdziwie interaktywną, zaawansowaną, ajaksową stronę WWW. jQuery nie zastępuje w żadnym wypadku biblioteki ASP.NET AJAX, ale uzupełnia ją o swoje potężne możliwości takie jak efektywne manipulowanie obiektami DOM czy animacje. Niniejszy artykuł nie wyczerpuje bogatej funkcjonalności biblioteki jQuery, ale mamy nadzieję, że zainteresował was na tyle, abyście chcieli zapoznać się bliżej z tą biblioteką i używać jej w swoich projektach.

ŁUKASZ SZEWCZAK

Pracuję jako analityk programista w firmie Risco Software. Z technologią .NET jestem związany od dwóch lat. Jestem członkiem warszawskiej grupy .NET.

Kontak z autorem: lukasz.szewczak@riscosoftware.pl:

R E K L A M A

Aplikacje desktopowe

Ext GWT (GXT) jako środowisko do tworzenia bogatych interfejsów użytkownika w przeglądarce WWW.

Do niedawna w interfejsach opierających się na przeglądarce uzyskanie pełnej funkcjonalności, którą oferowały aplikacje oparte na grubym kliencie, było bardzo trudne, jeśli wręcz nie niemożliwe. Środowisko GWT w połączeniu z biblioteką Ext GWT zmieniają diametralnie ten stan rzeczy. W niniejszym artykule pokażemy jak łatwo jest stworzyć taką aplikację.

Dowiesz się:
- Zdobędziesz podstawową wiedzę o bibliotece GXT i środowisku GWT;
- Jak wykorzystać mechanizm zdarzeniowy i model MVC z biblioteki GXT;
- Jakie są zalety proponowanego rozwiązania.

Powinieneś wiedzieć:
- Jak programować w języku Java;
- Znać podstawy budowy aplikacji sieciowych;
- Orientować się w mechanizmie MVC.

Poziom trudności

Od kilku lat da się zaobserwować bardzo dynamiczny rozwój aplikacji internetowych. Dzieje się tak za sprawą technologii AJAX, która umożliwia asynchroniczną komunikację aplikacji z serwerem. Dzięki niej możemy przeładować wybrany kawałek strony WWW, bez konieczności odświeżania całości. Interfejs użytkownika działa przez to szybciej, płynniej i wygodniej, czyli staje się bardziej *desktopowy*. Technologia ta opiera się na JavaScripcie, wymaga więc od programisty znajomości tego języka. Niestety JS często stwarza problemy z kompatybilnością z różnymi przeglądarkami. Istnieje sporo bibliotek ułatwiających tworzenie przenośnych, interaktywnych komponentów (ang. *widgetów*) ale wszystkie wymagają znajomości JavaScriptu i co jest charakterystyczne dla tego języka, nie rozwiązują trudności związanych z debugowaniem.

Wybawienie

I tu właśnie pojawia się produkt Google rozwiązujący przedstawione problemy. Google Web Toolkit (GWT), bo o nim mowa, to szkielet do tworzenia aplikacji sieciowych. Umożliwia on korzystanie z zalet asynchronicznej komunikacji nie tylko bez znajomości JavaScriptu, ale też i samej technologii AJAX. Wykorzystuje do tego język Java i zwalnia programistę z konieczności dbania o przenośność kodu, gdyż robi to za niego.

GWT kompiluje kod napisany w Javie i tworzy odpowiednie pliki JavaScript, CSS i HTML. Pliki te mają wersje dedykowane do większości popularnych przeglądarek WWW. Znika więc problem związany z kompatybilnością.

GWT daje możliwość przetestowania aplikacji w tak zwanym *hosted mode*. Kod Javy jeszcze przed kompilacją, jest uruchamiany przez JVM, co czyni proces wykrywania błędów łatwiejszym (tak jak w zwykłej Javie).

Ponadto narzędzie Google umożliwia tworzenie testów JUnit, łatwą lokalizację, posiada wsparcie dla historii (przycisk wstecz w przeglądarce) i zbiór podstawowych komponentów. Zbiór ten, mimo iż jest stosunkowo spory, nie wyczerpuje naturalnie wszystkich wymagań stawianych aplikacjom WWW. Stanowi on tylko szkielet, pod dalszą rozbudowę. Dlatego właśnie powstają biblioteki wykorzystujące GWT, które rozszerzają jego możliwości. Jedną z nich jest GWT Ext (GXT).

GWT Ext to biblioteka widgetów napisanych w Javie, które korzystają z wydajności, jaką daje GWT i oferują do tego bogate możliwość konfiguracji. Ponadto posiada ona wbudowaną obsługę tematów (ang. *themes*) za pomocą CSS, wsparcie dla obsługi XML-a, JSON-a i komunikacji z serwerem przez RPC. Ważną rzeczą jest też własny mechanizm MVC czyniący programowanie interakcji z użytkownikiem bardziej przejrzystym, łatwiejszym w zrozumieniu i późniejszym utrzymaniu.

Proponowane combo GWT + GXT jest rozwiązaniem dla tzw. bogatych interfejsów użytkownika (ang. *Rich application client* - RIA). Aplikacja stworzona w ten sposób swą funkcjonalnością praktycznie nie różni się od wersji desktopowej (tzw. gruby klient). Ładowana jest raz, przy wejściu na stronę, po czym działa dokładnie tak jak zwyczajna aplikacja okienkowa uruchomiona na naszym komputerze. Charakteryzuje się przy tym sporą multimedialnością objawiającą się między innymi w obsłudze efektów graficznych, czy mechanizmie Drag&Drop. Rozwiązanie oparte o GXT ma jednak przewagę nad grubym klientem, gdyż eliminuje problem przenośności między systemami operacyjnymi. Do prawidłowego działania aplikacji wystarczy przeglądarka WWW. W ramce wymienione zostały przeglądarki kompatybilne z GXT.

Stworzenie szkieletu projektu

Do stworzenia aplikacji z wykorzystaniem GWT i GWT Ext użyjemy środowiska NetBeans 6.0, które wyręczy nas w wykonywaniu nudnych czynności związanych z przygotowaniem struktury projektu i tworzeniem potrzebnych plików konfiguracyjnych.

Zaczynamy od sprawdzenia, czy mamy zainstalowany plugin GWT4NB. W tym celu klikamy menu *Tools>Plugins>Installed plugin*. Jeśli takowego nie ma na liście, przechodzimy do zakładki *Available plugins*, szukamy, zaznaczamy i klikamy przycisk *install*.

Środowisko GWT w najnowszej wersji ściągamy ze strony:

http://code.google.com/webtoolkit/download.html,

a bibliotekę GWT Ext ze strony:

http://extjs.com/products/gxt/download.php.

Kolejnym etapem jest stworzenie projektu. Wybieramy typ projektu *Web application* z

kategorii *web*. Wersję Javy ustawiamy na *Java EE 5*, a serwer na *Glassfish*, choć równie dobrze może być to każdy inny. W zakładce *frameworks* wybieramy *Google Web Toolkit* i wskazujemy ścieżkę do ściągniętego wcześniej środowiska GWT.

Po stworzeniu projektu musimy jeszcze dodać bibliotekę GWT Ext (plik *<ścieżka do GXT>/gxt.jar*) do naszej aplikacji oraz przegrać standardowy temat graficzny (katalogi *<ścieżka do GXT>/resources/css* i *<ścieżka do GXT>/resources/css*) do podkatalogu *WEB-INF* w projekcie.

W terminologii używanej przez Google, część aplikacji odpowiedzialna za uruchomienie interfejsu użytkownika w przeglądarce, to moduł GWT.

Każdy moduł musi posiadać tzw. punkt wejściowy/startowy (z ang. *entry point*). Jest to klasa, od której zaczyna się ładowanie aplikacji – pewnego rodzaju odpowiednik klasy *Main* z metodą *main()*, od której zaczyna się wykonywanie programu w zwykłej Javie. Klasa taka musi implementować interfejs `com.google.gwt.core.client.EntryPoint` zawierający metodę `public void onModuleLoad()`. Do tej właśnie metody przekazywane jest sterowanie, kiedy użytkownik próbuje załadować naszą aplikację w swojej przeglądarce.

Kolejnym wymaganiem jest obecność deskryptora modułu w strukturze katalogów z kodem źródłowym. Deskryptor ten to plik XML o rozszerzeniu *gwt.xml*, który określa, z jakich modułów środowiska GWT będzie korzystać nasza aplikacja, z jakich dodatkowych bibliotek (w naszym przypadku GWT Ext) jak również, którą klasę traktować jako entry point.

Deskryptor powinien znajdować się w pakiecie nadrzędnym w stosunku do pliku z implementacją entry point. Wszystkie klasy wchodzące w skład modułu, powinny być umieszczone w tym samym pakiecie co punkt startowy, bądź w pakietach podrzędnych. Właśnie ich kod w trakcie kompilacji zostanie przetłumaczony na JavaScript i HTML. W naszym przypadku tworzymy moduł obsługujący interfejs użytkownika. Istnieje jednak możliwość stworzenia modułów bez klasy z entry point, które można potem wykorzystać w innych modułach. Takimi modułami są właśnie biblioteki zewnętrzne jak GXT.

W naszym przypadku poza standardowym odwołaniem do modułu

```
com.google.gwt.user.User
```

w deskryptorze znajdzie się w odwołanie do biblioteki GWT Ext. Przedstawia to Listing 1.

Ostatnim obowiązkowym elementem aplikacji wykorzystującej GWT jest plik HTML, w którym w znacznikach <meta/> i <script/> umieszczone są odwołania do naszego modułu (odwołania wskazują na deskryptor *Main.gwt.xml)*. Jest też deklaracja tematu (ang. *theme*), który będzie używany przez naszą aplikację. U nas będzie to standardowy temat graficzny GXT, który wcześniej skopiowaliśmy do katalogu *WEB-INF*.

Model MVC

Ponieważ chcemy pokazać jak najwięcej możliwości graficznych GXT, by uświadomić, że nie są już one domeną grubych klientów, to w naszej aplikacji skupimy się na stronie klienckiej, odkładając komunikacje z serwerem na dalszy plan.

Aplikacja będzie obsługiwać zapisywanie informacji o osobach i wyświetlać je w tabelce. Podzielimy w tym celu okno główne na zakładki, by dodatkowo prześledzić interakcje między nimi i lepiej przedstawić dostępny w GXT mechanizm zdarzeniowy połączony z modelem MVC. Dodatkowo zademonstrujemy jak prosto i szybko można uzyskać ciekawe efekty.

MVC (*Model View Controller*) to mechanizm pozwalający na podział aplikacji na trzy współpracujące ze sobą warstwy – model danych, widok reprezentujący interfejs użytkownika i kontroler zarządzający komunikacją między poszczególnymi warstwami. Stworzymy taki model dla zakładki wyświetlającej dane o osobach. Pierwszym krokiem jest określenie danych, które będziemy wyświetlać w tabelce. Przyjmijmy, że będą to: imię, nazwisko, data urodzenia i ilość dzieci. Ilość dzieci służyć będzie do zobrazowania konfigurowalnej prezentacji danych w komórce tabeli, o czym w dalszej części artykułu.

Tabelka z danymi

Tworzymy plik `pl.com.xqg.client.beans.UserDataBeanGxt`. Jest to bean reprezentujący osobę zawierający settery i gettery dla przedstawionych atrybutów. Różnica między standardowym beanem Java (ang. *JavaBean*) polega na tym, że dziedziczy on po klasie

```
com.extjs.gxt.ui.client.data.BaseModel,
```

która implementuje interfejs

```
com.extjs.gxt.ui.client.data.ModelData.
```

Interfejs ten jest wymagany przez większość widgetów w GXT. W naszym beanie

Listing 1. *Deskryptor aplikacji Main.gwt.xml*

```
<?xml version="1.0" encoding="UTF-8"?>
<module>
   <!-- rdzeń biblioteki GWT -->
   <inherits name="com.google.gwt.user.User"/>
   <!-- biblioteka GXT -->
   <inherits name="com.extjs.gxt.ui.GXT"/>
   <!-- klasa implementująca entry point w naszym module -->
   <entry-point class="pl.com.xqg.client.MainEntryPoint"/>
</module>
```

Listing 2. *Plik startowy HTML*

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
   <head>
      <!-- odwołanie wskazujące nasz moduł (plik Main.gwt.xml) -->
      <meta name='gwt:module' content='pl.com.xqg.Main=pl.com.xqg.Main'>
      <title>Main</title>
      <!-- standardowy temat graficzny-->
      <link rel="stylesheet" type="text/css" href="css/ext-all.css" />
   </head>
   <body>
      <!-- odwołanie do naszego modułu i JavaScriptu wygenerowanego podczas kompilacji
                     przez GWT -->
      <script language="javascript" src="pl.com.xqg.Main/pl.com.xqg.Main.nocache.js">
                     </script>
   </body>
</html>
```

Przeglądarki obsługiwane przez GWT + GWT Ext

- Internet Explorer 6+,
- Firefox 1.5+ (PC, Mac),
- Safari 3+,
- Opera 9+ (PC, Mac).

atrybuty nie są zapisywane w zmiennych prywatnych, a w mapie klasy nadrzędnej.

Skoro mamy już model danych, to potrzebujemy widoku, który nam je zaprezentuje. W tym celu tworzymy klasę `pl.com.xqg.client.ui.tabs.UserTableTab`, która będzie zakładką zawierającą tabelkę z osobami. Plik ten został przedstawiony na listingu nr 3.

W GXT umiejscowienie widgetów opiera się głównie na różnego rodzaju kontenerach lub panelach. Te drugie, to bardziej rozbudowane wersje tych pierwszych. Obu elementom można przypisywać sposób rozmieszczenia elementów (ang. *layout*). Biblioteka posiada ich sporo, my jednak będziemy wykorzystywać głównie

`com.extjs.gxt.ui.client.widget.layout.FitLayout`,

który to rozciąga element wewnątrz kontenera na cały jego obszar. Zakładka będzie więc kontenerem, w którym umieścimy panel z tabelką. Właśnie dlatego `PeopleTableTab` dziedziczy po kontenerze

`com.extjs.gxt.ui.client.widget.LayoutContainer`.

Jest to dość *lekki* kontener, który umożliwia, mimo wszystko, ustawienie layoutu – w naszym przypadku FitLayout.

By stworzyć tabelkę, trzeba zdefiniować jej kolumny. Służy do tego klasa

`com.extjs.gxt.ui.client.widget.grid.ColumnConfig`.

Definiujemy w niej jakiemu polu z beana z danymi odpowiada kolumna, jej nazwę oraz rozmiar. Można też ustawić kilka innych parametrów, jak choćby, czy kolumna ma być sortowalna, widoczna/niewidoczna, automatycznie

Listing 3. *Zakładka z tabelką użytkowników*

```
public class UserTableTab extends LayoutContainer {

   //zbiór danych (ang.store) przechowujący dane użytkowników
   private ListStore peopleStore;

   public UserTableTab() {

      ContentPanel mainPanel = new ContentPanel();
      mainPanel.setHeading("Przykładowa tabelka, którą można
                     przesuwać");
      mainPanel.setSize(500,500);

      List<ColumnConfig> configs = new ArrayList<ColumnConfig>();
                        //lista z definicjami kolumn

       //definiujemy kolumny
      configs.add(new ColumnConfig(UserDataBeanGxt.NAME, "Imie",
                     100));
      configs.add(new ColumnConfig(UserDataBeanGxt.SURENAME,
                     "Nazwisko", 100));
      configs.add(new ColumnConfig(UserDataBeanGxt.BIRTH_DATE,
                     "Data urodzenia", 80));

      //definiujemy kolumnę wyświetlającą liczbę dzieci.
      ColumnConfig childrensColumn = new ColumnConfig(UserDataBean
                     Gxt.CHILDRENS_NO, "Liczba dzieci", 120);
       //dla tej kolumny definiujemy nasz własny renderer, ktory
                     będzie wyświetlał nie liczby, a tekst
      childrensColumn.setRenderer(new GridCellRenderer<UserDataB
                     eanGxt>() {

      public String render(UserDataBeanGxt model, String arg1,
                     ColumnData arg2, int arg3, int arg4,
                     ListStore<UserDataBeanGxt> arg5) {
            Number val = model.getAge();     //model zawiera
                     model reprezentujący wiersz w tabeli
            String renderStyle = "green";
            String renderValue = "Mało albo wcale";
                    //jeśli liczba dzieci jest większa od 3
wyświetlamy czerwony napis Dużo

             if (val != null && val.longValue() > 3) {
               renderStyle = "red";
               renderValue = "Dużo";
            }
            //to zostanie wstawione w komórkę tabeli
            return "<span style='color:" + renderStyle + "'>" +
                     renderValue + "</span>";
            }
      });
      configs.add(childrensColumn);

       //inicjujemy zasób danych dla naszej tabelki jakimiś
                     przykładowymi danymi
      peopleStore = new ListStore();

       //tworzymy model kolumn z naszych definicji
      ColumnModel cm = new ColumnModel(configs);

       //Tworzymy tabelkę korzystając z wcześniej stworzonych
                     zasobu danych i definicji kolumn
      Grid exapmleTable = new Grid(store, cm);
      exapmleTable.setStyleAttribute("borderTop", "none");
      exapmleTable.setAutoExpandColumn("name");
      exapmleTable.setBorders(true);

      mainPanel.add(exapmleTable);

       //ustawiamy jeszcze, by panel z tabelką miał możliwość
                     przeciągania
      Draggable d = new Draggable(mainPanel);
      d.setUseProxy(true);
      this.add(mainPanel);
   }

    /**
    * Metoda, która dodaje użytkownika do tabelki.
    * wyświetla też komunikat informujący, że użytkownik został
                     dodany.
    *
    * W naszym tutorialu metoda jest wywoływana przez widok
                     UserTableView
    *
    * @param userDataBeanGxt bean określający dane nowego
                     użytkownika
    */
   public void addNewUser(UserDataBeanGxt userDataBeanGxt) {
      peopleStore.add(userDataBeanGxt);
      MessageBox.alert("Informacja", "Dane zostały dodane do
                     tabelki", null);
   }
}
```

się rozszerzać itd. Nam wystarczy, że standardowo kolumnę można usuwać w trakcie działania aplikacji, przesuwać i sortować w obu kierunkach. Wszystkie te opcje są standardowe (dają się jednak wyłączyć) więc nie musimy nic implementować.

Omówienia wymaga kolumna wyświetlająca liczbę dzieci. Postanowiliśmy wyświetlać ją inaczej niż standardowo (liczbowo). Będzie to więc u nas zielony napis *mało lub wcale* w przypadku, gdy osoba ma mało dzieci lub czerwony napis *dużo*, gdy dzieci jest już sporo. By zdefiniować niestandardowe renderowanie komórki tabeli, trzeba do definicji kolumny dodać swój renderer. Reprezentuje go interfejs

`com.extjs.gxt.ui.client.widget.grid.GridCellRenderer`,

którego metodę

```
public String render(DataModel, String, ColumnData, int, int, Store)
```

trzeba zaimplementować. Jak się można łatwo domyśleć, metoda ta jest wywoływana podczas próby renderowania danej kolumny tabeli dla każdego z wiersza osobno. Interesuje nas pierwszy parametr, który reprezentuje model danych dla danego wiersza i wartość zwracana przez metodę. Na podstawie modelu danych z parametrów wejściowych metody, definiujemy i zwracamy zmienną typu string. Jest ona kodem HTML, który zostanie wstawiony do właśnie renderowanej komórki tabeli.

Listing 4. *Zakładka z formularzem do dodawania osoby*

```
public class PersonFormTab extends LayoutContainer {

   public PersonFormTab() {
      this.setLayout(new FitLayout());    //rozciągamy na całe
                     okienko
      final ContentPanel mainPanel = new ContentPanel();
      mainPanel.setHeaderVisible(false);

      //tworzymy panel z formularzem
      final FormPanel formPanel = new FormPanel();
      formPanel.setHeading("Formularz zgłoszeniowy");
      formPanel.setFrame(true);

       final TextField<String> nameTF = new TextField<String>();
      nameTF.setFieldLabel("Imie");

      //tworzymy textfielda na nazwisko - nie pozwalamy, by pole
                     było puste
      final TextField<String> surenameTF = new
                     TextField<String>();
      surenameTF.setFieldLabel("Nazwisko");
      surenameTF.setAllowBlank(false);

      final NumberField childrensNF = new NumberField();
      childrensNF.setFieldLabel("Liczba dzieci");

      final DateField birthDateDF = new DateField();
      birthDateDF.setFieldLabel("Data urodzenia");

      formPanel.add(nameTF);
      formPanel.add(surenameTF);
      formPanel.add(childrensNF);
      formPanel.add(birthDateDF);

      //przycisk do czyszczenia formularza
      formPanel.addButton(new Button("Wyczyść") {

         @Override
         protected void onClick(ComponentEvent arg0) {   //
                     metoda uruchamiana po kliknięciu przycisku
            super.onClick(arg0);
             //czyścimy wszystkie pola formularza
            nameTF.setValue(null);
            surenameTF.setValue(null);
            childrensNF.setValue(null);
            birthDateDF.setValue(null);
         }
      });

       //pobieramy dispatcher z GXT
      final Dispatcher dispatcher = Dispatcher.get();
            //przycisk dodający użytkownika, który będzie
inicjował zdarzenie ADD_USER_TO_TABLE

      formPanel.addButton(new Button("Dodaj do tabelki") {

         @Override
         protected void onClick(ComponentEvent arg0) {   //
                     metoda uruchamiana po kliknięciu przycisku
                     dodaj
            super.onClick(arg0);
            //trochę validacji.
                  //ponieważ wcześniej ustawiliśmy na polach,
w jakim przypadku są poprawne,

            //teraz wystarczy tylko wywołać funkcję isValid()
             if (!surenameTF.isValid()) {
               MessageBox.alert("Błędnie wypełnione dane",
                     "Muszisz podać nazwisko", null);
              return;
            }
            if (!birthDateDF.isValid()) {
            MessageBox.alert("Błędnie wypełnione dane", "Pole
                     data jest źle wypełnione", null);
               return;
            }

             //Wszystko ok.
            //pobieramy dane do beana

            UserDataBeanGxt userDataBeanGxt = new UserDataBeanGxt
                     (nameTF.getValue(), surenameTF.getValue(),
                     childrensNF.getValue(),
                     birthDateDF.getValue());
             //inicjujemy zdarzenie ADD_USER_TO_TABLE,
                     do ktorego dołączamy bean z nowym
                     użytkownikiem.
            dispatcher.dispatch(TutorialEvents.ADD_PERSON_TO_
                     TABLE.toInt(), userDataBeanGxt);

            }
         });
      mainPanel.add(formPanel);

      this.add(mainPanel);
   }
}
```

Tabelka w GXT może korzystać z danych, które dostarcza jej *zasób danych* (ang. *store*). Zasób ten stanowi u nas zmienna *peopleStore* typu

`com.extjs.gxt.ui.client.store.ListStore<UserDataBeanGxt>`.

Przechowuje ona informacje o osobach, które są wyświetlane w tabelce.

Pozostało nam już tylko zadeklarować model kolumn z definicji, które stworzyliśmy wcześniej i stworzenie gridu będącego naszą tabelką osób. Do jego inicjalizacji używamy modelu kolumn i *zasobu danych*.

Na koniec dodajemy naszą tabelkę do nowo stworzonego panelu a całość do zakładki.

W klasie `PeopleTableTab` zaimplementowaliśmy dodatkowo metodę `public void addNewUser(UserDataBeanGxt)`, która będzie wywoływana przez kontroler obsługujący tabelkę. Jej zadaniem jest dodanie beana z nową osobą do zasobu tabelki i wyświetlenie informacji dla użytkownika, że operacja przebiegła pomyślnie.

Mechanizm zdarzeniowy

GXT posiada mechanizm zdarzeniowy wspierający model MVC. Zdarzenia w tym systemie reprezentowane są przez liczby całkowite. Zalecane jest jednak stworzenie do tego celu klasy typu enum z odpowiednimi metodami zwracającymi indeksy do poszczególnych pól, co też zrobimy w przypadku naszej aplikacji. Tworzymy zatem prostą klasę `pl.com.xqg.client.mvc.TutorialEvents`, w której implementujemy następującą metodę:

```
public int toInt() {
 return this.ordinal();
}
```

Najważniejszą częścią mechanizmu zdarzeniowego jest dyspozytor (ang. *dispatcher*), który zarządza zdarzeniami. Użytkownik podczas interakcji z aplikacją może wygenerować zdarzenie. Jest ono dostarczane przez dyspozytora do nasłuchujących kontrolerów, które odpowiednio na nie reagują. Dyspozytor jest zaimplementowany w klasie

`com.extjs.gxt.ui.client.mvc.Dispatcher`.

Klasa ta umożliwia zarówno wygenerowanie zdarzenia wraz z dołączonym do niego modelem danych, jak i zarejestrowanie w systemie kontrolera nasłuchującego.

W naszej aplikacji zakładka z formularzem będzie generować zdarzenie, które zostanie przekazane dalej do kontrolera związanego z tabelką osób. Kontroler obsłuży zdarzenie, przekazując je do odpowiedniego widoku, w którym nastąpi odświeżenie zawartości tabelki. Zakładka dodająca użytkownika jest reprezentowana przez klasę `pl.com.xqg.client.ui.tabs.PersonFormTab` przedstawioną na listingu 4.

Zakładka ta, tak jak poprzednia, rozszerza klasę `LayoutContainer`. Zdefiniowany jest w niej panel `com.extjs.gxt.ui.client.widget.form.FormPanel` reprezentujący w GXT formularz HTML. Do tego panelu dodajemy widgety odpowiadające atrybutom osoby i ich typom; mamy więc:

`com.extjs.gxt.ui.client.widget.form.NumberField` – reprezentujący pole numeryczne,

`com.extjs.gxt.ui.client.widget.form.TextField` – reprezentujący pole tekstowe,

`com.extjs.gxt.ui.client.widget.form.DateField` – reprezentujący pole z datą i kalendarzem do jej wyboru.

Wszystkie wyżej wymienione widgety umożliwiają łatwe ustawienie walidacji, czy to w postaci wyrażeń regularnych (ang. *regex*), czy w postaci własnoręcznie zdefiniowanych walidatorów. Pokazujemy to na przykładzie pól związanych z datą i nazwiskiem. Zakładamy, że pole nazwisko będzie obowiązkowe, a data musi być poprawnie uzupełniona. W tym celu definiujemy przycisk *Dodaj* klasy `com.extjs.gxt.ui.client.widget.button.Button`, w którym nadpisujemy metodę `onClick()`. Metoda ta jest uruchamiana w momencie kliknięcia przycisku. Sprawdzamy w niej czy obydwa pola są poprawnie wypełnione poprzez bardzo proste wywołanie metody `isValid()`. Jeśli nie spełniają one stawianych im warunków, wyświetlamy odpowiednie komunikaty. Warto zwrócić uwagę, że GXT automatycznie podkreśla i wyświetla dymki w przypadku pól, do których wpisujemy złe wartości. Oczywiście zachowanie takie jest w pełni konfigurowalne.

W przypadku poprawnego wypełnienia danych osobowych, następuje odwołanie do dyspozytora i wygenerowanie zdarzenia `ADD_PERSON_TO_TABLE` z beanem reprezentującym nową osobę.

W razie problemu z kompilacją

Podczas kompilacji i próby debugowania przy wykorzystaniu pluginu GWT4NB mogą wystąpić problemy z pamięcią. Objawiają się one komunikatami *Java heap error* lub *Out of memory exception* podczas kompilacji. Rozwiązanie tego problemu jest proste - wymaga dodania parametru `maxmemory = "512M"` do węzłów *<java/>* w pliku *build-gwt.xml* (plik konfiguracyjny pluginu GWT4NB znajdujący się w katalogu *nbproject* projektu). Uwaga, jeśli użyjemy opcji *clean* lub *clean&build* na projekcie, to niestety czynność z dodaniem parametru należy powtórzyć.

Listing 5. *Kontroler związany z tabelką z osobami*

```
public class PeopleTableController extends Controller {

   PeopleTableView userTableView;

   public PeopleTableController() {
      registerEventTypes(TutorialEvents.ADD_PERSON_TO_TABLE.toInt()); //ustalamy na
                        jakie zdarzenia ma reagować nasz kontroller
   }

   @Override
   public void handleEvent(AppEvent event) {
      switch (TutorialEvents.fromInt(event.type)) {
         default:
            forwardToView(userTableView, event);
      }
   }

   /**
   * Metoda wywoływana automatycznie.
   * tu powinnien być stworzony widok.
   */
   public void initialize() {
      super.initialize();
      userTableView = new PeopleTableView(this);
   }
}
```

Kontroler i widok

Ostatnią częścią układanki jest stworzenie kontrolera i widoku. Kontroler został przedstawiony na listingu 5.

Zadaniem kontrolera jest nasłuchiwanie na zdarzenie `ADD_PERSON_TO_TABLE`, jego obsługa i przekazanie sterowania do widoku. W przypadku naszej nieskomplikowanej aplikacji obsługa sprowadza się do przekierowania zdarzenia wraz z danymi dalej do widoku. Dzieje się to w metodzie `handleEvent()`, która musi zostać zaimplementowana. Wymusza to klasa `com.extjs.gxt.ui.client.mvc.Controller`, w której jest ona zdefiniowana jako metoda abstrakcyjna. Ta klasa musi być klasą dziedziczoną przez każdy kontroler.

Zazwyczaj, to właśnie w kontrolerze następuje przetwarzanie danych, pobranie ich, bądź wysłanie z/do serwera w celu zapisu w bazie itd. To jest miejsce gdzie należy takie rzeczy implementować. Dla tych, którzy nie mieli styczności z czystym GWT, może wydać się to oczywiste i nierewolucyjne. Ci jednak, którzy mieli okazje w nim programować, wiedzą, na czym polega udręka i bałagan związany z niekontrolowanymi wywołaniami serwera umieszczonymi w warstwie prezentacji danych. Objawia się tu kolejna przewaga biblioteki GXT nad standardowymi rozwiązaniami GWT.

W kontrolerze należy także zaimplementować metodę `void initialize()`, w której zostanie utworzony odpowiadający mu widok. Metoda ta zostanie wywołana podczas rejestrowania kontrolera u dyspozytora.

Widok, podobnie jak kontroler, musi dziedziczyć z odpowiedniej klasy (`com.extjs.gxt.ui.client.mvc.View`). Implikuje to implementację konstruktora biorącego jako parametr obiekt typu `Controller` oraz metody `handleEvent(...)`. W metodzie tej obsługiwane są zdarzenia przekazane przez kontroler. W naszym przypadku, jest to pobranie z rejestru zakładki z tabelką oraz wywołanie na niej metody dodającej osobę z odpowiednimi danymi.

```
PeopleTableTab userTableTab = Registry.get("userTableTab");

//pobieramy zakładkę z tabelką

userTableTab.addNewUser((UserDataBeanGxt)event.data);

//informujemy ją, że trzeba dodać użytkownika
```

Składamy wszystko w jedną całość

Czym jest rejestr ? Jest to kolejny wewnętrzny mechanizm ułatwiający tworzenie aplikacji w GXT. Umożliwia on zapisanie obiektu pod określoną nazwą i odwołanie się do niej w dowolnym momencie.

Listing 6. *Klasa startowa aplikacji*

```
public class MainEntryPoint implements EntryPoint {

   public MainEntryPoint() {
   }

   /**
   * Metoda ta jest wywoływana automatycznie, podczas ładowania aplikacji
   * po stronie użytkownika w jego przeglądarce
   */
   public void onModuleLoad() {

      final Dispatcher dispatcher = Dispatcher.get(); //Dispatcher, zarządzający
                     zdarzeniami w mechaniźmie MVC z GXT.
      dispatcher.addController(new PeopleTableController()); //rejestrujemy kontroler
                     naszej tabelki z użytkownikami


      Viewport viewport = new Viewport();              //view port po to, by obsłużyć
                     zmianę rozmiaru okienka
      viewport.setLayout(new FitLayout());


      TabPanel tabs = new TabPanel();

      //zakładka z formularzem użytkownika
      TabItem singleTab = new TabItem();
      singleTab.setText("Dodaj osobę - formularz");

      singleTab.setClosable(false);
      singleTab.setScrollMode(Scroll.AUTO);

      singleTab.setLayout(new FitLayout());
      singleTab.add(new PersonFormTab());
      tabs.add(singleTab);

      //zakładka z tabelką z użytkownikami
      singleTab = new TabItem();
      singleTab.setText("Wyświetl osoby - tabelka");

      singleTab.setClosable(false);
      singleTab.setScrollMode(Scroll.AUTO);

      singleTab.setLayout(new FitLayout());
      PeopleTableTab userTableTab = new PeopleTableTab();
      Registry.register("userTableTab", userTableTab);
      singleTab.add(userTableTab);
      tabs.add(singleTab);

      //zakładka z panelami pokazująca kilka efektów graficznych
      singleTab = new TabItem();
      singleTab.setText("Kilka efektów - demo");
      singleTab.setClosable(false);
      singleTab.setScrollMode(Scroll.AUTO);
      singleTab.setLayout(new FitLayout());
      singleTab.add(new FXTab());

      tabs.add(singleTab);

      viewport.add(tabs);
      RootPanel.get().add(viewport);
      }
```

W przypadku naszej aplikacji zapisanie tabelki do rejestru nastąpiło w klasie implementującej entry point. Klasę tę przedstawia listing nr 6.

W omówionej już metodzie `onLoad()` tworzymy kontroler i rejestrujemy go u dyspozytora.

Następnie deklarujemy tak zwany viewport, który jest kontenerem dopasowującym swe rozmiary do okna przeglądarki. Jest to o tyle wygodne, że podczas zmiany rozmiaru okna, viewport automatycznie poprzesuwa znajdujące się w nim komponenty.

Kolejnym krokiem jest stworzenie panelu z zakładkami

```
com.extjs.gxt.ui.client.widget.TabPanel
```

i kolejnych zakładek reprezentowanych w GXT przez obiekty typu

