Windows Presentation Framework (WPF) – architektúra

A WPF előtt

- GDI – Graphics Device Interface
  - Egy API a kimeneti eszközökhöz
    - Monitor
    - Printer
  - Feladatai
    - Vonalak rajzolása
    - Fontok renderelése
  - Hátrány
    - Animáció készítése nehézkes
    - Nincs szinkronizáció a framebufferrel
    - Nincs 3D raszterizáció

A WPF előtt

- GDI+
  - Leváltja a GDI-t
  - Windows XP-től
  - Új funkciók
    - Élsimítás (2D)
    - Float koordináták
    - Gradient shading
    - JPG/PNG támogatás

A WPF előtt

- Felhasználói élmény: User Experience
  - Nincs designer támogatás (sem eszköz sem ember)
  - Nem kényelmes/bővíthető a programozói modell
- Nagyon régi technológiákra épít
  - GDI (1985): hardveresen támogatott a műveletek nagy része, de keveset tud
  - GDI+ (2001, XP): csak szoftveres, de van például élsimítás, transzparens megjelenés is
  - Direct2D (2008, Vista): hardveresen gyorsított de van software fallback
    - IE9
- Nem használja a mai grafikus kártyákat, rossz teljesítmény:
  - GDI: 10
  - GDI+: 1
  - DirectX/2D: 100-1000

GDI „problémák”

- Raszter alapú megoldás
  - A rajzolás eredménye tipikusan egy bitmappre kerül rá
  - A rajzolás imperatív modellt követ: a kiadott utasítások azonnal végrehajtódnak
- Nehézkes programozási modell
  - WM_PAINT, újrarajzolás, egy szálon
  - A rajzolás az alkalmazás „idejéből” megy, a teljes felhasználói élmény függ a futó alkalmazásoktól
- Jól működött korlátozott memória esetén
  - Biztonság: közvetlenül a képernyőre rajzolhat
- Egyéb problémák
  - Rosszul skálázódik: DPI függő
  - Egész szám koordináták, stb

Limitációk

- Limitált számú GDI objektum
  - Windows 95/98/Millenium
    - 1200
  - Windows 2000
    - 16384
  - Windows XP/Vista/7:
    - 10000 / process (konfigurálható)
    - 65536 / session
- GDI overflow

GDI / GDI+ programozói modell

- GDI: C
  1. GetDC: DeviceContext (rajzolási környezet)
  2. SelectObject (Brush / Pen) -> old
  3. DrawLine, …
  4. SelectObject (old)
- GDI+: C++
  1. GetDC
  2. Graphics(DC)
  3. New Pen / Brush
  4. Graphics.DrawLine(… pen, brush, …)

Az integráció hiánya

- Operációs rendszer szinten nincs integráció
  - Rajzolás (GDI/GDI+ / DirectX, OpenGL, …)
  - Eseménykezelés (Window alapú)
    - Nincs köze a megjelenített objektumokhoz
  - Alkalmazás (nincs adatkötés)
- Egyéb sziget technológiák
  - Képkezelés
  - Videó és hang
  - 3D grafika
  - Érintés, gesztusok, beszédfelismerés, …
  - Dokumentum kezelés

Programozói modell

- Vezérlők és a rajzolás nem objektum orientált
  - Nincs származtatás, egységbe zárás
    - Nincs újrafelhasználás, csak segéd függvények
  - Nincs polimorfizmus, nincsenek virtuális metódusok
    - Nehézkes eseménykezelés
- Nehéz a megjelenés testreszabása
  - Minden rajzolást újra kell írni
  - Imperatív rajzolási modell
- Nem komponálható vezérlők
  - A vezérlő nem meglévő másik vezérlőkből, elemekből áll össze

Agenda

- „Problems” with GDI
- The new declarative modell
- Threading (Dispatcher)
- Dependency / Attached properties
- Visuals
- Physical architecture
- DPI independent coordinates

Integráció

- Teljeskörű integráció
  - 2D / 3D grafika
  - Mediakezelés
  - Beszédfelismerés, érintéskezelés
- Hardveres gyorsítás
  - DirectX
  - Szoftveres render pipeline fallback
  - Új driver modell: WDDM, Vista-tól
- Felbontás független koordinátakezelés

Objektum orientált megközelítés

- Minden vezérlő .NET-es osztály
  - Komplex öröklődő osztály hierarchia
  - A saját osztályok is ebbe épülnek bele
  - Jobban testreszabható származtatással és a virtuális metódusok átírásával
  - A funkcionalitás tisztábban megjelenik az egyes hierarchia ágakon
- A vezérlőhöz nem kötődik megjelenés
  - „Lookless controls”: a megjelenés testreszabható
  - Cél: a vezérlő csak a viselkedésért felelős

Komponálhatóság

- A vezérlők tetszőleges tartalmat meg tudnak jeleníteni
  - Például: Button, Content
  - Content bármi lehet: szöveg, kép, más vezérlők:
    - StackPanel + szöveg + videó
- Az összetett vezérlők más vezérlőkből állnak
  - Az összeépítésre nagyon kevés megkötés van
  - Például: ListView
    - ScrollView a görgetéshez
    - Minden csomópont egy másik tetszőleges vezérlő

