# MetroUI

To czym my jako programiści się zamujemy średnio obchodzi przeciętnego użytkownika – aby zrobić na nim wrażenie liczy się efekt końcowy i właśnie tu widać największe zmiany – bo nietrudno nazwać rewolucją zupełnie nowe podejście do interfejsu użytkownika.

Metro UI to zupełnie nowa jakość aplikacji – na pierwszy rzut oka to jednolity i spójny styl (fonty – rodzina Segoe UI, spójny User Experience np. tytuły i przyciski operacji dostępne w jednym miejscu). To zupełnie nowe pomysły dotyczące zagospodarowania przestrzeni – każda aplikacja ma możliwość wyświetlania w 3 trybach: Fullscreen (domyślne wypełnienie całego ekranu), Filled (wypełnienie dostępnej przestrzeni) i Snapped (bardzo wąski pasek boczny). Dzięki temu interfejsy są dużo lepiej dopasowane do ekranów tabletów, urządzeń mobilnych gdzie zagospodarowanie przestrzeni nabiera jeszcze większego znaczenia niż na komputerach stacjonarnych.

Sam Microsoft reklamuje swoje innowacyjne podejście do interfejsów użytkownika jako „Fullscreen, immersive, interactive and responsive experience” – zobaczmy jak ono wygląda.

[demo]

Nie trudno oprzeć się wrażeniu, że nowy interfejs jest wyjątkowo zoptymalizowany pod kątem dotykowych ekranów, do tego stopnia, że użytkownik standardowych monitorów żałuje że jego panel nie jest dotykowy. Samo GUI jest schludne i proste, wręcz nawet może się wydawać ascetyczne. Widać jednak wyraźnie, że hasło „less is more” firma z Redmond wzięła sobie do serca.

Nie trzeba daleko poruszać się po całym systemie aby dostrzec różnicę – już lock screen i start screen znacznie odróżniają się od tego co znamy z Windows 7 (opowiastka o jakości, UX, spójnym GUI i wyglądzie – blablabla <improwizuj>) – panel sterowania, charms, przykładowe aplikacje – tweetorama, alarms, stock etc.

# Wnętrze

Oczywiście poza powierzchnią i warstwą typowo zewnętrzną interesują nas zmiany w samym wnętrzu (bebechy:P) a tych jest sporo:

[slajd z platformą]

**Podwójny UI Stack**

**XAML Managed = C#/VB**

**XAML WinRT - C++/C**

**HTML5 + CSS3 - JavaScript**

Ogólnie rzecz biorąc możemy zauważyć podział na dotychczasowe podejście, które współdzielą aplikacje desktopowe i webowe „starego typu”(niebieski) i podejście nowe, zgodne z filozofią aplikacji *Metro Style (zielony)*. Przyjrzyjmy się teraz bliżej warstwie Windows RunTime.

# WinRT

To przedewszystkim *Native bindings* – sposób na dostęp do niższych warstw API spójny dla wszystkich języków programowania – co ważne nie wyklucza **istnienia .NETu i profili aplikacji (PYTANIE XBOX)** – istnieją one równolegle z WinRT, co więcej WinRT opakowywuje standardowe API np. sensorów i udostępnia je aplikacjom wykorzystującym .NET Framework. Widać również, że do natywnych bibliotek został przeniesiony parser i cały podsystem obsługujący XAML.

No MessageLoops, No old dialogs, No GDI - Only DirectX! – Koszmar pisania w Win32API lub Forms staje się przeszłością – wywołania asynchroniczne zastępują eventy i delegacje w nienaturalnych zastosowaniach.

Łączenie języków poprzez WinRTComponent (komponent napisany w C++ jest widoczny w JavaScript i odwrotnie ;))

Portable libraries -> biblioteki przenośne między platformami (ARM, x86) – czyli dotychczasowy mit o przenośności C++ staje się prawdą.

Bardzo ważna rzecz związana z naszym pierwszym tematem (Async) to:

**Wszystkie wywołania z API które trwają >= 50ms są wywołaniami nieblokującymi (\*Async).**

Więc w pełni czerpiemy zalety z asynchronicznych nieblokujących wywołań które pozwalają nam tworzyć responsywne i szybkie interfejsy użytkownika.

Dodatkowo wśród nowych elementów budowy Metro Styled Apps warto rozpocząć od nowego profilu aplikacji specjalnie utworzonego tylko dla tego typu aplikacji. Samo rozdzielenie jest związane z wspominaną wcześniej warstwą WinRT.

Ponadto każda aplikacja ma do dyspozycji nowe środowisko wykonania– AppExecutionEnvironment, w którym możemy wyróżnić dwa konteksty – bezpieczny (ms-wwa://), zaszyty wewnątrz paczki oraz kontekst niezaufany, zewnętrzny (ms-wwa-web://) charakteryzujący się brakiem dostępu do warstwy API systemu, automatycznie zabezpieczanym kontekstem jeśli chodzi o wykonywanie i dołączanie zewnętrznych skryptów. Dodatkowo komunikacja pomiędzy kontekstem zaufanym a webowym odbywa się poprzez stanadardowy sposób znany wszystkim web-developerom – postMessage.

Nowością są również paczki i sposób dostępu do funkcjonalności API zadeklarowane poprzez twórcę aplikacji tzw. Manifest. Jeśli nie zdefiniujemy odpowiednich zależności i nie zadeklarujemy potrzeby dostępu do określonego fragmentu API otrzymamy wyjątek (dodatkowa rola security). Pewne elementy API są dostępne tylko poprzez brokerowane usługi (np. hasła z systemowej skrytki) – zaletą tego rozwiązania jest fakt, że to system zarządza bezpieczeństwem krytycznych dla użytkownika zasobów.

Ciekawostką jest fakt, że od teraz dla Metro Styled Apps instalatory tworzą się automagicznie z paczką podczas budowania naszej aplikacji, dzięki czemu można ją od razu przygotować do wrzucenia do Windows Store.

Ważnym elementem są kontrolki, które już wcześniej jako całościowy obraz zaprezentowaliśmy przy okazji nowego spójnego stylu Metro:

[slajd z kontrolkami]

blablabla <improwizuj>

IE 10 – kiedyś przekleństwo, dziś nowoczesna i bezpieczna przeglądarka…

Najbardziej zaawansowany silnik renderowania HTML5 i CSS3 obecnie na rynku (Irish, Verou).

Zaimplementowane nawet drafty standardów (multicolumn, box model, gradientym, transitions).

JIT dla JavaScript (Chakra z IE 9 na sterydach).

Aplikacje Metro napisane w JS korzystają z tego (ale nie są bundlowane z IE10).

[demo]

* Grid App (Encyclopedia) – prezentacja możliwości platformy WinRT i JavaScriptowych, ***manifest.***
* XHR -> Promises -> then, callback, error handling w jednym miejscu, obsługa asynchronicznych żądań.
* Dom Explorer, Debugowanie (linijka do odkomentowania aby łapać wyjątki), JS console – przeniesienie możliwości IE Web Developer Toolbar do Visual Studio 11 vNext
* Spójność patternów między językami (Promises = Async) – mimo różnic w językach i ich możliwościach najważniejsze jest to że jest zachowana spójność platformy i wzorców które są w niej zaprezentowane
* IE 10 – wygląd desktopowy i chromeless Metro UI – porównanie, precyzja myszy i optymalizacja dla dotykowych interfejsów i mniej precyzyjnych urządzeń wejściowych takich jak palce ;>
* Expression Blend 5 – prezentacja możliwości tworzenia GUI i ogólne przejście, zachwyt nad nowym GUI i interfejsem użytkownika.