```
com.extjs.gxt.ui.client.widget.TabItem.
```

Do każdego takiego obiektu przed dodaniem do panelu z zakładkami dodajemy omawiane już wcześniej klasy z formularzem i z tabelką z osobami.

Nie wspomnieliśmy jeszcze o trzeciej zakładce, która została dodana do naszej aplikacji. Implementuje ją klasa `pl.com.xqg.client.ui.tabs.FXTab`. Pokazuje ona jak w prosty sposób stworzyć panele, które da się rozciągać, przesuwać i stosować na nich efekty specjalne. Efekty specjalne są tworzone na referencji do obiektów typu `com.extjs.gxt.ui.client.core.El`, a konfigurowane przez klasę `Com.extjs.gxt.ui.client.fx.FxConfig`. W GXT każdy widget posiada metodę zwracającą obiekt typu `El`. Nie jesteśmy więc ograniczani w efektach specjalnych tylko do paneli.

Na tym kończy się tworzenie naszej aplikacji. Efekt jej działania został przedstawiony na rysunku nr 1.

Best practices i podsumowanie

Na koniec zwracamy uwagę na kilka elementów, które zostały pominięte w prezentowanym programie. Jednym z nich jest dobry nawyk nadawania *id* każdemu elementowi. Ułatwia to późniejsze debugowanie i analizę wygenerowanego HTML-a oraz dodawanie nowych stylów w CSS.

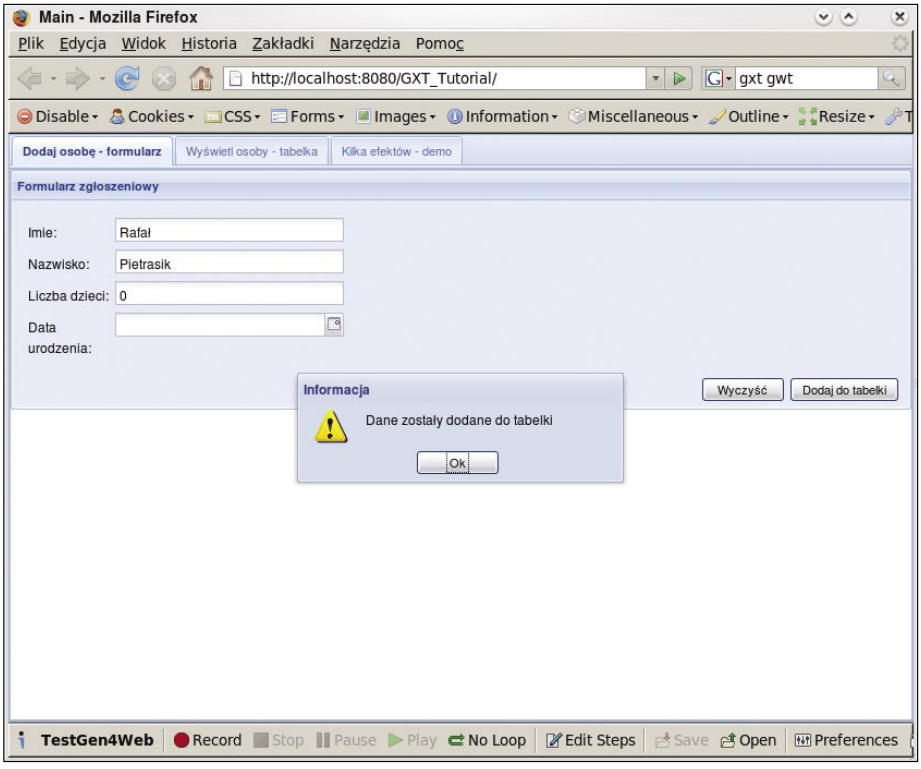
Kolejnym jest rozważne tworzenie widgetów. Należy pamiętać, że kod Java jest tłumaczony do JavaScriptu i może powodować znaczne spowolnienie przeglądarki lub w niektórych skrajnych przypadkach powodować problemy z pamięcią. Powinniśmy starać się tworzyć komponenty jak najbardziej reużywalne, eliminujące konieczność tworzenia ich od nowa przy każdym wywołaniu. Zamiast inicjowania/wypełniania (nie tworzenia) elementów w konstruktorze, powinno się tworzyć oddzielne metody. Na przykład: przypuśćmy, że po kliknięciu wiersza tabelki z osobami chcemy wyświetlić okienko z formularzem. Lepiej jest mieć jedno takie okienko, które będziemy wyświetlać i ukrywać, niż tworzyć je za każdym razem od nowa.

W prezentowanej aplikacji zakładki zostały stworzone w klasie entry point, by zobrazować mechanizm rejestru. Eleganckim rozwiązaniem byłoby jednak stworzenie każdej z zakładek, a przynajmniej tej z tabelką w odpowiadającym jej widoku.

Ostatnią rzeczą, na którą warto zwrócić uwagę jest możliwość korzystania z hosted mode, czyli ze specjalnego trybu ułatwiającego debugowanie. W środowisku NetBeans 6.0 w połączeniu z pluginem GWT4NB jest to o tyle łatwe, że wystarczy kliknąć przycisk *DEBUG*. Otworzą się nam wtedy dwa dodatkowe okna. W jednym będziemy mogli obserwować wynik działania aplikacji, a w drugim będą informacje o wyjątkach i logowane komunikaty. Hosted mode znacznie ułatwia pracę z GWT, bo po każdej zmianie w kodzie nie musimy kompilować wszystkiego od nowa – wystarczy, że odświeżymy okienko z aplikacją (klawisz F5 lub przycisk *Odśwież*).

Zachęcamy do dalszego zagłębienia się w temat GWT + GXT, gdyż połączenie to oferuje znacznie większe możliwości niż prezentowane w niniejszym artykule. Polecamy stronę projektu, szczególnie demo z przykładami. Warto też zaznaczyć, że biblioteka GXT posiada rozbudowaną dokumentację w postaci Javadoców oraz bogatą społeczność reprezentowaną przez forum na stronach projektu.

GWT nie rozwiązuje wszystkich problemów w aplikacjach opartych na przeglądarce. Ułatwia jednak proces ich tworzenia i stanowi silną konkurencję dla aplikacji desktopowych. Konkurencyjność tego rozwiązania zwiększa się z dnia na dzień. GWT i GXT są nieustannie rozbudowywane i ulepszane w oparciu o zgłoszenia pochodzące bezpośrednio od programistów. Stają się przez to coraz lepiej dostosowane do aktualnych wymagań i potrzeb. Niestety nie ma róży bez kolców. Wykorzystanie przeglądarki i obsługiwanego przez nie języka - JavaScriptu, czyni rozwiązania oparte o GWT wolniejszymi od normalnych interfejsów graficznych grubych klientów. Wszędzie jednak tam, gdzie wydajność nie ma głównego znaczenia, GWT z masą swoich zalet jest *prawie idealnym* środowiskiem dla naszej aplikacji.



Rysunek 1. *Działająca zakładka z formularzem osobowym*

W Sieci

- Strona projektu GWT Ext – *http://extjs.com/products/gxt/*,
- Strona środowiska GWT – *http://code.google.com/webtoolkit/*.

AUTOR: RAFAŁ PIETRASIK

Absolwent Politechniki Warszawskiej, od kilku lat zajmuje się pisaniem aplikacji sieciowych i projektowaniem interfejsów użytkownika. Pracuje w firmie XQG Software.

Kontakt z autorem: rafal@xqg.com.pl

Biblioteki/Programy

Python dla S60
Biblioteka niezbędna do uruchamiania aplikacji i skryptów napisanych w języku Python dla systemu Symbian.

Android SDK 1.0
Wersja SDK 1.0 dla systemu Google – Android. Każdy programista, który się powstrzymywał, od pobrania środowiska developerskiego dla Android z winy niedociągnięć, teraz nie ma wymówek.

Wszystko jest dostępne, działa i zostało przetestowane przez wielu programistów. Miłego kodowania.

Eclipse Ganymede SR1 + EclipseME
Eclipse to jedno z dwóch najczęściej wykorzystywanych IDE przez programistów piszących mobilne aplikacje w Javie ME.

Netbeans 6.5
Netbeans 6.5 to IDE wykorzystywane przez programistów piszących mobilne aplikacje w Javie ME

Materiały

GPS z .NET Compact Framework
Do artykułu dołączony kod źródłowy omawianej aplikacji i skompilowana aplikacja.

Aplikacje desktopowe w przeglądarce www
Do niedawna w interfejsach opierających się na przeglądarce uzyskanie pełnej funkcjonalności, które oferowały aplikacje oparte na grubym kliencie, było bardzo trudne, jeżeli wręcz nie niemożliwe. Środowisko GWT w połączeniu z biblioteką Ext GWT zmieniają diametralnie ten stan rzeczy.

PS Mobile
PS Mobile to system wspomagania zbierania zamówień przez mobilnych przedstawicieli handlowych – PreSelling-u. Materiały prezentują możliwości systemu.

Inne
Pozostałe programy przydatne przy programowaniu urządzeń mobilnych.

Redakcja nie udziela pomocy technicznej w instalowaniu i użytkowaniu programów zamieszczonych na płytach CD-ROM dostarczonych razem z pismem.

Jeśli nie możesz odczytać zawartości płyty CD, a nie jest ona uszkodzona mechanicznie, sprawdź ją na co najmniej dwóch napędach CD. W razie problemów z płytą, prosimy pisać pod adres: *cd@softtware.com.pl*

Wykładnicze mapy cieni

Charakterystyka i implementacja algorytmu miękkich cieni w grafice komputerowej czasu rzeczywistego

Mapy cieni są najczęściej wykorzystywaną techniką cieniowania w grach komputerowych. W niniejszym artykule postaram się przybliżyć technikę map cieni oraz zaimplementować algorytm miękkich cieni przy wykorzystaniu HLSL i biblioteki DirectX 9.0.

Dowiesz się:
- Co to są mapy cieni i jak one działają;
- Jak zaimplementować algorytm miękkich cieni.

Powinieneś wiedzieć:
- Posiadać znajomość algebry z zakresu przekształceń geometrycznych w 3D;
- Znać podstawy programowania w języku HLSL oraz DirectX w wersji 9.0;
- Znać podstawy programów cieniujących geometrię.

Poziom trudności

Realistyczne obrazy tworzone za pomocą grafiki komputerowej są ważnymi narzędziami w takich dziedzinach jak: nauka, rozrywka, symulacja, projektowanie, czy marketing. Wyznaczanie zacieniowanych powierzchni sceny wirtualnej jest nieodzownym elementem aplikacji graficznych czasu rzeczywistego (tj. gry komputerowe, symulatory lotu etc.). Cienie dodają wrażenia głębi obrazu i sprawiają, iż uzyskany obraz staje się bardziej fotorealistyczny. Najczęściej wykorzystywaną techniką wyznaczania cieni w grach komputerowych jest technika map cieni (lub mapowania cieni, ang. *shadow mapping*). Obecnie istnieje wiele modyfikacji algorytmu mapowania cieni. W niniejszym artykule postaram się przybliżyć nową technikę cieniowania, bazującą na algorytmie map cieni, umożliwiającą uzyskanie miękkich cieni dla zarówno prostych, jak i złożonych scen wirtualnych. Zaczniemy od wyjaśnienia, czym są mapy cieni, omówimy algorytm mapowana cieni w wersji podstawowej, a następnie przyjrzymy się prostej modyfikacji algorytmu, wprowadzając wykładnicze mapy cieni.

Wprowadzenie

Pojęcie map cieni zostało wprowadzone przez Lance Williama w 1978 roku w dokumencie zatytułowanym *Casting curved shadows on curved surfaces*. Od tego czasu technika ta stosowana jest zarówno w grafice czasu rzeczywistego, jak i w grafice prerenderowanej wykorzystywanej w branży filmów animowanych (np. Toy Story, Wall-E).

Mapy cieni są jednym z wielu algorytmów cieniowania w grafice komputerowej. Do najważniejszych zalet map cieni należy zaliczyć:

- wydajne dla dynamicznych i złożonych scen wirtualnych,
- algorytm nie wymaga skomplikowanych obliczeń i jest łatwy w implementacji,
- do przechowywania informacji o cieniach dla danego światła potrzebna jest tylko jedna tekstura,
- istnieje wiele modyfikacji algorytmu, które umożliwiają uzyskanie efektu realistycznie wyglądających miękkich cieni.

Niestety obok szerokiej listy zalet mapy cieni posiadają również wady:

- dokładność cieni jest ograniczona przez rozdzielczość mapy cieni,
- efekt aliasingu – zwłaszcza dla map cieni o niskiej rozdzielczości,
- obraz sceny musi zostać wygenerowany osobno dla każdego światła kierunkowego.

Pomimo wymienionych wyżej wad, mapy cieni są najczęściej wykorzystywaną i bardzo intensywnie rozwijaną techniką cieniowania w grafice komputerowej. Mała dokładność cieni oraz efekt aliasingu mogą zostać zniwelowane poprzez zwiększenie rozdzielczości lub przez szereg rozmaitych procesów związanych z filtrowaniem mapy cieni, tj. technika PCF (ang. *percentage close filtering*). W grach komputerowych najczęściej stosuje się cienie wygenerowane od światła słonecznego, które reprezentowane jest przez pojedyncze światło kierunkowe, zatem mapy cieni generowane są dla jednego światła na scenie.

Obok map cieni istnieją jeszcze inne techniki wyznaczania cieni na scenie wirtualnej. Do takich metod należą m.in.:

- cieniowanie z wykorzystaniem bufora szablonowego (ang. *stencil shadows*),
- cienie rzutowane (ang. *projected planar shadows*),
- przeliczone mapy światła (ang. *precomputed lightmaps*).

Cienie z wykorzystaniem bufora szablonowego sprawiają trudności przy wymaganym geometrycznym generowaniu obszarów cienia (ang. *shadow volumes*). Dodatkowo krawędzie cienia wygenerowane tą techniką są ostre i nienaturalne. Cienie rzutowane z kolei działają poprawnie jedynie dla płaskich powierzchni, co jest dość znaczącym ograniczeniem. Mapy światła, oprócz miękkich cieni, mogą przechowywać także informacje o przeliczonym oświetleniu pośrednim (tzn. uwzględniającym również światło odbite od innych powierzchni na scenie (ang. *global illumination*), jednakże wymagają one sporych nakładów obliczeniowych i nie są przystosowane do scen dynamicznych.

Jak działają mapy cieni?

Czym jest cień? Otóż cieniem będziemy nazywać obszar, do którego nie dociera światło na skutek przeszkód ustawionych na drodze promieni świetlnych, nieprzepuszczających światła. Wyobraźmy sobie scenę, na której znajdują się różne obiekty i jedno źródło światła (Rysunek 1.).W jaki sposób sprawdzić, czy dany ele-

ment sceny jest w cieniu? Intuicja podpowiada, iż należałoby sprawdzić geometrycznie czy na drodze od rozpatrywanego elementu sceny do źródła światła znajduje się jakiś inny element sceny wirtualnej blokujący promień światła.

Innymi słowy – wszystkie fragmenty obiektów widoczne z punktu widzenia światła są oświetlone, natomiast pozostałe elementy (schowane za obiektami oświetlonymi) będą znajdowały się w cieniu. Na tej właśnie obserwacji bazuje algorytm mapowania cieni. W pierwszej kolejności scena wirtualna jest renderowana z punktu widzenia światła do mapy głębokości (ang. *depth map*), którą będziemy określać mianem mapy cieni (ang. *shadow map*). Mapa cieni zawiera informację o odległości do obiektów widzianych z punktu widzenia światła i jest najczęściej przechowywana w postaci tekstury o formacie zmiennoprzecinkowym.

Następnie renderujemy scenę z punktu widzenia obserwatora, jednocześnie wykorzystując mapę cieni do porównania, czy dany obszar jest widoczny z punktu widzenia światła, czy też nie. W ten sposób określamy, które obszary sceny wirtualnej znajdują się w cieniu. Algorytm składa się jedynie z dwóch kroków:

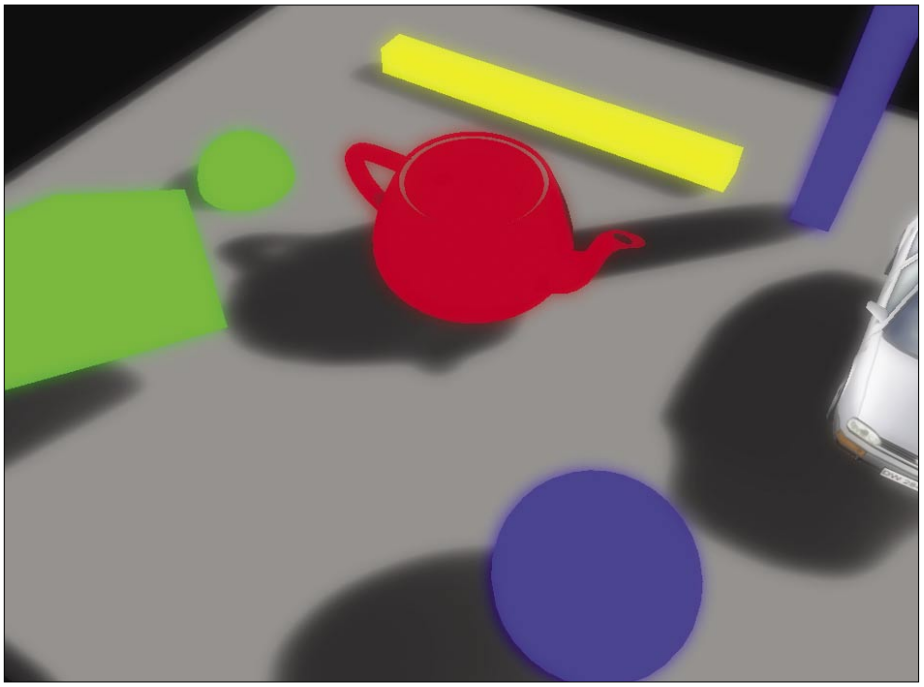
- generowanie mapy cieni z punktu widzenia światła,
- renderowanie sceny z punktu widzenia obserwatora z wyznaczaniem zacieniowanych obszarów.

Poniżej omówimy dokładnie te dwa elementy algorytmu a także przyjrzymy się szczegółom implementacji algorytmu z wykorzystaniem akceleracji sprzętowej kart graficznych.

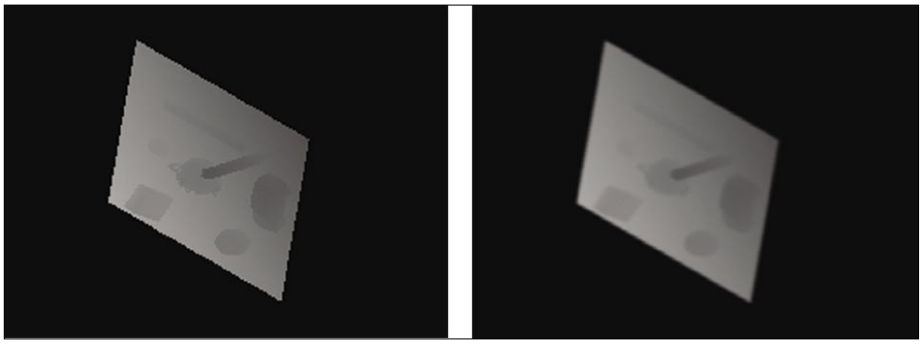
Generowanie mapy cieni

W celu wygenerowania mapy cieni musimy wyrenderować scenę z punktu widzenia światła oraz zapisać informację o głębokości do tekstury. Aby to zrobić, potrzebujemy najpierw utworzyć teksturę o formacie zmiennoprzecinkowym i zadanym wymiarze, np. 512x512 pikseli. Będziemy także potrzebować macierzy transformacji światła na wzór macierzy transformacji obserwatora. Dla świateł punktowych (ang. *omni light*) macierz transformacji światła powinna opisywać rzut perspektywiczny, natomiast dla świateł kierunkowych (ang. *directional light*) – rzut prostopadły. Macierz transformacji światła powinna również zawierać informację (podobnie jak macierz transformacji obserwatora) o położeniu i kierunku światła.

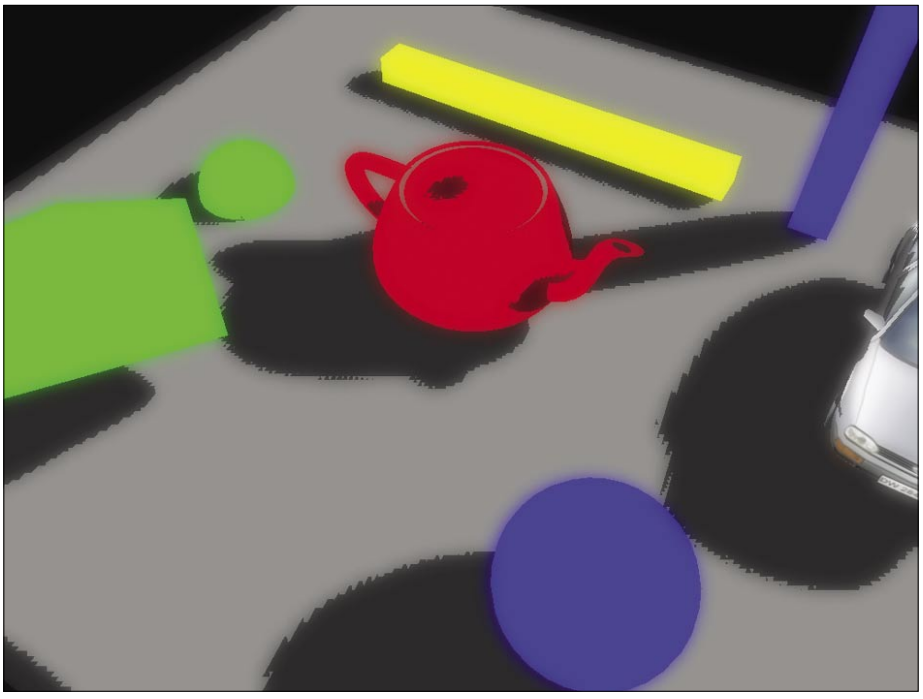
Podczas generowania obrazu do mapy cieni wszystkie wierzchołki obiektów na scenie są przekształcane przez macierz transformacji światła w celu pozyskania informacji o głębokości. W programie cieniującym piksele zamiast koloru zapisujemy interpolowaną pomiędzy wierzchołkami obiektu wartość współrzędnej *z*, która jest równoznaczna z odległością danego fragmentu obiektu od światła. Na Rysunku 2. przedstawiona jest wizualizacja mapy głębokości wygenerowanej z punktu widzenia światła dla przykładowej sceny.



Rysunek 1. *Cienie wygenerowane metodą wykładniczych map cieni*



Rysunek 2. *Wygenerowana mapa cieni z punktu widzenia światła; od lewej: podstawowa mapa cieni i rozmyta mapa cieni*



Rysunek 3. *Cienie wygenerowane podstawową metodą mapowania cieni; widoczne artefakty na brzegach obszarów zacieniowanych*

Obszary ciemniejsze oznaczają fragmenty znajdujące się bliżej światła (wartość współrzędnej z jest mniejsza). Kod w HLSL-u generujący mapę cieni jest przedstawiony na Listingu 1.

Listing 1. *Generowanie mapy cieni (HLSL)*

```
struct VS_SHADOW_OUTPUT
{
    float4 Position   : POSITION;
    float2 Depth    : TEXCOORD0;
};

VS_SHADOW_OUTPUT ShadowVS( VS_INPUT input )
{
    VS_SHADOW_OUTPUT output;

    /// Przekształcenie wierzchołków przez macierz transformacji światła
    output.Position = mul(input.Position, WorldMatrix);
    output.Position = mul(output.Position, LightViewProjMatrix);
    output.Depth.xy = output.Position.zw;

    return output;
}


float4 ShadowPS( VS_SHADOW_OUTPUT input ) : COLOR0
{
    /// Zapisanie wartości z/w w mapie głębokości
    return input.Depth.x / input.Depth.y;
}


technique zfill
{
    pass P0
    {
        VertexShader = compile vs_2_0 ShadowVS();
        PixelShader  = compile ps_2_0 ShadowPS();

            CullMode = CCW;
            ZEnable = true;
            ZWriteEnable = true;
    }
}
```

Rysunek 4. *Różnice w jakości wyznaczonych obszarów zacieniowanych, od lewej: (a) podstawową techniką mapowania cieni; (b) techniką PCF; (c) techniką wykładniczych map cieni*

Wyznaczanie zacieniowanych obszarów

Wyznaczanie zacieniowanych obszarów odbywa się podczas procesu generowania obrazu sceny z punktu widzenia obserwatora.

Program cieniujący wierzchołki (ang. *vertex shader*), oprócz typowej dla siebie operacji przekształcenia wierzchołków obiektów przez macierz transformacji obserwatora, wyznacza również wierzchołki obiektów w układzie światła (poprzez przekształcenie przez macierz transformacji światła). Otrzymane w ten sposób informacje przekazywane są do programu cieniującego piksele (ang. *pixel shader*), który jest odpowiedzialny za wyznaczenie odpowiednich wartości koloru dla wszystkich pikseli w docelowym widoku. Jednym z elementów wyznaczenia wartości koloru jest określenie, czy dany piksel reprezentujący fragment renderowanego obiektu powinien zostać zacieniowany. Program cieniujący piksele na ekranie porównuje głębokość każdego piksela w układzie współrzędnych światła (r) z głębokością danego piksela odczytaną z mapy głębokości (o). Możliwe przypadki to:

- `r = o` – na drodze do światła nie stoi żadna przeszkoda, zatem piksel nie jest zacieniowany (wartość koloru pozostaje bez zmian),
- `r > o` – na drodze do światła stoi przeszkoda, która cieniuje rozpatrywany piksel na ekranie (wartość koloru zostaje przemnożona przez współczynnik zacieniowania (ang. *shadow factor*), np. `sf = 0.3`).

Powtarzając powyżej wymienione czynności dla każdego piksela na ekranie, wyznaczamy zacieniowane obszary w zakresie pola widzenia obserwatora (jak pokazano na Rysunku 3.).

Czy można poprawić jakość cieni?

Jak możemy zaobserwować na Rysunku 3., jakość wygenerowanych cieni techniką mapowania cieni pozostawia wiele do życzenia. Główną przyczyną słabej jakości cieni jest mała rozdzielczość mapy cieni (w przykładzie 512 x 512 pikseli).

Cienie możemy poprawić, zwiększając rozdzielczość mapy cieni. Nie zawsze jednak możemy sobie na to pozwolić, gdyż wiąże się to z większym narzutem pamięciowym, a także ze spadkiem wydajności. Możemy również skorzystać z techniki PCF, która wykorzystuje interpolację dwuliniową (ang. *bilinear interpolation*) w celu uzyskania bardziej płynnego przejścia pomiędzy obszarami zacieniowanymi i oświetlonymi. Wartości z tekstury mapy cieni wykorzystujemy do porównania w procesie wyznaczania zacieniowanych obszarów sceny, co sprawia, iż w żaden sposób nie możemy tej tekstury zmodyfikować (np. przefiltrować). Możemy natomiast zbadać otoczenie piksela, próbkując kilka wartości z mapy cieni i w ten sposób określić stopień zacieniowania w danym punkcie poprzez uśrednienie otrzymanych wyników porównania. Na Rysunku 4. możemy zobaczyć nieznaczną poprawę jakości wygenerowanych cieni przy wykorzystaniu techniki PCF.

Innym sposobem na poprawienie jakości cieni jest przefiltrowanie bufora (np. techniką Gaussa) z wygenerowanymi cieniami, który wielkością odpowiada rozmiarom bufora ekranu. W tym podejściu należy wyrenderować scenę (mapę koloru obiektów) i zacieniowane obszary do osobnych tekstur. Pomocna tutaj okazuje się technika MRT (ang. *multiple render targets*), która umożliwia nam zapis do tych dwóch tekstur w jednym przebiegu. Następnie filtrujemy bufor z zacieniowanymi obszarami, co niweluje efekt aliasingu i ostrych krawędzi cieni. W ostatniej fazie rysujemy prostokąt na pełnym ekranie, uwzględniając wartości koloru pikseli z tekstury sceny i z rozmytej tekstury cieni.

Algorytm kaskadowych map cieni (CSM, ang. *cascaded shadow maps*) eliminuje w znacznym stopniu problem aliasingu poprzez zastosowanie dokładniejszych map cieni dla obiektów znajdujących się bliżej obserwatora i map mniej dokładnych dla obiektów odległych. CSM wykorzystują podział stożka widzenia obserwatora prostopadłymi płaszczyznami i generowanie oddzielnej mapy cieni dla każdego uzyskanego fragmentu stożka.

Lepszą jakość cieni możemy również uzyskać, wykorzystując najnowszą technikę wykładniczych map cieni.

Wykładnicze mapy cieni

Jak wspomnieliśmy wcześniej, algorytm map cieni porównuje głębokość każdego piksela w układzie współrzędnych światła z głębokością danego piksela odczytaną z mapy głębokości. W ten sposób otrzymujemy jedynie dyskretną wartość (0 lub 1), która określa, czy dany piksel jest w cieniu, czy też nie. Może moglibyśmy wykorzystać też informację o różnicy wartości, które porównujemy?

Wykładnicze mapy cieni (ESM, ang. *exponential shadow maps*) wyznaczają cienie za pomocą ciągłej funkcji wykładniczej

$$f(r,o)=e^{-c(y-o)}$$

gdzie c jest stałym współczynnikiem skalującym różnicę wartości głębokości piksela w układzie światła i wartości odczytanej z mapy cieni ($r–o$). Dzięki takiemu podejściu mamy możliwość skorzystania z przefiltrowanej mapy cieni i płynnego wyznaczenia stopnia zacieniowania rozpatrywanego piksela. W ten sposób uzyskujemy płynne przejścia z obszarów zacieniowanych do obszarów oświetlonych. Zmieniając wartość stałej c, sterujemy poziomem miękkości uzyskanych cieni (w przykładzie wartość stałej c wynosi 90).

Algorytm wykładniczych map cieni wprowadza dodatkowy krok polegający na filtrowaniu mapy cieni:

- generowanie mapy cieni z punktu widzenia światła,
- przefiltrowanie mapy cieni (np. techniką Gaussa),
- wygenerowanie obrazu sceny z punktu widzenia obserwatora z wyznaczaniem zacieniowanych obszarów przy pomocy funkcji wykładniczej.

Proces generowania mapy cieni dla algorytmu ESM z punktu widzenia światła pozostaje bez zmian w stosunku do wersji dla zwykłego algorytmu mapowania cieni, który omówiliśmy wcześniej. Nowością jest kolejny krok polegający na przefiltrowaniu (rozmyciu) mapy cieni, dzięki czemu uzyskujemy miękkie obwód-

Listing 2. *Rozmywanie mapy cieni metodą logarytmiczną (HLSL)*

```
struct VS_OUTPUT_SMBLUR
{
    float4 Position    : POSITION;
    float2 TexCoord[5] : TEXCOORD0;
};
/// Rozmiar mapy cieni (512x512)
#define SHADOW_MAP_SIZE 512

VS_OUTPUT_SMBLUR BlurShadowMapVS( float4 vPos : POSITION, float4 vTexCoord : TEXCOORD0,
                      uniform int taps, uniform float2 direction )
{
    VS_OUTPUT_SMBLUR output;

    float2 texelSize = 1.0f / float2(SHADOW_MAP_SIZE, SHADOW_MAP_SIZE);

    output.Position.xy = 2.0f * vPos.xy - 1.0f;
    output.Position.y *= -1;
    output.Position.zw = float2(0.5f, 1.0f);

    float2 step = direction * texelSize;
    float2 base = vTexCoord - ((((float)taps - 1)*0.5f) * step);

    for (int i = 0; i < taps; i++)
    {
        output.TexCoord[i] = base;
        base += step;
    }

    return output;
}
//-----------------------------------------------------------------------------
// Filtrowanie logarytmiczne
//-----------------------------------------------------------------------------
float log_conv ( float x0, float X, float y0, float Y )
{
    return (X + log(x0 + (y0 * exp(Y - X))));
}

float4 BlurShadowMapPS( VS_OUTPUT_SMBLUR input ) : COLOR0
{
    int i;

    float sample[5];
    for (i = 0; i < 5; i++)
        sample[i] = tex2D( ShadowMapSampler, input.TexCoord[i] ).x;

    const float c = (1.f/5.f);

    float accum;
    accum = log_conv( c, sample[0], c, sample[1] );
    for (i = 2; i < 5; i++)
        accum = log_conv( 1.f, accum, c, sample[i] );

    return accum;
}
```

Listing 3a. *Wyznaczanie obszarów zacieniowanych różnymi technikami (HLSL)*

```
struct VS_INPUT
{
    float4 Position     : POSITION;
    float3 Normal       : NORMAL0;
    float3 Tangent      : TANGENT;
    float4 Color        : COLOR0;
    float2 TexCoord     : TEXCOORD0;
};

struct VS_OUTPUT
{
    float4 Position         : POSITION;
    float4 Diffuse          : COLOR0;
    float4 TexCoord      : TEXCOORD0;
    float4 Normal           : TEXCOORD1;
    float4 PosInWorld    : TEXCOORD4;
    float4 PosInSun       : TEXCOORD5;
};


VS_OUTPUT RenderSceneVS( VS_INPUT input)
{
    VS_OUTPUT output;

    float4 pos_in_world = mul(input.Position, WorldMatrix);
    output.Diffuse = input.Color * DiffuseColor;

    output.Position = mul(pos_in_world, ViewProjMatrix);
    output.TexCoord = float4(input.TexCoord.xy, 0.0f, 0.0f);

    float3 n = normalize(mul(input.Normal.xyz, (float3x3)WorldMa
                      trix).xyz);
    output.Normal = float4(n.xyz, 1.0f);

    /// Pozycja w układzie świata
    output.PosInWorld = pos_in_world;

    /// Pozycja w układzie słońca
    output.PosInSun = mul(pos_in_world, LightViewProjMatrix);

    return output;
}

texture TexShadowMap;
sampler ShadowMapSampler = sampler_state
{
    Texture   = <TexShadowMap>;
    AddressU  = CLAMP;
    AddressV  = CLAMP;
    MinFilter = LINEAR;
    MagFilter = LINEAR;
    MipFilter = LINEAR;
    SRGBTEXTURE = FALSE;
};

#define SHADOWS_SIMPLE         0
#define SHADOWS_PCF       1
#define SHADOWS_ESM      2

#define SHADOWS_TECH    SHADOWS_ESM
```

```
float4 RenderScenePS( VS_OUTPUT input ) : COLOR0
{
    float4 oColor = input.Diffuse;

    float3 n = normalize(input.Normal.xyz);
    float3 l_dir = normalize(SunPos.xyz - input.PosInWorld.xyz);

    /// Jeżeli trójkąt jest zwrócony w stronę światła
    if (dot(l_dir.xyz, n.xyz) > 0.0f)
    {
      /// Wyznaczenie współrzędnych mapowania danego piksela w
                    mapie głębokości
      float2 sh = 0.5 * input.PosInSun.xy / input.PosInSun.w +
                    float2(0.5f, 0.5f);
      sh.y = 1.0f - sh.y;

      /// Wyznaczenie wartości Z w układzie światła
      float zw = input.PosInSun.z / input.PosInSun.w;

      /// Zmienne pomocnicze dla interpolacji dwuliniowej
      float2 SM_SIZE = float2(512 ,512);
      float2 unnormalized = sh * SM_SIZE;
      float2 fractional = frac(unnormalized);
      unnormalized = floor(unnormalized);

#if (SHADOWS_TECH == SHADOWS_SIMPLE)
      float shadow = (tex2D( ShadowMapSampler, unnormalized.xy/
                    SM_SIZE ) + EPSILON < zw) ? 0.0f: 1.0f;
#endif

#if (SHADOWS_TECH == SHADOWS_PCF)
      /// Porównanie 4 próbek w obrębie badanego piksela
      float4 shadow4;
      shadow4.x = (tex2D( ShadowMapSampler, (unnormalized.xy +
                    float2(0,0))/SM_SIZE ) + EPSILON < zw) ?
                    0.0f: 1.0f;
      shadow4.y = (tex2D( ShadowMapSampler, (unnormalized.xy +
                    float2(1,0))/SM_SIZE ) + EPSILON < zw) ?
                    0.0f: 1.0f;
      shadow4.z = (tex2D( ShadowMapSampler, (unnormalized.xy +
                    float2(0,1))/SM_SIZE ) + EPSILON < zw) ?
                    0.0f: 1.0f;
      shadow4.w = (tex2D( ShadowMapSampler, (unnormalized.xy +
                    float2(1,1))/SM_SIZE ) + EPSILON < zw) ?
                    0.0f: 1.0f;

      /// Interpolacja dwuliniowa pomiędzy wynikami 4 porównań
      float shadow = lerp(lerp( shadow4.x, shadow4.y,
                    fractional.x ),
                       lerp( shadow4.z, shadow4.w,
                    fractional.x ),
                       fractional.y );
#endif

#if (SHADOWS_TECH == SHADOWS_ESM)
    /// Próbkowanie 4 wartości obrębie badanego piksela
    float4 shadow4;
    shadow4.x = zw - tex2D( ShadowMapSampler, (unnormalized.xy
                    + float2(0,0))/SM_SIZE );
    shadow4.y = zw - tex2D( ShadowMapSampler, (unnormalized.xy
                    + float2(1,0))/SM_SIZE );
```

ki wokół obszarów zacieniowanych (Rysunek 2b.). Bardzo dobre rezultaty przy filtrowaniu mapy cieni otrzymuje się, korzystając z metody logarytmicznej przedstawionej na Listingu 2. Filtrowanie odbywa się przy wykorzystaniu tymczasowej tekstury w formacie zmiennoprzecinkowym, rozmiarami odpowiadającej mapie cieni. Kolejny etap algorytmu polega na wyznaczeniu obszarów zacieniowanych przy wykorzystaniu przefiltrowanej mapy cieni. Podobnie jak wcześniej wyznaczamy głębokość piksela w układzie światła (r) oraz odczytujemy głębokość z mapy cieni (o). Tym razem jednak nie porównujemy tych dwóch wartości, tylko wyznaczamy stopień zacieniowania piksela, korzystając ze wzoru. Dodatkowo możemy skorzystać z techniki PCF, podobnie jak przy zwykłych mapach cieni. W wyniku otrzymujemy miękkie cienie bez widocznego aliasingu i innych artefaktów wpływających negatywnie na krawędzie cieni. Kod programu cieniującego piksele różnymi technikami (podstawową, PCF i wykładniczych map cieni) przedstawiono na Listingu 3. Metoda wykładniczych map cieni w połączeniu z techniką PCF daje bardzo dobre rezultaty, co możemy zaobserwować na Rysunku 4. W razie konieczności możemy również wygenerować cienie techniką ESM do oddzielnego bufora i następnie skorzystać z filtrowania pełnoekranowego jeszcze bardziej zmiękczając cienie. Dodatkowo możemy zastosować technikę kaskadowych map cieni zmniejszając efekt aliasingu.

Podsumowanie

Przedstawiliśmy szczegółowy opis oraz implementację nowoczesnego algorytmu wykładniczych map cieni służącego do wyznaczania miękkich cieni w czasie rzeczywistym. Wykładnicze mapy cieni wprowadzają niewielkie zmiany w stosunku do swojej wersji pierwotnej. Sprawia to, iż zaprezentowany algorytm jest szybki i wydajny nawet na nieco starszych kartach graficznych.

Wykładnicze mapy cieni sprawdzają się zarówno dla opisujących wnętrza np. pomieszczenia budynków (ang. *indoor scenes*) jak i dla złożonych scen modelujących świat zewnętrzny (teren, lasy, roślinność, budynki, ang. *outdoor scenes*). Przedstawiony algorytm jest elastyczny i możliwy w zastosowaniu z innymi metodami poprawiającymi jakość cieni (np. PCF, czy kaskadowe mapy cieni).

Wszystkie przykładowe programy zostały przetestowane na komputerze z procesorem AMD Athlon XP 2500+ 1.8 GHz, 512 MB pamięci RAM oraz kartą graficzną nVidia 6600 GT.

W Sieci

- *http://pixelstoomany.wordpress.com/category/shadows/exponential-shadow-maps/* – model matematyczny wykładniczych map cieni;
- *http://www.mpi-inf.mpg.de/~tannen/papers/gi_08_esm.pdf* – artykuł opisujący algorytm wykładniczych map cieni;
- *http://en.wikipedia.org/wiki/Shadow_mapping* – definicja podstawowych pojęć związanych z mapowaniem cieni;
- *http://developer.nvidia.com/object/doc_shadows.html* – zbiór artykułów dotyczących różnych metod cieniowania w grafice komputerowej;
- *http://msdn.microsoft.com/en-us/library/bb147372(VS.85).aspx* – przykład (wraz z kodem źródłowym) prezentujący działanie algorytmu map cieni.

Definicje

- *Algorytm* – ściśle określony ciąg czynności, których wykonanie prowadzi do rozwiązania jakiegoś zadania;
- *Fotorealizm* – sposób oddawania rzeczywistości powodujący wrażenie, jakby została ona sfotografowana;
- *Silnik graficzny* – część kodu aplikacji odpowiedzialna za tworzenie grafiki na ekranie komputera;
- *Interpolacja dwuliniowa* (ang. *bilinear filtering*) – wyznaczanie wartości koloru piksela na podstawie wartości czterech sąsiednich pikseli;
- *Aliasing* – zjawisko zniekształcenia obrazu w procesie rasteryzacji wynikające ze zbyt małej częstości próbkowania.

PAWEŁ ROHLEDER

Autor pracuje na stanowisku programisty/projektanta grafiki komputerowej 3D w firmie Techland Sp. z o.o. Jest również słuchaczem studiów doktoranckich na Politechnice Wrocławskiej.

Kontakt z autorem: prohleder@wp.pl

Listing 3b. *Wyznaczanie obszarów zacieniowanych różnymi technikami (HLSL)*

```
    shadow4.z = zw - tex2D( ShadowMapSampler, (unnormalized.xy
                      + float2(0,1))/SM_SIZE );

    shadow4.w = zw - tex2D( ShadowMapSampler, (unnormalized.xy
                      + float2(1,1))/SM_SIZE );

    shadow4 = max(shadow4, 0.0f) * saturate(dot(n, l_dir));

    /// Zastosowanie funkcji wykładniczej dla pobranych próbek
                      (wzór 1)

    shadow4 = exp(-90.0f * shadow4);

    /// Interpolacja dwuliniowa pomiędzy 4 próbkami
    float shadow = lerp(lerp( shadow4.x, shadow4.y, fractional.x
                      ),
                        lerp( shadow4.z, shadow4.w,
                      fractional.x ),

                        fractional.y );
#endif
```

```
        /// Przemnożenie wartości koloru przez współczynnik
                          zacieniowania
        oColor.xyz *= saturate(shadow + 0.3f);
    }

    return oColor;
}


technique std
{
    pass P0
    {
        VertexShader = compile vs_2_0 RenderSceneVS();
        PixelShader  = compile ps_2_0 RenderScenePS();