Deklaratív rajzolási modell

- A megjelenítendő tartalmat objektumok írják le
  - A programozó az objektumokat hozza létre és azok tulajdonságait állítja be
  - Az objektumok az alkalmazás életciklusa alatt végig léteznek és elérhetőek a program számára
    - Vs. GDI - mentés Windows metafile-ba
- Alap vizuális elemek
  - Geometriai formák: vonal, téglalap, poligon, stb.
  - Szöveg
  - Videó, hang
  - Effektek

Demo

Testreszabható megjelenés

- Deklaratívan, objektumokkal leírható az egyes vezérlők megjelenése
  - Vonal, effektek, stb.
- Leírhatók a különböző állapotokhoz tartozó változások
  - Például letiltott vezérlő
- Az állapotok eseményekhez köthetők
  - Például megnyomják a gombot
- A változások animálhatók

Agenda

- „Problems” with GDI
- The new declarative modell
- Threading (Dispatcher)
- Dependency / Attached properties
- Visuals
- Physical architecture
- DPI independent coordinates

STA – egy szálú működés

- GUI: vezérlők állapotai, eseménykezelés
- Többszálú megközelítés:
  - Az állapot módosító metódusok párhuzamosan futhatnak – nagyon sok zárolás, konzisztencia probléma
  - Eseménykezelők futása – manuálisan kiváltott események párhuzamosak lesznek a felhasználóval
- Megpróbálták
  - Túl lassú
  - Nehéz programozói modell
  - Nagyon hibák – nem determinisztikus deadlockok

Architektúra – osztályok

- Minden objektum (vezérlő stb) egy szálhoz van rendelve, csak arról a szálról elérhető
  - Így van WinFormsban és XWindowban is
  - A DispatcherObject tartja nyilván
- System.Threading.DispatcherObject
  - A legtöbb WPF osztály őse
  - Párhuzamosság ellenőrzés: VerifyAccess, CheckAccess
  - Dispatcher : prioritásos üzenetsor

Dispatcher - használat

```
ThreadStart start = delegate
{
    status.Te = "...";
};
new Thread(start).Start();
```

```
ThreadStart start = delegate
{
  Dispatcher.Invoke(
   DispatcherPriority.Normal,
   new Action<string>(SetStatus),
   "...");
};
new Thread(start).Start();
```

```
DispatcherOperation op =
  Dispatcher.BeginInvoke(
  DispatcherPriority.Normal,
  new Action<string>(SetStatus),
  "... (Async)");

  DispatcherOperationStatus status = op.Status;
  while (status != DispatcherOperationStatus.Completed)
  {
      status = op.Wait(TimeSpan.FromMilliseconds(1000));
      if (status == DispatcherOperationStatus.Aborted)
      { // Alert }
  }
```

Agenda

- „Problems” with GDI
- The new declarative modell
- Threading (Dispatcher)
- Dependency / Attached properties
- Visuals
- Physical architecture
- DPI independent coordinates

DependencyProperty

- A property rendszer további jellemzői
  - Egységes, kiszámítható funkcionalitás
  - Optimális tárolás
  - Aszimmetrikus teljesítmény
- Érték precedencia (egyszerűsített !)
  1. Animáció
  2. Explicit értékállítás XAML-ből vagy programozottan
  3. Stílus
  4. Öröklés: a logikai fában lévő szülőtől
  5. Alapértelmezett érték

A property keretrendszer

- Keretrendszer szolgáltatások
  - Adatkötés, változás értesítési mechanizmussal (INPC)
  - Animáció, sablon, stílus, trigger, öröklés, default érték, …
- WPF specifikus property rendszer
  - *DependencyObject* : alaposztály
  - *DependencyProperty* : a property leírója

```
public static readonly DependencyProperty BackgroundProperty =
    DependencyProperty.Register(„Background”, typeof( Brush ),
      typeof( Control ), … FrameworkPropertyMetadataOptions.AffectsRender,
                                        …Brushes.White, … validationCallback ) );
…
        public Brush Background
        {     get { return (Brush)this.GetValue( BackgroundProperty ); } }
```

Attached Property

- Általában olyan objektumhoz tartozó beállítás amit a más (pl Parent) használ, például:
  - Szülő: *DockPanel* osztály
  - Információ: melyik oldalra dokkoljon
  - AttachedProperty: *DockProperty*, Dock.Left, stb.
  - Bármelyik gyerek elemen be lehet állítani

```
<DockPanel>
    <Button DockPanel.Dock=„Left" />
```

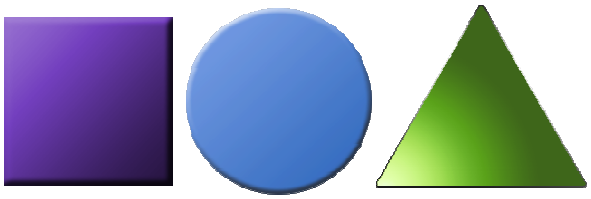
```
DockProperty = DependencyProperty.RegisterAttached( … );
DockPanel.SetDock( uiElement, Dock.Left );
uiElement.SetValue( DockPanel.DockProperty, Dock.Left );
```

