SURF

SVM

Visual studio

Accord. Net

Wzorce projektowe (MVVM, ViewMode Locator, Messenger, Command, Repository)

Przegląd istniejących rozwiązań

BoW

HSL

RGB

C# i .NET

Opis kontrolek

WPF

Windows Presentation Foundation jest silnikiem graficznym dostarczanym przez firmę Microsoft.

Jego premiera nastąpiła w 2006 roku, gdy stał się częścią platformy programistycznej .NET w wersji 3.0.

Służy do budowania aplikacji okienkowych dla systemu opracyjnego Windows.

Nowe rozwiązanie umożliwiło w pełni odseparować interfejs użytkownika od logiki biznesowej.

WPF został oparty o kolekcję DirectX zawierającą interfejsy programistyczne aplikacji (z ang. *application programming interfaces*, APIs). Zestaw ten wspomaga generację grafiki, dźwięku oraz innych elementów związanych z aplikacjami multimedialnymi.

ARCHiTEKTURA

Architektura silnika WPF została oparta zarówno o kod zarządzany, jak i o kod natywny.

Większość elementów składowych WPF znajduje się w kodzie zarządzanym, tak jak publiczne API dostępne dla deweloperów.

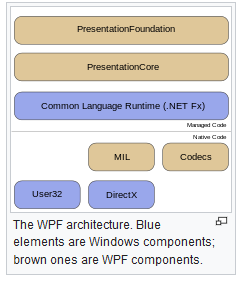
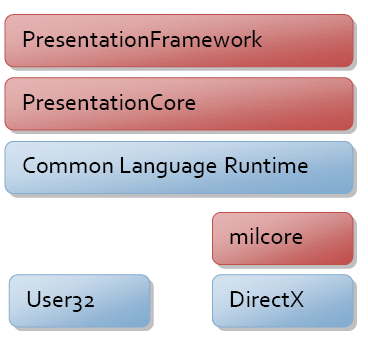
Publiczne API dostępne dla deweloperów jest dostępne jedynie z poziomu kodu zarządzanego

Silnik kompozycji, który renderuje aplikacje WPF jest umieszczony w kodzie niezarządzanym pod nazwą Media Integration Layer (w skrócie MIL) i jest składową biblioteki milcore.dll. Współpracuje on bezpośrednio z DirectX i zapewnia podstawową obługę dla powierznich 2D oraz 3D,

Poprzednikami WPF’a były interfejsy skupione wokół dwóch bibliotek współdzielonych (z ang. Dynamic link library, w skrócie dll): User32 oraz GDI32, które zapewniały obługę okien oraz grafiki dla

Przed premierą

WPF został zaprojektowany niezależnie od dotychczasowego silnika renderującego GDI, który wraz z wiekiem stał się przestarzały oraz nieefektywny.



* PresentationFramework – biblioteka implementująca elementy do prezentacji dla końcowego uzytkownika tj. rozkład kontrolek, wyświetlanie animacji, skalowanie aplikacji.
* PresentationCore – podstawowa biblioteka w technologii WPF. Dostarcza wraper dla MIL z poziomu kodu zarządzanego oraz impementuje bazowe usługi dla każdej aplikacji WPF. W skład tych usług wchodzi przede wszystkim system zarządzania wiadomościami , którego implementację stanowi obiekt typu Dispacher.
* Media Integration Layer, MIL – komponent działający w kodzie niezarządzanym w celu zapewnienia wydajnej współpracy z DirectX. Zawiera silnik kompozycji, który odpowiada za podstawową obsługę renderowania powierzchni 2D oraz 3D.
* Kodeki – zbiór programów odpowiedzialnych do przekształcania strumienia danych do postaci multimedialnej.
* DirectX – kolekcja zawierająca interfejsy programistyczne aplikacji (z ang. *application programming interfaces*, APIs). Zestaw ten wspomaga generację grafiki, dźwięku oraz innych elementów związanych z aplikacjami multimedialnymi
* User32 – komponent Microsoft Windows dostarczający bazowe funkcjonalności do tworzenia prostych interfejsów użytkownika. Aplikacje WPF zawierają obiekt typu Dispacher, który używa systemu zarządzania wiadomościami dostępnymi w User32.
* Common Language Runtime, CLR – wspólne środowisko uruchomieniowe. Podstawowy komponent .NET. Pelni wiele kluczowych roli tj. uruchomienie aplikacji, zarządzanie pamięcią. Dodatkowo zajmuje się również konwersja języka IL do kodu maszynowego. Elementem bazowym środowiska CLR jest standardowy zestaw typów danych, który jest wykorzystywany przez wszyskie języki programowania oparte o CLR.

Silnik WPF udostępnia system własności dla obiektów, które dziedziczą z DependencyObject. Obiekt ten monitoruje wszytkie zależności pomiędzy własnościami i jest w stanie wykonywać odpowiednie akcje bazujac na ich zmianach. Własności implementują mechanizm informujący o zmianach (z ang. Change notifications), który wywołuje wbudowane zachowania (z ang. Behaviors) w przypadku wykrycia jakiejkolwiek zmiany. Dodatkowo isniej możliwość definiowania własnych zachowań w celu propagowania informacji o zmianie własności do innych elementów . System zarządzania rozkładem elementów w obrzarze interfejsu użytkownika wykorzystuje powyższy zbior zachowań do przeliczania nowego rozkładu w przypadku zmiany własności. Dzięki temu architektura systemu WPF spełnia deklaratywny paradygmat programowania, w którym praktycznie wszystko, począwszy od ustawania wielkości kontrolek do tworzenia animacji może zostać osiągnięte poprzez zmianę własności. Takie zachowanie umożliwia tworzenie aplikacji WPF w XAML (z ang. Extensible Application Markup Language) – deklaratywnym języku znaczników, gdzie przy pomocy atrybutów oraz słów kluczowych tworzone jest bezpośrednie połączenie z własnościami oraz klasami technologii WPF.

Każdy element interfejsu aplikacji WPF dziedziczy z abstrakcyjnej klasy Visual. Obiekty tej klasy dostarczają interfejs do drzewa kompozycji zarządzanego przez MIL. Kazdy element WPF tworzy oraz dodaje przynajmniej jeden węzeł kopozycji do drzewa. Węzły te zawierają przede wszytkim istrukcje renderowania takie jak przycinanie elementu bądź transformacja wizualna. Zatem cała aplikacja może być traktowana jako kolekcja węzłów kompozycji, które są przechwowywane w buforze pamięci. Okresowo MIL przechodzi po strukturze drzewa i wykonuje instrukcje renderowania dla każdego węzła. Powoduje to tworzenie kopozytu na powierzchni DirectX, która następnie jest wyświetlana na ekranie. MIL wykorzystuje algorytm malarza, w którym wyświetlanie elementów na monitorze rozpoczyna się od tych najbardziej odległych (tło). Takie zachowanie umożliwia renderowanie złożonych efektów takich jak rozmycie czy transparentność. Dodatkowo proces rysowania jest sprzętowo wspomagany przy pomocy GPU.

Każda z aplikacji WPF statuje z dwoma wątkami: pierwszy służy do obsługi interfejsu użytkownika, a drugi działa w tle do obsługi