        CullMode = CCW;
        ZEnable = true;
        ZWriteEnable = true;
    }
}
```

Mobilne PKI

Autoryzacja i uwierzytelnianie użytkowników

Koncepcja Infrastruktury Klucza Publicznego (PKI) znana jest od lat, jednak wciąż nie potrafi znaleźć naprawdę powszechnego zastosowania. W naszym artykule przyjrzymy się implementacji PKI na urządzeniach mobilnych, na przykładzie wdrożenia w jednej z polskich służb mundurowych. Wytłumaczymy, co to LAP i LASS, dlaczego łatwo jest używać CSP oraz jakie niespodzianki czekają na piszących oprogramowanie na karty SIM.

Dowiesz się:
- co to jest LAP i LASS,
- jak stworzyć własne CSP,
- co to jest Java Card,
- dlaczego mobilny podpis elektroniczny ma szansę zrewolucjonizować podejście do PKI.

Powinieneś wiedzieć:
- coś o programowaniu w językach C i Java (znajomość tematyki kart SIM oraz podpisu elektronicznego może okazać się przydatna).

Poziom trudności

Infrastruktura Klucza Publicznego (z ang. *Public Key Infrastructure* – PKI) to znana od wielu lat koncepcja wykorzystania i zastosowania w praktyce zalet i dobrodziejstw zaproponowanej w 1976r. przez Diffiego i Hellmana kryptografii wykorzystującej asymetryczne algorytmy szyfrujące. Mimo wszystkich swoich zalet, najwyższego oferowanego poziomu bezpieczeństwa, gotowości technologicznej, wielu analiz i niezliczonych publikacji poświęconych implementacji PKI, koncepcja ta (także w Polsce) wciąż nie uzyskała statusu rozwiązania powszechnego i ogólnie dostępnego. Kolejne światowe wdrożenia mają ciągle charakter eksperymentalny i obejmują raczej niewielkie grupy użytkowników

Wyboista droga do PKI

W warunkach polskich, wśród czynników ograniczających faktyczne upowszechnienie idei podpisu elektronicznego, poza barierami psychologicznymi oraz podejrzliwością wobec nowych rozwiązań, a zwłaszcza tych rozwiązań, które pociągać mogą za sobą bardzo brzemienne skutki formalnoprawne, wymienić należy trzy elementy:

- Kontekst oraz uwarunkowania prawne determinowane w głównej mierze przez ustawę o podpisie elektronicznym z 18 września 2001 roku – niestety znajdujące się tam definicje i rozwiązania nie poprawiły powszechnego zrozumienia, a w konsekwencji nie doprowadziły do zaakceptowania i wdrożenia, omawianej technologii.
- Uwarunkowania ekonomiczne – w powszechnym odczuciu koszt certyfikatów kwalifikowanych potrzebnych do prawnego uznania podpisu elektronicznego jest zbyt wysoki (rzędu 200 PLN rocznie), co w połączeniu z ograniczonym do niego zaufaniem i niepełnym zrozumieniem jego znaczenia i działania powoduje, że ciągle niewiele osób decyduje się na jego zakup,
- Ograniczenia związane z wygodą, komfortem oraz ergonomią korzystania z rozwiązań aktualnie dostępnych na rynku – osoba chcąca korzystać na co dzień z dobrodziejstw PKI musi nabyć, zainstalować i skonfigurować dodatkowe elementy sprzętowe, takie jak specjalne karty kryptograficzne, dedykowane do nich czytniki itd.

Czy zatem podpis elektroniczny – rozwiązanie, które może usprawnić tak wiele obszarów naszej aktywności – nie ma szans na to by stać się naturalnym, ogólnie dostępnym i powszechnie akceptowanym rozwiązaniem w naszym kraju? Na szczęście wydaje się, że tak nie jest. W ostatnim czasie zarówno na prawnym, jak i na technologicznym froncie podjęto działania, które sprawić mają, iż podpis elektroniczny zyska należny mu status i rzeczywiście wejdzie pod strzechy.

Z punktu widzenia uregulowań prawnych trwają intensywne prace nad nowelizacją ustawy o podpisie elektronicznym. Obecnie prowadzone są ustalenia międzyresortowe i pozostaje mieć nadzieję, iż znowelizowana ustawa nie tylko dostosuje polskie prawodawstwo do prawodawstwa unijnego, ale też, unikając wad i błędów swojej poprzedniczki, przyczyni się do lepszego zrozumienia i ułatwi akceptację rozwiązań podpisu elektronicznego.

W niniejszym artykule skupiamy się natomiast na rozwiązaniu, które eliminuje pozostałe ze wspomnianych ograniczeń spowalniających bądź uniemożliwiających upowszechnienie się w naszym społeczeństwie rozwiązań związanych z PKI i podpisem elektronicznym. O stopniu zaawansowania i innowacyjności tego rozwiązania niech świadczy fakt uhonorowania go (i jego autorów) tytułem Innovatora Małopolski 2008.

Mobilna infrastruktura klucza publicznego

Koncepcja mobilnego PKI zakłada wykorzystanie jako nośnika materiału kryptograficznego specjalnych kart SIM, które omawiane są szerzej w dalszej części artykułu, oraz

dowolnego telefonu komórkowego. Za użyciem telefonu przemawia nasycenie rynku, przekraczające 100% populacji, oraz fakt, że (według badań koncernu Visa) ponad 70% z nas zabiera telefon ze sobą udając się na nocny spoczynek oraz używa go zamiast budzika i zegarka. Użycie kart SIM i telefonu pozwala więc użytkownikowi uniknąć kosztów dodatkowego sprzętu, a zamiast czytników, kabli i kolejnych kart używać tylko swojego telefonu!

Zaproponowano zatem rozwiązanie, które eliminuje w znacznej mierze niewygody i ograniczenia rozwiązań, które dotychczas obecne były na rynku, zapewniając oczywiście najwyższy poziom bezpieczeństwa. Świadczy o tym m.in. fakt, iż jedna z polskich służb mundurowych już od ponad roku stosuje jedną z implementacji omawianego rozwiązania do zapewnienia autoryzacji dostępu do urządzenia przenośnego, a następnie do zabezpieczania transmisji danych i uwierzytelniania dostępu do krytycznych danych operacyjnych. W dalszej części artykułu skupimy się na omówieniu wybranych aspektów technologicznych tego właśnie rozwiązania.

Ogólna architektura rozwiązania

Na Rysunku 1. przedstawiono ogólną architekturę wdrożonego rozwiązania. Przy użyciu oprogramowania dostępnego na urządzeniu mobilnym (np. systemowego modułu uwierzytelniania, przeglądarki internetowej), które wykorzystuje specjalne biblioteki kryptograficzne, użytkownik uzyskuje dostęp do strzeżonych systemów informatycznych. Artykuł ten omawia tylko część komponentów zaprezentowanej architektury, umieszczonych na urządzeniu mobilnym.

W dalszej części opisujemy koncepcję, sposób działania oraz techniki implementacji: modułu uwierzytelniania użytkowników LAP, bibliotek kryptograficznych CSP oraz apletów JavaCard odpowiedzialnych za przechowywanie kluczy i certyfikatów oraz wykonywanie przy ich użyciu operacji kryptograficznych.

Local Authentication Plugin

Wykorzystywane w opisywanym rozwiązaniu urządzenia PDA (takie jak np. Symbol MC70 czy Intermec CN3 (Rysunek 2) są wyposażone w system operacyjny Windows Mobile. Zabezpieczenie dostępu do urządzenia odbywa się przy wykorzystaniu komponentu architektury systemu operacyjnego zwanego LASS (*Local Authentication SubSystem*). Jest to moduł, który pozwala na zastosowanie zaawansowanych mechanizmów i polityk uwierzytelniania użytkowników urządzenia wyposażonego w mobilną wersję Windows.

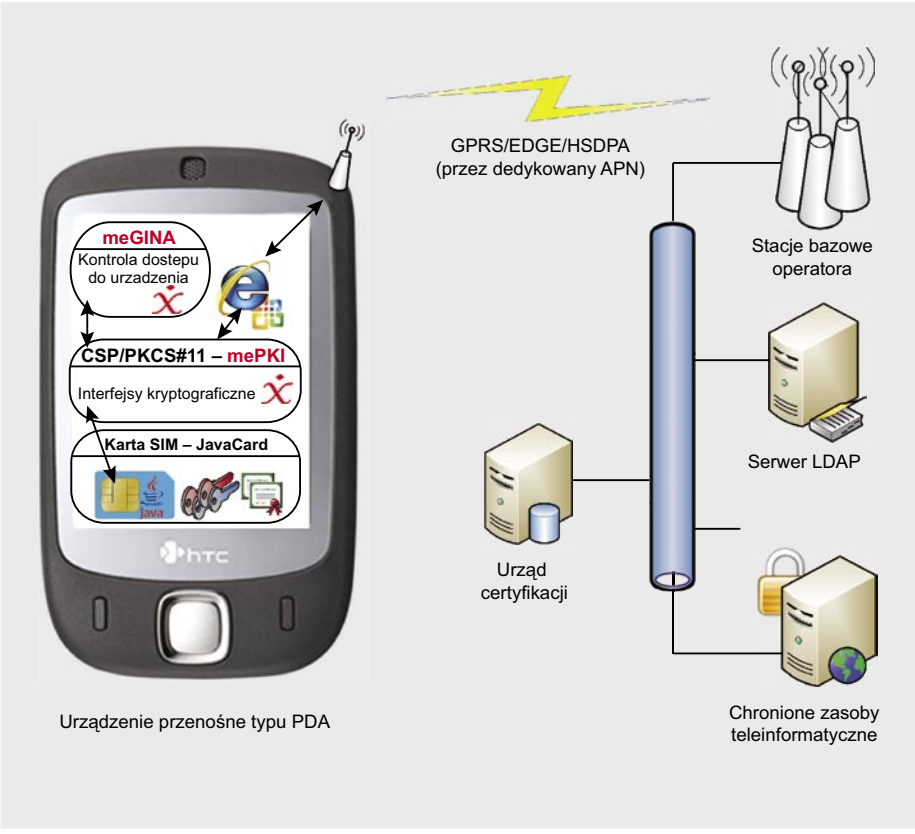
Zdefiniowanie takiego mechanizmu polega na instalacji modułu zwanego LAP (*Local Authentication Plugin*), który odpowiedzialny jest właśnie za proces uwierzytelniania użytkowników.

Z technicznego punktu widzenia, moduł LAP to dynamicznie ładowana biblioteka (DLL), eksportująca określone funkcje, które wywoływane są przy uruchamianiu systemu w celu uwierzytelnienia użytkownika. Instalowany moduł LAP musi dodatkowo posiadać specjalny podpis cyfrowy – bez niego moduł nie zostanie w ogóle załadowany przez system operacyjny. Podpis ten można uzyskać w przeprowadzanym przez Microsoft procesie weryfikacji konkretnego modułu.

Tworząc własny moduł LAP można odwoływać się do szeregu serwisów udostępnianych przez środowisko Windows Mobile. W przypadku omawianego rozwiązania, najbardziej przydatną możliwością jest odwoływanie się do tzw. CryptoAPI – warstwy systemu Windows odpowiedzialnej za udostępnianie usług kryptograficznych. Dostawcy bibliotek kryptograficznych rejestrują w CryptoAPI swoje biblioteki (tzw. CSP – *Cryptographic Service Provider*), z których można korzystać w całym systemie operacyjnym.

Cryptographic Service Provider

CSP jest analogicznym do istniejącego standardu PKCS#11 autorstwa korporacji RSA standardem definiującym powszechnie używane struktury i operacje kryptograficzne. Implementacja własnego CSP polega na utworzeniu modułu DLL z wyeksportowanymi funkcjami służącymi do rozpoczęcia



Rysunek 1. *Schemat architektury opisywanego w artykule rozwiązania*

Rysunek 2. *Przykładowe urządzenie przenośne Intermec CN3, wykorzystywane w opisywanym wdrożeniu, wraz z oprogramowaniem do kontroli dostępu do urządzenia – Mobile Experts meGINA*

i zakończenia pracy z modułem, generowania, odczytywania i zapisywania kluczy kryptograficznych i certyfikatów, sporządzania skrótów, podpisów i szyfrowania danych. Tak jak i w przypadku LAP, biblioteki CSP muszą być zweryfikowane i cyfrowo podpisane przez Microsoft.

Rysunek 3. przedstawia przykładowy kod używający CSP (dla przejrzystości bez dokładnego sprawdzania błędów). Jak widać, odwołanie do konkretnego dostawcy usług kryptograficznych odbywa się w funkcji `CryptAcquireContext` poprzez podanie jego nazwy symbolicznej.

Następnie wykorzystywana jest metoda enumeracji tzw. kontenerów (funkcja `CryptGetProvParam` z parametrem `PP_ENUMCONTAINER`). Pojedynczy kontener służy w standardzie CSP do przechowywania pary kluczy oraz przypisanego do niej certyfikatu.

Kryptograficzne karty SIM – programowanie w Java Card

Najniższą warstwą opisywanego rozwiązania, która pozwala na korzystanie z mechanizmów PKI, jest kryptograficzna karta SIM. Na karcie rezyduje specjalny komponent, zwany apletem, który umożliwia wykonywanie operacji personalizacyjnych karty (np. generacji kluczy szyfrujących, eksportu/importu certyfikatów) oraz operacji kryptograficznych (np. podpisywanie czy szyfrowanie danych).

Aplety przygotowywane są w technologii Java Card, czyli w specjalnej wersji języka Java, która dostarcza interfejsu do kryptoprocesora oraz pamięci karty SIM. Interfejs programistyczny tej technologii jest dość ograniczony – dla przykładu można wspomnieć, że klasa String jest dostępna dopiero w wersji 3.0 specyfikacji Java Card, a karty ją implementujące nie są jeszcze popularne (na Rysunku 4. przedstawiliśmy przykładowy łańcuch znaków Hello World w wersji dla Java Card 2). Do dyspozycji programisty są przede wszystkim elementy pozwalające na zarządzanie kodami PIN, generowanie kluczy symetrycznych (np. DES) oraz asymetrycznych (np. RSA) oraz szyfrowanie danych.

Komunikacja z apletem na karcie odbywa się poprzez tzw. pakiety APDU (*Application Protocol Data Unit*). Interfejs pozwalający na komunikację z kartą poprzez APDU jest dostępny zarówno dla czytników PC/SC podłączanych do stacjonarnych komputerów (są to znane większości z nas czytniki kart kryptograficznych), jak i dla czytników kart zainstalowanych w urządzeniach mobilnych różnego typu.

Bezpieczeństwo rozwiązania

Przy okazji dyskusji o kartach kryptograficznych nie można zapomnieć o dodatkowych elementach bezpieczeństwa, jakie wprowadzane są do infrastruktury wdrożonego systemu poprzez samo zastosowanie kart SIM i apletów Java Card. Po pierwsze, zapisanie oraz odczytanie apletu na karcie SIM nie jest możliwe bez znajomości specjalnych kluczy szyfrujących chroniących kartę (KiC oraz KiD) – klucze te zna tylko operator sieci, w której dana karta działa, i są one objęte jego bardzo restrykcyjną polityką bezpieczeństwa.

Po drugie, sama konstrukcja apletu uniemożliwia odczytanie kluczy prywatnych wygenerowanych i zapisanych na karcie – nie istnieje więc ryzyko ich utraty. Oczywiście, zarówno dostęp do karty SIM, jak i do wszystkich krytycznych operacji wykonywanych przez aplet (np. sporządzenie podpisu przy użyciu klucza prywatnego) są zabezpieczone kodami PIN.

Przyszłość to mobilePKI

Przedstawione wyżej, wdrożone już rozwiązanie, wykorzystuje tylko część potencjału, jaki niesie ze sobą integracja świata mobilnego z technologią PKI. Klucz prywatny przechowywany na karcie SIM może być wykorzystany do szeregu innych czynności i stać się uniwersalnym środkiem dostarczającym niezawodnego mechanizmu bezpieczeństwa w bardzo wielu zastosowaniach. Mobilny podpis elektroniczny mobilePKI stwarza możliwość integracji infrastruktury klucza publicznego z praktycznie dowolnym systemem lub aplikacją, a dodatkowo w znacznym stopniu upraszcza korzystanie z mechanizmów zapewniających najwyższy dostępny obecnie poziom bezpieczeństwa. Przy wykorzystaniu omawianej technologii, a zatem korzystając wyłącznie z telefonu, bez którego i tak nie ruszamy się z miejsca, można np. podpisać dokument Microsoft Word czy wiadomość email w programach pocztowych Outlook czy Thunderbird. Prostą operacją na naszym telefonie można zastąpić też wpisywanie jednorazowych haseł potwierdzających operacje bankowe czy ujednolicić procedurę dostępu do bezpiecznych zasobów firmowych poprzez przeglądarkę internetową. Wszystko to możliwe jest dzięki wykorzystaniu oprogramowania oraz bibliotek kryptograficznych analogicznych do opisywanych w tym artykule. Wierzymy, że zastosowanie mobilnego podpisu elektronicznego sprawi, że koncepcja PKI stanie wreszcie przed szansą szerokiego zastosowania.

O AUTORACH

Leszek Siwik, współzałożyciel i jeden z wiceprezesów MobileExperts sp. z o.o. oraz Krzysztof Lewandowski i Adam Woś – architekci i zarządzający m.in. projektem mobilnego PKI realizowanym w spółce. Wszyscy trzej specjalizują się w zagadnieniach szeroko pojętego oprogramowania mobilnego, a w szczególności tematyką wykorzystania urządzeń mobilnych w płatnościach, bankowości oraz do zapewnienia bezpieczeństwa informacji.


```
HCRYPTPROV prov; BOOL ret = TRUE; BYTE data[200]; DWORD dataLen = sizeof(data); int cnt = 1;

ret = CryptAcquireContext(&prov, NULL, L"MobileExperts mobilePKI", PROV_RSA_FULL, 0);
while (ret) {
    ret = CryptGetProvParam(prov, PP_ENUMCONTAINERS, data, &dataLen, (cnt == 1 ? CRYPT_FIRST : 0));
    if (ret) {
        HCRYPTPROV temp; HCRYPTKEY hKey;
        wprintf(L"Kontener %d: '%s'\n", cnt++, data);
        CryptAcquireContext(&temp, (LPCWSTR) data, L"MobileExperts mobilePKI", PROV_RSA_FULL, 0);
        ret = CryptGetUserKey(temp, AT_SIGNATURE, &hKey);
        if (ret) {
            DWORD certLen = 0;
            CryptGetKeyParam(hKey, KP_CERTIFICATE, NULL, &certLen, 0);
            wprintf(L"Znaleziono klucz w kontenerze %s, długość certyfikatu %d\n", data, certLen);
            // itd...
            CryptDestroyKey(hKey);
        }
        CryptReleaseContext(temp, 0);
    } else if (GetLastError() == ERROR_NO_MORE_ITEMS) {
        wprintf(L"Koniec listy kontenerów...\n") ;
        break;
    } else {
        // inny błąd...
    }
}
CryptReleaseContext(prov, 0);
```

Rysunek 3. *Przykład kodu pobierającego informacje o dostępnych w CSP kontenerach oraz kluczach i certyfikatach w nich zapisanych*

```
byte[] helloWorld = {
    (byte) 0x00, (byte) 'H', (byte) 0x00, (byte) 'e',
    (byte) 0x00, (byte) 'l', (byte) 0x00, (byte) 'l',
    (byte) 0x00, (byte) 'o', (byte) 0x00, (byte) ' ',
    (byte) 0x00, (byte) 'W', (byte) 0x00, (byte) 'o',
    (byte) 0x00, (byte) 'r', (byte) 0x00, (byte) 'l',
    (byte) 0x00, (byte) 'd', (byte) 0x00, (byte) '!' };
```

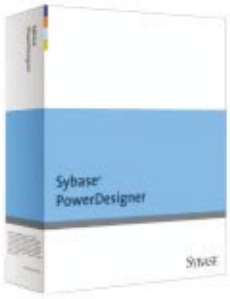
Rysunek 4. *Przykładowy łańcuch znaków Hello World! zapisany w formacie narzuconym przez API Java Card 2. Łańcuch jest zakodowany w UCS-2, stąd występujące co drugi bajt zera*

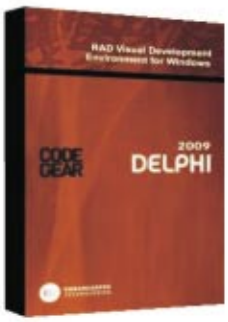
TTS Company

Qt Reseller

Największy wybór profesjonalnego oprogramowania w Polsce !

... w ofercie produkty ponad 200 producentów ...





www.OprogramowanieKomputerowe.pl

Code less. **Create more.** **Deploy everywhere.**

Więcej informacji: (022) 868 40 42

Sprzedaż ▸▸ **Dystrybucja** ▸▸ **Import na zamówienie**

TTS Company Sp. z o.o. ul.Słowicza 53 02-170 Warszawa e-mail: tts@tts.com.pl www.tts.com.pl

Mobilny Magazyn

Łamigłówki architektoniczne

Autor przedstawia dylematy związane z doborem właściwej architektury systemu komputerowego do obsługi magazynu. Zadaniem jest przeniesienie wybranych procesów logistycznych na urządzenia mobilne przy równoczesnej integracji z tradycyjnym systemem zarządzania magazynem a szerzej z systemem klasy ERP.

Dowiesz się:

- Jakie trudności są związane z tworzeniem rozwiązań wykorzystujących urządzenia mobilne oraz z integracją z istniejącym kodem (legacy code).
- Jak zaprojektować elastyczną i skalowalną architekturę systemu komputerowego dla omawianego zadania.

Powinieneś wiedzieć:

- Posiadać znajomość podstaw technologii .NET.

Poziom trudności

PS_2000 to system zarządzania przedsiębiorstwem. Przez lata obrastał nowymi funkcjonalnościami, modyfikacjami, wariantami spowodowanymi potrzebami różnych klientów. Wydawałoby się nic specjalnego – system jak wiele innych na rynku. Pewnego jednak poranka, trudno w tej chwili dociec w jaki sposób (potrzeba matką wynalazków?), zrodził się pomysł rozbudowy systemu zarządzania magazynem w PS_2000 o moduł obsługujący urządzenia mobilne. Obecnie wszystkim nam wydaje się, że używanie urządzeń mobilnych w magazynie to oczywista oczywistość. Oprogramowanie na urządzeniu mobilnym bezpośrednio komunikujące się z systemem PS_2000 i wykorzystujące kody kreskowe porządkuje i znacząco upraszcza procesy magazynowe. Ponadto daje możliwość pełnego nadzorowania ruchów magazynowych czy kompletacji tak, że towary opuszczają magazyn zgodnie z zamówieniem (w uzgodnionym czasie, z towarem określonej serii i o właściwej dacie ważności). Rozmawiając obecnie z pracownikami magazynów Farmacol S.A. – bo o tej firmie właśnie tu mowa, wielokrotnie spotykam się ze stwierdzeniem, że nie wyobrażają sobie obecnie pracy bez tych urządzeń.

Założenia

Wróćmy jednak do początków naszej przygody z urządzeniami mobilnymi (potocznie przez nas zwanymi terminalami). Zadanie postawione przed nami było złożone:

- Urządzenie mobilne ma komunikować się w czasie rzeczywistym z systemem PS_2000, tak aby każda zmiana na urządzeniu miała bezpośrednie odzwierciedlenie w systemie i odwrotnie;
- Praca w magazynie odbywa się w sposób ciągły (24h/dobę przez 7dni w tygodniu), więc oprogramowanie musi być niezawodne, a instalacja uaktualnień nie może dezorganizować pracy magazynu;
- Każdy terminal musi być identyfikowalny, każdy użytkownik jest autentykowany i autoryzowany w oprogramowaniu na terminalu;
- Wybrane oprogramowanie z dokładnością do wersji ma działać na konkretnych urządzeniach mobilnych;
- Rozwiązanie musi być skalowalne – początkowo obsługiwało około 100 urządzeń, obecnie ponad 200, a kolejne magazyny i oddziały czekają na wdrożenie;
- Musi istnieć możliwość monitorowania liczby zalogowanych użytkowników i obciążenia systemu.

Prócz trudnych wyzwań, które powyżej krótko opisałem, dochodzą również ograniczenia, które wpływają na sposób rozwiązania:

- Urządzeniami mobilnymi używanymi w magazynie są terminale firmy Motorola pracujące pod kontrolą systemu operacyjnego Windows CE;
- Komunikacja urządzeń ze światem zewnętrznym odbywa się drogą radiową – WiFi;
- System PS_2000 został napisany w C++ Builder i jedyną metodą integracji z jego modułami jest bezpośrednia wymiana komunikatów przez TCP/IP;
- Urządzenia mobilne mają bardzo ograniczoną wielkość pamięci trwałej (flash), w której są przechowywane pliki. Programy po pełnym zimnym restarcie są usuwane z pamięci urządzenia.

Od początku nie mieliśmy wątpliwości, że to rozwiązanie powinno powstać w technologii Microsoft .NET. Pragnęliśmy również stworzyć rozwiązanie modułowe, rozproszone, z łatwą możliwością skalowania.

Rozwiązanie

Ostatecznie zaproponowaliśmy system, którego architekturę można przedstawić za pomocą Rysunku 1.

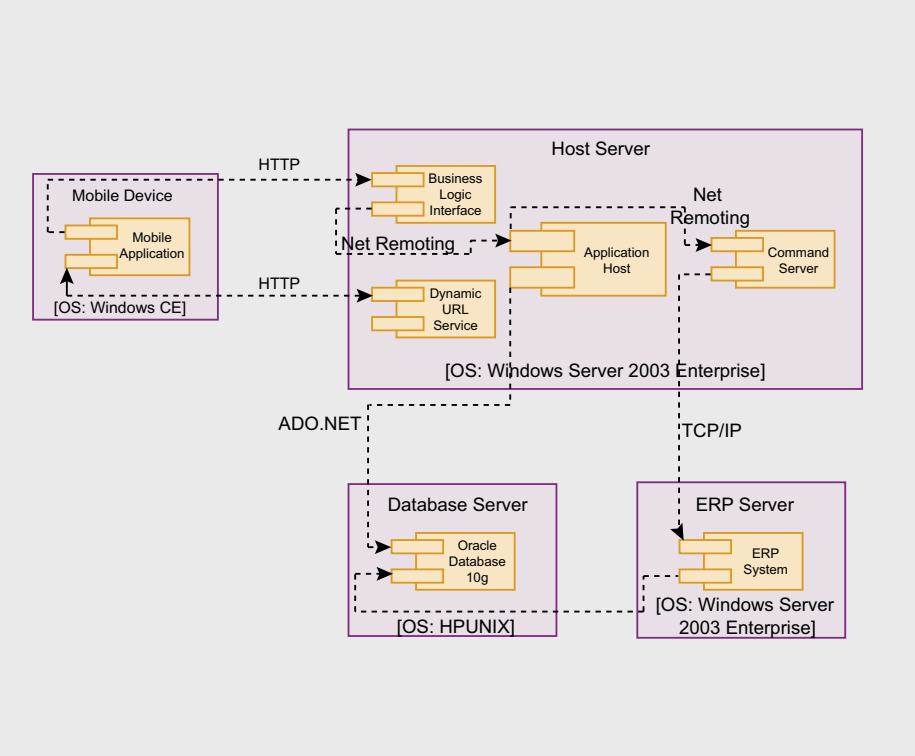
Na każdym urządzeniu mobilnym zostaje początkowo zainstalowane oprogramowanie Uploader, służące jedynie do instalacji pozostałych programów. Programy takie jak Komora przyjęć, Przesunięcia Międzymagazynowe, Stacja wydań itp. są programami wielowarstwowymi, których cienki klient pracujący na terminalu łączy się z serwerem aplikacji. W poszczególnych modułach ładowanych przez serwer aplikacji została zaimplementowana logika biznesowa programów i warstwa dostępu do danych. Moduły logiki biznesowej aplikacji często wywołu-

ją operacje z systemu ERP (tutaj PS_2000). Wywołanie tych operacji odbywa się z wykorzystaniem serwera poleceń (*command server*). Serwer poleceń może być połączony z wieloma instancjami modułu systemu ERP, dlatego jego głównym zadaniem jest kolejkowanie, balansowanie obciążenia i dystrybucja poleceń.

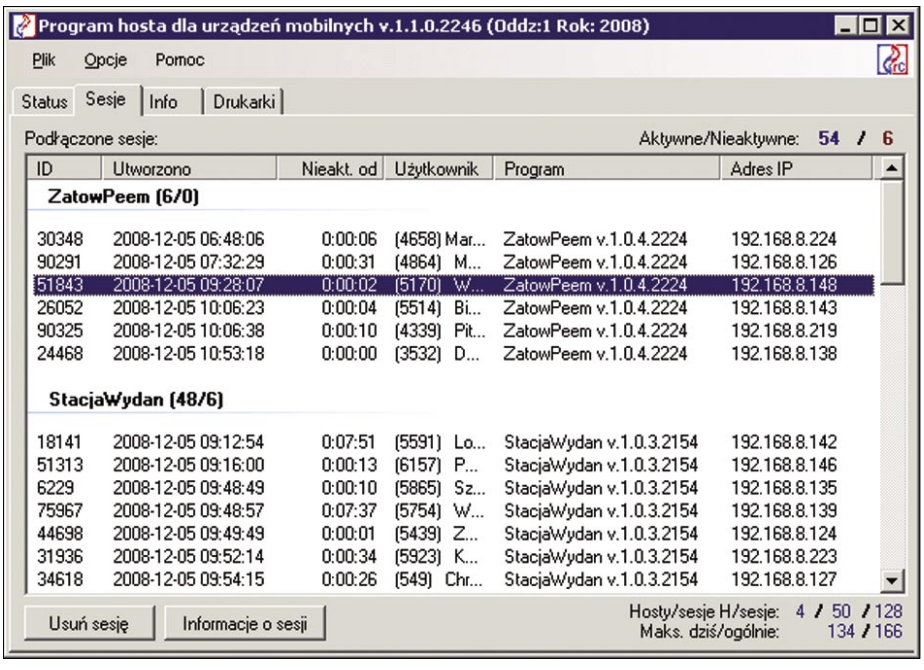
Szczególne znaczenie ma XML Web Service: Dynamic URL. Aplikacja uruchamiając się na urządzeniu mobilnym woła Dynamic URL. Na podstawie ID terminala i nazwy oraz wersji aplikacji Dynamic URL zwraca adres serwisu Business Logic Interface. Rozwiązanie takie pozwala na znaczną konfigurację systemu i skalowalność. W zależności od potrzeb na tym samym komputerze lub fizycznie innej maszynie, można zaimplementować kolejny serwis Business Logic Interface, gdy np. chcemy, aby nowa wersja programu działała jedynie na wybranych terminalach. Skalowalność jest też zapewniona na następnym poziomie. Mianowicie zarówno komunikacja między Business Logic Interface a serwerem aplikacji oraz serwerem aplikacji i serwerem poleceń odbywa się z wykorzystaniem *.NET Remoting*. Daje to oczywiście możliwość wdrożenia każdego z tych serwerów na oddzielnych maszynach.

Wspomniałem na początku artykułu o wymaganiu związanym z niezawodnością i wydajnością rozwiązania. Szczególnie narażone na potencjalne problemy wydawała nam się komunikacja z system ERP. Dlatego w serwerze poleceń został zaimplementowany system monitoringu (Rysunek 3). Monitorowane są czasy wykonania poszczególnych poleceń, liczby operacji oczekujących w kolejkach, ale również istnieje możliwość zablokowania wykonania konkretnego polecenia, czy testowanie poleceń. W serwerze aplikacji kontrolowana jest ilość równocześnie pracujących użytkowników z daną aplikacją (Rysunek 2). Dane związane z działaniem aplikacji, dla każdego użytkownika są przechowywane w obiektach sesji serwera aplikacji. Web serwis Busines Logic Interface pozostaje bezstanowy i jest tylko prostą warstwą pośredniczącą – fasadą serwera aplikacji. Jak w każdym rozwiązaniu opartym o sesje użytkownika czasem istnieje potrzeba przechowywania informacji wspólnych dla wielu sesji. Powstał w tym celu oddzielny moduł (nie umieszczony na diagramie). Niezależny moduł był niezbędny aby zapewnić wymianę informacji między sesjami pochodzącymi z potencjalnie wielu serwerów aplikacji. Moduł ten został zrealizowany jako serwis Windows.

W artykule przedstawiłem architekturę wdrożonego systemu zarządzania magazynem z wykorzystaniem urządzeń mobilnych. Rozwiązanie jest nowoczesne, zapewnia skalowalność i możliwość dowolnej konfiguracji modułów. W końcu, co najważniejsze, spełnia oczekiwania klienta.



Rysunek 1. *Architektura rozwiązania*

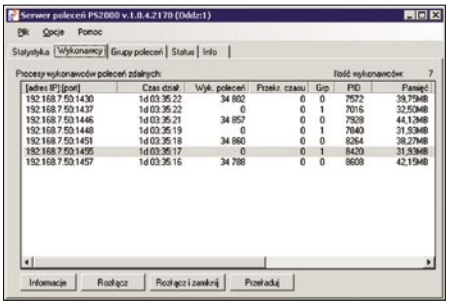


Rysunek 2. *System monitoringu dla serwera aplikacji*

RAFAŁ DEJA

Autor jest doktorem nauk technicznych z zakresu informatyki. Specjalizuje się w projektowaniu rozwiązań webowych opartych na technologiach .NET. Jest Architektem Rozwiązań w Polsoft Engineering Sp. z o.o. oraz wykładowcą na Wyższej Szkole Biznesu.

Kontakt z autorem: rdeja@polsoft.pl.



Rysunek 3. *System monitoringu dla serwera poleceń*

MOTOROLA and the Stylized M Logo are registered in the US Patent & Trademark Office. All other product or service names are the property of their respective owners. © Motorola, Inc. 2008

Jak to się robi w Krakowie

Dla każdego, kto nie miał do czynienia z programowaniem urządzeń mobilnych, specyfika tego zajęcia sprowadza się do kilku znanych haseł, jak real-time, środowisko embedded itp. Wszystko to prawda, jednak wiele z tych haseł straciło ważność, inne zyskały na znaczeniu, a w dodatku pojawiły sie nowe, nigdy niebrane pod uwagę w kontekście systemów embedded.

Dowiesz się:

- Jakie są charakterystyczne cechy programowania urządzeń mobilnych;
- Jak przekładają się one na codzienną pracę na przykładzie radiotelefonów systemu TETRA.

Poziom trudności

Dzisiejsze urządzenia mobilne obejmują tak wiele różnych zagadnień programistycznych, że praktycznie każdy znajdzie tu coś dla siebie. Programista rozproszonych systemów bazodanowych nie zrealizuje może swojego największego projektu, ale z pewnością spędzi trochę czasu, próbując zintegrować silnik swojej bazy z konkretną platformą systemową. Właśnie te szerokie możliwości są charakterystyczne dla współczesnych urządzeń mobilnych. Z jednej strony potrafią zaoferować bardzo rozbudowany, graficzny interfejs użytkownika, czekający tylko na coraz to nowe aplikacje, z drugiej zaś realizują skomplikowaną logikę łączności bezprzewodowej.

Proces tworzenia aplikacji użytkownika na taki system praktycznie nie różni się od tego znanego z komputerów PC. Trzeba dokładnie poznać możliwości konkretnego systemu operacyjnego, ustalić wymagania – i można zaczynać pisanie nowej gry, kalendarza, edytora czy przeglądarki obrazów. Istnieje kilka środowisk developerskich wspomagających tworzenie takich aplikacji (jak choćby WINDEV Mobile czy NetBeans Mobile) a same systemy operacyjne dostarczają szeregu niezbędnych bibliotek, ukrywając sporą część implementacji. Naturalnie liczba systemów operacyjnych używanych w urządzeniach mobilnych mogłaby przyprawić o zawrót głowy, jednak nowe urządzenia coraz częściej korzystają z tych wiodących na rynku.

Na tym tle krakowskie centrum Motoroli cechuje się pewną specyfiką, związaną z obszarem prowadzonej działalności. Wbrew obiegowym skojarzeniom, nie znajdziemy w nim zespołów odpowiedzialnych za rozwój oprogramowania dla telefonów komórkowych, gdyż polski oddział tej firmy skoncentrowany jest na tworzeniu rozmaitych elementów systemów łączności radiowej przeznaczonych (głównie, choć nie wyłącznie) dla służb bezpieczeństwa publicznego. Interesujący nas obszar urządzeń mobilnych reprezentuje w niej ponad stuosobowa grupa architektów, programistów i testerów, posiadająca niemal całkowitą odpowiedzialność za proces tworzenia oprogramowania dla radiotelefonów systemu TETRA.

TErrestrial Trunked RAdio, bo tak właśnie rozwija się ten skrót, jest otwartym standardem cyfrowej komunikacji radiowej opracowanym i rozwijanym w ramach prac European Telecommunications Standardisation Institute (ETSI). Sieci telekomunikacyjne tego typu są tworzone – z wykorzystaniem infrastruktury i terminali różnych producentów – w Europie, Azji czy na Bliskim Wschodzie. Największy system tego typu to Airwave, pokrywający swoim zasięgiem terytorium Wielkiej Brytanii i używany tam m.in. przez policję i straż pożarną.

Jak łatwo się domyślić, kluczowymi aspektami działania radiotelefonów systemu TETRA są niezawodność i bezpieczeństwo. Urządzenia te muszą zapewnić pełną funkcjonalność nawet (czy też – zwłaszcza) w sytuacjach kryzysowych, ale też gwarantować poufność i integralność transmisji. Cele te realizowane są z jednej strony poprzez spełnianie rygorystycznych standardów dotyczących sprzętu, a z drugiej przez implementację różnorodnych typów szyfrowania danych i autentykacji terminali, jak również sieci, w której radia się rejestrują.

Nieco mniej oczywistym elementem tworzenia oprogramowania dla tego typu urządzeń jest złożoność i różnorodność samego standardu TETRA, do której należy dodać specyficzne dla poszczególnych producentów rozszerzenia nie objęte standardem. Na bazie dwóch typów komunikacji: Trunked Mode (TMO – z infrastrukturą) i Direct Mode (DMO – bezpośrednia komunikacja między radiotelefonami) może funkcjonować szeroka gama serwisów opartych zarówno o transmisję głosu jak i danych. Mieszczą się w tym zarówno głosowe połączenia jeden-do-wielu, pełnodupleksowe rozmowy jeden-do-jednego (obsługujące rozbudowany system priorytetyzacji), transmisja wiadomości tekstowych, danych pakietowych, pozycjonowanie GPS i wiele innych. Zakres stwarzanych przez standard możliwości sprawia, że żaden z dostawców terminali TETRA nie dysponuje jego pełną implementacją. Z kolei różnorodny profil klientów (od wojska po komunikację publiczną) generuje zapotrzebowanie na wprowadzanie kolejnych funkcjonalności i rozwiązań sprzętowych.

Profil zmian implementowanych w krakowskim centrum odzwierciedla tę różnorodność: sięga on od wymiany kluczy dla algorytmów szyfrowania poprzez interfejs radiowy po stworzenie protokołu komunikacyjnego pomiędzy radiotelefonem samochodowym a nowym typem głowicy kontrolnej; od funkcjonalności bezpośrednio związanych z warstwą aplikacji, jak obsługa no-

wych zestawów czcionek, po czysto sprzętowe – np. komunikację z nowymi typami akcesoriów; od wprowadzenia obsługi wiadomości tekstowych w trybie DMO po modyfikacje dostosowujące oprogramowanie do działania na terminalu spełniającym standard ATEX (dopuszczony do działania w strefie zagrożonej wybuchem). Często nad jednym feature'em pracują zarówno programiści zajmujący się cyfrowym przetwarzaniem sygnałów, specjaliści od sprzętu, osoby dobrze znające standard i programiści warstwy aplikacyjnej. Ale bez względu na to, czy zmiana dotyczy sprzętu czy interfejsu użytkownika, specyfika programowania systemów wbudowanych dotyka tej pracy niemal zawsze – choć na różne sposoby.

Na poziomie tworzenia warstwy aplikacji wiedza z zakresu systemów czasu rzeczywistego nie jest szczególnie istotna, MIDLet nie musi z reguły uruchamiać się w ściśle określonym czasie. Należy jednak uświadomić sobie jedną zasadniczą różnicę pomiędzy urządzeniem mobilnym a komputerem osobistym. Jeśli w tym drugim zabraknie pamięci na daną aplikację, to najzwyczajniej w świecie powiększamy przestrzeń dyskową. Jeśli aplikacja będzie działała zbyt wolno, to znaczy, że już czas na nowy procesor bądź pamięć RAM. Przycinający się obraz w trakcie filmu lub gry - nowa karta graficzna. Takich możliwości rozbudowy 'on-demand' urządzenie mobilne jest praktycznie pozbawione (podobnie jak większość urządzeń embedded). Nie oznacza to jednak, że końcowa aplikacja uruchamiana w telefonie komórkowym będzie ograniczona funkcjonalnie z powodu jednego z powyższych ograniczeń. Nic z tych rzeczy – musi ona oferować pełny zakres usług w konkretnych warunkach.

Przy braku pamięci na stosie, nie można powiększać go w nieskończoność. Być może należy przeorganizować stos oraz umieścić potrzebne dane w innym miejscu. Jeśli aplikacja zbyt wolno działa, to niewykluczone, że wykonuje wiele niepotrzebnych operacji (jak np. sortowanie posortowanej już tablicy, czy odświeżanie ciągle aktualnego ekranu). Każdy krok, każda operacja powinny być dokładnie przeanalizowane, aby zapewnić jak najefektywniejsze wykorzystanie dostępnych zasobów oraz by nie obciążać nadmiernie całego systemu. Ten rodzaj analizy nie jest szczególnie trudny podczas tworzenia kodu, ale niezwykle czasochłonny na późniejszym etapie. A ponieważ obok wprowadzania nowych platform Motorola utrzymuje i rozwija oprogramowanie dla starszych modeli radiotelefonów (co oczywiste, klienci są o wiele bardziej skłonni do zakupu software'u z nowymi funkcjonalnościami niż wymiany dużej – niekiedy idącej w tysiące – liczby terminali), również ciągła optymalizacja istniejącego kodu pozostaje istotnym elementem kolejnych releasów i swoistym polem do popisu dla programistów, bez względu na to, czy ich obszarem działania jest MMI czy DSP.

Odpowiadając na pytanie, czy aplikacje pisane na urządzenia mobilne należą właśnie do tych najwyższych lotów można zadać kolejne: "Czy sądzicie, że w sprawie prędkości działania np. waszego telefonu komórkowego nic więcej nie trzeba już robić?" Prędkość działania całego systemu to jest właśnie jedna z tych rzeczy, których nie można lekceważyć podczas tworzenia oprogramowania. I chociaż istnieją możliwości zmiany taktowania procesorów, wprowadzania coraz to szybszych jednostek obliczeniowych w kolejnych wersjach sprzętowych – to zabiegi te są dość mocno ograniczone przez jeden bardzo ważny element, który znacząco odróżnia urządzenia mobilne od pozostałych urządzeń embedded. O tym jednak, za chwilę.

Każde urządzenie mobilne musi się jakoś komunikować ze światem zewnętrznym, czy to poprzez łącze szeregowe, bluetooth, Wi-Fi, czy w końcu siecią komórkową. Logikę niektórych protokołów można kupić od zewnętrznego dostawcy. W przypadku radiotelefonów Motoroli można tu wskazać WAP realizowany jako zewnętrzna biblioteka. Z kolei TETRA PDA, nowa, zorientowana na transmisję danych platforma sprzętowa, w warstwie aplikacyjnej w całości opiera się na produktach innych dostawców.

Co jednak, jeśli żaden producent nie oferuje kompletnych rozwiązań w interesującym nas obszarze? Na tym etapie wiedza z zakresu systemów czasu rzeczywistego bardzo sie przyda. To właśnie na tutaj najczęściej można stanąć oko w oko z zagadnieniami teorii współbieżności, takimi jak zagłodzenie, deadlock, priority inheritance. Rozwiązywanie zaawansowanych problemów w tym obszarze potrafi dać wiele przyjemności, jak również przysporzyć wielu zmartwień. Debugowanie kodu na tym etapie nie jest jeszcze niesamowitym wyczynem, wiele urządzeń posiada szereg interfejsów zewnętrznych czy wspiera standard JTAG (co jest też udziałem motorolowych terminali TETRy). Każdy jednak rodzaj debugowania zmienia warunki pracy systemu, co znacznie utrudnia badanie problemów zależności czasowych. Może się na przykład okazać, że bardzo dobrze znana i przetestowana logika nie zadziała poprawnie po wymianie układu pamięci na szybszy i większy. Zapewnienie obsługi zdarzeń w zdefiniowanym i skończonym czasie zmusza niejednokrotnie do stosowania najróżniejszych rozwiązań, niestety również wbrew dobremu stylowi i podstawom praktyk programistycznych. Wywoływanie metod innego taska w sposób jawny, kłóci się przecież z tak piękną ideą systemu message driven, ale jeśli jest jedynym możliwym rozwiązaniem, to nie pozostawia innego wyboru. Każdy przypadek trzeba rozważyć indywidualnie, znalezienie balansu pomiędzy skutecznością a dobrym stylem nie jest wcale takie łatwe. Bardzo często powyższe ograniczenia nie pozwalają zaimplementować funkcjonalności tak, jak byśmy sobie tego życzyli, tak jak nam podpowiada nasz instynkt programistyczny. Jesteśmy spragnieni kodu, z którego bylibyśmy dumni, który nie wymaga poprawek, refaktoringu, i którym można by pochwalić się innym, równie spragnionym programistom (udało się komuś napisać taki kod?). Wznosimy się na wyżyny naszych możliwości, aby chwilę później skonfrontować pierwotne plany z rzeczywistością i uznać wyższość ograniczeń sprzętowych. I ten kompromis pomiędzy wiedzą, chęcią i możliwościami jest również nieodłączną częścią programowania – a w szczególności rozwijania – urządzeń mobilnych.

Dla programistów z lekkim zacięciem elektronicznym idealnym wręcz obszarem wydaje się być warstwa obsługująca sprzęt. Pisanie sterowników nie jest zadaniem trywialnym zarówno w systemach embedded, jak i komputerach osobistych. Znajomość potencjałów poszczególnych układów i zależności pomiędzy nimi pozwala wycisnąć z nich maksimum ich

Rysunek 1. *Różne rodzaje radiotelefonów TETRA*

możliwości. W procesie projektowania nowego układu samodzielnie trzeba zaproponować sprzęt spełniający wymagania przed układem stawiane. Także krakowski zespół – mimo że jego profil jest czysto software'owy – jest częścią procesu decyzyjnego przy opracowywaniu rozwiązań sprzętowych dla nowych platform i modyfikacji platform istniejących.

Kolejnym elementem, na który warto zwrócić uwagę jest właśnie wieloplatformowość. Jeśli firma dobrze się rozwija, to prędzej czy później wprowadzi na rynek kolejne wersje swojego produktu, być może dedykowane potrzebom konkretnego klienta. Oczywiście naturalnym staje się maksymalny re-use istniejącego kodu. Utrzymywanie wielu wersji będzie kosztować mniej im więcej będzie wspólnego kodu. Tutaj również może ujawnić się nowe wymaganie: jeśli kolejna platforma będzie oparta na procesorze o odwrotnym endiannessie, to kod niezależny od endiannessu będzie nieoceniony. Jeżeli chodzi o terminale Motoroli, to obecnie rozwijanych jest siedem platform sprzętowych, startują prace nad kilkoma kolejnymi, a trzy starsze modele jeszcze nie zakończyły swojego żywota. Całkowicie lub w dużej części powstawały w Krakowie: wspomniana nowa głowica kontrolna i radiotelefon ATEX oraz model Covert (jak sama nazwa wskazuje, jest on przeznaczony dla służb nieprzepadających za ostentacyjnym afiszowaniem się tym, że są służbami). Warto też wspomnieć tutaj o kolejnym aspekcie pracy programistycznej w środowisku wieloplatformowym: kluczowym i nietrywialnym zagadnieniem staje się w takiej sytuacji configuration management. Od tego, na ile przemyślany jest system wersjonowania i procedura integracji pracy kilkudziesięciu programistów, od wydajności systemu budowania software'u – zależy produktywność całego zespołu.

Rysunek 2. *Radio systemu TETRA, model MTP850*

Mimo iż podzespoły elektroniczne stają się coraz szybsze i tańsze wielu producentów urządzeń mobilnych nie wymienia ich przy pierwszej nadarzającej się okazji. Jeśli projektujemy całkowicie nową platformę, która ma zastąpić dotychczas istniejącą, to możliwości wydają się być niemal nieograniczone. Co stoi na przeszkodzie? Tutaj dochodzimy do sedna naszych rozważań. Obok oczywistych ograniczeń związanych z kosztami, bardzo poważnym ograniczeniem jest bateria. To ona decyduje o mobilności każdego urządzenia. Nie ma znaczenia, jak wyrafinowane funkcje będzie pełnił system i jak szybko poradzi sobie z najtrudniejszymi zadaniami, jeśli wszystko to odbędzie się kosztem zwiększonej eksploatacji baterii, czyli w rzeczywistości utraty części mobilności.

Kupilibyście telefon o powiększonych możliwościach, który trzeba by ładować w połowie dnia? Jeśli nie byłoby alternatywy, to oczywiście tak, ale w większości przypadków jest to bardzo ważny parametr. Urządzenia mobilne to nie tylko telefony komórkowe (te akurat charakteryzują się całkiem dobrymi parametrami czasów pracy i czuwania). To również urządzenia telemetryczne, gameboys, audio/video recorders i inne. Gdy już raz określimy wymagania co do czasu pracy systemu i przyzwyczaimy końcowego użytkownika, późniejsza degradacja tego parametru kosztem zwiększenia możliwości jest bardzo często niemożliwa. Zaczyna się wtedy rozwiązywanie ograniczeń sprzętowych w oprogramowaniu i jest to ten aspekt pracy z systemami mobilnymi, który będzie towarzyszyć ich twórcom do samego końca.

KAMIL KOWALSKI

Wykształcenie w dziedzinie telekomunikacji, związany z programowaniem radiotelefonów Motoroli od prawie 5 lat. Odpowiedzialny za rozwijanie protokołu komunikacji z infrastrukturą.
Kontakt z autorem: kamil.kowalski@motorola.com

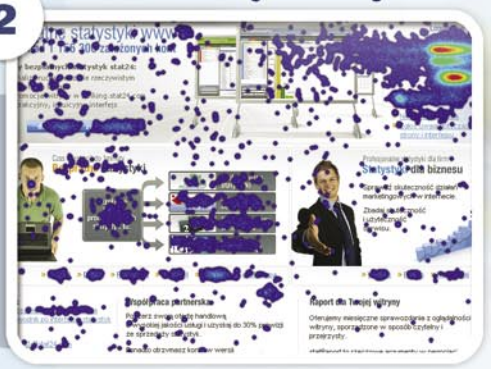
ARTUR CHRUŚCIEL

Z wykształcenia informatyk, w Motoroli pracuje od 2005 roku. Od początku w zespole tworzącym oprogramowanie dla radiotelefonów systemu TETRA. Zajmuje się głównie warstwami związanymi z protokołem komunikacji po interfejsie radiowym w trybie DMO (bez infrastruktury).
Kontakt z autorem: artur.chrusciel@motorola.com

Profesjonalne statystyki zupełnie za darmo?

1 Chcesz wiedzieć kto odwiedza Twoją stronę?

2 W które miejsca użytkownicy klikają najczęściej?



3 Chcesz zamieścić licznik na swojej stronie?

Sprawdź!

Odwiedź stronę www.stat24.com
i zarejestruj bezpłatne konto dla swojej strony.

Bezpłatne statystyki stat24

Poznaj swoją internetową widownię

stat24

Wąż w komórce

Telefony komórkowe obecnie to nie tylko proste urządzenia do wykonywania połączeń głosowych czy korzystania z wiadomości tekstowych i multimedialnych. To zaawansowane mikrokomputery, pełniące często role elektronicznych terminarzy (PDA), wyposażone nierzadko we własny system operacyjny.

Dowiesz się:
- O podstawach systemu operacyjnego Symbian;
- Jak rozpocząć tworzenie aplikacji dla telefonów komórkowych przy użyciu języka Python.

Powinieneś wiedzieć:
- Jak połaczyć telefon z komputerem za pomocą połączenia kablowego lub Bluetooth;
- Jak instalowac i uruchamiac aplikacje w telefonie z Symbianem.

Poziom trudności

Jednych z systemów operacyjnych stosowanych w telefonach komórkowych jest Symbian. System ten został stworzony w oparciu o środowisko EPOC, wykorzystywane w PDA firmy Psion PLC. W skład konsorcjum o takiej samej nazwie, wchodzą takie firmy jak Samsung, Motorola, Siemens, Sony Ericsson i Nokia. I właśnie ta ostatnia firma jest głównym popularyzatorem telefonów z systemem Symbian, modeli ze średniej i górnej półki, których użytkownikami są w dużej mierze klienci biznesowi.

Symbian (najnowsza stabilna wersja to 9.5) daje programiście zbiór bibliotek i funkcji, które umożliwiają pełny dostęp do systemu operacyjnego (włączając w to funkcje niskopoziomowe), peryferiów telefonu i interfejsu użytkownika. W przypadku telefonów Nokia istnieje dodatkowo podział na tzw. serie w zależności do rodzaju interfejsu użytkownika. Obecnie wyróżnia się telefony: Series 60, Series 80 i Series 90. W porównaniu do innych mobilnych systemów operacyjnych Symbian umożliwia programiście na największe możliwości i kontrolę funkcji telefonu.

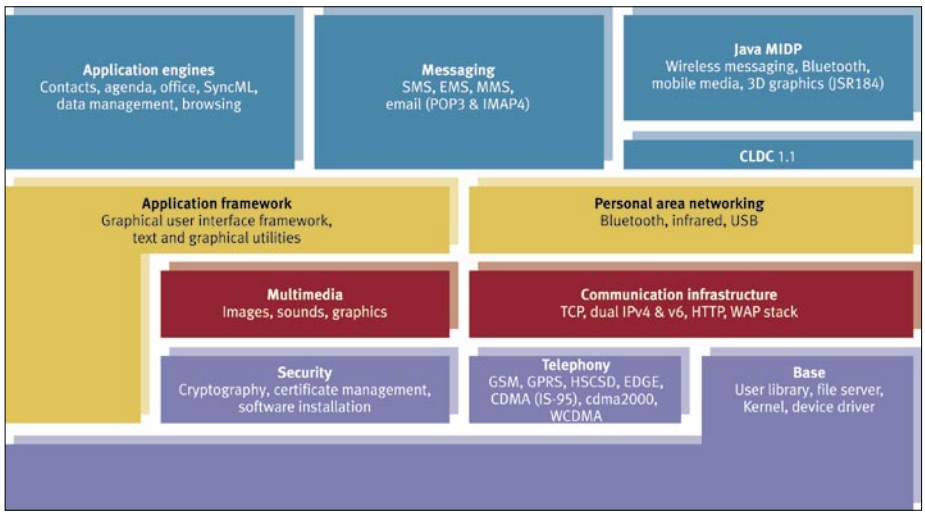
Nokia udostępnia programistom zestaw narzędzi (SDK i emulator) do tworzenia aplikacji dla Symbian OS w języku C++. Mimo że sam język jest znany dość dużej liczbie developerów, to umiejętność programowania dla platformy Symbian OS jest obarczona wysoką *barierą wejścia* z uwagi na dość zawiłe budowanie szkieletu aplikacji, tworzenie GUI, przymus zadbania o przydzielanie i zwalnianie zasobów czy wreszcie sam proces budowania aplikacji.

Inżynierowie z Nokii postanowili więc uprościć nieco proces tworzenia aplikacji dla Symbian OS. Tak więc w 2004 roku powstał projekt o nazwie kodowej Amaretto, który miał na celu uczynić programowanie aplikacji dla Symbiana szybszym i łatwiejszym, a uzyskane w ten sposób aplikacje miały były w pełni funkcjonalne i mogłyby korzystać z większości modułów systemu operacyjnego. W październiku tego samego roku ukazał się port języka Python przeznaczony do uruchamiania skryptów działających w telefonach Nokia Series 60 a projekt zmienił nazwę na *Python for S60* (*pyS60*).

Python dla platformy S60 umożliwia tworzenie aplikacji dla telefonów Nokia Series 60 w postaci skryptów Pythona o wysokim poziomie integracji z funkcjami Symbian OS. Tworzone aplikacje mają taki sam wygląd jak te wykonane przy użyciu C++ SDK. Niewątpliwym plusem takiego rozwiązania jest szybkość tworzenia oprogramowania i łatwość testowania. Obecna wersja pyS60 oparta jest na wersji 2.2.2 Pythona i wspiera takie elementy platformy Symbian jak: dostęp do natywnego GUI, łącza Bluetooth, kamery, książki adresowej, skrzynki SMS, sieci danych, modułu telefonicznego, lokalizacji z wykorzystaniem nadajników BTS itp.

Kilka słów o Pythonie

Python jest obiektowo zorientowanym językiem interpretowanym powstałym w 1991 (jego twórca Guido van Rossum pracuje obecnie dla Google), używany jest najczęściej do tworzenia serwisów internetowych, jak i do tworzenia narzędzi do administrowania systemami operacyjnymi. Dostępne są wersje interpretera języka dla wielu systemów operacyjnych (Windows, Unix, MacOS). Składnia Pythona jest dość ory-



Rysunek 1. *Architektura Symbian OS*

ginalna: zakres pętli czy instrukcji warunkowej jest wyznaczony za pomocą wcięć, jednak dzięki temu programy są przejrzyste i czytelne. Python posiada dynamiczny system typów danych oraz automatyczny system zarządzania pamięcią. Python oferuje zestaw kolekcji takich jak: listy, krotki i słowniki, funkcje rzutowania typów danych (sprawdzane w umiarkowanym stopniu), a także funkcje operujące na łańcuchach znaków. Programujący wcześniej w takich językach jak Perl z pewnością szybko odnajdą się w Pythonie. Ponieważ istotą artykułu nie jest nauka Pythona, pomocne adresy internetowe zostały umieszczone w ramce *W sieci*.

Zestaw startowy

Aby rozpocząć zabawę (i nie tylko) w programowanie dla Symbian OS z wykorzystaniem Pythona potrzebne będą następujące elementy:

- Telefon Nokia Series 60 drugiej lub trzeciej edycji (lista telefonów na *http://www.forum.nokia.com/devices/matrix_all_1.html*).
- Komputer z jednym z systemów z grupy Windows, Unix, MacOS.
- Kabel połączeniowy USB do telefonu lub interfejs Bluetooth w komputerze (do transferu skryptów).

Z sekcji *Download* strony WWW projektu pyS60 (*http://sourceforge.net/projects/pys60*) należy pobrać dwa pliki z rozszerzeniem *SIS* (archiwum Symbian) – *PythonForS60_1_4_4_3rdEd.sis* i *PythonScriptShell_1_4_4_3ndEd.SIS* lub *PythonForS60_1_4_4_2ndEd.SIS* i *PythonScriptShell_1_4_4_2ndEd.SIS* (w zależności od typu posiadanego telefonu).

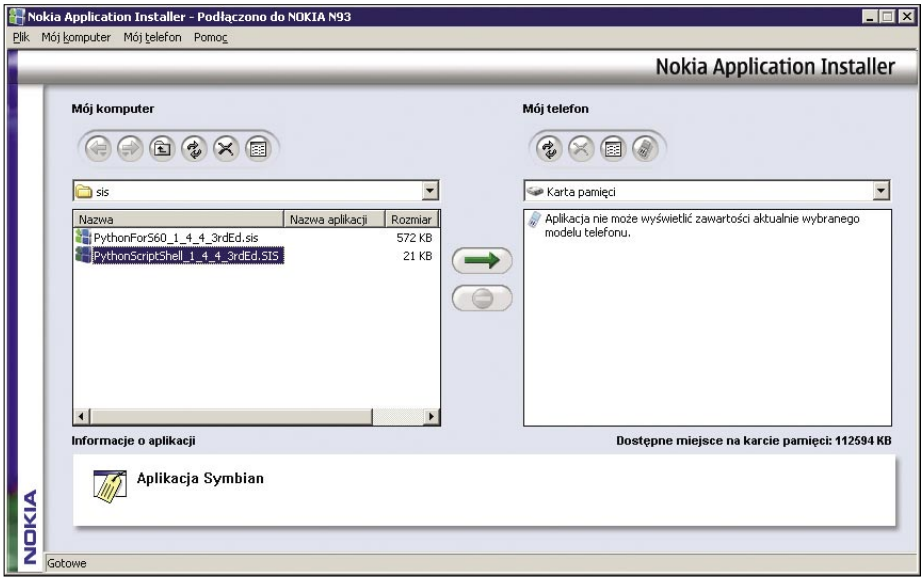
Aby ułatwić sobie pisanie skryptów Pythona, warto zainstalować edytor ConTEXT, PythonWin czy EditPlus choć nic nie stoi na przeszkodzie, aby korzystać z poczciwego Notatnika czy pobrać interpreter Pythona ze strony *http://www.python.org* i korzystać z jego IDE. Programiści korzystający z Eclipse'a mogą pobrać plugin PyDev.

Przygotowane skrypty czy też pobrane pliki SIS muszą zostać załadowane do telefonu, z pomocą przyjdzie tu Nokia PC Suite (*http://www.nokia.pl/A4745165*) – darmowe oprogramowanie umożliwiające m.in. instalację oprogramowania za pomocą kabla USB czy Bluetooth.

Prace przygotowawcze zaczynamy od instalacji interpretera i konsoli Pythona w telefonie. W tym celu uruchamiamy Nokia PC Suite i łączymy się z telefonem za pomocą jednego ze wcześniej wymienionych sposobów. Po szczegóły związane z połączeniem via USB czy Bluetooth odsyłam do instrukcji obsługi konkretnego telefonu. Klikamy ikonę *Instaluj aplikacje*. Wskazujemy w lewym oknie dyskową lokalizację wcześniej pobranych plików SIS, klikamy przycisk oznaczony strzałką i postępujemy zgodnie z instrukcjami dotyczącymi przebiegu instalacji pojawiającymi się na ekranie monitora oraz telefonu. Powyższy proces wykonujemy dla obydwu archiwów Symbian, przy czym jako pierwszy należy instalować plik zgodny z maską *PythonForS60_1_4_4*.sis*.

Po zakończonym procesie instalacji w menu *Osobiste* telefonu powinna pojawić się dobrze znana ikona Pythona. To znak, że środowisko testowe dla naszych skryptów zostało poprawnie przygotowane.

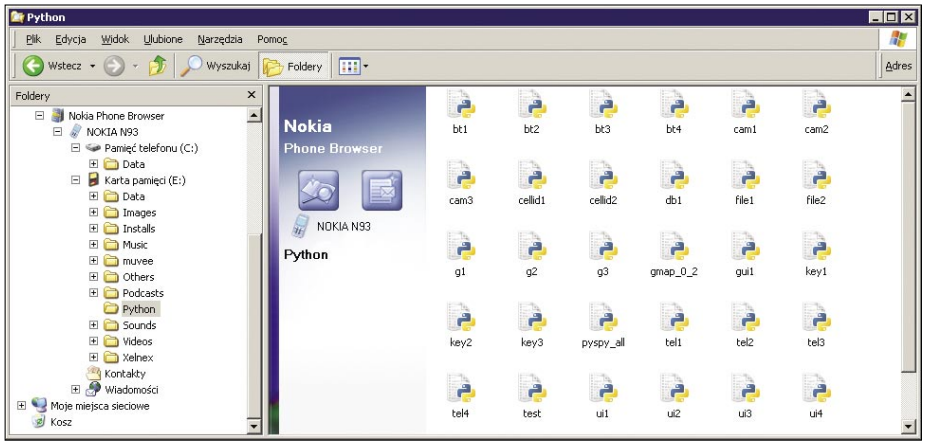
Kliknięcie tej ikonki powinno spowodować uruchomienie shella interpretera Python.



Rysunek 2. *Instalacja aplikacji przy pomocy Nokia Application Installer*

Rysunek 3. *Ikona interpretera Python w telefonie*

Rysunek 4. *Konsola interpretera*



Rysunek 5. *Skrypty Pythona w katalogu na karcie pamięci telefonu*

Na koniec konfiguracji środowiska testowego należy zadbać, aby na karcie pamięci lub w pamięci telefonu znajdował się katalog o nazwie *Python*, w takiej lokalizacji interpreter będzie poszukiwał skryptów do wykonania. Katalog taki możemy utworzyć za pomocą menedżera plików wywoływanego z menu głównego aplikacji Nokia PC Suite. Również za pomocą tego modułu możemy dokonywać niezbędnych operacji na plikach, takich jak kopiowanie plików między dyskami a telefonem.

Hello world!

Naszą przygodę z pyS60 rozpoczniemy od nieśmiertelnego programu typu *Hello word*. Krótki, kilkuwierszowy skrypt będzie pobierał tekst z okna dialogowego, a następnie wyświetlał go w koniunkcji ze słynną sentencją. Otwieramy więc ulubiony edytor programistyczny i wpisujemy 3 linijki (Listing 1.).

Zapisujemy na dysku pod nazwą *hellowold.py*, a następnie kopiujemy do telefonu do katalogu o nazwie *Python*. Uruchamiamy interpreter w telefonie, z menu *Opcje* wybieramy pozycję *Run script*, a następnie z listy skryptów zapisany wcześniej plik *hellowold.py*. Wynik działania aplikacji jest widoczny na rysunkach 6. i 7.

Pora na omówienie kodu. Pierwszą linią kodu importujemy do skryptu bibliotekę odpowiedzialną za dostęp do elementów UI. W kolejnym wierszu wywołujemy funkcję `query()` realizującą wyświetlenie pola tekstowego z możliwością edycji. Jako parametr funkcji podana jest treść komunikatu wyświetlanego nad editboxem. Wynik działania funkcji zwracany jest do zmiennej data, która przechowuje wprowadzony łańcuch znaków. Ostania linia skryptu to wywołanie funkcji wyświetlającej okno informacyjne z podanym tekstem. Ponieważ w Symbianie kodowanie znaków odbywa się z wykorzystaniem Unikodu, przed literałami należy podać literkę *u* w celu właściwego rzutowania tekstu. Użyta w przykładzie funkcja `query()` jako drugi parametr przyjmuje typ wprowadzanej wartości. Obok *text* dostępne są: *number, date, time, query, code*; z kolei drugi parametr funkcji `note()` przyjmować może również wartości *error* lub *conf*. Dobrym źródłem wiedzy na temat funkcji i ich parametrów jest tekst *PyS60 API and Python reference* dostępny na stronach projektu pyS60.

Rysunek 6. *Okno wprowadzania danych*

Python Pizza

W celu szerszego zapoznania się z możliwościami i modułami Pythona dla Symbian OS, przygotujemy przykładową, w pełni funkcjonalną aplikację umożliwiającą zamawianie pizzy przez telefon – tym razem bez użycia głosu. Działanie aplikacji polegać będzie na prezentacji użytkownikowi formularza złożonego z róż-

Listing 1. *Najprostsza aplikacja*