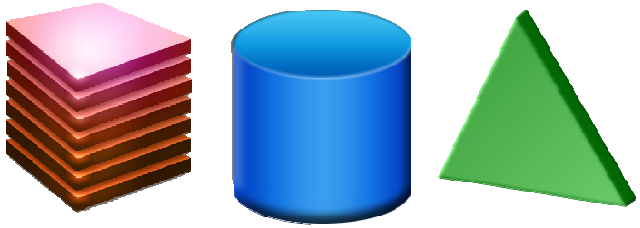
- Általános, kiterjeszthető

Agenda

- „Problems” with GDI
- The new declarative modell
- Threading (Dispatcher)
- Dependency / Attached properties
- Visuals
- Physical architecture
- DPI independent coordinates

WPF megjelenítési csatornák





Segoe UI

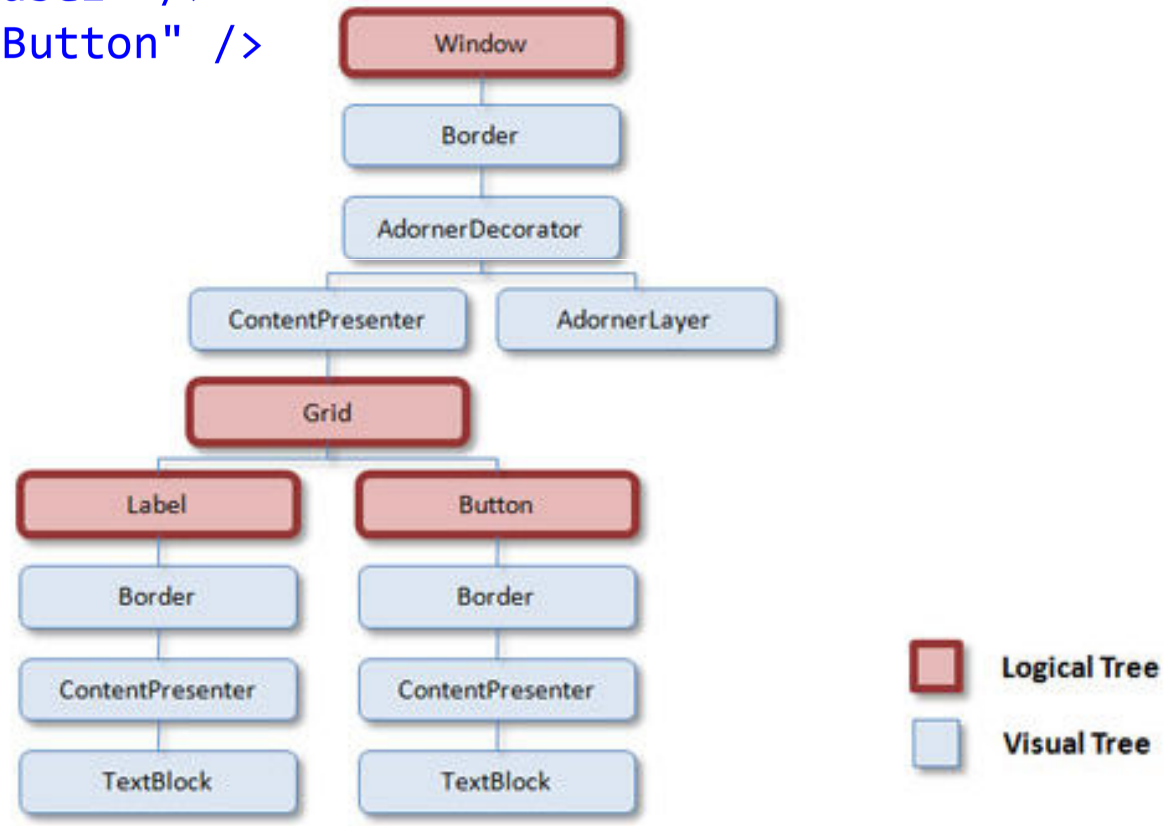
Szöveg

A vezérlőktől a grafikus primitívekig

- Logical tree: a vezérlők hierarchiája
  - Alapvetően a XAML definiálja, tartalom modell
- Visual tree: a megjelenítésért felelős Visual-ök fája
  - Visual: a megjelenés vektorgrafikus összállításáért felel
- A UIElementek tipikusan az OnRender metódusban adják meg megjelenésüket
  - A rajzolt primitíveket a Visual osztály tárolja el
  - A rajzolás a DrawingContexttel történik

Visual tree és logical tree

```
<Window>
   <Grid>
      <Label Content="Label" />
      <Button Content="Button" />
   </Grid>
</Window>
```



DrawingContext

- A DrawingContext tipikus metódusai:
  - DrawLine, DrawEllipse, DrawGeometry, ...
  - DrawText, DrawGlyphRun
  - DrawVideo, PushEffect, PushOpacityMask, ...
    - Paraméterként animáció is megadható !
- A DrawingContext készíti el és tárolja a grafikus primitíveket a Visual osztályban
- A Visualtól Drawing osztályokként lehet lekérdezni a tárolt vektorgrafikus tartalma

Visualból örökölt osztályok

- DrawingVisual: alakzat, kép, szöveg renderelés
- DrawingVisual3D: 3D
- ContainerVisual: más Visualöket tartalmaz