```
import appuifw
data = appuifw.query(u"Type your name:", "text")
appuifw.note(u"Hello world from " + str(data), "info")
```

Listing 2. *Tworzenie menu aplikacji*

```
import appuifw
import e32

def exit_key_handler():
    app_lock.signal()

# funkcje obsługujące zdarzenia menu
def menu1():
    appuifw.note(u"Menu 1", "info")

def menu2a():
    appuifw.note(u"Menu 2 - opcja A", "info")

def menu2b():
    appuifw.note(u"Menu 2 - opcja B", "info")

# utwórz menu apikacji
appuifw.app.menu = [(u"Menu 1", menu1),
                    (u"Menu 2", ((u"Opcja A", menu2a),
                                 (u"Opcja B", menu2b)))]

appuifw.app.exit_key_handler = exit_key_handler
app_lock = e32.Ao_lock()
app_lock.wait()
```

Listing 3. *Szkielet pętli głównej aplikacji*

```
running = 1
while running:
    ##odśwież ekran
    refresh_screen()
        ##sprawdz stan klawiszy

    check_keys()
    ##odczekaj chwilę
    e32.ao_sleep(0.5)
```

Listing 4. *Szkielet pętli głównej aplikacji*

```
# utworzenie tzw. Active Object typu Ao_lock
app_lock = e32.Ao_lock()

# przypisanie obsługi zamknięcia aplikacji
appuifw.app.exit_key_handler = exit_key_handler

# aplikacja oczekuje na zdarzenia od UI
# do momentu wywołania app_lock.signal()
# kiedy to powoduje zakończenie działania aplikacji
app_lock.wait()
```

nego rodzaju pól, z którego dane po zapisaniu będą odpowiednio formatowane i wysyłanie w formie SMS-a do pizerii.

Na samym początku kodu deklarujemy za pomocą dyrektywy *import*, jakich będziemy używać modułów, możemy również określić wybrane funkcje modułów poprzez składnie *from nazwa_modułu import nazwa_funkcji1, nazwa_funkcji2*. W opisywanym przykładzie niezbędne będą funkcje operujące na UI telefonu (*appiufw*), umożliwiające wysyłanie wiadomości SMS (*messaging*), a także konwertujące czas na tekst (`localtime`, `strftime`). A ponieważ Python umożliwia programowanie obiektowe, utworzymy klasę `PizzaFormView`, w której umieścimy metody wykorzystywane przez aplikację. Oczywiście kod zawierać będzie również *sekcję startową* inicjalizującą całą aplikację. W konstruktorze klasy inicjalizowana jest zmienna `_iIsSaved` informująca czy użytkownik zapisał zmiany w formularzu. Następnie występuje deklaracja list przechowujących rodzaje typów pizzy oraz dodatków do nich. Kierując się dobrze rozumianą ergonomią, będziemy wymagali od użytkownika wyboru wyżej wymienionych wartości tylko z predefiniowanej listy. Na końcu konstruktora definiowana jest również lista definiująca pola formularza, tym razem składającą się z tzw. krotek. W odróżnieniu od typowej listy, krotki (ang. *tuple)* nie mogą być modyfikowane w trakcie działania aplikacji.

Metoda `setActive()` odpowiada już za utworzenie właściwego formularza z wykorzystaniem funkcji appuifw.Form, przypisanie do pola *save_hook* obiektu *_iForm* funkcji, która będzie notyfikowana o zapisie bądź odrzuceniu zmian wartości pól i wyświetlenie całości poprzez wywołanie metody `execute()`. Klasa zawiera również szereg funkcji odpowiedzialnych za pobieranie informacji o wprowadzonych wartościach do poszczególnych pól formularza: `getPizzaName()`, `getAmount()` itd.

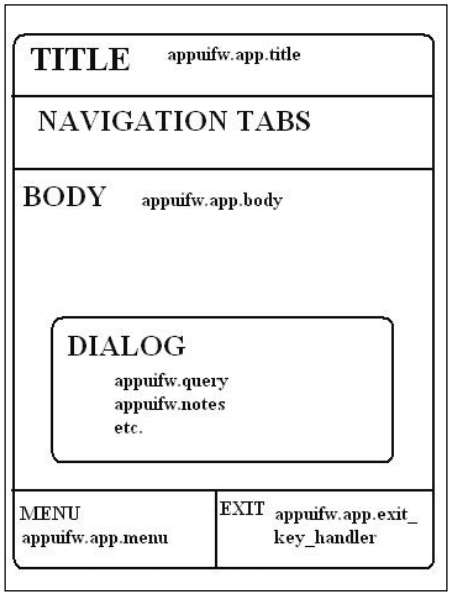
Sekcja startowa skryptu ustawia odpowiedni tytuł aplikacji u góry okna, inicjalizuje obiekt klasy `PizzaFormView` i go wykonuje. Od tego momentu aplikacja przechodzi w tryb interakcji z użytkownikiem, a w momencie naciśnięcia przycisku *Wróć*, pojawia się monit z zapytaniem o zapis wprowadzonych danych, skutkiem czego będzie ustawienie odpowiedniej wartości logicznej zmiennej `_iIsSaved`.

Jeśli zmiany zostaną potwierdzone, nastąpi odczytanie poszczególnych pól formularza. W przypadku pól z wyborem, na podstawie indeksu musimy zlokalizować wartość tekstową z listy, a dla pola z maską czasu – dokonać konwersji na czytelną dla człowieka postać z użyciem funkcji `localtime()` i `strftime()`. Końcowym elementem jest odpowiednie sformatowanie łańcucha znaków, a następnie wysłanie SMS-em pod wskazany numer funkcją `messaging.sms_send(numer.txt)`. Aplikacja kończy się komunikatem informacyjnym. Odbiorca (pizzeria) otrzyma wiadomość podobną do tej z rysunku 10.

Rysunek 7. *Monit informacyjny*

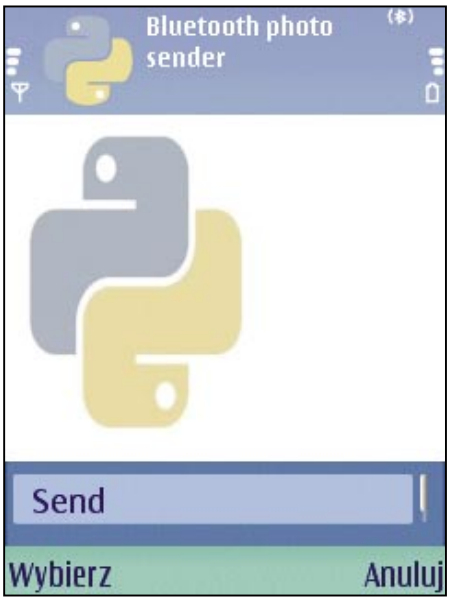
Rysunek 10. *Treść wysyłanego przez aplikację SMS-a*

Rysunek 8. *Interfejs użytkownika aplikacji*



Rysunek 11. *Obszary ekranu aplikacji*

Rysunek 9. *Monitor z potwierdzeniem*



Rysunek 12. *Ekran aplikacji*

Odrobina teorii

Najlepszą praktyką podczas nauki programowania jest tworzenie niewielkich aplikacji. Nie zaszkodzi jednak opisać kroki, jakie należy wykonać w celu stworzenia aplikacji z wykorzystaniem pyS60.

Ekran

Ekran telefonu komórkowego można podzielić obszary przedstawione na Rysunku 11.

Każdy z nich reprezentowany jest przez odpowiednią zmienną, tak więc, aby ustawić tytuł aplikacji, należy przypisać wartość do *appuifw.app.title*, a obsługę zakończenia aplikacji podpiąć do *appuifw.app.exit_key_handler*.

Import modułów

Na początku aplikacji należy zaimportować moduły lub dane funkcje z modułów.

Określenie wielkości ekranu aplikacji i tytułu aplikacji.

Zmienna `appuifw.app.screen` może przyjmować wartości: *normal* (widoczny pasek tytułu i etykiety klawiszy funkcyjnych), *large* (widoczne tylko etykiety klawiszy funkcyjnych) lub *full* (tryb pełnoekranowy). Aby ustawić tytuł aplikacji, należy przypisać wartość do zmiennej `appuifw.app.title`.

Menu aplikacji

Do zmiennej `appuifw.app.menu` należy przypisać listę krotek, które będą składać się z dwóch elementów: nazwy pozycji menu oraz nazwy funkcji obsługującej wybranie pozycji menu. Oczywiście dozwolone jest zagnieżdżanie krotek. (Listing 2)

Pętla główna aplikacji

Utworzenie pętli głównej polega na odpowiedniej konstrukcji pętli *while*; zgodnie z przykładem z Listingu 3.

Tuż przed powrotem do warunku sprawdzającego warto odczekać chwilę wywołując metodę `e32.ao_sleep()`, zatrzymującą wykonywanie aplikacji bez blokowania zdarzeń systemowych czy interfejsu użytkownika. Dla aplikacji, które nie muszą mieć pętli głównej programu, gdyż realizują prosty scenariusz polegający na wprowadzeniu danych, wykonaniu operacji i poinformowaniu o wyniku użytkownika wystarczy konstrukcja przedstawiona na Listingu 4.

Rysunek 13. *Lista urządzeń BT*

Bluetooth i moduł telefoniczny

Symbian OS oferuje programiście bardzo szeroki dostęp do takich peryferiów jak moduł telefoniczny czy interfejs Bluetooth. Pierwszy z przedstawionych programów realizuje wysyłanie pliku graficznego przez Bluetooth z wykorzystaniem protokołu OBEX. Oprócz znanych z poprzednich skryptów modułów, w tym przykła-

Listing 5. *Wysyłanie plików przez Bluetooth*

```
import e32, socket, appuifw, graphics

##lokalizacja pliku graficznego do wysłania
IMAGE = u"e:\\Images\\python-logo.jpg"
##funkcja wysyłająca grafikę przez Bluetooth
def send_photo():
    try:
            ## skanowanie urządzeń BT w pobliżu
                address, services = socket.bt _ obex _ discover()
    except:
            ## komunikat wyjątku
            appuifw.note(u"OBEX Push not available", "error")
            return
    ## wyszukianie protokołu OBEX Object Push
    if u'OBEX Object Push' in services:
            ## odczyt kanału
            channel = services[u'OBEX Object Push']
                ## wysłanie pliku na podany adres urządzenia i poprzez
                ## wskazany numer kanału
            socket.bt_obex_send_file(address, channel, IMAGE)
            appuifw.note(u"Photo sent", "info")
    else:
            appuifw.note(u"OBEX Push not available", "error")

## obsługa klawisza Wyjdź
def quit():
    app_lock.signal()

## ekran aplikacji pracuje w trybie graficznym
## pobranie uchwytu ekranu do operacji graficznych
canvas = appuifw.Canvas()

## przypisanie do zmiennej
appuifw.app.body = canvas

## załadowanie grafiki do pamięci
img = graphics.Image.open(IMAGE)

## wyświetlenie na ekranie (bez skalowania)
canvas.blit(img, scale = 0)

## określenie funkcji obsługującej
## zakończenie aplikacji
appuifw.app.exit_key_handler = quit

## ustaw tytuł aplikacji
appuifw.app.title = u"Bluetooth photo sender"
## definicja menu
appuifw.app.menu = [(u"Send", send_photo)]
## utworzenie Active Object (odbiór zdarzeń z UI)
app_lock = e32.Ao_lock()
app_lock.wait()
```

dzie konieczne jest zadeklarowanie dodatkowo pakietów *socket* i *graphics,* przez co otrzymamy odpowiednio dostęp do funkcji operujących na gniazdach oraz do funkcji graficznych. W tym przykładzie po raz pierwszy uzyskamy dostęp do ekranu aplikacji metodą `appuifw.Canvas()` *a* wynik tej operacji zostanie przypisany do zmiennej `appuifw.app.body`. W dalszej kolejności zostanie załadowany do pamięci plik graficzny, a następnie zostanie on wyświetlony bez skalowania na ekranie (Rysunek 12).