Visual osztály

- System.Windows.Media.Visual
  - Nyilvántartja a rajzolt grafikus primitíveket
  - Rajzolás a képernyőre
  - Kapcsolat a managed API és a milcore között
    - *Composition node*
  - Metaadatok
    - Vágás, transzformáció, bounding rect, hittesting
  - Fa szerkezet => *Visual Tree*
    - Z-order, alfa csatorna, stb

Szöveg megjelenítés

- DrawingContext . DrawText(szöveg, pozició)
  - Szöveg megjelenítés MilCore szinten, DirectX-szel
- FormattedText: a formázott szöveget reprezentál
  - Betűtípus, méret, fékövér, dőlt, aláhúzás, stb.
  - Brush: tetszőleges kitöltés
  - A fenti tulajdonságok akár betűnként változhatnak
  - Méretek, max szélesség, sortáv, igazítás, trimming, ...
  - Lekérdezések: méretek, soronkénti szélességek
  - *BuildGeometry*: alakzat készítése a szövegből

Grafika készítése

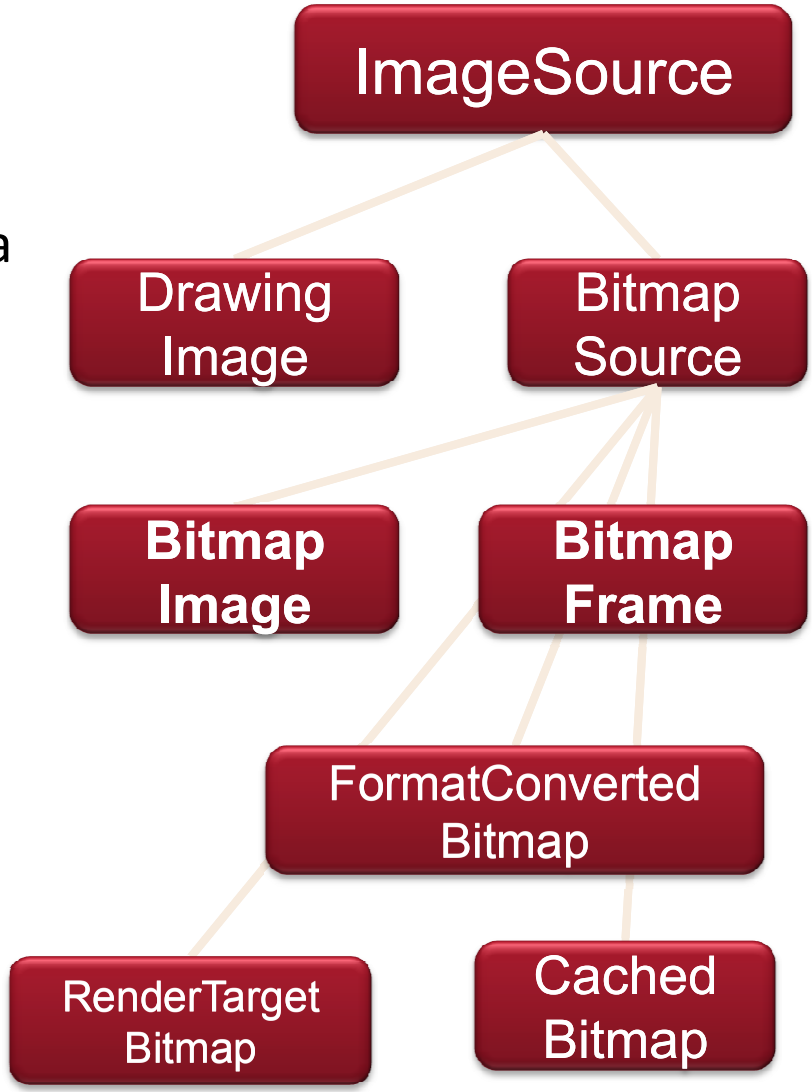
1. Közvetlenül a DrawingContext metódusaival
   - UIElement esetén override-olni kell az OnRender-t
2. DrawingVisual: ‘alakzat’ megjelenítésére képes
   - Drawing osztályok készítésével, például VideoDrawing, GeometryDrawing, ...
   - A GeometryDrawing 2D alakzatokat tartalmaz
     - Brush és Pen az alakzatok kontúrjához és kitöltéséhez
     - Például LineGeometry: végpont koordináták
3. Vagy Shape oszályokkal, például Line, Rectangle
   - Eseménykezelés, layout, transzformációk, stb.

Pen és Brush

- Pen (vagy Stroke): kontúr vonalak formázása
  - Vonalvastagság, végpontok és találkozások
  - Vonalkitöltés, ami egy Brush
- Brush: alakzat kitöltésének formázása
  - SolidColorBrush, Linear/RaidalGradientBrush
  - ImageBrush: kép (tipikusan bitmap)
  - DrawingBrush: alakzat, videó, szöveg – skálázódik
  - VisualBrush: egy vezérlő aktuális képe
  - Tile, stretch, transform, stb.

Kép kezelés

- ImageSource alaposztály
  - Szélesség, magasság, metaadatok
  - ImageBrush, ImageDrawing használja
- DrawingImage
  - vektorgrafikus alakzat van rajta
- BitmapSource: rasztergrafikus kép
- BitmapImage: kép forrás, pl. fájl
  - XAML-ben jól használható
- <Image>: kép megjelenítése
  - XAML-ben
  - ImageSource adja meg a forrást



Párhuzamok

- Leíró osztályok – adat jellegűek
  - LineGeometry, GeometryGroup, MeshGeometry3D
- Grafikus primitíveket leíró osztályok
  - Megjelenítéshez szükséges attribútumok (pl. Brush)
  - MediaPlayer, ImageSource, FormattedText, Drawing (pl. GeometryDrawing), GeometryModel3D, ...
- Framework szintű osztályok: layout, események
  - Shape, MediaElement, ModelUIElement3D
- DrawingContext – DeviceContext, Graphics, HDC

Elnevezési konvenciók

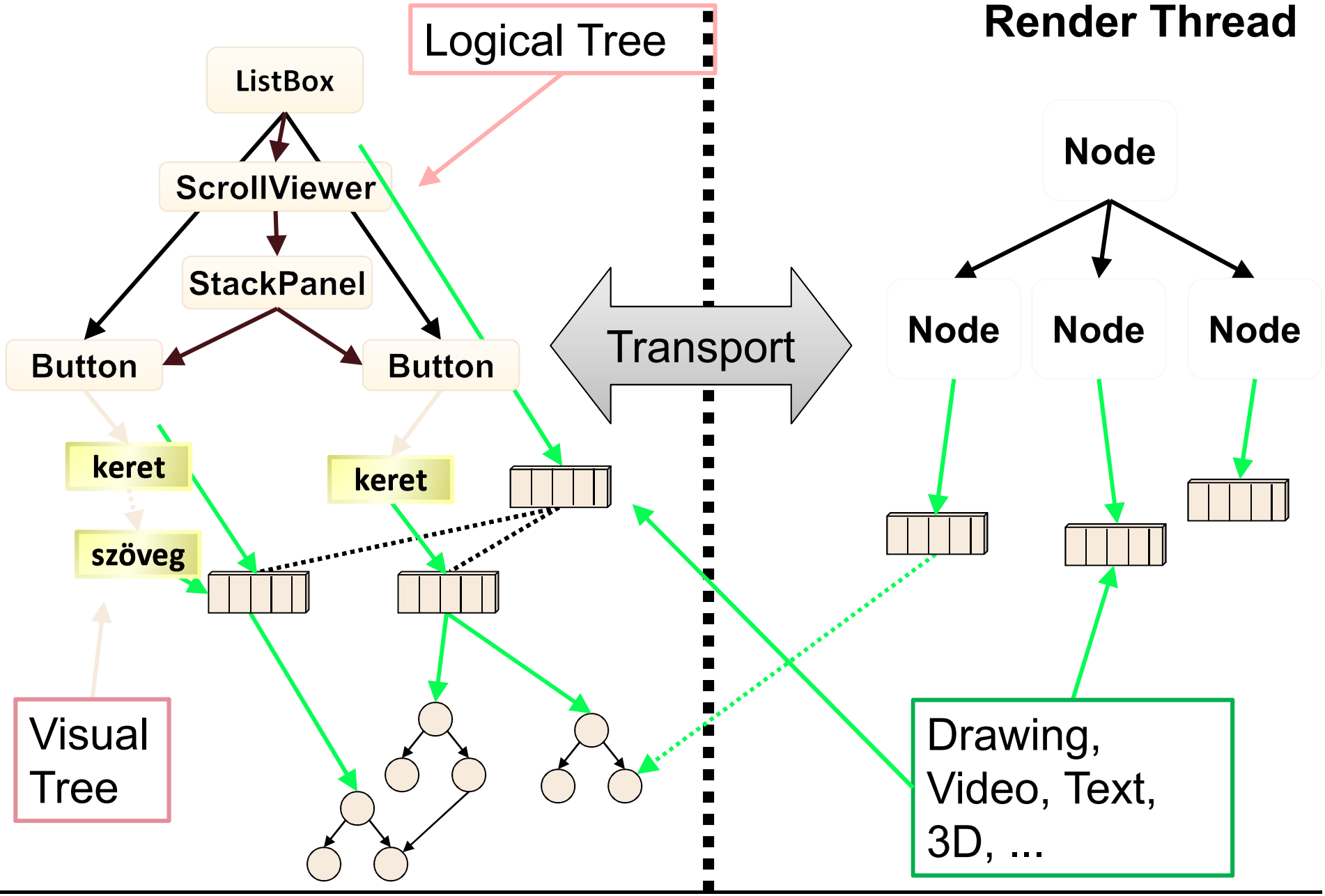
- *Drawing*Visual
  - „olyan Visual osztály amiben *drawing* van”
  - ‘*alakzatot*’ megjelenítő Visualból származó osztály
- ImageSource: kicsit csalóka, csak Image-nek hívják
  - BitmapImage (-Source): raszteres kép
  - DrawingImage: alakzatot megjelenítő vektorgrafikus kép
- ImageDrawing: képet reprezentáló ‘alakzat’
- GeometryDrawing: alakzatot reprezentáló ‘alakzat’
- LineGeometry: vonalat reprezentáló geometria