```
canvas = appuifw.Canvas()
appuifw.app.body = canvas
img = graphics.Image.open(IMAGE)
canvas.blit(img, scale = 0)
```

W Sieci

- Strona projektu PyS60 – *http://sf.net/projects/pys60*;
- PyS60 Library Reference – *http://downloads.sourceforge.net/pys60/PythonForS60_1_4_4_doc.pdf*;
- Wszystko o Symbianie – *http://www.symbian.com/*;
- Wszystko o Pythonie (kursy, artykuły, forum) – *http://www.python.org.pl/*;
- Plugin Pythona do Eclipse'a – *http://pydev.sourceforge.net/*;
- E-book *Dive Into Python* – *http://diveintopython.org/*;
- Python Wiki Nokia – *http://wiki.forum.nokia.com/index.php/Category:Python*;
- Nokia OpenSource – *http://wiki.opensource.nokia.com/projects/PyS60*;
- pyS60's Snippets – *http://snippets.dzone.com/tag/pys60*;
- Forum Nokia – *http://discussion.forum.nokia.com/forum/forumdisplay.php?f=102*;
- Protokół OBEX – *http://en.wikipedia.org/wiki/OBEX.*

Listing 6. *Rejestrator rozmów*

```
import appuifw
import e32
import telephone
import audio
#ścieżka do pliku z nagraniem
AUDIOFILE = "E:\\Sounds\\call.wav"
sound = None
recording = False
## funkcja przechwytująca status połączenia
def handleCall((callState, number)):
   #status zakończenia rozmowy
   if callState == telephone.EStatusDisconnecting:
      if recording == True:
         print "end of call"
         sound.stop()
         sound.close()
         recording = False
   #odebranie rozmowy
   if callState == telephone.EStatusAnswering:
      if number == "":
         print "incoming unknown call-id"
         recording = True
         sound = audio.Sound.open(AUDIOFILE)
         sound.record()
      else:
         print "call from " + number

def quit():
   app_lock.signal()
## określenie funkcji obsługującej
## zakończenie aplikacji
appuifw.app.exit_key_handler = quit
#ustaw tytuł aplikacji
appuifw.app.title = u"Call Manager"
#określ funkcję odczytującą status rozmowy
telephone.call_state(handleCall)
## utworzenie Active Object (odbiór zdarzeń z UI)
app_lock=e32.Ao_lock()
app_lock.wait()
```

Po wybraniu odpowiedniej opcji z menu aplikacji rozpocznie się proces wyszukiwania znajdujących się w pobliżu urządzeń z interfejsem Bluetooth za pomocą funkcji `socket.bt_obex_discover()` (Rysunek 13).

Po wybraniu danego urządzenia zwrócona zostanie struktura, na którą składa się unikatowy numer interfejsu Bluetooth urządzenia oraz informacja o nazwie usługi i numerze kanału, na którym jest ona dostępna. W dalszej części następuje sprawdzenie czy usługa nosi nazwę *OBEX Object Push* – jeśli tak, odczytywany jest numer kanału i wywoływana jest funkcja `socket.bt_obex_send_file()` wysyłająca plik.

Kolejną aplikacją demonstrującą dostęp do funkcji modułu telefonicznego jest rejestrator rozmów, który będzie automatycznie nagrywał każdą przychodzącą rozmowę z zastrzeżonego numeru telefonu.

Najważniejszym punktem aplikacji jest wskazanie jako dla `telephone.call_state()` funkcji, która będzie odpowiedzialna za identyfikację stanu połączenia oraz numeru telefonu. W prezentowanym przykładzie funkcja `handleCall()` bada parametr *callState* - lista stałych opisujących status połączenia znajduje się w dokumencie *PyS60 Library Reference* – i podejmuje akcję rejestrowania połączenia do pliku WAV. Nagrywane są wyłącznie połączenia, dla których nie można zidentyfikować numeru dzwoniącego (parametr *number* funkcji jest pusty). Aby uzyskać dostęp do funkcji rejestrujących dźwięk, konieczne jest skorzystanie z modułu *audio*, wywołanie metody `record()` podczas trwania rozmowy powoduje rozpoczęcie jej nagrywania.

Słowem zakończenia

Mam nadzieję, że zaprezentowany artykuł gładko wprowadził czytelników w temat programowania w Pythonie dla Symbian OS. Oczywiście jest to tylko wprowadzenie do tematu, ale mimo to myślę, że już dziś, po przeczytaniu tego artykułu, zwiększyła się rzesza developerów chętnych do programowania w tej technologii. Z pewnością nie można przejść obok szybkości tworzenia i debugowania aplikacji, dużej liczby dodatkowych modułów, możliwości kompilacji skryptów do postaci archiwum Symbian (pliki SIS) czy wsparcia stale rosnącej społeczności korzystającej z Pythona.

MARIAN WITKOWSKI

Marian Witkowski jest miłośnikiem oraz entuzjastą wykorzystywania technologii mobilnych w życiu codziennym. Cenne doświadczenie w zakresie usług mobilnych nabył podczas współpracy w latach 2001-2006 z firmą One-2-One S.A. zajmującą się integracją usług dodanych w telefonii komórkowej. Prowadzi własną firmę świadczącą usługi informatyczne w zakresie tworzenia mikroprocesorowych systemów czasu rzeczywistego, technologii mobilnych oraz układów automatyki.

Kontakt z autorem: mw@it-mobile.net

GPS z .NET Compact Framework

Chip GPS coraz częściej znajduje się w niedrogich telefonach oraz urządzeniach PDA z zainstalowanym systemem Windows CE lub Mobile. Z dobrą baterią można takie urządzenie zabrać nawet na całodzienną wędrówkę i późniejsze pytania *gdzie byłeś?* skwitować gotowym zapisem trasy.

Dowiesz się:

- Jak napisać prosty program na urządzenie przenośne wykorzystując Microsoft .NET Compact Framework.
- Jak komunikować się z Chipem GPS poprzez port szeregowy.
- Jak przeliczyć współrzędne przesyłane przez GPS na format UTM.

Powinieneś wiedzieć:

- Posiadać znajomość podstaw .NET i C#.
- Jak korzystać z Microsoft Active Sync.

Poziom trudności

Niedawno otrzymałem nowy telefon T-Mobile MDA Compact III, znany również jako HTC 3300, wyposażony w chip GPS SiRF STAR III oraz Windows Mobile 6. Natychmiast postanowiłem wykorzystać nawigację GPS do swoich całodziennych wędrówek i zamiast studiować możliwości, cenniki i instrukcje obsługi dostępnego oprogramowania nawigacyjnego zdecydowałem się na napisanie prostego programu pokazującego aktualną pozycję, którą potem można by skonsultować z drukowaną mapą, i tak zawsze obecną w plecaku i dużo wygodniejszą w porównaniu ze skromnymi możliwościami wyświetlacza 240x320. Dodatkowy regularny zapis pozycji miał gwarantować wieczorną rozrywkę w gronie przyjaciół przed ekranem Google Earth.

Szybka konsultacja z Google zdradziła istnienie zgrabnej biblioteki programistycznej GPS Intermediate Driver, zawartej w Microsoft Windows Mobile SDK, dobrze udokumentowanej, opatrzonej przejrzystymi przykładami zarówno w C, jak i C# oraz ... niedziałającej z moim telefonem i systemem operacyjnym ([WEB01]).

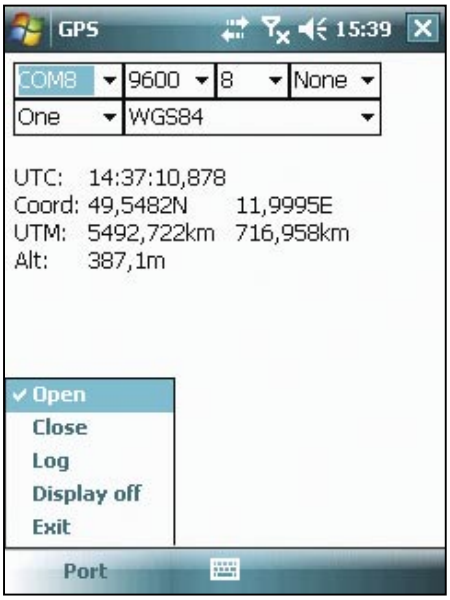
Na szczęście z GPS-em można komunikować się za pomocą zwykłego portu szeregowego i można obyć się bez dodatkowych API.

Obsługa portu szeregowego

Do obsługi portu szeregowego w .NET służy klasa `Serial.IO.Ports.SerialPort`. Za pomocą własności `PortName`, `BaudRate`, `Databits`, `Parity`, `StopBits` ustala się parametry komunikacji szeregowej, statyczna metoda `GetPortNames()` zwraca nieposortowaną tablicę nazw portów szeregowych dostępnych w systemie, a metody `Open()` oraz `Close()` otwierają i zamykają port szeregowy. Do odczytu portu służy kilka wersji metody `Read()` `ReadByte()`, `ReadChar()` oraz `ReadLine()`. Wszystkie te metody są blokujące, praca programu zostaje wstrzymana aż do czasu pojawienia się jakiejkolwiek informacji na porcie, w przypadku metody `ReadLine()` aż do pojawienia się znaku końca linii. Własność `ReadTimeout` klasy określa czas w milisekundach, po którym wygenerowany zostanie wyjątek `System.TimeoutException`, jeśli na porcie nie pojawi się żadna informacja, względnie znak końca linii. Standardowa wartość tej własności to stała `InfiniteTimeout`.

Wybór właściwej metody do odczytu portu szeregowego ma ogromny wpływ na efektywność, czy w ogóle możliwość działania programu. Chipy GPS generują duże ilości informacji i np. metoda `ReadLine()` kończyła się wyjątkiem z enigmatycznym wyjaśnieniem o przepełnieniu dostępnego bufora, natomiast odczyt bajtu po bajcie za pomocą metody `ReadByte()` skutecznie powstrzymywał telefon od robienia czegokolwiek poza czytaniem portu.

Każda przyjazna dla użytkownika aplikacja, a szczególnie aplikacja przeznaczona dla tak powolnych urządzeń jak PDA, powinna umożliwić w każdym momencie jej przerwanie czy też uruchomienie innego programu lub funkcji. Oznacza to, że po pierwsze należy koniecznie ustawić własność `ReadTimeout` na wartość mniej więcej między 500-2000 ms, nie za mało, aby nie przemęczać małego komputerka i nie za dużo, aby jednak zapewnić reakcję programu w przewidywalnym czasie. Po drugie odczyt portu powinien odbywać się w oddzielnym, najlepiej przeznaczonym tylko do tego celu wątku. Znawcy platformy .NET i ogólnie programowania systemów MS Windows wypomną mi jeszcze istnienie w klasie `SerialPort` zdarzenia `DataReceived`, wywołującego zdefiniowanego delegata w momencie pojawienia się na porcie informacji. Niestety, tu mój telefon sprawił mi niemiłą niespodziankę, ponieważ port szeregowy GPS-u nie zgłaszał żadnych zdarzeń, ani w .NET, ani w C (porównaj również [WEB02]), podczas gdy np. na porcie szeregowym do komunikacji z komputerem desktop zdarzenia generowane były bez zarzutu.



Rysunek 1. *Program sgps.exe*

Listing 1. zawiera uproszczoną wersję klasy otwierającej i odczytującej port szeregowy COM4, COM4 to port programowego dostępu do GPS na HTC3300 Najważniejszą część klasy stanowi metoda `readFkt()`. Wywołanie

```
len = this.m_serial.Read(this.m_buf, 0,
                         Serial.BUFLEN);
```

powoduje odczytanie maksymalnie 4096 bajtów, przy czym dokładna ilość odczytanych bajtów zwracana jest do zmiennej `len`. Jeśli w ciągu 2 sekund na porcie GPS-u nie pojawi się ani jeden bajt, wtedy generowany jest wyjątek `TimeoutException`, który w tej metodzie jest po prostu ignorowany. Przed przystąpieniem do kolejnego odczytu sprawdzana jest wartość zmiennej `m _ doread`, którą można ustawić na `false` w implementacji własności `Read`. W ten sposób w każdej chwili można przerwać odczyt portu i metoda `readFkt()` wraz ze swoim wątkiem zakończy działanie najpóźniej po około 2 sekundach.

Po udanym odczycie w tablicy `m_buf` znajduje się maksymalnie 4096 bajtów zawierających potencjalnie wiele linii tekstu oddzielonych za pomocą znaków karetki, końca linii lub (najczęściej) obydwu. Jeśli ustawiliśmy wcześniej zmienną `m_rdlinecb` za pomocą własności `RdLineCb`, a zatem jeśli zdefiniowaliśmy funkcję `callback` (delegata) do obsługi odebranych linii tekstu, to metoda przekształca tablicę bajtów z `m_buf` na listę łańcuchów znaków i przekazuje ją jako parametr do naszego delegata. W ten sposób klasa pozwala na elastyczne zdefiniowanie własnej obsługi każdej odczytanej linii tekstu lub po prostu na zaniechanie robienia z tą informacją czegokolwiek. W tym ostatnim wypadku program

Listing 1. *Klasa do obsługi portu szeregowego*

```
using System;
using System.Collections;
using System.IO.Ports;
using System.Text;
using System.Threading;
public delegate void ReadLineDelegate(ArrayList arr);
public class Serial {
  private const int BUFLEN = 4096;
  private const int DEFAULT_TIMEOUT = 2000;
  private SerialPort m_serial;
  private Thread m_thr;
  private ReadLineDelegate m_rdlinecb;
  private StringBuilder m_sbline;
  private byte[] m_buf;
  private bool m_doread;
  public Serial() {
    this.m_buf = new byte[Serial.BUFLEN];
    this.m_sbline = new StringBuilder(Serial.BUFLEN);
    this.m_serial = new SerialPort();
    this.m_serial.ReadTimeout = Serial.DEFAULT_TIMEOUT;
    this.m_serial.PortName = "COM4";
    this.m_thr = null;
    this.m_rdlinecb = null;
    this.m_doread = false;
  }
  public ReadLineDelegate RdLineCb {
    set { this.m_rdlinecb = value; }
  }
  public virtual void Open() {
    this.Close();
    this.m_serial.Open();
    this.Read = true;
  }
  public void Close() {
    this.Read = false;
    if (this.m_serial.IsOpen) this.m_serial.Close();
  }
  public bool Read {
    get {
      if (this.m_thr == null) return false;
      return true;
    }
    set {
      if (value) {
        if (this.m_serial.IsOpen) {
          this.Read = false;
          this.m_doread = true;
          this.m_thr = new Thread(this.readFkt);
          this.m_thr.IsBackground = true;
          this.m_thr.Start();
        }
      }
      else {
        this.m_doread = false;
        if (this.m_thr != null) {
          this.m_thr.Join();
          this.m_thr = null;
        }
      }
    }
  }
  private void readFkt() {
    ArrayList arr;
    string s;
    int len, i;
    char  c;
    if (this._serial.IsOpen) {
      while (this.m_doread) {
        try {
          len = this.m_serial.Read(this.m_buf, 0, Serial.BUFLEN);
          if (len > 0) {
            if (this.m_rdlinecb != null) {
              arr = new ArrayList();
              for(i = 0; i < len; i++) {
                c = Convert.ToChar(this.m_buf[i]);
                if (c == '\r' || c == '\n') {
                  if (this.m_sbline.Length > 0) {
                    s = this.m_sbline.ToString();
                    arr.Add(s);
                    this.m_sbline.Length = 0;
                  }
                }
                else this.m_sbline.Append(c);
              }
              if (this.m_doread && arr.Count > 0) this.m_
                      rdlinecb(arr);
            }
          }
        }
        catch(TimeoutException) { }
      }
    }
  }
}
```

oparty o opisywaną klasę może posłużyć np. do rozładowania baterii PDA.

Sentencje NMEA 0183

National Marine Electronics Association, w skrócie NMEA, to organizacja zajmująca się ujednoliceniem standardów elektroniki stosowanej w marynarce ([WEB03]). Ogromna większość chipów GPS generuje tzw. sentencje NMEA 0183 i ogromna większość z nich czyni to standardowo tuż po otwarciu portu. Doskonały zestaw dokumentacji przeróżnych implementacji chipsetu SiRF znajduje się pod adresem [WEB04]. Opis samych łańcuchów NMEA można znaleźć pod adresem [WEB05]. Najczęściej wykorzystywaną sentencją NMEA jest tzw. string GGA, zawierający zarówno aktualne współrzędne, jak i wysokość nad poziomem morza. Sentencja ta ma format przedstawiony na Listingu 2. i składa się z poniższych pól oddzielonych przecinkami:

- Każda sentencja NMEA 0183 zaczyna się od znaku dolara i dwóch dowolnych znaków, za którą następuje trzyliterowe oznaczenie sentencji. W przypadku SiRF tuż za dolarem znajdują się znaki SP a GGA to oznaczenie typu sekwencji.
- Czas UTC.
- Długość geograficzna w formacie stopnie, minuty, sekundy, części dziesiętne sekund. Np. 1234.5678 oznacza 12 stopni, 34 minuty i 56.78 sekundy.
- N (północ) lub S (południe)
- Szerokość geograficzna w podobnym formacie jak długość.
- E (wschód) lub W (zachód)
- Wskaźnik jakości sygnału GPS.
- Ilość widocznych satelitów.
- Niedokładność pozioma w metrach.
- Wysokość nad poziomem morza.
- Jednostka wysokości nad poziomem morza, M dla metrów.
- Różnica między elipsoidą WGS84 i poziomem morza.
- Jednostka dla 11, M dla metrów.
- Wiek danych różnicowych GPS.
- ID stacji odniesienia.
- Suma kontrolna.

Listing 2. *Format sentencji GGA*

```
0      1         2       3 4        5 6 7  8   9  10 |  12 13  14   15
|      |         |       | |        | | |  |   |   | |  | |   |    |
$--GGA,hhmmss.ss,llll.ll,a,yyyyy.yy,a,x,xx,x.x,x.x,M,x.x,M,x.x,xxxx*hh
```

Przykład

```
$GPGGA,200329.000,4932.7248,N,01159.2531,E,
1,06,1.5,401.0,M,47.4,M,,0000*5A
```

oznacza czas UTC 20:03:29, 49°32'72.48"N, 11°59'25.31"E, 401 metrów nad poziomem morza.

Z powyższych pól interesujące są długość, szerokość geograficzna, wysokość nad poziomem morza oraz jednostki, czyli pola 2, 3, 4, 5, 9 oraz ewentualnie jeszcze 10, choć jednostką wysokości

Listing 3. *Metoda Utils.fromGGA()*

```
sing System;
using System.Globalization;
public class Utils {
  private static double dms2dec(double dms) {
    double deg, min, deg100, part;

    deg = Math.Floor(0.01 * dms);
    deg100 = 100 * deg;
    min = Math.Floor(dms - deg100);
    part = dms - deg100 - min;
    return deg + (min + part)/ 60;
  }
  public static bool fromGGA(out double utcTime,
                             out double n_lat,
                             out double e_long,
                             out double altitude,
                             string     gga) {
    const NumberStyles ST = NumberStyles.AllowDecimalPoint |
                            NumberStyles.AllowLeadingSign |
                            NumberStyles.AllowLeadingWhite |
                            NumberStyles.AllowTrailingWhite;
    string[] sa;
    double lat, lon, tim, alt;
    int i;
    bool retval;
    retval = false;
    utcTime = 0.0;
    n_lat = 0.0; e_long = 0.0; altitude = 0.0;
    if (string.IsNullOrEmpty(gga) || gga.Length <= 6 ||
        gga[0] != '$' || gga[3] != 'G' || gga[4] != 'G' ||
                         gga[5] != 'A') {
      goto end;
    }
    sa = gga.Split(Utils.GGASEPS);
    if (sa.Length < 10) goto end;
    for(i = 0; i < 10; i++) if (string.IsNullOrEmpty(sa[i]))
                              goto end;
    try {
      tim = Double.Parse(sa[1], ST, CultureInfo.InvariantCulture);
      lat = Double.Parse(sa[2], ST, CultureInfo.InvariantCulture);
      lon = Double.Parse(sa[4], ST, CultureInfo.InvariantCulture);
      alt = Double.Parse(sa[9], ST, CultureInfo.InvariantCulture);
    }
    catch(Exception) {
    }
    lat = Utils.dms2dec(lat);
    lon = Utils.dms2dec(lon);
    switch(sa[3].Trim()) {
      case "N" :
      case "n" : break;
      case "S" :
      case "s" : lat = -lat; break;
      default : goto end;
    }
    switch(sa[5].Trim()) {
      case "E" :
      case "e" : break;
      case "W" :
      case "w" : lon = -lon; break;
      default : goto end;
    }
    utcTime = tim;
    n_lat = lat; e_long = lon; altitude = alt;
    retval = true;
    end:
    return retval;
  }
}
```

są zawsze metry. Listing 3. przedstawia przykład metody interpretującej sentencję GGA.

Metoda sprawdza w najprostszy sposób format przekazanego łańcucha znaków i usiłuje zinterpretować pole drugie, trzecie, piąte i dziewiąte jako liczby określające długość, szerokość geograficzną oraz wysokość nad poziomem morza. Proszę zwrócić uwagę na zastosowanie stałej `System.Globalization.CultureInfo.InvariantCulture` przekazanej do metody `Parse()`. W naszej strefie kulturowej część dziesiętna liczby zmiennoprzecinkowej oddzielana jest od całkowitej za pomocą przecinka, podczas gdy łańcuchy NMEA stosują w tym miejscu typową dla krajów anglosaskich kropkę. Bez zastosowania stałej `InvariantCulture` decyzja o użyciu kropki czy przecinka powzięta by została na podstawie aktualnych ustawień regionalnych komputera i z dużym prawdopodobieństwem wybór padłby na niewłaściwy w tym miejscu przecinek. Odczytanie współrzędnych z sentencji NMEA może się często nie udać, np. gdy odbiornik GPS straci kontakt z satelitami lub jeszcze go nie nawiązał. Metoda `Utils.dms2dec()` zamienia współrzędne z sentencji NMEA na postać dziesiętną, tzn. np. współrzędna 01159.2531 (11°59'25.31") zostanie przeliczona na 11,987552°. Dalej przyjmuje się, że współrzędne północne i wschodnie są dodatnie, natomiast południowe i zachodnie zwracane są jako współrzędne ujemne. Odczytany czas, współrzędne oraz wysokość metoda `Utils.fromGGA()` zwraca w parametrach wyjściowych `utcTime`, `n_lat`, `e_long` oraz `altitude`.

Współrzędne UTM

UTM (*Universal Transverse Mercator Projection*) to współrzędne określające odległość w metrach lub kilometrach od równika oraz od najbliższego z rozmieszczonych co sześć stopni południków (-3°, 3°, 9°, 15°, 21° itd.). Współrzędne UTM mogą różnić się typem zastosowanej elipsoidy rotacyjnej, przy czym najpopularniejszy jest system WGS84. Wiele dokładnych map, czyli tych najbardziej przydatnych do wędrówek po bezdrożach, oraz planów miast zawiera właśnie siatkę współrzędnych UTM zamiast współrzędnych biegunowych. Ponadto mając współrzędne UTM można w miarę łatwo określić odległość do innego punktu.

Sposób przeliczenia z i na współrzędne UTM został doskonale opisany na stronie profesora Steve'a Dutcha [WEB06]. Profesor Dutch nie tylko opisuje konwersję między systemami współrzędnych, ale również pod adresem [WEB07] udostępnia arkusz Microsoft Excel implementujący algorytm konwersji, nieco różniący się zresztą od teoretycznego opisu pod adresem [WEB06]. Na podstawie tego arkusza i za wyraźnym pozwoleniem profesora pozwoliłem sobie zaimplementować metodę `Utils.toUTM()` przeliczającą współrzędne geograficzne na UTM. Zainteresowanych odsyłam do kodu załączonego do artykułu.

Format GPX

Format GPX pozwala na zapis ścieżki, po której poruszaliśmy się z urządzeniem GPS. Pliki w formacie GPX rozumiane są również bezpośrednio przez Google Earth, potrafiący tak zapisane trasy dosyć spektakularnie odtworzyć. Dokładny opis formatu znajduje się pod adresem [WEB08]. Listing 4. przedstawia przykładowy plik w formacie GPX.

Element time to czas startu zapisu, stemple czasowe zapisywane są w plikach GPX w formacie `yyyy-MM-ddTHH:mm:ssZ`. Element `bounds` zawiera maksymalne i minimalne długości i szerokości geograficzne występujące we wszystkich elementach `wpt`. Współrzędne zapisywane są w sposób dziesiętny. Elementy `wpt` zawierają jeden punkt ścieżki, element `ele`każdego elementu `wpt` zawiera wysokość na poziomem morza, `time` – czas zapisu punktu, `name` i `desc` to czytelna nazwa i opis punktu. Wartość `Dot` dla elementów `sym` i `type` określa pojedynczy punkt ścieżki.

Pliki XML, a do tego typu należy również format GPX, najwygodniej utworzyć w .NET za pomocą klasy `System.Xml.XmlWriter`, metody `WriteStartElement` i `WriteEndElement` zapisują do zdefiniowanego pliku lub strumienia pojedynczy element XML, metoda `WriteAttributeString` – atrybut elementu XML. Za pomocą statycznej metody `Create` można utworzyć nowy obiekt klasy, metoda `Close` zamyka zapisywany plik lub strumień XML. Podobnie jak w przypadku sentencji GGA należy zapewnić zapis liczb zmiennoprzecinkowych z kropką jako separatorem części dziesiętnej.

Listing 5 przedstawia implementację klasy generującej pliki *gpx*. Metoda klasy `Gpsx` tworzy nowy obiekt klasy `XmlWriter` i zaczyna zapis do podanego pliku, umieszczając tam element główny pliku *gpx* oraz element `time` wraz z czasem rozpoczęcia zapisu. Metoda `addLocation` zapisuje pojedynczy element `wpt` wraz z podanymi współrzędnymi i wysokością. Jako czas zapisywany jest aktualny czas systemowy. Elementy `name` oraz `description` ustawiane są na aktualną wartość licznika, przez co nazwą każdego punktu ścieżki będzie po prostu liczba od zera w górę. Elementy `sym` oraz `type` zawierają wspomnianą wartość `Dot`. Metoda `close` zapisuje element `bounds` wraz z maksymalnymi i minimalnymi współrzędnymi znalezionymi podczas kolejnych wywołań metody `addLocation`, po czym zapisywany jest kończący znacznik XML dla elementu głównego .

Usypianie i wyłączanie ekranu

Na pewno żaden użytkownik nie pozwoli namówić się na klikanie raz na kilka minut w ekran lub na wciskanie któregoś z klawiszy PDA. Niestety, jeśli użytkownik tego nie będzie robił, to PDA zostanie po prostu uśpiony, nasz program ani nie pokaże, ani nie zapisze aktualnej pozycji do pliku, a chip GPS zgubi satelity i będzie po obudzeniu PDA potrzebował przynajmniej kilku minut, aby je ponownie znaleźć. Oczywiście można zmienić parametry zarządzania energią PDA i zabronić mu się usypiać, ale wtedy wystarczy, że zapomnimy o tym, zostawimy urządzenie

Listing 4. *Przykład pliku GPX*

```
<?xml version="1.0" encoding="utf-8"?>
<gpx xmlns="http://www.topografix.com/GPX/1/0">
  <time>2008-11-25T18:37:18Z</time>
  <wpt lat="49.5480166666667" lon="11.999805">
    <ele>444.9</ele>
    <time>2008-11-25T18:37:19Z</time>
    <name>0</name>
    <desc>0</desc>
    <sym>Dot</sym>
    <type>Dot</type>
  </wpt>
  ...
  <wpt lat="49.5453083333333" lon="11.9878866666667">
    <ele>418.3</ele>
    <time>2008-11-25T18:51:59Z</time>
    <name>347</name>
    <desc>347</desc>
    <sym>Dot</sym>
    <type>Dot</type>
  </wpt>
  <bounds minlat="49.5445216666667"
          minlon="11.9876866666667"
          maxlat="49.5482366666667"
          maxlon="12.0002133333333" />
</gpx>
```

włączone na noc i po baterii. Jest jeszcze inne rozwiązanie. Nie musimy regularnie klikać w ekran, aby PDA myślał, że wciąż aktywnie na nim pracujemy. Wystarczy, że wyzerujemy licznik bezczynności za pomocą funkcji C `SystemIdleTimerReset()` znajdującej się w bibliotece `coredll.dll` ([WEB09]). Funkcje C używa się w .NET za pomocą technologii zwanej P/Invoke (Platform Invoke) doskonale opisanej choćby w [WEB10] i której opis stanowczo wykracza poza ramy tego artykułu. Użycie funkcji `SystemIdleTimeReset()` z poziomu C# sprowadza się do zdefiniowania jej jako metody klasy za pomocą polecenia `System.Runtime.InteropServices.DllImport`, patrz Listing 6.

Klasa `Utils` definiuje metodę `SystemIdleTimerReset()` za pomocą P/Invoke. Publiczna metoda `dontSuspend()` wywołuje `SystemIdleTimerReset()`, ale tylko wtedy, kiedy program wykonywany jest na urządzeniu mobilnym, wersje MS Windows dla komputerów PC, nie wspominając o innych systemach operacyjnych, na których programy .NET obsługiwane są za pomocą Mono czy Portable.NET, nie znają tej funkcji i jej pierwsze wywołanie wygenerowałoby wyjątek. Teraz tylko wystarczy regularnie wywoływać metodę `Utils.dontSuspend()` i PDA nie zostanie uśpiony.

Zerowanie licznika bezczynności ma jednak pewien niemiły efekt uboczny: powoduje, że prądożerczy ekran PDA również nie jest wyłączany. Ekran można wyłączyć za pomocą dosyć kiepsko udokumentowanej funkcji C `ExtEscape()` ([WEB11])

Wystarczy zatem wywołać metodę `Utils.displayOff()`, choćby z metody obsługującej zdefiniowany do tego celu klawisz lub punkt menu GUI, aby wyłączyć ekran. Ekran rozjaśnia się ponownie po wciśnięciu któregoś z klawiszy PDA.

Obsługa GPS

Czas, aby poskładać zaprezentowane wcześniej elementy i odczytać choćby jedną współrzędną z GPS. Listing 7. przedstawia klasę `SerialGPS`, pochodną od wcześniej opisanej klasy `Serial`, definiującą metodę `parseNMEA()`, która służy jako callback (`delegat m_gpscb`) wywoływany dla linii tekstu odczytanych z portu szeregowego w metodzie `readFkt()` klasy `Serial`. Dla każdej linii tekstu przekazanej w tablicy metoda `parseNMEA()` wywołuje metodę `Utils.fromGGA()` odczytującą współrzędne i wysokość nad poziomem morza z sentencji GGA. W tablicy przekazanej do `parseNMEA()` może znajdować się nawet kilka poprawnych sentencji GGA, dlatego metoda `Utils.fromGGA()` wywoływana jest w pętli, tak aby po zakończeniu pętli współrzędne ostatniej sentencji znalazły się w zmiennych `lat2`, `lon2` i `alt2`. Jeśli znaleziono przynajmniej jeden poprawny łańcuch GGA (`isValid`

Listing 5. *Klasa Gpsx do generacji plików gpx*

```
using System;
using System.Globalization;
using System.IO;
using System.Xml;
public class Gpsx {
  private const string TIMEFORMAT = "yyyy-MM-ddTHH:mm:ssZ";
  private const string SYM = "Dot";
  private XmlWriter m_writer;
  private double m_minlat;
  private double m_minlon;
  private double m_maxlat;
  private double m_maxlon;
  private int m_counter;
  public Gpsx() {
    this.m_writer = null;
    this.m_minlat = 180.0; this.m_minlon = 180.0; this.m_maxlat
                    = -180.0; this.m_maxlon = -180.0; this.m_
                    counter = 0;
  }
  ~Gpsx() { this.close(); }
  public void open(string filename) {
    XmlWriterSettings settings;
    this.close();
    settings = new XmlWriterSettings(); settings.Indent = true;
    this.m_writer = XmlWriter.Create(filename, settings);
    this.m_writer.WriteStartElement("gpx", "http://
                    www.topografix.com/GPX/1/0");
    this.m_writer.WriteElementString("time", DateTime.Now.ToStri
                    ng(TIMEFORMAT));
  }
  public void close() {
    if (this.m_writer != null) {
      this.m_writer.WriteStartElement("bounds");
      this.m_writer.WriteAttributeString("minlat", this.m_minlat
                      .ToString(CultureInfo.InvariantCulture));
      this.m_writer.WriteAttributeString("minlon", this.m_minlon
                      .ToString(CultureInfo.InvariantCulture));
      this.m_writer.WriteAttributeString("maxlat", this.m_maxlat
                      .ToString(CultureInfo.InvariantCulture));
      this.m_writer.WriteAttributeString("maxlon", this.m_maxlon
                      .ToString(CultureInfo.InvariantCulture));
      this.m_writer.WriteEndElement();
      this.m_writer.WriteEndElement();
      this.m_writer.Close();
      this.m_writer = null;
    }
    this.m_counter = 0;  this.m_minlat = 180.0; this.m_minlon
                    = 180.0; this.m_maxlat = -180.0; this.m_
                    maxlon = -180.0;
  }
  public void addLocation(double nlat, double elon, double ele)
                    {
    string name;
    if (this.m_writer != null) {
      name = this.m_counter.ToString();
      this.m_counter++;
      this.m_writer.WriteStartElement("wpt");
      this.m_writer.WriteAttributeString("lat", nlat.ToString(Cu
                      ltureInfo.InvariantCulture));
      this.m_writer.WriteAttributeString("lon", elon.ToString(Cu
                      ltureInfo.InvariantCulture));
      this.m_writer.WriteElementString("ele", ele.ToString(Cultu
                      reInfo.InvariantCulture));
      this.m_writer.WriteElementString("time", DateTime.Now.ToSt
                      ring(TIMEFORMAT));

      this.m_writer.WriteElementString("name", name);
      this.m_writer.WriteElementString("desc", name);
      this.m_writer.WriteElementString("sym", Gpsx.SYM);
      this.m_writer.WriteElementString("type", Gpsx.SYM);
      this.m_writer.WriteEndElement();
    }
    if (nlat < this.m_minlat) this.m_minlat = nlat; if (nlat >
                    this.m_maxlat) this.m_maxlat = nlat;
    if (elon < this.m_minlon) this.m_minlon = elon; if (elon >
                    this.m_maxlon) this.m_maxlon = elon;
  }
}
```

będzie wtedy `true`), to znalezione współrzędne porównywane są ze współrzędnymi uzyskanymi podczas ostatniego wywołania tej metody. Jeśli współrzędne lub wysokość zmieniły się w zauważalny sposób, to wywoływany jest kolejny callback tym razem z wartością współrzędnych. Nieco później przekonamy się, że tą funkcją callback będzie metoda prezentująca współrzędne w kontrolce GUI. Po każdym wywołaniu metoda `parseNMEA()` wywołuje metodę `Utils.dontSuspend()` aby odwieźć PDA od wyłączenia się.

GUI

Ostatnią, choć zdaniem chyba każdego kierownika projektu informatycznego najważniejszą częścią programu jest interfejs użytkownika. W opisywanym tu programie do odczytu GPS *sgps.exe* interfejs użytkownika to pojedyncza forma zrealizowania w klasie `SerialGPSForm` pochodnej od `System.Windows.Forms.Form` (porównaj plik *SerialGPSForm.cs* w załączonych źródłach). W górnej części formy znajdują się kontrolki typu `ComboBox m_ports`, `m_baud`, `m_databits`, `m_parities` i `m_stopbits` do wyboru i definicji parametrów portu szeregowego, inicjowane nazwami portów szeregowych dostępnych w systemie (`Serial.GetPortNames()`), kilkoma częściej używanymi wartościami prędkości transmisji lub wielkości słowa danych (`m_baud i m_databits`) czy wszystkimi dostępnymi wartościami enumeracji (jak w polach `m_parities` i `m_stopbits`). W ostatniej kontrolce combo `m_system` można wybrać standard używany przy przeliczeniu współrzędnych geograficznych na UTM. Poniżej znajdują się pola `m_lblutctim`, `m_lbllat`, `m_lbllon`, `m_lbleutm`, `m_lblnutm` (Obsługa współrzędnych UTM została pominięta w przedstawionym opisie zarówno klasy SerialGPS jak i formy SerialGPSForm aby nie powiększać i tak już długich listingów prezentowanych w artykule. Pełna implementacja klas wraz z pełną obsługą UTM znajduje się w załączonym kodzie źródłowym), `m_lblalt`, wszystkie typu `System.Windows.Forms.Label`, służące do wyświetlenia ostatnio odebranych z GPS: czasu UTC, współrzędnych geograficznych oraz wysokości nad poziomem morza. Program kontroluje się za pomocą menu *Port* zawierającego następujące punkty:

- *Open* – otwórz port szeregowy i zacznij odczyt sentencji NMEA
- *Close* – zakończ odczyt portu i zamknij go
- *Log* – włącz lub wyłącz zapis współrzędnych do pliku gpx.
- *Display off* – wyłącza ekran PDA
- *Exit* – kończy program. W przeciwieństwie do Larry'ego Roofa, jednego z autorów [ROOF01], uważam, że aplikacja PDA powinna mieć menu *Exit* do ostatecznego jej zakończenia i np. bardzo brak mi tego menu w istniejących programach, jak Communication Manager czy Active Sync.

Listing 6. *Metody użytkowe klasy Utils*

```
using System.Runtime.InteropServices;
using System.IO;
class Utils {
  [DllImport("coredll.dll")]
  private static extern void SystemIdleTimerReset();
  [DllImport("coredll.dll")]
  private static extern IntPtr GetDC(IntPtr hwnd);
  [DllImport("coredll.dll")]
  private static extern int ExtEscape(IntPtr hdc, int nEscape, int cbInput, byte[]
                      lpszInData, int cbOutput, IntPtr lpszOutData);
  public static bool isMobile() {
    if (Environment.OSVersion.Platform == PlatformID.WinCE)
      return true;
    return false;
  }
  public static void dontSuspend() {
    if (Utils.isMobile()) { try {
       Utils.SystemIdleTimerReset(); } catch(Exception) {} }
  }
  public static void displayOff() {
    const int SETPOWERMANAGEMENT = 6147;
    const int VIDEOPOWEROFF = 4;
    const int SIZ = 12;
    const int DPMSVER = 1;
    IntPtr  hdc;
    byte[]  vpm;

    if (Utils.isMobile()) {
      hdc = Utils.GetDC(IntPtr.Zero);  vpm = new byte[SIZ];
      BitConverter.GetBytes(SIZ).CopyTo(vpm, 0);
      BitConverter.GetBytes(DPMSVER).CopyTo(vpm, 4);
      BitConverter.GetBytes(VIDEOPOWEROFF).CopyTo(vpm, 8);
      Utils.ExtEscape(hdc, SETPOWERMANAGEMENT,
                              SIZ, vpm, 0, IntPtr.Zero);
    }
  }
  public static string Exepath {
    get {
      Uri u;
      string s;
      s = Assembly.GetExecutingAssembly().GetName()
            .CodeBase;
      u = new Uri(s); if (u.IsFile) s = u.LocalPath;
      return s;
    }
  }
  public static string getDateFileName(string ext) {
    StringBuilder sb;
    string exe, dir, fn;
    exe = Utils.Exepath;
    dir = Path.GetDirectoryName(exe);
    fn = Path.GetFileNameWithoutExtension(exe);
    sb = new StringBuilder(fn);
    sb.Append(DateTime.Now.ToString("yyyyMMddHHmmss"));
    sb.Append(ext);
    return Path.Combine(dir, sb.ToString());
  } // getUniqFileName()
  public static string getExtFileName(string ext) {
    return Path.ChangeExtension(Utils.Exepath, ext);
  } // getExtFileName()
}
```

Port szeregowy GPS reprezentowany jest przez zmienną prywatną `m_serial` typu `SerialGPS`. W metodzie `menuOpen_Click()` obsługującej menu *Port/Open* port szeregowy otwierany jest z parametrami odczytanymi z kontrolek typu combo. Metoda `menuCloce_Click()` zamyka port, wywołując metodę `Close()` obiektu `m_serial`, czyli ostatecznie wywołując metodę `Serial.Close()`.

Włączanie lub wyłączanie zapisu współrzędnych do pliku gpx zrealizowane zostało w metodzie `menuLog_Click()` oraz we własności `Log` klasy `SerialGPSForm`. Podczas rozpoczęcia zapisywania do pliku, czyli w momencie przypisania własności `Log` wartości prawdy, wywoływania jest metoda `Gpsx.open()`, zakończenie zapisu do pliku realizowane jest poprzez wywołanie `Gpsx.close()`, mające miejsce w momencie przypisania własności `Log` wartości fałszu. Dla ułatwienia nazwa pliku gpx generowania jest automatycznie przez program jako pełna ścieżka do pliku wykonywalnego (assembly) programu uzupełniona o aktualną datę systemową i rozszerzenie gpx. W ten sposób wynikowe pliki gpx nie nadpisują się nawzajem przy kolejnym włączaniu i wyłączaniu zapisu, można je w miarę jednoznacznie zidentyfikować i znaleźć w znanym miejscu – tam gdzie znajduje się plik wykonywalny naszego programu. Poszukiwanie ścieżki do pliku programu zrealizowano we własności `Utils.Exepath` jako przedstawionej na Listingu 6. Własność `CodeBase` użyta w metodzie zwraca string typu URI (*file://katalog/plik.exe*) z lokalizacją pliku wykonywalnego programu, który za pomocą własności `System.Uri.LocalPath` przekształcany jest na ścieżkę zwykle używaną w systemie (*\katalog\plik.exe*). Metoda `Utils.getDateFileName()` tworzy nazwę pliku gpx. W metodzie tej pełna ścieżka do pliku wykonywalnego rozkładana jest za pomocą metod klasy `System.IO.Path` na katalog i nazwę bez rozszerzenia. Do nazwy bez rozszerzenia dodawany jest łańcuch znaków aktualnej daty oraz podane rozszerzenie *ext* i za pomocą metody `System.IO.Path.Combine()` budowana jest ponownie pełna ścieżka do pliku.

Wyłączenie ekranu w punkcie menu *Port/Display off* zrealizowano w metodzie `menuDoff_Click()` wywołującej przedstawioną na Listingu 6. metodę `Utils.displayOff()`. Wyłączenie programu w menu *Port/Exit* zrealizowano w metodzie `menuExit_Click()` jako wywołanie metody `System.Windows.Forms.Close()`. Przed opuszczeniem programu należy jednak koniecznie zamknąć ewentualnie otwarty port szeregowy i zakończyć ewentualny zapis do pliku gpx. Wprawdzie odpowiednie operacje zamknięcia zrealizowane zostały również w destruktorach klas zarówno `SerialGPS` jak i `Gpsx`, dużo pewniej jest jednak zakończyć operacje samemu niż liczyć na mechanizm odśmiecania wywołujący nasze destruktory. Tuż przed zamknięciem formy `SerialGPSForm` wywoływana jest jej metoda `onClose()`. Dzieje się tak dzięki zdefiniowaniu tej metody wcześniej w konstruktorze jako obsługi zdarzenia `Closing` za pomocą `this.Closing += this.onClose`. W tej metodzie możemy umieścić kod sprzątający po naszej aplikacji, w naszym wypadku wywołujemy wspomniane metody zamykające obiekty obsługi portu szeregowego i zapisu ścieżki.

Listing 7. *Klasa SerialGPS*