Agenda

- „Problems” with GDI
- The new declarative modell
- Threading (Dispatcher)
- Dependency / Attached properties
- Visuals
- Physical architecture
- DPI independent coordinates

Alkalmazás, UIThread

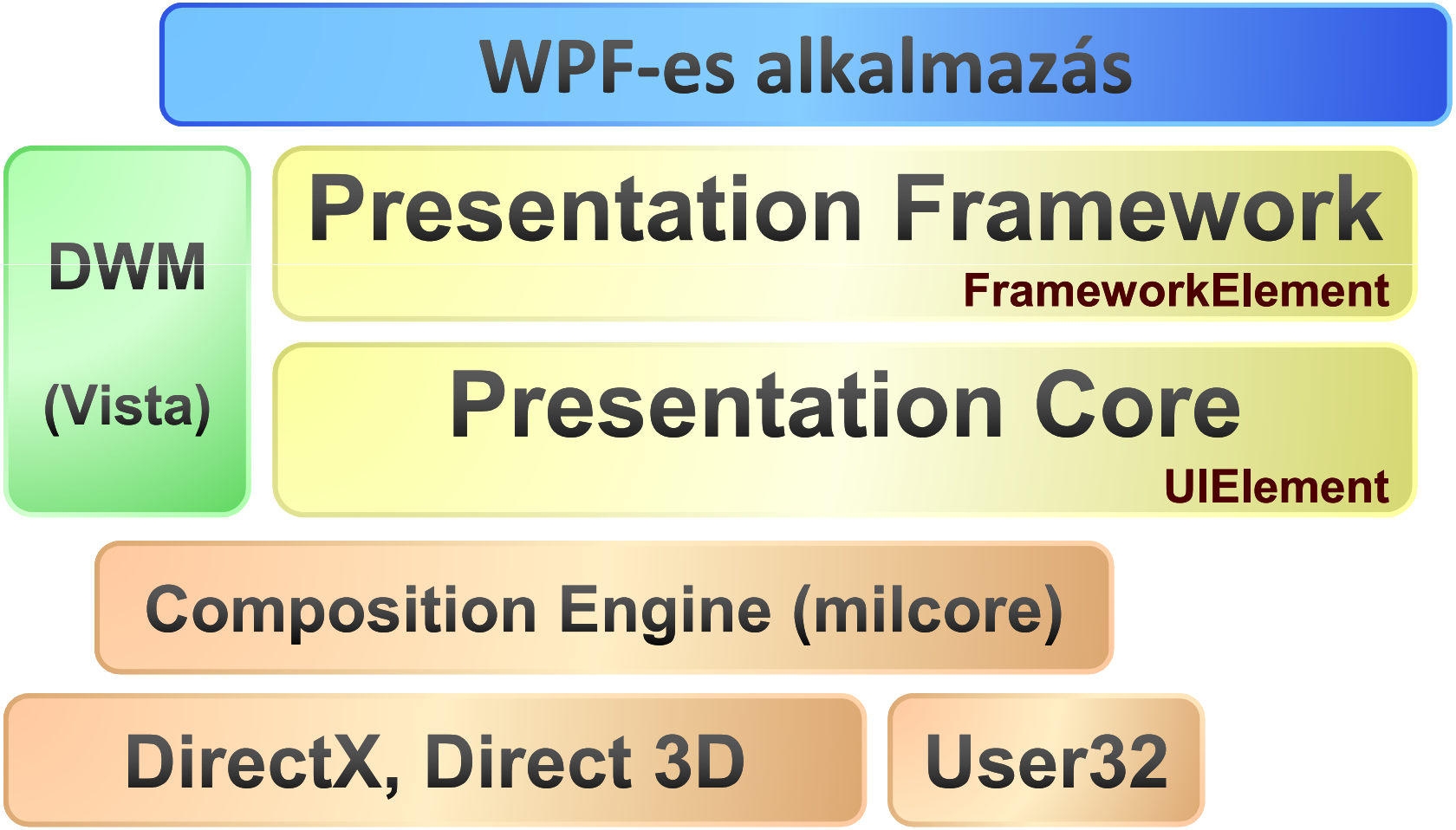
MILCORE, Channels



Vektorgrafikus megjelenítés

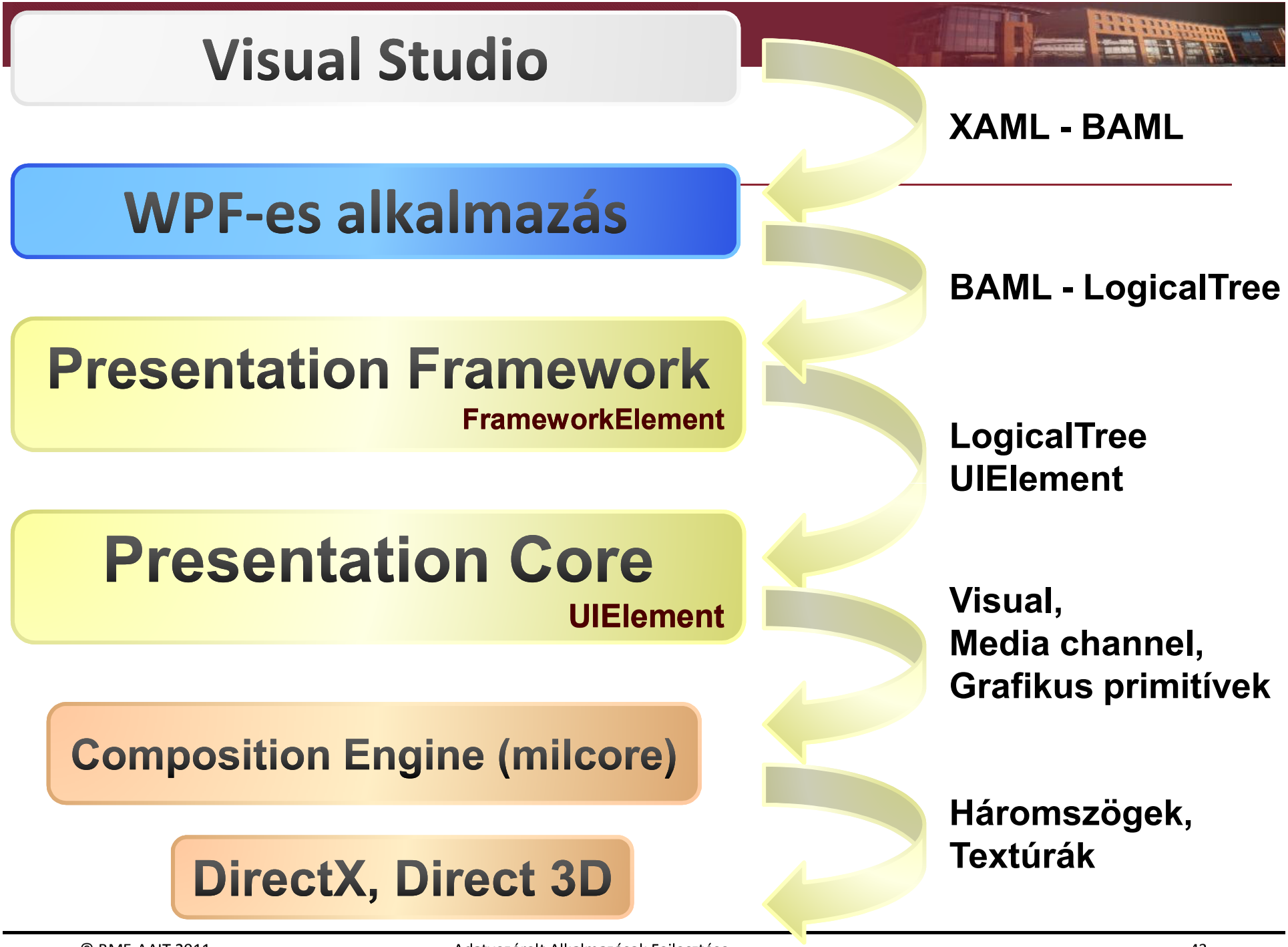
- WPF: vektorgrafikus felhasználói felület modell
  - Az alkalmazásban nincs WM_PAINT !
  - Minden vezérlő grafikus primitívekből épül fel
    - Win32/WinForms: csak a vezérlők épülnek hierarchiába.
    - *WPF esetén a vezérlőket alkotó grafikus elemek is objektumok és hierarchiába szerveződnek !*
    - Integráció: 3D objektumok, szöveg, kép, videó, effekt !
  - A grafikus elemek hierarchiába rendeződnek, fát alkotnak → megjelenés deklaratív modellje, VisualTree
  - A grafika végül egy ***DirectX***-es felületen jelenik meg

Architektúra, avagy hogyan kerül egy képpont a képernyőre ?



WPF architektúra

- PresentationCore: megjelenítés, esemény kezelés
  - MilCore: grafikus primitívek DirectX alapú renderelése
  - *Grafikus primitívek: média-, alakzat- és képkezelés*
  - *Visual: a renderelés felügyelt interfésze, hit-testing*
  - UIElement: pozicionálás, eseménykezelés, animáció
- PresentationFramework: layout, adatkötés, vezérlők