```
using System;
using System.Collections;
using System.IO.Ports;
public delegate void CoordDelegate(double utcTime,
                    double n_lat, double e_long, double altitude);
public class SerialGPS : Serial {
  private double[] m_lastcoord;
  private ReadLineDelegate m_gpscb;
  private CoordDelegate m_coordcb;
  public SerialGPS() {
    int i;
    this.m_coordcb = null;
    this.m_system = SerialGPS.DEFAULT_SYSTEM;
    this.m_lastcoord = new double[3];
    for(i = 0; i < this.m_lastcoord.Length; i++)
       this.m_lastcoord[i] = 0.0;
    this.m_gpscb = new ReadLineDelegate(this.parseNMEA);
  }
  public CoordDelegate CoordCb {
    set {
      this.m_coordcb = value;
    }
  }
  private void parseNMEA(ArrayList arr) {
    double tim1, lat1, lon1, alt1;
    double tim2, lat2, lon2, alt2;
    bool   isValid;
    if (this.m_coordcb != null) {
      tim2 = 0.0; lat2 = 0.0; lon2 = 0.0;
      utmn2 = 0.0; utme2 = 0.0; alt2 = 0.0;
      isValid = false;
      foreach(object o in arr) {
        if (Utils.fromGGA(out tim1, out lat1, out lon1, out alt1,
                          o.ToString())) {
          tim2 = tim1; lat2 = lat1; lon2 = lon1; alt2 = alt1;
          isValid = true;
        }
      }
      if (isValid &&
          (Math.Abs(lat2 - this.m_lastcoord[0]) > 0.0002 ||
           Math.Abs(lon2 - this.m_lastcoord[1]) > 0.0002 ||
           Math.Abs(alt2 - this.m_lastcoord[3]) > 0.5)) {
        this.m_lastcoord[0] = lat2; this.m_lastcoord[1] = lon2;
        this.m_lastcoord[2] = alt2;
        this.m_coordcb(tim2, lat2, lon2, alt2);
      }
      Utils.dontSuspend();
    }
  }
  public override void Open() {
    this.Close();
    this.RdLineCb = this.m_gpscb;
    base.Open();
    this.Read = true;
  } // Open()
}
```

Metoda `onClose()` dokonuje jeszcze zapisu aktualnej konfiguracji formy. Przy konfiguracji chodzi o aktualne ustawienia portu szeregowego oraz systemu konwersji na UTM wybranego w polach combo na górze formy. Zapisując te ustawienia w pliku tuż przed opuszczeniem formy i odczytując je z pliku tuż po jej uruchomieniu (np. w konstruktorze) chronimy użytkownika przed koniecznością wybierania wciąż tych samych wartości po każdym uruchomieniu programu. Zapis i odczyt konfiguracji zrealizowany został w metodach `SerialGPSForm.writeCfg()` (Listing 8.) i `SerialGPSForm.readCfg()`. Plik konfiguracyjny składa się z szeregu linii tekstu, z których każda zawiera aktualną wartość kontrolek combo. Dosyć istotna jest obsługa błędów – lub jej niemal całkowity brak – w obydwu metodach, które wywoływane tuż na początku lub tuż przed samym końcem działania aplikacji nie powinny generować denerwujących komunikatów błędów lub – co gorsza – przerywać działanie programu. Możliwymi błędami w przypadku plików konfiguracyjnych może być brak pliku, niemożność utworzenia go z powodu zapełnienia woluminu lub ekskluzywnego dostępu, zły format pliku konfiguracyjnego zapisanego np. przez poprzednią wersję aplikacji itp. Szczególnie metoda `readCfg()` powinna zwrócić w wypadku błędu w miarę rozsądne wartości standardowe, zamiast ubolewać nad wygenerowanym wyjątkiem.

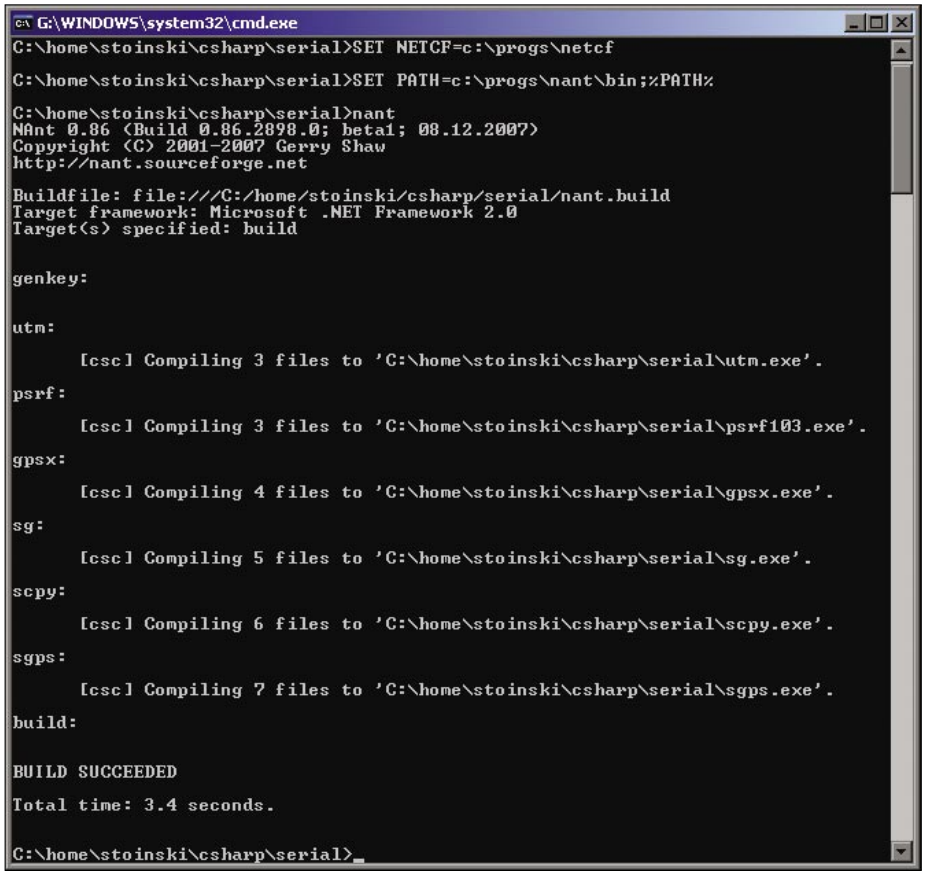
Nazwa pliku konfiguracyjnego zwracana przez `Utils.getExtFileName()` (Listing 6.) to pełna nazwa pliku wykonywalnego programu z rozszerzeniem zamienionym z *.exe na .cfg*.

Najważniejszą i jednocześnie najtrudniejszą częścią interfejsu użytkownika naszego programu było przedstawienie współrzędnych otrzymanych z klasy `SerialGPS`. Metoda `parseNMEA()` wywołuje zdefiniowanego delegata przy każdej poprawnej i nieco różniącej się od poprzedniej współrzędnej. Takiego delegata implementuje w formie metoda `gpscoord()` przedstawiona w Listringu i zgłoszona obiektowi `m_serial` za pomocą zastosowanego w konstruktorze klasy wywołania

```
this.m_serial.CoordCb = new CoordDelegate(t
                        his.gpscoord);
```

Metoda `gpscoord()` zapisuje otrzymane współrzędne do pliku gpx, jeśli za pomocą menu *Port/Log* uaktywniono taki zapis, następnie zapamiętuje otrzymane współrzędne w zmiennych prywatnych klasy `m_nlat`, `m_elon`, `m_utctim` i `m_alt` i ostatecznie za pomocą metody `Form.Invoke()` wywołuje kolejną metodę do wyświetlenia parametrów zapamiętanych w zmiennych prywatnych klasy (Listing 11.).

Dlaczego to tak skomplikowane? Dlaczego nie ustawimy pól `m_lblutctim`, `m_lbllat`, `m_lbllon` i `m_lblalt` już w metodzie `gpscoord()`,



Rysunek 2. *Kompilacja za pomocą NAnt*

Listing 8. *Zapis pliku konfiguracyjnego formy*

```
using System.IO;
  private void writeCfg() {
    StreamWriter s;
    s = null;
    try {
      s = new StreamWriter( Utils.getExtFileName(".cfg"));
      s.WriteLine(this.m_ports.Text);
      s.WriteLine(this.m_baud.Text);
      s.WriteLine(this.m_databits.Text);
      s.WriteLine(this.m_parities.Text);
      s.WriteLine(this.m_stopbits.Text);
      s.WriteLine(this.m_system.Text);
    }
    catch(Exception excp) {}
    if (s != null) s.Close();
  }
```

Listing 9. *Metoda SerialGPSForm.gpscoord()*

```
  private void gpscoord(double utcTime, double n_lat, double e_long, double altitude) {
    try {
      this.m_utctim = utcTime; this.m_alt = altitude;
      this.m_nlat = n_lat; this.m_elon = e_long;
      if (this.Log) {
        this.m_gpsx.addLocation(n_lat, e_long, altitude);
      }
      this.Invoke(this.m_gpscoordcb);
    }
    catch(Exception excp) {}
  }
```

tylko zadajemy sobie trud wywołania jeszcze jednej metody, i to nie bezpośrednio tylko za pośrednictwem `Form.Invoke()`? Otóż implementacja *Windows.Forms* ma spore problemy z kodem wielowątkowym, a w tym wypadku mamy do czynienia z dwoma wątkami. O ile forma `SerialGPSForm` stworzona została i obsługiwana jest w wątku głównym aplikacji, to metoda callback `gpscoord()` wywoływana jest z wątku odczytującego port szeregowy GPS i stworzonego we własności `Serial.Read`. Modyfikacja wartości kontrolek formy w wątku innym niż ten obsługujący formę kończy się w najlepszym wypadku wygenerowaniem wyjątku. Opisane jest to np. w dokumentacji własności `Form.InvokeRequired`, pod adresem [WEB12]. Jeśli `InvokeRequired` zwróci wartość `true`, a w naszym wypadku zawsze zwróci, to jedyną możliwością przypisania naszych współrzędnych kontrolkom formy jest wywołanie metody `Invoke()`. `Invoke()` sprawia jeszcze dodatkowy problem: jest to operacja blokująca i bardzo łatwo zakleszczająca się z wywołaniem np. `Thread.Join()` zastosowanym w implementacji własności `Serial.Read()`. Rozwiązaniem problemu może być zastosowanie kilku prób wywołań metody `Join(int32)` z podaniem maksymalnego czasu oczekiwania zamiast jednego blokującego wywołania `Join()`.

Kompilacja kodu

Z plików źródłowych dołączonych do artykułu tworzone są następujące aplikacje:

- *sg.exe* – program oparty o System.Windows.Forms pokazujący wszystkie informacje pojawiające się na porcie szeregowym. Może być np. użyty do podglądu sentencji NMEA generowanych przez chip GPS.
- *scpy.exe* – kopiuje dane z jednego portu szeregowego na drugi. Np. kopiując z portu COM4 na COM8 na HTC 3300 można przekierować strumień GPS na port połączenia USB między komputerem desktop i PDA i analizować ten strumień na nieco wydajniejszym i wygodniejszym w obsłudze komputerze PC.
- *utm.exe* – aplikacja konsolowa przeznaczona dla komputerów desktop do testowania konwersji między współrzędnymi geograficznymi i UTM oraz do testowania sentencji CGA. Wywołanie utm.exe (mono utm.exe na systemach wyposażonych w mono) podaje bliższe informacje o sposobie użycia programu.
- *psrf103.exe* – również aplikacja konsolowa do generacji sentencji PSRF103, rozumianych przez niektóre odmiany chip-ów SiRF i służących do sterowania ilością i częstotliwością generowanych przez chip sentencji. Niestety, mój telefon skutecznie odmawiał przyjęcia jakiejkolwiek informacji na porcie szeregowym COM4 i niezależnie od wysłanego łańcucha PSRF103 generował tą samą ogromną ilość informacji, dlatego postanowiłem nie zajmować się tym problemem na łamach artykułu. Zainteresowanych odsyłam do przestudiowania zawartości metod Utils.psrf103(), Utils.nmeaChksum() oraz dokumentacji chip-ów pod adresem [WEB04]
- *gpsx.exe* – to również konsolowa aplikacja do zamiany plików typu flat na format gpsx.
- *sgps.exe* – to nareszcie opisywana w artykule aplikacja do obsługi GPS.

Listing 10. *Skrypt do kompilacji programu sgps.exe*

```
@ECHO OFF
SETLOCAL
IF "%NETCF%" == "" SET NETCF=%ProgramFiles%\Microsoft.NET\SDK\CompactFramework
IF "%NETFK%" == "" SET NETFK=%SystemRoot%\Microsoft.NET\Framework\v2.0.50727
IF "%CSC%" == "" SET CSC=%NETFK%\csc.exe
SET CSCOPTS=/nologo /debug /noconfig /nostdlib /target:exe /win32icon:ico\Marta2.ico
                    /keyfile:serial.snk
SET REFERENCE=/reference:"%NETCF%\mscorlib.dll"
SET REFERENCE=%REFERENCE% /reference:"%NETCF%\System.dll"
SET REFERENCE=%REFERENCE% /reference:"%NETCF%\System.Drawing.dll"
SET REFERENCE=%REFERENCE% /reference:"%NETCF%\System.Data.dll"
SET REFERENCE=%REFERENCE% /reference:"%NETCF%\System.Windows.Forms.dll"
SET REFERENCE=%REFERENCE% /reference:"%NETCF%\System.Xml.dll"
SET RESOURCE=/resource:ico\Marta1.ico,Marta1.ico
SET RESOURCE=%RESOURCE% /resource:ico\Marta2.ico,Marta2.ico
SET SOURCE= AssemblyInfo.cs Gpsx.cs Serial.cs SerialGPS.cs SerialGPSForm.cs
                    SerialGPSGUI.cs Utils.cs
echo "%CSC%" %CSCOPTS% /out:sgps.exe %REFERENCE% %RESOURCE% %SOURCE%
"%CSC%" %CSCOPTS% /out:sgps.exe %REFERENCE% %RESOURCE% %SOURCE%
ENDLOCAL
```

Listing 11. *Metoda SerialGPSForm.updGPSControls()*

```
private void updGPSControls(object sender, EventArgs e) {
   try {
    this.m_lblutctim.Text = this.m_utctim.ToString("00:00:00.000");
    this.m_lbllat.Text = this.m_nlat < 0? (-this.m_n_lat).ToString("00.0000S"):
                                          this.m_nlat.ToString("00.0000N");
    this.m_lbllon.Text = this.m_elon < 0? (-this.m_elon).ToString("00.0000W"):
                                          this.m_elon.ToString("00.0000E");
    this.m_lblalt.Text = this.m_alt.ToString("0.0m");
  }
  catch(Exception excp) {}
}
  this.m_gpscoordcb = new EventHandler(this.updGPSControls);
```

Przed kompilacją programu konieczne jest zainstalowanie .NET Compact Framework 2.0 Redistributable (NETCFSetupv2.msi, *http://downloads.microsoft.com*). W przeciwnym razie program skompiluje się jako program dla komputera desktop i nie będzie działał na urządzeniach PDA. Wprawdzie aplikacje typu .NET są przenośne i powinny działać praktycznie z każdym interpreterem, jednak zastosowanie tzw. mocnych nazw i wymaganie ich podczas uruchomienia skutecznie powstrzymuje np. kod skompilowany w środowisku mono przed uruchomieniem go na urządzeniu przenośnym. Można sobie z tym wprawdzie poradzić poprzez podmianę klucza publicznego mocnych nazw, patrz również [WEB13]. Jeśli jednak nie jesteśmy zdani na kompilowanie kodu w środowisku Mono czy PortableNET i mamy do dyspozycji zarówno system operacyjny, jak i kompilator Microsoftu, to zastosowanie .NET Compact Framework Redistributable będzie miało jeszcze jedną ogromną zaletę: już w czasie kompilacji zostaniemy poinformowani o nieprzemyślanym zastosowaniu klasy, własności lub metody, obecnej wprawdzie w .NET Framework dla komputerów desktop, ale w związku z ograniczonymi zasobami PDA nie zaimplementowanej w .NET Compact Framework. Metody zaimplementowane w NETCF zaznaczone są w dokumentacji MSDN małym symbolem telefonu komórkowego, jeśli jednak pisze się z pamięci lub na podstawie książkowych przykładów, łatwo zagalopować się i zastosować kod niedopuszczalny dla małych urządzeń przenośnych. Bez współpracy kompilatora tego

typu błąd mógłby objawić się dopiero po latach, i to np. na PDA naszego najbardziej wymagającego klienta.

Do źródeł programów dołączony jest plik nant.build, plik konfiguracyjny dla programu NAnt ([WEB14]), narzędzia do sterowania kompilacją programów .NET. Po zainstalowaniu Nant-a, .NET Compackt Framework i ustawieniu zmiennej środowiskowej NETCF na katalog zainstalowanego NETCF wywołanie w katalogu źródeł powinno stworzyć nam bezbłędnie wszystkie wymienione pliki exe. Wywołanie `nant doc` dodatkowo stworzy w katalogu *doc* dokumentację HTML z komentarzy umieszczonych w kodzie źródłowym.

Z kompilacją można sobie poradzić również bez Nant-a. Oprócz .NET Compact Framework potrzebujemy jedynie kompilator C#, który firma Microsoft dostarcza wraz z .NET Framework. Można zatem spokojnie założyć, że kompilator C# znajduje się praktycznie na każdym komputerze wyposażonym w średnio nawet aktualny system MS Windows, w dosyć trudnym do znalezienia katalogu

%SystemRoot%\Microsoft.NET\Framework\v2.0.50727.

Listing 10. przedstawia przykład skryptu `compile-sgps.cmd` kompilującego aplikację `sgps.exe`:

Proszę zwrócić uwagę na opcje `/noconfig` i `/nostdlib` wywołania kompilatora *csc.exe*. Dzięki nim program nie będzie konsolidowany ze standardowymi bibliotekami .NET 2.0, zamiast nich zostaną dołączone wymienione opcjami `/reference` biblioteki .NET Compact Framework.

Skrypt powinien być umieszczony w tym samym katalogu, co pliki źródłowe. Za pomocą polecenia `PUSHD` skrypt najpierw zmienia katalog do katalogu źródłowego i sprawdza, czy są ustawione zmienne NETCF (katalog *.NET Compact Framework*), NETFK (*.NET Framework*) i CSC (ścieżka do kompilatora). Jeśli nie, to ustawia je na standardowe wartości, które należy zapewne nieco dostosować w zależności od aktualnej konfiguracji systemu. Następnie wywołuje kompilator wraz ze wszystkimi źródłami, zasobami i parametrami linii poleceń. Po udanym wywołaniu w katalogu skryptu powinien pojawić się plik *sgps.exe*.

Uruchomienie programu na PDA

Aby uruchomić nasze programy na PDA, należy je najpierw tam przekopiować. Użytkownicy Microsoft Windows powinni w tym celu zainstalować aplikację Microsoft Active Sync albo dostarczaną wraz z PDA na CD towarzyszącym dokumentacji do urządzenia albo dostępną w strefie download serwisu *www.microsoft.com*. Po podłączeniu, zdefiniowaniu i zsynchronizowaniu się PDA z komputerem możemy dostać się do systemu plików PDA za pomocą dołączonego do AcitveSync specjalnego eksploratora plików.

Za pomocą przesuwania lub kopiowania i wklejania możemy przenieść skompilowany wcześniej plik naszej aplikacji z komputera PC na PDA. Na PDA możemy uruchomić nasz program, klikając na niego na ekranie PDA w eksploratorze plików. [ROOF01] w rozdziale 21 podaje opis tworzenia własnego pliku instalacyjnego cab, wraz ze skrótami w *\Windows\Start Menu\Programs*. Zainteresowanych odsyłam do książki lub polecam zapoznanie się z programem CAB Wizard pod adresem [WEB15].

Użytkownicy systemów uniksowych mogą zdefiniować połączenie PPP między komputerem desktop a PDA, tak jak to opisano w [WEB16], przynajmniej, dopóki projekt SynCE jest jeszcze w powijakach. Po nawiązaniu połączenia IP między komputerem uniksowym i PDA można transferować pliki choćby za pomocą FTP, HTTP czy samby, konfigurując odpowiedni serwis na komputerze UNIX.

Ostatni problem, z którym musimy sobie poradzić, jest wybranie właściwego portu szeregowego GPS po uruchomieniu aplikacji. Niestety, nie istnieje jednoznaczna procedura pozwalająca określić, który z dostępnych – z reguły – około dziesięciu portów szeregowych na PDA jest portem GPS. Oczywiście możemy po kolei otwierać wszystkie porty, np. korzystając z aplikacji *sg.exe*. Chip GPS po otwarciu jego portu powinien zacząć praktycznie natychmiast wysyłać sentencje NMEA. Jeśli próba otwarcia portu się nie powiedzie lub jeśli w ciągu maksymalnie kilku sekund na ekranie sg.exe nie pojawi się łańcuch znaków wyglądający na sentencję NMEA, to oznacza, że otworzyliśmy niewłaściwy port.

Na moim WHTC3300 problem ten rozwiązał za mnie sam system operacyjny telefonu. Po prostu nie mogę bezpośrednio otworzyć portu szeregowego GPS, mogę dostać się do niego tylko za pomocą zainstalowanego sterownika GPSID. Wprawdzie na początku pisałem, że API GPSID nie funkcjonuje na moim telefonie jak należy i w ciągu kilku stron artykułu ten stan nie uległ zmianie. GPSID oprócz niedziałającej API kopiuje jeszcze dane z portu GPS na tzw. programowy port szeregowy. Ta funkcja sterownika GPSID akurat działa bez zarzutu. To poprzez taki programowy port szeregowy opisywany w artykule program pobierał sentencje NMEA. Z moim wydanym przez niemiecką się komórkową T-Mobile HTC3300 dostarczany jest program konfiguracyjny GPSID, który można znaleźć pod *Einstellungen/System/Externes GPS*. W zakładce *Programme* nazwa portu programowego zdefiniowana jest w polu *GPS-Programmanschluss*. Standardowo jest to COM4.

Kilka dobrych sposobów rozpoznania, który port szeregowy PDA do czego służy, można znaleźć w kodach źródłowych programu GPSProxy ([WEB17]). W pliku *GPSProxy/FormComPortSelect.cs* w metodzie `EnumerateComPorts()` autorzy programu przeszukują klucze *Drivers\Active* i *Software\Microsoft\Bluetooth\Serial\Ports* w celu znalezienia jakiegokolwiek opisu portu lub znalezieniu portów Bluetooth. Rejestr Windows CE możemy przeglądać za pomocą programu PHM Registry Editor ([WEB18]).

W Sieci

- WEB01: WM5 to WM6 GPSID Problem on HTC P3300 – *http://www.eggheadcafe.com/software/aspnet/31057319/wm5-to-wm6-gpsid-problem.aspx;*
- WEB02: Native Mobile – *http://nativemobile.blogspot.com/2007_01_01_archive.html;*
- WEB03: National Marine Electronics Association (NMEA) – *http://www.nmea.org;*
- WEB04: Dokumentacja SiRF – *http://gpsd.berlios.de/vendor-docs/sirf/;*
- WEB05: NMEA 0183 – *http://gpsd.berlios.de/NMEA.txt;*
- WEB06: Converting UTM to Latitude and Longitude (Or Vice Versa) – *http://www.uwgb.edu/DutchS/UsefulData/UTMFormulas.HTM;*
- WEB07: Spreadsheet For UTM Conversion – *http://www.uwgb.edu/DutchS/UsefulData/UTM-Conversions1.xls;*
- WEB08: GPX: The GPS exchange format – *http://www.topografix.com/gpx.asp;*
- WEB09: SystemIdleTimerReset – *http://msdn.microsoft.com/en-us/library/aa930260.aspx;*
- WEB10: Platform Invoke Tutorial – *http://msdn.microsoft.com/en-us/library/aa288468.aspx;*
- WEB11: ExtEscape w pinvoke.net – *http://www.pinvoke.net/default.aspx/coredll/ExtEscape.html;*
- **ROOF01: Larry Roof, Dan Fergus, , 2003**
- WEB12: InvokeRequired – *http://msdn.microsoft.com/de-de/library/system.windows.forms.control.invokerequired.aspx;*
- WEB13: Mono and the Compact Framework in Love, Part Two – *http://evain.net/blog/articles/2006/03/31/mono-and-the-compact-framework-in-love-part-two;*
- WEB14: NAnt - A .NET Build Tool – *http://nant.sourceforge.net;*
- WEB15: CAB Wizard – *http://msdn.microsoft.com/en-us/library/aa924359.aspx;*
- WEB16: Chris De Herrera's Windows CE Website – *http://www.poctepcfaq.com/wce/linux-serial.htm*
- WEB17: GPS Proxy – *http://sourceforge.net/projects/gpsproxy/*
- WEB18: PHM Registry Editor – *http://www.phm.lu/Products/PocketPC/RegEdit/*

DANIEL STOIŃSKI

Informatyk w firmie Nagler & Company (http://www.nagler-company.com).

Kontakt z autorem: stoyac@gmx.de.

Android vs. Java ME

Platforma Sun dla urządzeń mobilnych istnieje już od dłuższego czasu. Podbiła rynek dzięki implementacji na wielu urządzeniach różnych producentów (obecnie istnieje około 2000 różnych typów modeli z Java ME). Główną ideą Sun było zapewnienie uniwersalności i jednolitości aplikacji wytworzonych na platformę Java ME.

Dowiesz się:
- Podstawowych informacji o platformach Java-Me oraz Android;
- Jakie różnice można dostrzec w architekturach porównywanych technologii.

Powinieneś wiedzieć:
- Podstawowa znajomość języka Java;
- Ogólna filozofia technologii Java.

Poziom trudności

Zaczęło się od konfiguracji CLDC oraz profilu MIDP 1.0, które wprowadzały podstawowy zbiór interfejsów i klas dostępnych w tej wersji maszyny KVM (ang. *Kilobyte Virtual Machine*). Po drodze dochodziły opcjonalne biblioteki API rozszerzające możliwości podstawowej maszyny wirtualnej. Dzięki nim można było przykładowo uzyskać dostęp do aparatu fotograficznego, wysyłać dane przez SMS lub Bluetooth. Jednak model *sandbox* maszyny wirtualnej okazał się dość odporny na zmiany. Rozszerzenie jej o pewne funkcjonalności API nie było możliwe. Jedynym rozwiązaniem była zmiana modelu telefonu, w którym pożądane cechy były dostępne. Nagromadzenie opcjonalnych bibliotek spowodowało, że pewien ich zakres został wciągnięty w nowy profil MIDP 2. Postęp technologiczny spowodował, że urządzenia miały coraz większą moc oraz nowe cechy. Z tego względu powstawały kolejne opcjonalne biblioteki wydawane także przez producentów tylko dla ich modeli telefonów. Producenci implementowali maszyny wirtualne, co niejednokrotnie prowadziło do odmiennego zachowania się aplikacji w pewnych sytuacjach. Dochodziły do tego także różne ograniczenia - na wielkość rozmiaru aplikacji czy wykorzystanie pamięci sterty, które oczywiście zależało od dostawcy urządzenia. Pociągało to za sobą rozdzielenie kodów źródłowych oraz zasobów (np. różne rozdzielczości obrazków) na określone platformy. Historia prowadzi do jednego wniosku, że stworzenie jednej aplikacji na wszystkie urządzenia sprawiało kłopoty. W obecnej chwili większość atrakcyjnych aplikacji wymaga procesu *portowania*. Zazwyczaj na początku tworzy się aplikację na grupę podobnych technologicznie urządzeń. Następnie jest *porting*, czyli przeniesienie aplikacji na inne urządzenia. Jednak nie rozwiązuje to problemu, że pewne operacje mogą być wykonywane tylko na ograniczonej grupie urządzeń, która akurat wspiera daną funkcjonalność.

Z tego względu Sun wprowadził także mocniejszą platformę Java ME konfiguracji CDC. Jednak nie zdobyła ona takiej popularności jak CLDC. Zbiór klas był bogatszy a możliwości bliskie maszynie wirtualnej pracującej na komputerach osobistych JRE. Problem braku pewnych funkcjonalności został rozwiązany przez możliwość rozszerzenia maszyny wirtualnej przez interfejs JNI. Gdy brakowało interfejsu API, na przykład dla bluetootha, to można było dopisać bibliotekę w języku natywnym. Co prawda przywiązywało to aplikację do konkretnego urządzenia i bez portingu na inne urządzenie się nie obyło. Jednak w większości przypadków nie byliśmy ograniczeni do zbioru interfejsów API dostarczonych przez producentów.

Konkurencja jednak nie spała i wkroczyła na rynek z platformą, która na pierwszy rzut oka wydaje się bardzo atrakcyjna. Oparta na systemie operacyjnym i własnej implementacji maszyny wirtualnej. Wykorzystuje jako język programowania Javę w bogatszym zestawie i o dziwo nie wprowadza tak restrykcyjnych ograniczeń, jak w Java ME (maksymalny rozmiar pliku jar czy heap). Oczywiście chodzi tutaj o fenomen dziecka Google, czyli Android. Niedawno ukazało się pierwsze urządzenie a aplikacje na platformę powstawały na długo przed jego premierą. Czy rzeczywiście platforma Android jest lepsza od Java ME w obecnych wersjach? Sun co prawda zapowiada wprowadzenie profilu MIDP 3, który zwiększa potencjał aplikacji. Jednak Google swoje idee zrealizowało już na urządzeniu a kolejne są przygotowywane. Czy Sun nie spóźni się z premierą swojego najnowszego produktu i nie przegra wojny na starcie? Jakie możliwości obecnych i przyszłych wersji Java ME są konkurencyjne w porównaniu do Android? Na pierwsze pytanie odpowiedź przyniesie życie i pokaże, w którym kierunku podąży społeczeństwo. Na ostatnie pytanie postaram się odpowiedzieć w serii artykułów, porównując wymienione platformy z punktu widzenia ich możliwości technicznych. Niniejszy artykuł stanowi wprowadzenie do cyklu artykułów badających Java ME oraz Android na kilku poziomach. Poczynając od podstaw, czyli filozofii technologii, w której skupiać się będziemy na cechach i architekturze platform. Zostanie wzięta pod uwagę anatomia aplikacji, aby programista mógł zrozumieć zachowanie się aplikacji w określonych sytuacjach.

W kolejnych artykułach przechodzić będziemy do zagadnień praktycznych i zaczniemy od szkieletu aplikacji. Krótkie testy oraz benchmarki urządzeń przedstawią nam krótki zarys możliwości obliczeniowych platform. Zbadany zostanie dostęp do kodów źródłowych, dokumentacji oraz ograniczeń wynikających z licencji. Wsparcie ze strony portali developerskich oraz system dystrybucji oprogramowania

ułatwiający rozpowszechnianie aplikacji będzie także celem porównania. Opisane zostaną również dotychczasowe problemy, które zostały napotkane w Java ME, a także odpowiem na pytanie czy udało się rozwiązać je w Google Android. Artykuły będą się skupiały na porównaniu na kilku obszarach:

- Interfejs użytkownika;
- Bezpieczeństwo;
- Multimedia;
- Komunikacja;
- Wybrane biblioteki API.

Oczywiście w każdym artykule, w zależności od zastosowań, zostanie opisane, w jaki sposób wykorzystać dostarczone narzędzia. W celu zapewnienia odpowiedniej jakości oprogramowania zostaną zaprezentowane biblioteki oraz aplikacje, które ułatwią zadania działom QA (ang. *quality assurance*).

Filozofia technologii

Android jest platformą dla urządzeń mobilnych zawierającą system operacyjny, maszynę wirtualną oraz zbiór podstawowych aplikacji. Programiści mogą tworzyć aplikacje głównie w języku Java, wykorzystując dostarczone przez Google SDK. Natomiast według Sun Java ME zapewnia solidne oraz elastyczne środowisko dla aplikacji działających na urządzeniach przenośnych. Dostarczaniem maszyny wirtualnej oraz systemu operacyjnego zajmują się producenci urządzeń mobilnych sugerując się specyfikacją Suna. Aplikacje działają w zamkniętym środowisku zwanym *piaskownicą* (ang. *sandbox*). Mają w niej ograniczony dostęp do zasobów urządzenia, które mogą stanowić potencjalne zagrożenie. Zatem przyjrzyjmy się bliżej które z podejść okazało się lepsze oraz w których miejscach są do siebie podobne.

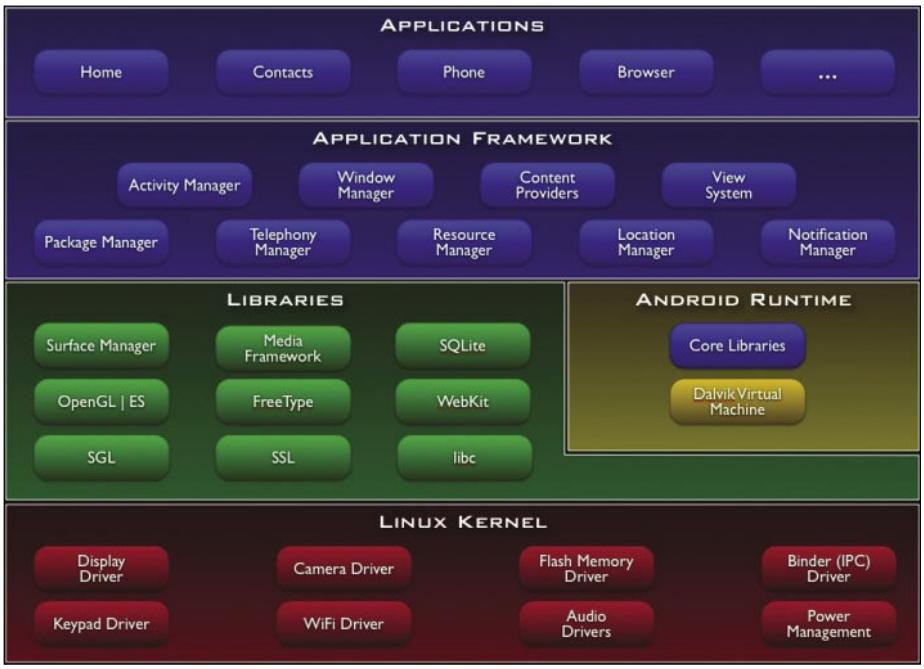
System operacyjny

Google Android pracuje na własnym systemie operacyjnym Linux (kernel 2.6), który odpowiada za bezpieczeństwo, zarządzanie pamięcią oraz procesami, sieć i dostęp do sterowników urządzeń. Stanowi warstwę dostępu pomiędzy sprzętem a resztą oprogramowania. Natomiast w Java ME dostarczaniem systemu operacyjnego zajmują się producenci urządzeń. Nie ma jasno określonej (tak jak to ma miejsce w Android) specyfikacji określającej wymagania dla systemu. Zazwyczaj jest tak, że dostawca urządzenia wprowadza swój własny system, na którym pracuje maszyna wirtualna Java ME. Istnieją jednak systemy operacyjne takie jak Symbian lub Windows Mobile, które dostępne są na wielu modelach urządzeń niezależnie od dostawcy.

Maszyna wirtualna

Większość wirtualnych maszyn jak Java VM, Microsoft .Net bazuje na stosie. Argumenty oraz wyniki przechowywane są w specjalnym obszarze pamięci zwanym stosem. Dla programu dostępny jest jedynie szczyt struktury stosu. Tylko ze szczytu można pobrać wartość lub na nim ją położyć. Głębokość stosu jest różna w zależności od zastosowań, z reguły większa niż 8 poziomów.

Natomiast Android wykorzystuje maszynę wirtualną opartą na rejestrach . Istnieje kilka powodów, dla których ten model został wy-



Rysunek 1. *Architektura platformy Android (http://code.google.com/android/what-is-android.html)*

Listing 1. *Klasa Activity*

```
public class Activity extends ApplicationContext {
    protected void onCreate(Bundle savedInstanceState);
    protected void onStart();

    protected void onRestart();
    protected void onResume();
    protected void onPause();
    protected void onStop();
    protected void onDestroy();
}
```

Listing 2. *Klasa MIDletu*

```
public class BasicMIDlet extends MIDlet {
    public BasicMIDlet(){
        //Konstruktor
    }

    protected void destroyApp( boolean unconditional )
              throws MIDletStateChangeException {

    }

    protected void pauseApp(){

    }

    protected void startApp()
              throws MIDletStateChangeException {

    }
}
```

brany dla Dalvika. Wcześniejsza optymalizacja kodu, co w ograniczonych środowiskach takich jak urządzenia mobilne jest rozwiązaniem korzystnym. Argumenty oraz wyniki przechowywane są w rejestrach, a rzadziej w pamięci danych. Są zaszyte wewnątrz maszyny wirtualnej, które wspierają obliczenia elementarne. Ich liczba jest zazwyczaj niewielka od 4 do 32.

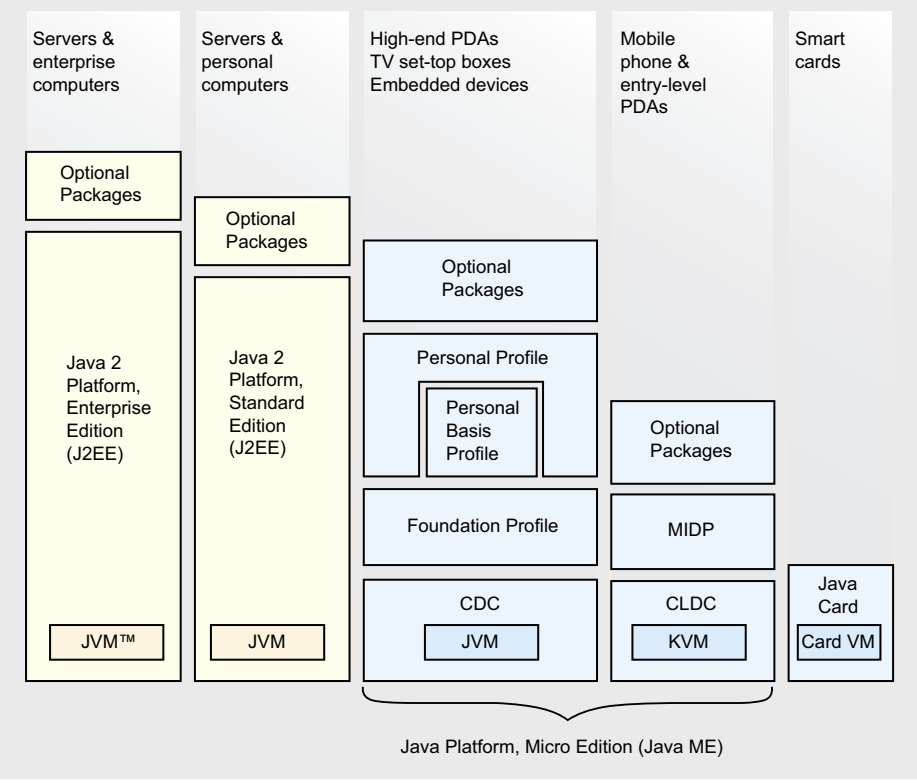
Głębsza analiza działania wymienionych maszyn doprowadza do wniosku, że maszyna bazująca na rejestrach pozwala na szybszy czas uruchamiania kosztem większego rozmiaru kodu skompilowanych programów. Na maszynie Suna operandy do operacji przechowywane są na stosie. Zatem nie jest wymagane ich wyspecyfikowanie jako argumentów funkcji. Natomiast w maszynie opartej na rejestrach wymagane jest podanie operandów razem z instrukcją. Powoduje to przyrost wielkości kodu. Na podstawie artykułu *Virtual Machine Showdown: Stack Versus Registers* dostępnego na stronie internetowej (patrz ramka *W Sieci*) maszyna ta wymaga 47% mniej uruchomionych instrukcji. Efektem tego jest fakt, że maszyna działa 26,5% szybciej kosztem przyrostu około 25% kodu. Oczywiście w artykule bazowano na własnej implementacji maszyn wirtualnych i przedstawione podejście jest czysto naukowe. Dla porównania wydajności maszyn wirtualnych w kolejnym artykule zostaną zaprezentowane wyniki testów benchmark, które przeprowadziłem na kilku urządzeniach. Najpierw jednak przyjrzymy się innym cechom wymienionych technologii, aby lepiej zinterpretować uzyskane rezultaty.

Maszyny KVM dostarczane przez producentów są implementacją, która jest zgodna ze specyfikacją Suna. Dodatkowo producenci w ramach własnej implementacji dodają bonusowe biblioteki. Wynika to z jednej strony z tego, że każdy dostawca urządzenia posiada swoje własne oryginalne cechy i chce, aby były prawidłowo obsłużone. Z drugiej strony Sun bardzo wolno wprowadza nowe standardy w Java Me i niecierpliwość dostawców powoduje, że wydają oni własne. Istnieją maszyny, które są wbudowane w system operacyjny lub istnieją jako niezależne aplikacje tego systemu. Do wbudowanych maszyn w system operacyjny producenta można zaliczyć urządzenia z systemem Symbian OS czy Motorola Linux. Więcej informacji na temat emulatorów i wsparcia programistów można znaleźć na portalach developerskich. Natomiast jako dołączaną aplikację można tutaj zaliczyć system operacyjny Windows Mobile i jego maszyny wirtualne (patrz ramka *W Sieci*).

Google natomiast zintegrował swoją maszynę wirtualną z system operacyjnym Linux. Co prawda wykorzystuje ona jako język programowania Java i na tym całe powiązanie się kończy. Ponieważ implementacja maszyny wirtualnej jest całkowicie niezależna od JVM Suna. Wynika to z licencji Open Source, na której bazuje Android. Chcąc uwolnić się od restrykcji nałożonych przez Sun, zaimplementowali własną maszynę wirtualną. Wprowadzili także możliwość obsługi paczek opcjonalnych, która podobnie jak w Java ME będzie uzależniona od dobrej woli producentów. W obecnej chwili trudno jest powiedzieć nad swobodą interpretacji specyfikacji do implementacji systemu, ponieważ jest jedno urządzenie. Po wielu doświadczeniach udało się zaobserwować, że pewne *bugi*, które występują na emulatorze, obecne są także na urządzeniu. Zatem przynajmniej starają się odwzorować faktyczny stan postępu prac developerskich Google. Z jednej strony to źle, że są błędy z drugiej dobrze, że producenci na własną rękę ich nie naprawiają. Jak ogólnie wiadomo, gdy każdy doda coś swojego, to później można się spodziewać różnych zachowań systemu. Pociąga to za sobą rozdzielenie kodu lub użycie preprocesora, dokładnie jak to ma miejsce w Java ME. Dużym plusem jest wgląd w kod źródłowy zarówno maszyny wirtualnej, jak i samego systemu operacyjnego.

Każda aplikacja w Androidzie zostaje uruchomiona w swoim własnym procesie oraz instancji maszyny wirtualnej. Architektura Dalvika umożliwia efektywne uruchamianie wielu maszyn wirtualnych. Nasuwa się pytanie, dlaczego nie wykorzystano architektury OSGi, która pozwala na uruchamianie wielu aplikacji w jednej maszynie wirtualnej. Jedną z odpowiedzi może być zapewnienie stabilności całego systemu. W OSGi w przypadku krytycznego błędu w jednej z aplikacji nastąpi zamknięcie całej maszyny wirtualnej z innymi aplikacjami. Model Dalvika chroni stan pozostałych aplikacji i odseparowuje je od siebie. Należy zwrócić w tym miejscu uwagę, że pomiędzy aplikacjami istnieje mechanizm wymiany komunikatów, który zostanie opisany w dalszej części tego artykułu.

Natomiast w Java ME zależy to od rodzaju urządzenia, na którym działa. Najczęściej, jeżeli jest to system operacyjny taki jak Symbian lub Windows Mobile to pozwala na uruchamianie różnych aplikacji w wielu instancjach maszyny wirtualnej. W pozostałych bywa różnie i czasami zdarzało się, że mogła pracować tylko jedna instancja maszyny wirtualnej i jedna aplikacja. Aplikacje z tzw. zestawu MIDletów mogą pracować tylko w jednej maszynie wirtualnej (obecnego profilu MIDP 2).



Rysunek 2. *Architektura platformy Java ME (http://java.sun.com/javame/technology/index.jsp)*

Architektura systemów

Android został podzielony na kilka warstw, poczynając od systemu operacyjnego. W tej najniższej warstwie znajdują się przede wszystkim sterowniki do urządzeń takich jak ekran, kamera, klawiatura, WiFi, audio itd. Następnie jest maszyna wirtualna oraz zbiór bibliotek C/C++ podzielonych w zależności od zastosowania . Powyżej znajdują się komponenty szkieletu aplikacji, przez które programiści mogą odwoływać się do opisanych wyżej natywnych bibliotek. Końcowym przystankiem są już same aplikacje, które wykorzystują infrastrukturę dostarczonych komponentów.

Szerzej widoczne komponenty oraz biblioteki atrakcyjne z punktu widzenia tworzenia aplikacji zostaną opisane w kolejnych artykułach (Rysunek 1.).

Jedynym urządzeniem, które ukazało się pod koniec października 2008 jest HTC Dream G1 w sieci T-Mobile (patrz ramka *W Sieci*). Producent nie określa ograniczeń, jak to miało miejsce w Java ME, co do wykorzystania pamięci. Zatem przypuszczać należy, że do dyspozycji mamy całą fizyczną jej ilość. Został także stworzony specjalny model telefonu dla programistów. Posiada on odblokowany SIM oraz program rozruchowy, który umożliwia uruchamianie innych wersji Androida (patrz ramka *W Sieci*).

W Java ME jak można zobaczyć na Rysunku 2., w najniższej warstwie znajduje się maszyna wirtualna. KVM dla urządzeń słabszych oraz JVM dla mocniejszych. System operacyjny jak to opisano wcześniej, jest dostarczany przez producentów konkretnego urządzenia. Następnie znajduje się rdzeń każdej maszyny wirtualnej, konfiguracja oraz profile, które określają podstawowy zbiór interfejsów oraz klas. Opcjonalne biblioteki są dodatkami, których obecność zależy również od producenta.

Ze względu na masę urządzeń, u których opcjonalne biblioteki były lub nie, Sun zdecydował się wprowadzić kategorie. Kategorie zgrupowały pewne opcjonalne biblioteki i zostały oznaczone odpowiednią specyfikacją JSR. Z tego względu grupuje się urządzenia ze względu na wersję CLDC (1.0 lub 1.1) lub MIDP (1.0 lub 2.x). Kolejną grupę JTWI (Java Technology for the Wireless Industry), które w minimalnej wersji zawierają następujące specyfikacje CLDC 1.0 (JSR 30), MIDP 2.0 (JSR 118), and WMA 1.1 (JSR 120), CLDC 1.1 (JRS 139) and MMAPI (JSR 135). Ostatnią najbardziej bogatą w możliwości grupą jest MSA (Mobile Service Architecture). W jej skład wchodzi większość opcjonalnych bibliotek podzielonych w zależności od zastosowań. Większość z nich doczekała się urzeczywistnienia na urządzeniach - ich szczegółowa charakterystyka zostanie opisana na późniejszym etapie (rysunek 3.).

Wymagania stawiane urządzeniom narzucone przez Sun jako minimalne (patrz ramka w sieci) określone są następująco:

- Typ procesora: głównie ARM (16 lub 32 bitowy);
- Częstotliwość taktowania: minimalna 50 MHz;
- Wielkość pamięci RAM: około 300 KB (włączając MIDP), typowo 600 KB;
- ROM/Flash min. 1MB, typowo > 1.5 MB.

Można się spotkać z dokumentami, które szczegółowo opisują, jakie są wymagania w

Listing 3. *Standardowa klasa aplikacji Personal Profile*

```
public class MyClass {
    public static void main( String[] args ){

    }
}
```

Listing 4. *Aplet*

```
<applet code="BasicApplet.class"
        width="300" height="200">
</applet>
public class BasicApplet extends java.applet.Applet {
    public BasicApplet(){

    }

    public void init(){

    }

    public void start(){

    }

    public void stop(){

    }

    public void destroy(){

    }
}
```

Listing 5. *Klasa Xlet*

```
import javax.microedition.xlet.*;

public class BasicXlet implements Xlet {
    private XletContext context;

    public BasicXlet(){

    }

    public void initXlet( XletContext context )
           throws XletStateChangeException {

        this.context = context;

    }

    public void destroyXlet( boolean unconditional )
           throws XletStateChangeException {