  - FrameworkElement: stílus, storyboard, triggerek
  - Layout panelek: StackPanel, Grid, DockPanel, ...
  - Vezérlők: Button, TextBox, Slider, ...


Visual Studio
WPF-es alkalmazás
Presentation Framework
FrameworkElement
Presentation Core
UIElement
Composition Engine (milcore)
DirectX, Direct 3D
XAML - BAML
BAML - LogicalTree
LogicalTree
UIElement
Visual,
Media channel,
Grafikus primitívek
Háromszögek,
Textúrák

Aero, WPF és a DWM

- Aero Glass – Vista Home Basic feletti változatok felhasználói felülete
  - Áttetsző „non-client” terület – pixel shader
  - Win + TAB : flip 3D – 3D felületek
  - Live thumbnails
- Ki a felelős?
  - A Desktop Window Manager (DWM)
    - Natív komponens

Desktop Window Manager (DWM)

- A teljes desktop egy visual tree
- Minden toplevel ablak két csomópont
  - „non-client” és a „client” terület
  - A „client” terület WPF esetén az a *megosztott* DirectX-es felület amire az alkalmazásban futó milcore.dll rajzol
  - Régi alkalmazásnál egy memória puffer
    - A hDC oda rajzol, nem közvetlenül a primary buffer-re
    - Miért? Pixel formátum konverzió, XOR, egyéb írás/olvasás műveletek lassúak lennének. GDI nem tud DirectX felületre közvetlenül írni.
    - A memória puffer időnként átkerül a videómemóriába, textúraként használja a DWM úgy, mint WPF esetén