    }

    public void pauseXlet(){

    }

    public void startXlet() throws XletStateChangeException {

    }
```

zależności od wersji MIDP czy konfiguracji CLDC (patrz ramka *W Sieci*).

Postęp technologiczny spowodował, że najnowsze urządzenia wykraczają daleko poza ramy przedstawione przez Sun. Szczegóły, na jakie programiści zwracają uwagę, a które zostały narzucone przez producentów to:

- Maksymalny rozmiar sterty (ang. *max. heap size*) określające maksymalną dostępną pamięć RAM na aplikację;
- Maksymalny rozmiar jar (ang. *max jar size*) określające maksymalny rozmiar aplikacji, jaka może być zainstalowana na urządzeniu;
- Maksymalny rozmiar pamięci stałej (ang. *max storage size*) oraz maksymalny rozmiar rekordu;
- Rozmiar ekranu;
- Biblioteki interfejsów API.

Różne wartości, które producenci określają, doprowadziły do włączenia procesu *portingu* w wytwarzaniu aplikacji. Szerzej proces ten zostanie opisany w kolejnych częściach publikacji.

Najnowsze urządzenia wprowadzają dynamiczną wartość ograniczeń - zależną od ilości zasobów przydzielonych przez system operacyjny.

Na rynku dostępnych jest około 95. producentów urządzeń J2ME oraz około 2000 modeli wliczając w to branding (modele różniące się tylko wyglądem).

Istnieją także implementacje, które rozszerzają pewne mobilne systemy operacyjne jak Windows Mobile czy PalmOS. Linki do popularnych portali developerskich znajdują się w ramce *W Sieci*.

Sun wprowadził nową specyfikację zwaną MSA2 (Rysunek 4.), w której doczekaliśmy się kilku innowacji. Przede wszystkim połączenie konfiguracji CDC oraz CLDC. Kolejną nowością jest wprowadzenie nowego profilu MIDP 3.0. Oficjalnie trzecia wersja MIDP 3.0 jest projektem Open Source prowadzonym przez grupę firm pod przewodnictwem Motorola (patrz ramka *W Sieci*). W chwili pisania artykułu nie ma jeszcze urządzenia, jednak specyfikacja precyzuje nową architekturę i możliwości tego profilu. Specyfikacja określa minimalne wymagania dla urządzeń adaptujących MIDP 3, jest to:

- 1MB dla nieulotnej pamięci dla implementacji MIDP poza wymaganiami dla CDC i CLDC;
- 512 KB pamięci nieulotnej dla pamięci trwałej;
- 1 MB dla pamięci ulotnej (*java heap runtime*);
- Jeden lub więcej logicznych interfejsów sieciowych.

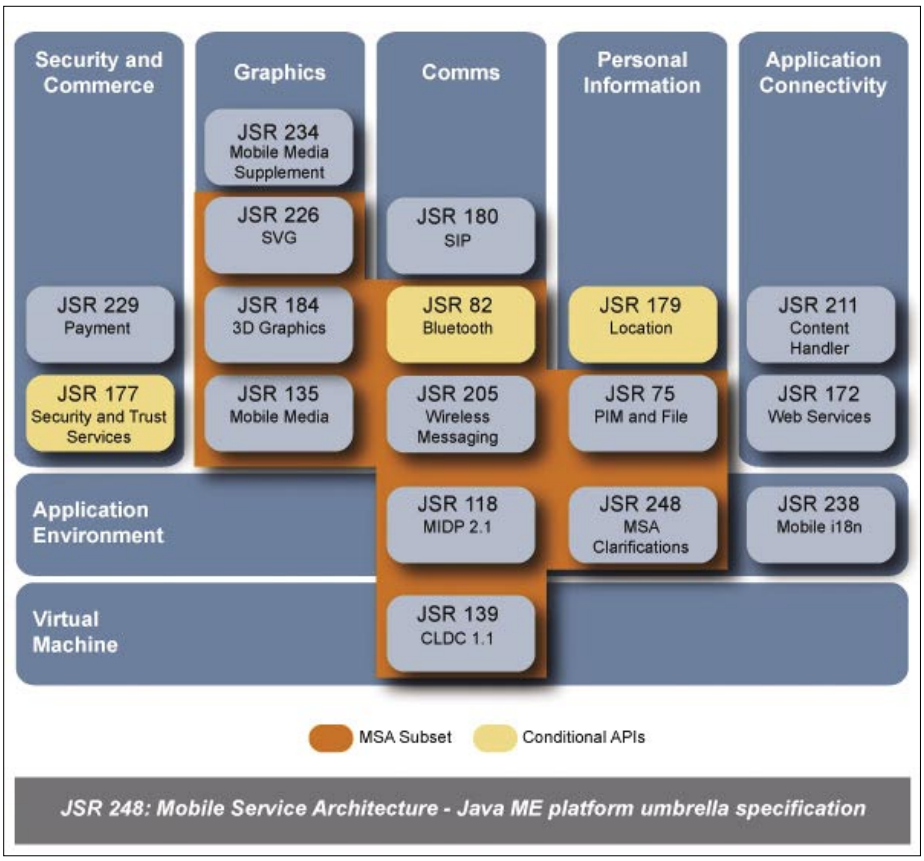
Konfiguracja CDC wprowadzona została dla urządzeń wyposażonych w około 2MB pamięci RAM oraz około 2,5 MB ROM i pracujących na 32 bitowych procesorach. Specyfikacja CDC 1.0 zawiera podzbiór klas platformy Java 2 Standard Edition wersji 1.3.1. Natomiast kolejna wersja CDC 1.1.2 (JSR 218) bazuje na J2SE w wersji 1.4.2. Konfiguracja CDC została uzupełniona o profil Fundation Profile, wzbogacając go o bogate operacje I/O oraz szybką sieć. Na końcu znalazł się profil Personal Profile dostępny w podstawowej i pełnej wersji. Personal Basis Profile wprowadził podzbiór klas AWT do budowy interfejsu oraz model aplikacji nazwanych Xlet.

Aplikacje Xlet, które są w dużym stopniu podobne do MIDletów. Wprowadzają dobrze zdefiniowany cykl życia aplikacji, który bardziej szczegółowo zostanie opisany w kolejnej części artykułu.

Pełna wersja Personal Profile wprowadza natomiast pełną implementację AWT oraz aplikacje oparte na modelu Java Applet. Biblioteka AWT dobrze znana jest programistom Java SE, za pomocą jej komponentów można wytwarzać graficzny interfejs użytkownika. Urządzenia na konfigurację CDC można policzyć na palcach jednej ręki:

- Nokia 9300;
- Nokia 9500;
- Sony-Ericsson /M600;
- Sony Ericsson P990i;
- Implementacje na Windows Mobile.

W dalszej części skupimy się na możliwościach oferowanych dla aplikacji przez obie platformy. Porównanie będzie polegało najpierw na przyrównaniu stanu dotychczasowego, czyli Android do profilu MIDP 2 oraz PersonalProfile. Następnie, jeśli jakaś funkcjonalność znalazła się w nowym profilu MIDP 3 zostanie ona porównana z Androidem.



Rysunek 3. *Architektura MSA (http://java.sun.com/javame/technology/msa/)*

Anatomia aplikacji

Android

Aplikacje w Android mogą być zrealizowane z następujących bloków:

- Activity;
- Intent;
- Broadcast recivier;
- Service;
- Content Provider.

Activity

Stanowi najczęściej używany komponent do budowania aplikacji. Zwykle jest to pojedynczy ekran interfejsu użytkownika. Wiele aplikacji wykorzystuje jednak wiele widoków do prezentowania swojej treści i można je zaimplementować jako osobne activity. Przechodzenie do innego ekranu polega na starcie nowego acitvity, poprzedni ekran przechodzi wówczas w stan pauzy i umieszcza-

ny jest na stosie historii. Android zachowuje historię dla każdej aplikacji uruchomionej z ekranu domowego. Użytkownik może przechodzić między widokami w historii a w razie potrzeby usuwać je. W pewnych przypadkach activity mogą przekazywać komunikaty między sobą.

Intent

Wchodzi w skład mechanizmu umożliwiającego wymianę komunikatów zdarzeń (intencji) w Android. Zawiera zbiór pól opisujących, co aplikacja chciałaby zrobić. Wywołany komunikat intent jest przesyłany do mechanizmu zwanego intent resolver. Wykorzystuje on filtr (intent filters) ustawiony w aplikacjach, aby znaleźć odbiorcę danej wiadomości. IntentFilter jest opisem, jakie komunikaty intent może obsłużyć activity (lub BroadcastReceiver, Service). Pola w intent zawierają określoną akcję, kategorię, dane typu String, dane typu MIME, docelową klasę oraz inne wartości. Najczęściej wykorzystywany jest mechanizm w przejściach między widokami aplikacji. Jeden acitvity uruchamia inny z określonym intenetm.

Intent receiver (Broadcast intent receiver)

Wykorzystują go aplikacje, które oczekują na komunikaty typu broadcast. Najczęściej aplikacja reaguje na zewnętrzne zdarzenia jak np. nadchodzące połączenie lub inne przerwania. Aplikacja nie musi być uruchomiona, aby odpowiedzieć na określone zdarzenie, system zatroszczy się o jej uruchomienie. Istnieje także możliwość wysyłania komunikatów typu broadcast z tego typu aplikacji do innych.

Intent filter

Activity oraz intent receivers zawierają jedną lub więcej filtrów wpisanych w pliku manifest określające, na jakie zdarzenia reagują. Dla activity Android szuka najbardziej odpowiadającej aplikacji, która będzie w stanie odebrać Intent. Natomiast dla komunikatów broadcast wysyła go do wszystkich odbiorców, dla których filtr pasuje do danego Intentu.

Service

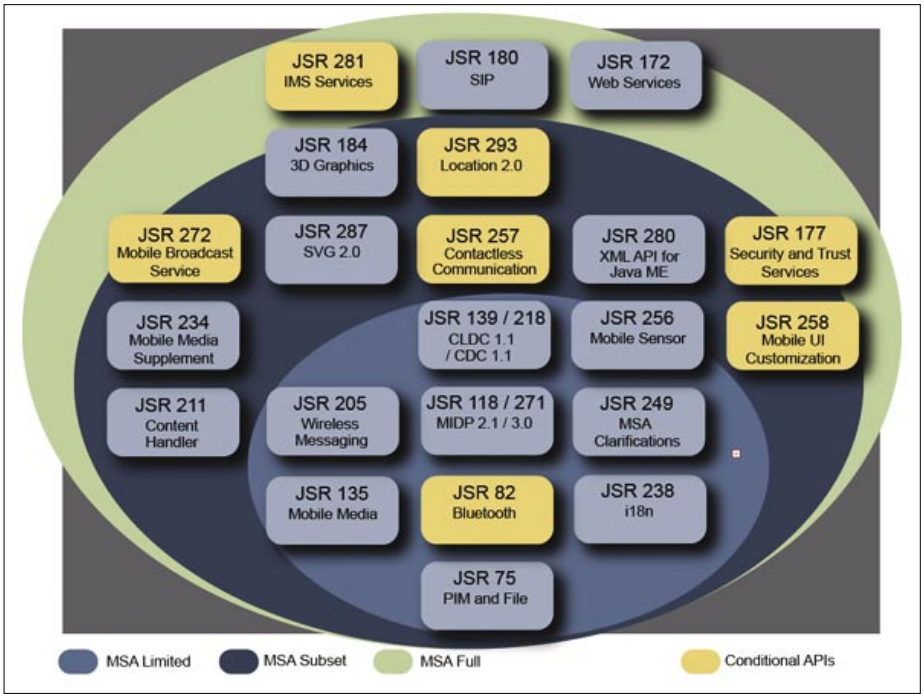
Program wykonywany w tle bez interfejsu użytkownika realizujący określone zadania jak np. odgrywanie muzyki czy monitoring sieci. Istnieje możliwość połączenia się z serwisem, wykorzystując do tego komunikaty Intent.

ContentProvider

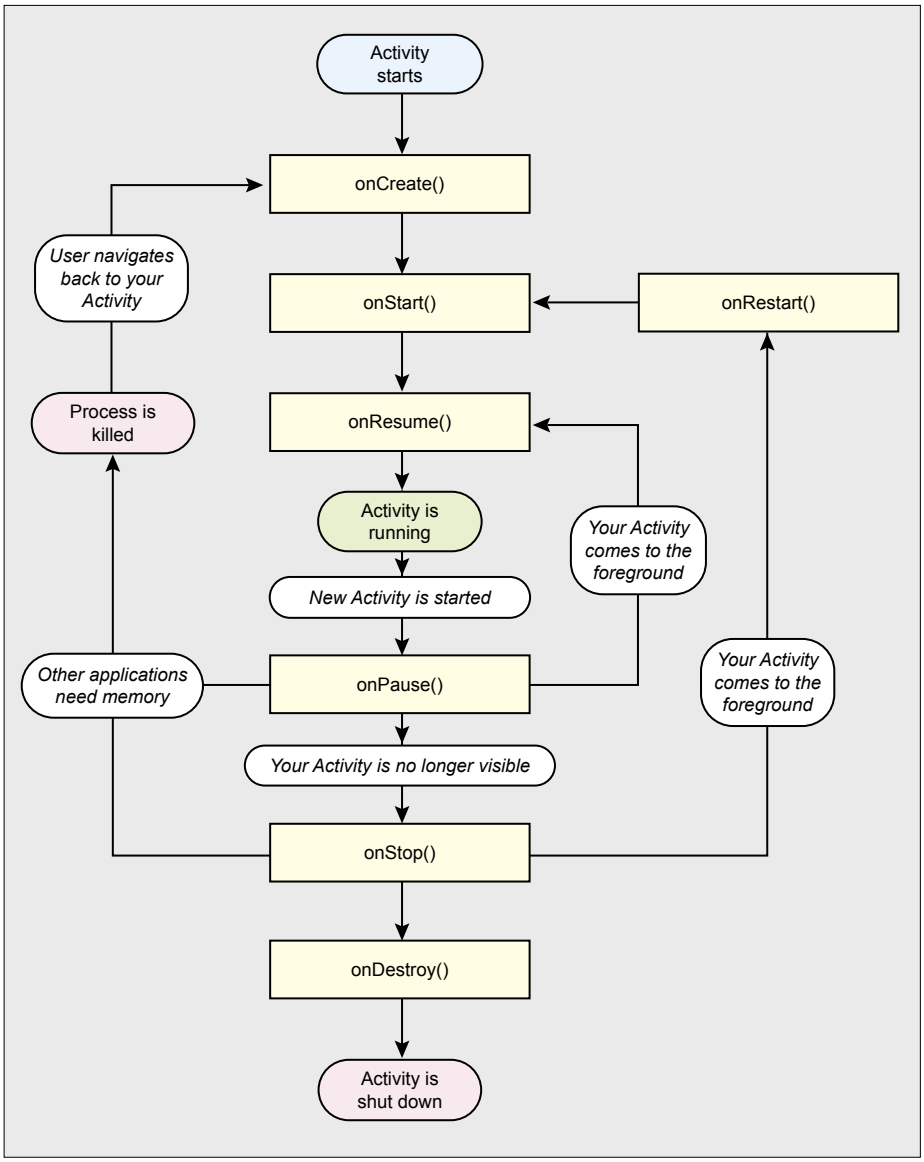
Klasa implementująca zestaw metod umożliwiających innym aplikacjom na dostęp do jej danych. Największe zastosowanie ma w przypadku, gdy występuje potrzeba współdzielenia danych między aplikacjami.

Cykl życia aplikacji

W większości przypadków każda aplikacja Android działa na własnym procesie Linuxa. Proces jest tworzony dla aplikacji, kiedy pewien kod musi być uruchomiony. Po zakoń-



Rysunek 4. *Architektura MSA2 (http://java.sun.com/developer/technicalArticles/javame/msa2-intro/)*



Rysunek 5. *Cykl życia Activity (http://code.google.com/android/reference/android/app/Activity.html)*

czeniu wykonywania określonego kodu system zwraca pamięć innym aplikacjom.

Fundamentalną cechą Androida jest to, że cykl życia aplikacji nie jest bezpośrednio kontrolowany przez aplikację. Zamiast tego jest determinowany przez system kombinacją części aplikacji, o których system wie, że są uruchomione, jak bardzo ważne są czynności dla użytkownika, które wykonuje oraz ile pamięci jest dostępnej w systemie. Należy tutaj podkreślić, że bardzo ważne jest zrozumienie funkcjonowania podstawowych komponentów aplikacji (Activity, Services, BroadcastReceivers), aby w przyszłości nie doszło do sytuacji, że system niszczy proces, który wykonuje istotne zadania.

Jednym z przykładów podstawowego błędu, jakie podaje Google, jest start wątku BroadcastReceivers, kiedy odbiera komunikat Intent w metodzie `BroadcastReceiver.onReceive()`

i wyjście z funkcji. Po pierwsze system bierze pod uwagę, że Broadcast receiver nie będzie już aktywny oraz to że jego proces nie będzie już potrzebny, pomimo tego, że inne komponenty aplikacji są w nim aktywne. Rozwiązaniem tego problemu jest uruchomienie serwisu (Service) z BroadcastReciviera co spowoduje, że system będzie wiedział, że proces ma wciąż aktywne zadania i go nie usunie.

Aby określić, które procesy powinny być zabijane (ang. *killed*), kiedy jest niski poziom wolnej pamięci, Android umieszcza je w hierarchii ważności. Bazuje ona na komponentach uruchomionych w nich, jak i stanie ich samych. Typy procesów są następujące (aby ułatwić zrozumienie zagadnienia na Rysunku 5. został przedstawiony cykl życia Activity):

- Procsy pierwszego planu (ang. *foreground process*) to proces, z którym użytkownik w danej chwili pracuje. Proces może pracować na pierwszym planie z kilku powodów:
  - jest Activity, z którym użytkownik jest w interakcji (jest to czas pomiędzy wywołaniem metody onResume() a onPause - w dalszej części będzie to wyjaśnione - Listing 1.);
  - jego BroadcastReceiver jest uruchamiany (wywołanie metody `onReceive()`);
  - posiada serwis, który obecnie wykonuje fragment kodu na jedną odpowiedź.

Takich procesów będzie w systemie niewiele i będą one w ostateczności zniszczone, jeżeli stan pamięci jest zbyt niski, aby żaden z procesów nie mógł kontynuować działania. Generalnie założono, żeby zapewniona była interakcja z użytkownikiem.

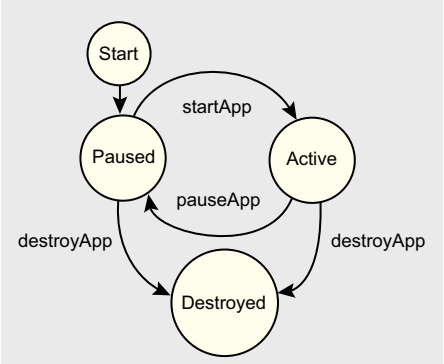
- *Procesy widoczne* (ang. *visible process*) są to te procesy, które są widoczne na ekranie, jednak nie są procesami foreground. Dzieje się tak w przypadku, gdy aplikacja utraciła focus, jednak jest wciąż widoczna. Na przykład została częściowo zasłonięta okienkiem dialogowym. Następuje wtedy przejście w stan pauzy (jej metoda `onPause()` została wywołana), jednak cały czas jest ona gotowa do powrotu w stan działania (przechowuje w pamięci swoje informacje). W ekstremalnych sytuacjach, jeżeli procesy foreground będą potrzebowały pamięci, aplikacje mogą zostać zabite.
- *Procesy serwisu* (ang. service process) system utrzymuje działające serwisy do czasu, gdy nie ma wystarczającej ilości pamięci dla powyższych dwóch typów.
- *Procesy pracujące w tle* (ang. *background process*) zaliczamy do tej kategorii wszystkie aplikacje, które nie są widoczne dla użytkownika (metoda onStop() została wywołana). W dalszym ciągu stan oraz zmienne są przechowywane w pamięci, jednak w przypadku alarmowego stanu pamięci zostaną natychmiast zabite. Zazwyczaj takich uruchomionych procesów jest dużo i są one przechowywane na liscie LRU (ang. *Least Recently Used*), aby zagwarantować, że proces najbardziej odwiedzany będzie usunięty na końcu.
- *Procesy bezczynne* (ang. *empty process*) nie zawierają żadnych aktywnych komponentów systemu. Są najszybciej zabijanymi procesami w systemie w przypadku braku pamięci. Powodem ich zastosowania jest zwiększenie czasu uruchamiania w przypadku, gdy komponent aplikacji będzie następnym razem uruchomiony. Sytuacja, kiedy proces może stać się empty, jest opisany wcześniej problemem w przypadku wyjścia z funkcji `onReceive()` BroadcastReceiver.

Model aplikacji Android

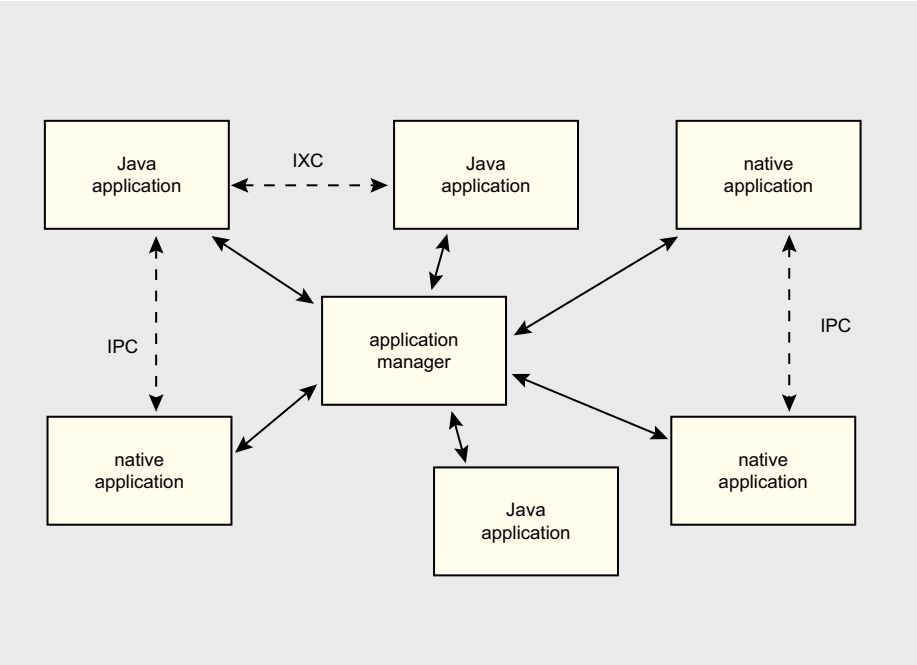
W większości systemów operacyjnych istnieje mocne powiązanie pomiędzy obrazem, z którego uruchamiamy aplikację (w Windows *.exe*), procesem oraz powiązaną z aplikacją ikoną.

W Android podejście jest bardziej elastyczne i warto poznać jak pewne części mogą współpracować ze sobą:

- Paczka aplikacji *.apk* zawiera kod aplikacji oraz jej zasoby. To jest plik, który jest dystrybuowany oraz pobierany przez użytkownika, gdy użytkownik instaluje aplikację.
- Zadania (ang. *task*) z reguły jest to to, co użytkownik spostrzega jako aplikacja, którą może uruchomić. Najczęściej



Rysunek 6. *Cykl życia MIDletu (http://developers.sun.com/mobility/midp/articles/fsm/)*



Rysunek 7. *Proces komunikacji między Xletami (http://java.sun.com/j2me/docs/cdc_appmgmt_wp.pdf)*

zawiera ikonę i jest dostępne na najwyższym poziomie elementów tak, żeby można to było szybko przywrócić na pierwszy plan.
- Proces (ang. *process*) opisany wcześniej niskopoziomowy proces kernela Linuxa. Najczęściej cały kod w aplikacji .apk jest uruchamiany w jednym procesie. Chociaż odpowiednia konfiguracja pozwala na uruchamianie w kilku procesach. Poprawia to stabilność oraz bezpieczeństwo Android, gdy pewien niezaufany lub awaryjny kod umieścimy w osobnym procesie. Redukcja obciążenia przez wykonywanie kodu z wielu .apk w jednym procesie. Pomaga to systemowi na umieszczanie wymagającego kodu w osobnym procesie, gdzie system może go spokojnie zabić bez obawy o inne części aplikacji.
- Wątki (ang. *threads*) każdy proces może zawierać jeden lub więcej wątków uruchomionych w nim. W większości sytuacji Android unika tworzenia nowych wątków w procesie, trzyma aplikacje w pojedynczym do czasu, gdy ona sama utworzy kolejny. Należy w tym miejscu zwrócić uwagę, że instancje Activity, BroadcastReceiver, Services są tworzone tylko z głównego wątku procesu.

MIDP 2

Aplikacje Java ME MIDP to kolekcja jednego lub więcej klas MIDletów umieszczonych w archiwum JAR znanych również jako zestaw MIDletów. W archiwum znajdują się także wszystkie pozostałe klasy oraz zasoby (obrazki, dźwięki) . W parze z plikiem *JAR* idzie plik *JAD*, w którym określa się podstawowe parametry, jak nazwa aplikacji, wersje MIDP oraz CLDC oraz inne, które bardziej szczegółowo będą opisane w kolejnych częściach. MIDlet podobnie jak Activity posiada swój cykl życia składający się z czterech stanów:

- *Start* – ma miejsce podczas uruchamiania aplikacji Java ME.
- *Pause* – występuje, gdy użytkownik traci możliwość interakcji z aplikacją. Jednak aplikacja jest cały czas uruchomiona. Przykładowo taka sytuacja zachodzi w przypadku nieoczekiwanych przerwań jak przychodząca rozmowa telefoniczna lub sam użytkownik przełącza się na inną aplikację, jeżeli system operacyjny mu na to pozwala.
- *Active* – sytuacja, gdy użytkownik ma bezpośrednią styczność z interfejsem użytkownika.
- *Destroyed* – stan, gdy aplikacja zostaje zakończona.

Rysunek 6. prezentuje przejścia pomiędzy kolejnymi stanami.

Kontekst MIDletu

Aplikacja może mieć interakcję ze środowiskiem operacyjnym. Przez `getAppProperty()` (lub `getProperty()`) można otrzymać parametry inicjalizujące aplikację (w plikach *JAD*, *MANIFEST* lub systemowe wartości). Aplikacja może znaleźć się w stanie zawieszenia z różnych powodów, jak np. przychodząca rozmowa telefoniczna lub samoczynne przejście w stan pauzy przez wywołanie `notifyPaused`. Chciałbym zwrócić tutaj uwagę, że metoda `pasueApp` nie jest wywoływana po `notifyPaused`. System zakłada, że MIDlet został przygotowany do przejścia w ten stan. Dlatego często spotyka się wywołanie przed metodą `notifyPaused` – `pauseApp`. Przejście z powrotem w stan aktywny odbywa się przez wywołanie metody `resumeRequest()`.

`ResumeRequest()` odpytuje system zarządzania o reaktywację MIDletu. Jak tylko będą dostępne zasoby, to następuje wywołanie metody startApp. Rodzi się tutaj pewne niebezpieczeństwo związane z ponownym wywołaniem metody `startApp()`. Jeżeli w ciele metody zostały zainicjalizowane obiekty, to może dojść do sytuacji ich zdublowania. Należy więc w miarę możliwości część inicjalizacji przeprowadzić w konstruktorze, a w metodzie `startApp` kontrolować ilość wywołań metody (np. flaga). Dlatego w ciele metody `pauseApp` powinny nastąpić działania przygotowujące MIDlet do przejścia w stan zawieszenia. Powinny również zostać zwolnione wykorzystywane zasoby lub ich część bądź zawieszone działania pętli oraz timerów. Zalecaną praktyką w zależności od sytuacji jest niszczenie w `pauseApp` tych obiektów, które były stworzone w `startApp` a zwłaszcza tych, które wymagają dostępu do ekranu. Specyfikacja maszyny wirtualnej nie wymaga, aby ekran był dostępny już w konstruktorze, zależy to od implementacji KVM.

Metoda `notifyDestroyed` powoduje natychmiastowe zamknięcie aplikacji bez wywoływania kodu w ciele metody `destroyApp`. Jeżeli metoda `destroyApp` wywołana jest przez system z parametrem true, to powoduje to natychmiastowe przejście w stan destroyed, niezależnie od przebiegu metody `destroyApp`. W przypadku false, MIDlet chce zakomunikować, że chce pozostać w obecnym stanie i może wyrzucić wyjątek `MIDletStateChangeException`. To żądanie może zostać uszanowane i wywołanie metody może nastąpić w późniejszym czasie . Jeśli podczas metody `startApp` wystąpi wyjątek inny niż `MIDletStateChangeException`, metoda `destroyApp` będzie wykonana (Listing 2.).

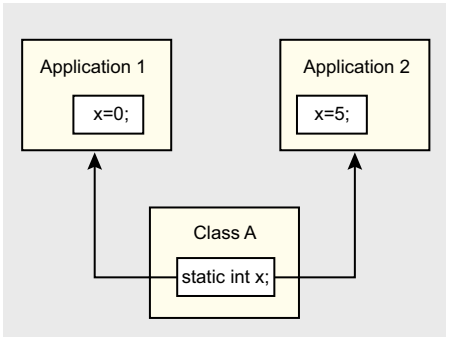
Generalnie porównując możliwe stany MIDletu z Activity Androida można dopatrzyć się podobieństw. Ze względu na to, że Android zarządza swoimi procesami i w zależności od stanu przyznaje określony priorytet, doszło w nim kilka pośrednich stanów.

AMS

Całym środowiskiem aplikacji JavaME oraz cyklem życia MIDletu zarządza system AMS (ang. *application management software*). Egzekwuje on politykę bezpieczeństwa, pozwolenia, stany aplikacji oraz dostarcza klasy systemowe oraz harmonogram. Serwisy oraz mechanizm wymiany komunikatów jak Intent nie zostały jednoznacznie określone w Java ME. Pośrednio można posłużyć się mechanizmem PushRegistry, który zapewnia system AMS. Umożliwia on uruchamianie aplikacji w odpowiedzi na określone zdarzenia (np. przychodzące połączenia, SMS). Po otwarciu odpowiedniego kanału komunikacyjnego można stworzyć system do wymiany komunikatów. Więcej na temat tego mechanizmu można znaleźć w jednym z artykułów, który został opublikowany w Software Developer's Journal 08/2008. W przypadku zestawu MIDletów, kiedy urządzenie pozwala na uruchamianie ich jednocześnie, to zazwyczaj pracują w tej samej maszynie wirtualnej. Oznacza to, że wykorzystują te same obiekty oraz interfejsy. Daje to możliwość zbudowania interfejsu komunikacji między aplikacjami z koniecznością synchronizacji części wspólnej. Niestety nie zostało to tak zgrabnie zrealizowane, jak na platformie Android w postaci Intentów.

Współdzielenie danych prywatnych aplikacji, które zapewnia ContentProvider w Android, między aplikacjami w JavaME możliwy jest tylko w obrębie zestawu MIDletów.

Rysunek 8. *Idle screen (http://www.slideshare.net/paulsu/midp-30-master-class/)*



Rysunek 9. *Dostęp do zmiennej statycznej w konkurencyjnych MIDletach MIDP 3.*

Jeżeli aplikacje znajdują się w dwóch różnych plikach *jar*, to MIDlety nie mają bezpośredniego dostępu do ich lokalnych baz (RMS).

Personal Profile (CDC)

Dla JavaME PP istnieje kilka możliwości budowy aplikacji:

- tradycyjna aplikacja (znana z JSE);
- applet;
- xlet;
- midlet.

Tradycyjna aplikacja

W głównej klasie aplikacji definiuje się statyczną metodę main, której szuka maszyna wirtualna, aby ją wywołać podczas startu. Program działa do momentu wywołana metody `System.exit` lub zabicia procesu z poziomu systemu operacyjnego (Listing 3.). Z wielu powodów tradycyjny model aplikacji sprawdza się w systemach, gdzie nie ma potrzeby przejmowania się zasobami (pamięcią, procesorem) systemu. Gdy aplikacja jest chwilowo niepotrzebna, zostaje np. zminimalizowana bez potrzeby przejmowania się jej stanem pauzy. W przypadku profilu MIDP, gdzie każdy bajt jest cenny, ten typ aplikacji jest nieoptymalny.

Applet

Applet jest rodzajem wbudowanej aplikacji działającej w środowisku przeglądarki internetowej. Jest ściśle powiązany ze stroną internetową, ponieważ zawiera ona obszar interfejsu użytkownika. Użytkownik może przechodzić przez strony w przeglądarce bez potrzeby informowania o tym appletu (Listing 4.). System zarządzania zadba, aby aplikacja przeszła w stan pauzy lub zniszczenia. Cykl życia apletu składa się z następujących stanów:

- *Loaded* – instancja apletu została utworzona, jednak nie przeprowadzono żadnej inicjalizacji. Nie powinno się tworzyć w konstruktorze żadnych zmiennych, ponieważ w tym momencie kontekst systemu operacyjnego nie został przygotowany;
- *Stopped* – aplet został zainicjalizowany, jednak jest nieaktywny w danej chwili. Kiedy kontekst jest gotowy, następuje wykonanie metody init(). W tej metodzie powinny znajdować się wszelkie informacje startowe;
- *Started* – aplet jest aktywny;
- *Destroyed* – zamknięcie i wyczyszczenie pamięci po aplikacji.

Kontekst apletu pozwala na interakcję ze swoim środowiskiem operacyjnym. Dzięki kontekstowi można wykonać wiele czynności na przeglądarce lub systemie operacyjnym (odegranie muzyki, komunikaty, otwarcie nowego okna przeglądarki).

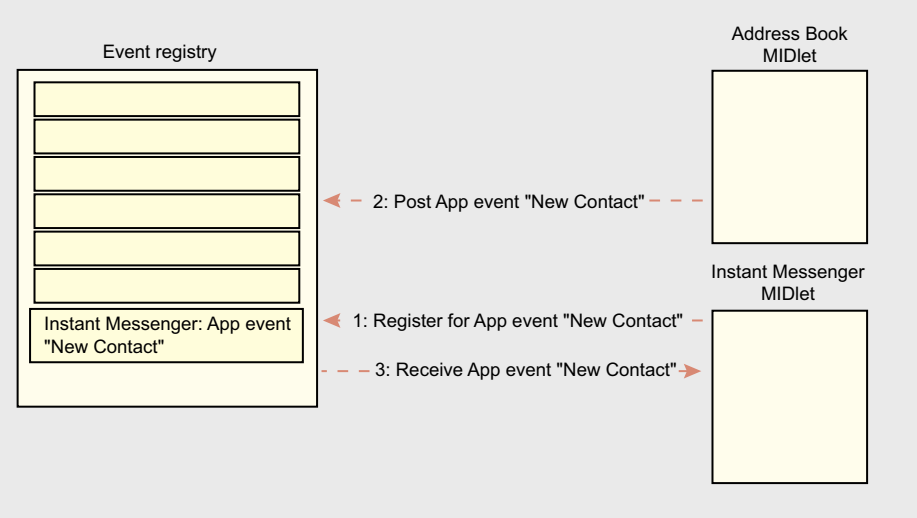
MIDlety także mogą znaleźć się na platformie JavaME CDC, jednak w większości przypadków ich klasy oraz maszyny wirtualne są odseparowane od siebie.

Xlety

Xlety stanowią kolejny typ aplikacji zarządzanych i podobnie jak MIDlety wymagają systemu AMS. Oryginalnie dostępne w Java TV API zostały zaadaptowane do Personal Basic Profile oraz Personal Profile (Listing 5.). Cykl życia Xletu składa się z następujących stanów:

- *loaded* – podobnie jak aplety instancja została skonstruowana, ale niezainicjowana;
- *paused* – stan pauzy Xletu;
- *active* – aktywna aplikacja;
- *destroyed* – zamknięta aplikacja.

Wraz z Xletami został wprowadzony mechanizm wymiany komunikatów IXC (*inter-Xlet communication*). IXC jest prostą odmianą mechanizmu RMI, w której aplikacja mogła zdalnie wywoływać metody obiektów (Rysunek 7.).



Rysunek 10. *Zdarzenia w MIDP 3.*

MIDP 3

Obserwując poczynania grupy zajmującej się rozwijaniem technologii MIDP 3, można wypisać kilka innowacyjnych rozwiązań, które były bardzo oczekiwane w poprzednich wersjach. Zostały wprowadzone kategorie aplikacji takie jak standardowe MIDlety, IdleScreen MIDlet oraz Screen Savers MIDlet (wygaszacze ekranu). IdleScreen MIDlet jest ekranem, który jest prezentowany użytkownikowi, kiedy żadna aktywność nie jest zaobserwowana na urządzeniu. Typowo taki ekran nie jest psuty, ale wypełniony komponentami UI, które prezentują informacje jak na przykład siłę sygnału, poziom baterii, serwisy na urządzeniu, nagłówki wiadomości (Rysunek 8.). Automatyczne uruchamianie MIDletów wraz ze startem systemu. Należy zwrócić tutaj uwagę na mechanizm PushRegistry, który nadal pozostaje.

Konkurencyjne MIDlety

Zapewniają niezależne działanie aplikacji, biorąc pod uwagę:

- Izolację danych, które będzie polegała na tym, że MIDlety zostaną uruchomione we własnym kontekście . Klasy oraz statyczne dane nie będą dzielone między MIDletami, nawet jeśli są w tym samym zestawie. Natomiast cechy jak pozwolenia (ang. *persmissions*), dostęp do RMS oraz zasoby pozostaną dla zestawu dzielone. Zatem należy się spodziewać, że dla przykładu zmienna statyczna nie będzie zmienną globalną dla zestawu jak to miało miejsce dotychczas. Każdy MIDlet będzie posiadał swoją własną kopię niedostępną dla drugiego (Rysunek 9.).
- Obsługa błędów w MIDletach będzie polegała podobnie jak w platformie Android. Błąd w jednej aplikacji nie powinien mieć wpływu na drugą.
- MIDP 3 powinien zapewniać MIDletom politykę szeregowania, aby nie dopuścić do tzw. *zagłodzenia*, czyli sytuacji, gdy któraś aplikacja zbyt długo oczekuje na zasoby. Podobnie jak w Android wprowadzą priorytety dla aplikacji, aby zapewniać im dostęp w zależności od ich ważności. Szczególnie nacisk jest kładziony, jeżeli aplikacja ma focus, czyli użytkownik bezpośrednio na niej działa.

Komunikacja między MIDletami.

Inter Midlet Communication wprowadza asynchroniczny dwukierunkowy strumieniowy protokół komunikacji. Podobnie jak protokół gniazd umożliwia ustawienie dwóch połączeń jako klient lub serwer. Dostęp do tych połączeń będzie kontrolowany przez politykę bezpieczeństwa. Klient może wyspecyfikować

URL do jednego serwera lub użyć maski wildcard, aby dostać się do jakiegokolwiek serwera wspierającego dany serwis. Aplikacje mogą wykorzystać PushRegistry, aby mogły być uruchomione w przypadku nadchodzącego połączenia. Serwer IMC ma możliwość filtrowania klientów, którzy się do niego chcą połączyć na podstawie domeny, producenta, podpisu.

Mechanizm notyfikacji o zdarzeniach

W MIDP 3 zdefiniowano dwa typy zdarzeń: systemowe i między aplikacjami. Systemowe komunikaty są informacją od systemu o zaistniałym przerwaniu lub innym zdarzeniu. App-to-app są zdarzeniami między aplikacjami, które się zarejestrowały w celu ich odbioru (Rysunek10.) .

Współdzielone biblioteki

LIBlety są niezależną paczką kodu oraz zasobów, które mogą być dzielone przez MIDlety. Mogą także być zależne od innych LIBletów. Wszystkie te biblioteki są zależne od kontekstu uruchomionego MIDletu. W LIBletach zdefiniowane są zależności między MIDletami oraz między nimi samymi. Mogą ograniczyć czas pobierania aplikacji, jeżeli ta zadeklaruje, że wykorzystuje daną bibliotekę (więcej patrz ramka W Sieci). Inne cechy nowego profilu zostaną opisane w kolejnych artykułach gdzie będą porównywane z możliwościami Android.

Podsumowanie

Przedstawiony artykuł ukazuje pierwszą bitwę między platformą Android oraz Java ME. W obszarze konfrontacji znalazły się ogólne koncepcje cech funkcjonalnych porównywanych technologii. Jak można było zaobserwować, istnieje duże podobieństwo na pewnych szczeblach architektury. Wprowadzono maszynę wirtualną oraz zbiór bibliotek odpowiedzialnych za określone zadania. Każda aplikacja posiada swój cykl życia oraz istnieją mechanizmy wymiany komunikatów. Jednak rzuca się w oczy fakt, że Google zrealizowało wszystkie funkcjonalności w ramach jednej platformy. Natomiast Sun porozbijał swoje na kilka platform Java ME (takich jak PersonalProfile, MIDP 2 oraz MIDP 3). Uzależnione to było od mocy obliczeniowej dostępnych w ówczesnym czasie urządzeń, które nie mogły zrealizować pewnych zadań. Ale postęp technologiczny nie stanął w miejscu, a polityka innowacyjności Suna chyba tak. Od kilku lat urządzenia mogą pochwalić się coraz szybszymi procesorami oraz większą pamięcią. Google wykorzystało tę sytuację i udało się mu wprowadzić platformę, która skupia wszystkie najlepsze cechy pozostałych trzech platform Java ME. Tak naprawdę z Androidem może konkurować jedynie konfiguracja MSA2. Skupia ona wszystkie dotychczasowe osiągnięcia Java-ME oraz wprowadza nowy profil MIDP 3. Jednak do tej pory nie doczekaliśmy się urządzenia z tą platformą. Przekonanie producentów do wprowadzenia nowej architektury Sun będzie coraz trudniejsze. Już w obecnej chwili można zaobserwować wiele deklaracji wytwórców urządzeń mobilnych wdrożenia Androida. Przewagą liczby urządzeń, na których dostępna jest Java ME, na razie może poszczycić się Sun. Z tego względu na pewno przez jakiś czas będą powstawały aplikacje na te urządzenia. Dlatego w kolejnych artykułach pójdziemy dalej i porównany dane technologie w kolejnych obszarach:

- Możliwości języka programowania;
- Licencje oraz dostęp do kodów źródłowych;
- Typowe problemy;
- System dystrybucji oprogramowania;
- Bezpieczeństwo;
- Interfejs użytkownika;
- Multimedia i gry;
- Komunikacja;
- Wybrane biblioteki;
- Optymalizacja oraz narzędzia.

W następnej części zostanie wprowadzony szkielet aplikacji, którą będę rozwijał przez następne odcinki kampanii wojennej. Dzisiejszy odcinek wygrywa Android ze względu na eleganckie zintegrowanie funkcjonalności oczekiwanych od dłuższego czasu przez programistów.

W Sieci

Telefon Android:

- *http://www.androidal.pl/t-mobile-g1htc-dream-podzespoly/176;*
- *http://www.fkn.pl/1,351,1860762,1,wiadomosc.html;*
- *http://www.t-mobileg1.com/not-found.aspx?aspxerrorpath=/g1-announcement.aspx;*
- *http://www.androidal.pl/t-mobile-g1-dla-deweloperow/209.*

Wymagania urządzenia Sun:

- *http://java.sun.com/j2me/docs/pdf/CLDC-HI_whitepaper-February_2005.pdf);*
- *http://java.sun.com/products/cldc/wp/KVMwp.pdf, http://pl.wikipedia.org/wiki/MIDP http://midlety.net.*

Portale developerskie:

- *http://developer.samsungmobile.com/Developer/index.jsp;*
- *https://developer.sonyericsson.com/site/global/home/p_home.jsp;*
- *http://www.forum.nokia.com/;*
- *http://developer.motorola.com/;*
- *http://developer.openwave.com/dvl/;*
- *http://www-01.ibm.com/software/wireless/weme/;*
- *http://na.blackberry.com/eng/developers/.*

Java ME:

- *https://opensource.motorola.com/sf/frs/do/downloadFile/projects.jsr271/frs.jsr_271_early_draft_release.jsr_271_public_review_release/frs1659?dl=1;*
- *https://opensource.motorola.com/sf/projects/jsr271;*
- *http://java.sun.com/javame/index.jsp.*

VM:

- *http://java.sun.com/products/cldc/wp/KVMwp.pdf;*
- *http://www.sagecertification.org/events/vee05/full_papers/p153-yunhe.pdf.*

Android:

- *http://forum.xda-developers.com/forumdisplay.php?f=446;*
- *http://groups.google.com/group/android-developers;*
- *http://www.androidal.pl/;*
- *http://code.google.com/android/.*

OSGi:

- *http://www.osgi.org/About/Technology?section=2.*

ALBERT WACHOWICZ

Albert Wachowicz pracuje na stanowisku Technology Unit Manager w BLStream wchodzącym w skład Grupy BLStream. Grupa BLStream powstała, by efektywniej wykorzystywać potencjał dwóch, szybko rozwijających się producentów oprogramowania – BLStream i Gamelion. Firmy wchodzące w skład grupy specjalizują się w wytwarzaniu oprogramowania dla klientów korporacyjnych, w rozwiązaniach mobilnych oraz produkcji i testowaniu gier.

Kontakt z autorem: albert.wachowicz@gmail.com.

Czy nowoczesna firma może funkcjonować bez najnowszej technologii?

Po przeczytaniu poniższej historii, z pewnej firmy odpowiedź nasuwa się sama.

Zarzewski pędził jak szalony. Wiedział, że teraz może się wszystko odmienić. Silnik szybko wchodził na obroty, a kolejne biegi zmieniały się, gdy tylko było to możliwe. Drzewa migały za oknami podobnie jak myśli, których nie mógł wszystkich ogarnąć.

Od 10 lat był przedstawicielem, bywały lepsze i gorsze okresy, ale teraz miał szansę odzyskać najlepszego klienta. Co chwila spoglądał na siedzenie, w tej stercie kartek leżało, jak gdyby nic, zamówienie od Zoweksu. To zamówienie może wszystko zmienić. Jeśli uda mu się je dostarczyć szybko do firmy, a towar dotrze do klienta, odzyskają zaufanie, kolejne zamówienia będą już tylko formalnością, a oni zawsze zamawiają dużo. Pozostaje tylko pytanie, czy mamy towary, które są na zamówieniu. Dlaczego akurat, gdy tego najbardziej potrzebował, Majewska nie odbiera telefonów? Każdego dnia wydzwaniali przecież krocie, żeby potwierdzić stany magazynowe, pewnie akurat dzwonili cały czas inni. Teraz to już nie ważne, ma zamówienie, trzeba tylko wpaść na magazyn i zablokować sobie ten towar, dostawa musi być jutro.

– Gdzie jest lofix? Gdzie do cholery jest ten lofix? – Kurzajowski popatrzył na niego spode łba. –Biegaj, skakaj, i tak nie znajdziesz w pięć minut. – za dużo nas tu jest, każdy stawia towar, gdzie wydaje mu się najlepiej, ale jak trzeba coś znaleźć, trzeba zorganizować wielką obławę, to jak poszukiwanie zbiega. Pracujemy na dwie zmiany, każdy wydaje i przyjmuje, nie ma już nikogo, kto ogarnia to wszystko…

Zarzewski pędził przez magazyn, każdego magazyniera pytał, czy wie gdzie jest lofix. To był najważniejszy towar na zamówieniu, wiedział, że jeśli go jutro nie dostarczą, nie będzie następnych zamówień. Te zamówienia mogą być przełomem nie tylko dla niego, ale dla całej firmy.

Na końcu pod ścianą zobaczył Otoczewskiego, który był ostatnią deską ratunku. Ściągał coś z regału i układał na palecie. Wiele lat temu zatrudniali się razem, byli nawet przyjaciółmi, potem poszli w innych kierunkach. Zdarza się. Otoczewski odwrócił się, był dziwnie zadowolony. - O, pan szanowny handlowiec, jaki rozogniony pewnie jakieś ważne zamówienie – odwrócił się i zachichotał.

– Marek! to nie chwila na żarty, wiesz gdzie jest lofix?

– A co, świat się zawali, bo ważny klient, a dostawa to już musi być na jutro, co?

– Heheheheheh. Otoczewski zaczynał się droczyć.

– To naprawdę jest ważne, muszę wiedzieć gdzie to jest, od tego bardzo wiele zależy.

– Powiedzieć Ci gdzie to jest? – powiedział i spojrzał na zegarek. – Teraz jest pewnie jakieś 50 kilometrów od Warszawy i wygodnie ma w tej ciężarówce. Wszystko zostało spakowane i odesłane zwrotem do dostawcy. Wiem, bo sam wysyłałem.

– Szeroko się uśmiechnął.

To było dla Zarzewskiego za dużo. Zacisnął pięść, która po chwili spotkała się ze szczęką dawnego przyjaciela. Otoczewski upadł, a Zarzewski wcale się nie uspokoił, zaczął wrzeszczeć.

– Zapieprzam cały dzień. Na oślep robię zamówienia, nie wiedząc czego ile mamy. Wypruwam z siebie flaki. Świecę oczami przed klientami. A potem przyjeżdżam tu i widzę jak ruszacie się jak muchy w smole, wydajecie nie to, co na dokumentach, sami nie wiecie gdzie znaleźć jakikolwiek towar, a do tego macie to wszystko gdzieś. Niech Cię szlag trafi. Co się z Tobą stało?!

– Odwal się. – Otoczewski powoli wstawał – Też mam tego dosyć, skąd mam wiedzieć gdzie wstawił towar koleś z poprzedniej zmiany, no skąd? A jak jesteś taki mądry, to sam wydawaj prawidłowo towary, jeżeli najpierw go musisz szukać, a potem widzisz na WZ tylko fragment nazwy, tych towarów są teraz tysiące, dla mnie sztuka jest sztuką, inaczej bym zwariował. A ten twój lofix, gdyby zamówienie było wcześniej, na pewno by nie wyjechał. Jak Ci się nie podoba, namów szefa, żeby wreszcie kupił jakiś system, który sobie z tym poradzi, a jak nie, to zostaw mnie w spokoju…

Zastosowanie systemów opartych o rozwiązania mobilne typu palmtopy Pocket PC czy terminale radiowe z zainstalowanym Windows CE umożliwia rozwiązanie wielu problemów związanych z przetwarzaniem, zamówień czy logistyką magazynową. Obecnie urządzenia mobilne znajdują także zastosowanie poza handlem i logistyką magazynową, np. w firmach transportowych, leśnictwie i innych branżach, w których przemieszczający się pracownicy gromadzą dane, przetwarzają je lub rejestrują swoje działania.

Przedstawiciele handlowi

Systemy serii PS_Mobile wspomaga pracę przedstawicieli handlowych. Zadaniem programu jest dostarczanie przedstawicielom informacji potrzebnych w codziennej pracy, a których aktualność ma zasadnicze znaczenie dla procesów handlowych. System pobiera dane z systemu handlowego i w dogodny sposób wyświetla potrzebne informacje.

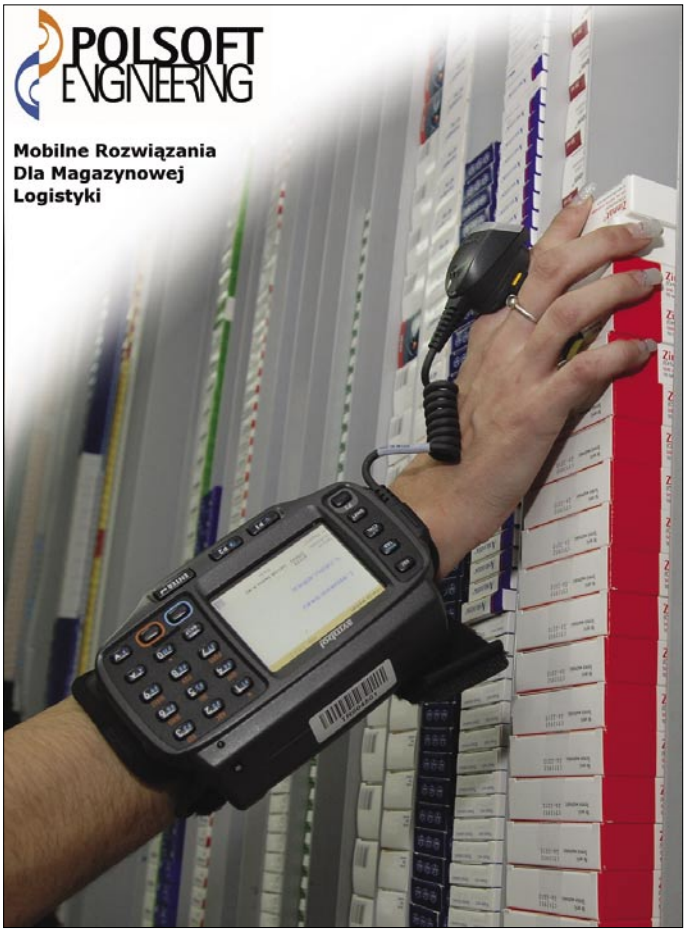
W trakcie przyjmowania zamówienia często krytyczna jest wiedza o stanach magazynowych, tak by przedstawiciel mógł potwierdzić klientowi, że realizacja zamówienia jest możliwa. Istotne jest także informowanie klienta o dokładnych cenach produktów, także wynikających ze wcześniejszych szczegółowych warunków cenowych przyznanych klientowi czy okresowych promocji. W trakcie przyjmowania zamówienia, może być także istotne, kiedy, w jakiej ilości i cenie klient wcześniej kupował wybrany pro-

dukt. Rejestracja historii sprzedaży na urządzeniu może być także pomocna w przypadku wyszukiwania produktów. W sytuacji, gdy przedstawiciel posiada w ofercie kilka lub kilkadziesiąt tysięcy produktów ograniczenie listy do towarów kupowanych przez danego klienta może bardzo przyspieszyć edycję zamówienia, a często zwiększyć jego wartość, gdyż przedstawiciel może zasugerować klientowi ponowienie zakupów wybranych towarów, o których klient może nie pamiętać.

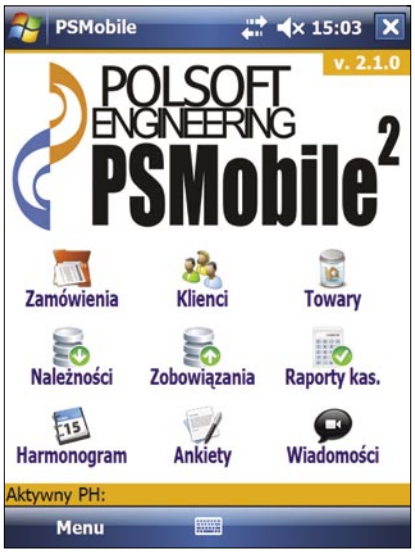
System jest także pomocny w realizacji dodatkowych zadań. Wyświetlane na palmtopie rozrachunki zgodne ze stanem rozrachunków z centrali pozwalają zwiększyć skuteczność windykacji. Istniejący moduł Kasa pozwala na rejestracje wpłat gotówkowych od klientów, umożliwia automatyczne tworzenie dokumentów KP i daje możliwość ich wydruku na drukarkach mobilnych. Dodatkowo, centrala ma stały dostęp do informacji o stanie kasy u poszczególnych przedstawicieli.

Innym często stosowanym udogodnieniem jest zbieranie informacji miękkich z terenu. Pracownik w centrali definiuje ankiety zawierające odpowiednie pytania, które mogą być przyporządkowane do poszczególnych klientów lub grup. Handlowcy gromadzą na palmtopach i automatycznie, podobnie jak zamówienia, przesyłają do centrali. W panelu administracyjnym dostępne są wyniki zobrazowane dodatkowo wykresami, pozwalające szybko i precyzyjnie ocenić, potrzeby klientów, procent zadowolenia itd.

Zastosowanie systemu pozwala zwiększyć wartość zamówień, znacznie przyspieszyć czas upływający od momentu przyjęcia zamówienia do dostarczenia do klienta, zwiększyć skuteczność przedstawicieli, którzy są w stanie realizować więcej wizyt. Przenoszenie zamówień bezpośrednio do systemu pozwala wyeliminować błędy powstające przy przepisywaniu zamówień. Dodatkową często zauważalną zmianą jest większa skuteczność ściągania należności.



Rysunek 1. *Zastosowanie mobilnych terminali radiowych w logistyce magazynowej.*



Rysunek 2. *Menu główne aplikacji wspomagającej pracę handlowca w terenie*

Mobilny magazyn

Zastosowanie mobilnych rozwiązań na magazynie pozwala na niemal całkowitą eliminację dokumentów papierowych. Magazynier posługując się terminalem radiowym w każdym miejscu magazynu ma dostęp on-line do systemu. Zarówno poszczególne lokalizacje na magazynie, jak i produkty mogą być oznaczone kodami kreskowymi, które pozwalają na zapisanie niezbędnych informacji koniecznych do zidentyfikowania produktów, nawet jeśli konkretny produkt musi być identyfikowany z dokładnością do daty produkcji lub serii.

System ułatwia przyjmowanie towaru do magazynu, wydawanie, przeprowadzanie inwentaryzacji i inne operacje magazynowe. Przygotowanie towaru do wydania odbywa się na podstawie dokumentu WZ wyświetlanego na ekranie urządzenia. System wskaże prawidłową lokalizację towaru, a także zweryfikuje przy pomocy kodu kreskowego wydanie odpowiedniego produktu. Już w momencie skanowania kodu kreskowego produktu informacje na temat tej operacji są widoczne w systemie centralnym, nawet jeśli praca odbywa się w magazynie odległym od centrali o setki kilometrów.

Zastosowanie rozwiązań mobilnych na magazynie pozwala znacznie zwiększyć zdolność do wydawania i przyjmowania towarów bez konieczności zatrudniania dodatkowych pracowników.

Inne zastosowania

Mobilne oprogramowanie w połączeniu z zaletami urządzeń przenośnych znajduje zastosowanie w wielu innych obszarach. Zastosowanie urządzeń mobilnych może być pomocne również w zarządzaniu flotą - w przypadku firm transportowych, dla których istotne jest nie tylko aktualne położenie pojazdów, ale też informacje, które powinny docierać do firmy od kierowców, dotyczące przewożonego towaru, stanu licznika, miejsc docelowych itd.

ADAM TYLA

Koordynator ds. Współpracy z Partnerami w firmie Polsoft Engineering.
Kontakt z autorem: atyla@polsoft.pl

KLUB PRO

Oferta skierowana dla firm

Jeżeli Twoja firma jest prenumeratorem Software Developer's Journal za comiesięczną dopłatą 50 PLN +VAT możesz dodatkowo otrzymać reklamę.
Wystarczy tylko, aby profil Twojej firmy pokrywał się z naszym magazynem.

Wyślij do nas: logo firmy, dane kontaktowe i informację o firmie

Reklama przez 12 kolejnych numerów tylko za 600 PLN +VAT.

Jeżeli nie posiadasz jeszcze prenumeraty możesz ją zamówić w atrakcyjnej cenie.

Dla nowych prenumeratorów specjalna oferta – 690 PLN.

Skontaktuj się z nami:

sdj@software.com.pl tel. 22 427 36 91
pren@software.com.pl *http://buyitpress.com*

Opera Software

Opera Software's vision is to deliver the best Internet experience on any device. We are offering browser for PC/desktops and embedded products that operates across devices, platforms and operating systems. Our browser can deliver a faster, more stable and flexible Internet experience than its competitors.

http://www.opera.com

Architektury systemów IT

Twórca frameworków JUVE i serwera aplikacji AVAX oferuje usługi, doradztwo, rozwiązania do tworzenia nowoczesnych, dużych systemów i rozwiązań informatycznych/internetowych, integrujące architektury ery post-J2EE/.NET, wykorzystujące MDD/MDA dla dziedzin – bankowość, telekomunikacja, handel, e-commerce, ERP/Workflow/CRM, rozwiązania internetowe, portalowe.
www.mpsystem.com mpsystem@mpsystem.com

FRONTLINE STUDIOS

Frontline Studios – amerykańsko-polski twórca gier konsolowych oraz PC – szuka utalentowanych koderów (bardzo dobra znajomość C/C++; Java, Delphi, UML), doświadczonych grafików (2D, 3D) oraz zespołów developerskich. Oferujemy długoterminową pracę nad poważnymi projektami. Nie przegap tej okazji!

kadry@frontstudios.com

Future Processing

Future Processing to dynamiczna firma technologiczna działająca na globalnym rynku oprogramowania. Jesteśmy zespołem wysokiej klasy specjalistów posiadających wiedzę i doświadczenie niezbędne do realizacji ambitnych projektów informatycznych. Jeśli programowanie to Twoja pasja dołącz do nas! (możliwość pracy zdalnej).

http://www.future-processing.pl

Kei.pl

Kei.pl działa na rynku usług hostingowych od 2000 roku. Do naszych zadowolonych Klientów z dumą możemy zaliczyć wiele przedsiębiorstw sektora MSP, instytucji oraz osób prywatnych. W ofercie Kei.pl znajdują się pakiety hostingowe, a także usługi dla wymagających Użytkowników – platformy *e-Biznes oraz serwery fizyczne.*

http://www.kei.pl

WSISiZ w Warszawie

INFORMATYKA ZARZĄDZANIE
studia stopnia I i II (stacjonarne i niestacjonarne) specjalności: inżynierskie, magisterskie i licencjackie. Szczegółowe plany studiów, opisy poszczególnych specjalności – zapraszamy na stronę uczelni.

http://www.wit.edu.pl

KLUB PRO

StatConsulting

StatConsulting to firma o znaczącej pozycji na rynku usług analitycznych oraz Data Mining. Nasza oferta obejmuje m.in. modele scoringowe, rozwiązania Fraud Detection oraz modelowanie ryzyka. Tworzymy także własne rozwiązania informatyczne na potrzeby projektów Data Mining, Data Quality, zarządzania ryzykiem itd.

http://www.statconsulting.com.pl

TTS Company Sp. z o.o.

Sprzedaż i dystrybucja oprogramowania komputerowego. Import programów na zamówienie. Ponad 200 producentów w standardowej ofercie. Chcesz kupić oprogramowanie i nie możesz znaleźć polskiego dostawcy? Skontaktuj się z nami – sprowadzimy nawet pojedyncze licencje.

http://www.OprogramowanieKomputerowe.pl

Transition Technologies S.A.

Firma w branży high-tech od 1991. Producent oprogramowania dla przemysłu (ponad 350 referencji z instalacji na całym świecie). Usługi z zakresu: hurtownie danych i Business Intelligence, rozwiązania eBusiness, optymalizacja, integracja danych, aplikacji oraz procesów biznesowych, portale korporacyjne, consulting i outsourcing usług IT, zaawansowana automatyka cyfrowa.

http://www.tt.com.pl

EPRO

Naszą misją jest projektowanie najwyższej jakości dedykowanych systemów IT, które cechuje wysoka niezawodność, wydajność, ergonomiczność i intuicyjność w obsłudze oraz administracji. Głównym elementem oferty EPRO jest oprogramowanie sklepu internetowego oraz identyfikacja wizualna.

tel. 085 743 66 38

http://www.epro.com.pl

IT SOLUTIONS

Wdrożenia i szkolenia z zakresu:

- SQL Server
- SharePoint Services
- MS Project / Server
- Tworzenie aplikacji w technologii .NET

http://www.itsolutions.biz.pl

marcin.pytlik@itsolutions.biz.pl

Softline rozwiązania mobilne

Wiodący producent systemów mobilnych, dostawca aplikacji użytkowych dla biznesu (Symbian OS, Windows Mobile, J2ME) zaprasza do współpracy. Zostań naszym partnerem. Dołącz do zespołu.

http://www.softline.com.pl

Proximetry Poland Sp. z o.o.

Proximetry Poland Sp. z o.o. jest polskim oddziałem amerykańskiej firmy Proximetry Inc. – dostawcy systemów zarządzania sieciami bezprzewodowymi opartymi na technologiach WiFi i WiMAX. Naszą misją jest dostarczenie klientom rozwiązań poprawiających jakość usług (QoS) dostarczanych drogą radiową. Dołącz do najlepszych i zostań członkiem naszej ekipy!

http://www.proximetry.com

SWD Software Sp. z o.o

System operacyjny czasu rzeczywistego RTOS QNX. Oficjalny dystybutor w Polsce.
Zakres działalności: lokalizacja produktów QNX, dostawy sprzętu i oprogramowania, doradztwo przedsprzedażowe, wsparcie techniczne, certyfikowane szkolenia, opracowania na zamówienie.

http://www.swdsoft.pl

Systemy bankowe, ISOF

HEUTHES istnieje na rynku od 1989 r. Obok systemów informatycznych dla banków, oferuje nowoczesne oprogramowanie do obsługi firm. System ISOF jest udostępniany klientom w trybie SaaS lub licencji. Pracuje na platformie Linux i zawiera m.in. takie moduły jak CRM, DMS, Magazyn, Sprzedaż, Logistyka oraz Rachunkowość.

http://www.isof.pl

Roczna prenumerata

tylko 250,-

Software Developer's Journal (poprzednio Software 2.0) jest miesięcznikiem głównie dla programistów, którzy liczą, że dostarczymy im gotowe rozwiązania, oszczędzając im czasu i pracy. Jesteśmy czytani przez tych, którzy chcą być na bieżąco informowani o najnowszych osiągnięciach w dziedzinie IT i nie chcą, żeby jakiekolwiek istotne wydarzenia umknęły ich uwadze. Aby zadowolić naszych czytelników, prezentujemy zarówno najnowsze rozwiązania, jaki starsze, sprawdzone technologie.

Kontakt

1. Telefon
+48 22 427 36 91
+48 22 427 36 79
+48 22 427 36 50

2. Fax
+48 22 244 24 59

2. Online
pren@software.com.pl

3. Adres
Bokserska 1
02-682 Warszawa
Polska

Zamówienie prenumeraty

Zadzwoń
+48 22 427 36 93
lub
zamów
mailowo!

Prosimy wypełniać czytelnie i przesyłać faksem na numer:
00 48 22 244 24 59
lub listownie na adres:
Software-Wydawnictwo Sp. z o. o.
ul. Bokserska 1
02-682 Warszawa
Polska
E-Mail: *pren@software.com.pl*
Przyjmujemy też zamówienia telefoniczne:
0048 22 427 36 91
0048 22 427 36 79
0048 22 427 36 50

Jeżeli chcesz zapłacić kartą kredytową,
wejdź na stronę naszego sklepu internetowego *www.buyitpress.com.*

Imię i nazwisko .....

Nazwa firmy.....

Dokładny adres .....

.....

Telefon .....

E–mail .....

ID kontrahenta .....

Numer NIP firmy .....

Fax (wraz z nr kierunkowym) .....

□ automatyczne przedłużenie prenumeraty

Prenumerujesz – zyskujesz

- oszczędność pieniędzy
- szybka dostawa
- prezenty
- bezpieczna płatność on–line

Tytuł	Ilość numerów	Ilość zamawianych prenumerat	Od numeru pisma lub miesiąca	Cena
Software Developer's Journal (1 płyta CD) – dawniej Software 2.0	12			250 PLN

Software Developer's JOURNAL
new ideas & solutions for professional programmers

W NASTĘPNYM NUMERZE SOFTWARE DEVELOPER'S JOURNAL 3/2009 (171)

Kolejny numer SDJ będzie poświęcony Testowaniu oprogramowania

Ponadto w magazynie:

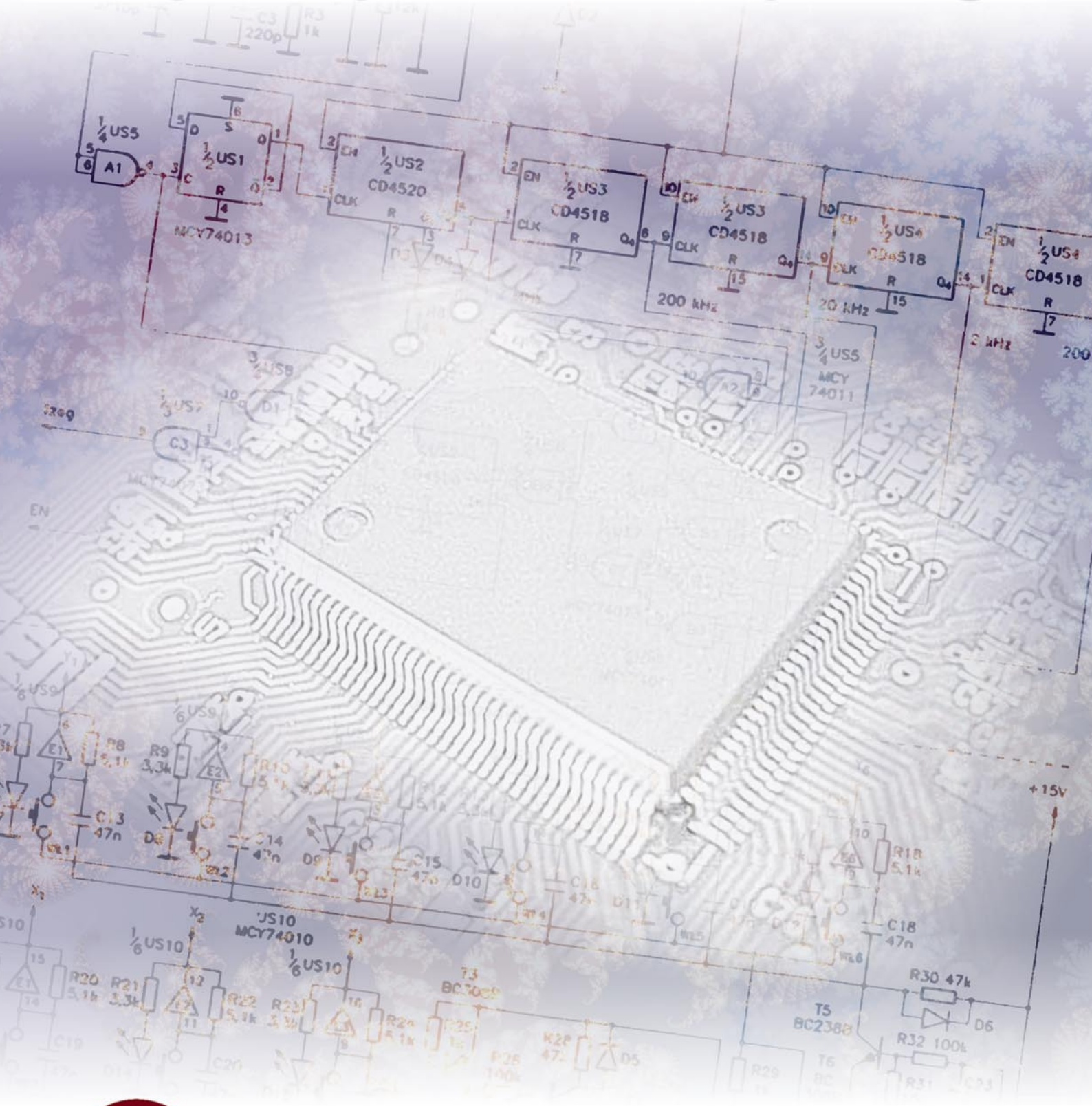
- **Testowanie aplikacji ASP. NET - WebAii**
- **Integracja ciągła – proces, który na bieżąco śledzi gotowość projektu do utworzenia wersji instalacyjnej**
- **Spring Java Configuration Project – alternatywa dla tradycyjnej konfiguracji XML**
- **Weryfikacja przez model z narzędziem SPIN**

W sprzedaży w drugiej połowie lutego

Redakcja zastrzega sobie możliwość zmiany zawartości pisma.

Software Developer's JOURNAL
new ideas & solutions for professional programmers

- Układy Embedded
- Sterowniki mikroprocesorowe
- Systemy czasu rzeczywistego



IT.mobile
BETTER SOFTWARE

Piotrkowska 19/4, Gdansk
info@it-mobile.pl
0-58 741 59 00

MOTOROLA

Centrum Oprogramowania Motoroli w Krakowie jest jednym z wielu centrów technologicznych Motoroli na świecie. Tworzone w nim są systemy komunikacyjne w standardzie TETRA/APCO dla służb bezpieczeństwa publicznego, jak również rozwijane jest oprogramowanie dla stacji bazowych telefonii komórkowej w systemach GSM, CDMA i UMTS.

INVENT
DISCOVER
CREATE

Codziennie przekonujemy się, że nie ma rzeczy niemożliwych. Poznaj nas bliżej. Dołącz do naszego zespołu!

UNLEASHYOURMOTO

www.MotorolaCareers.com/Poland