Windows Display Driver Model (WDDM)

- Az új videó driver modell, DirectX 10.0
- WDDM nélkül nincs DWM !
  - Aero glass, thumbs, flip3d, stb.
- Főbb funkciók
  - Videómemória virtualizáció
    - Nem fogy el, csak lassú lesz ☺
  - Osztott DirectX felület támogatás (shared DX surface)
    - A WPF-es alkalmazás saját felületét a DWM közvetlenül használja textúraként megjelenítéshez
  - Megszakítható GPU műveletek
  - Újrainduló videódriver !

Agenda

- „Problems” with GDI
- The new declarative modell
- Threading (Dispatcher)
- Dependency / Attached properties
- Visuals
- Physical architecture
- DPI independent coordinates

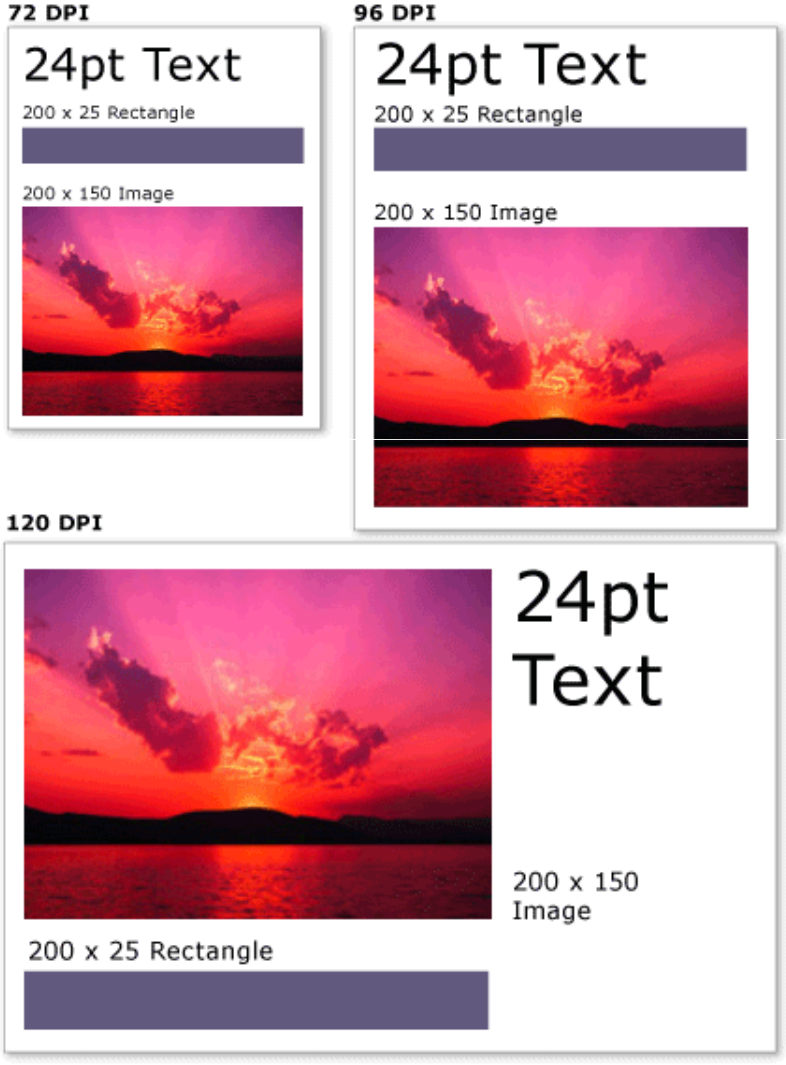
High DPI támogatás

- Ha a Windows nem 96 DPI-n fut, akkor *minden korábbi* alkalmazást úgy futtat, mintha 96 DPI lenne, és a képét felnagyítja
  - Módosítja az egér koordinátákat, stb.
  - Csak a kész DirectX-es képet tudja nagyítani …
  - … így a szöveg elmosódik ☹
  - Kikapcsolható ! ☺
    - Alkalmazás Property oldala, Compatibility fül, „Disable display scaling on high DPI settings”
    - Fejlesztők: manifeszt fájllal vagy API hívással

DPI semleges programozási modell

- Cél: ablak méretezése felbontástól függetlenül !
  - Ehhez a Windows-ban megfelelő DPI-t kell beállítani
  - Például: 17”-os LCD, 1600x1200 → 120 DPI
- WPF: *mindent* csak képpont független egységben lehet megadni (Device Independent Pixel, DIP)
  - A hüvelyk 1/96-od része
  - Az egységhez tartozó pixelek számát a Windows DPI beállítása határozza meg
    - Alapértelmezett a 96 DPI, így egy egység pont egy pixel
    - Megfelelő beállításnál a DIP valódi hosszt jelent
  - *Mindenütt*: egér koordináta, vezérlő elemek mérete, betűméret, grafika, rendszer méretek, stb.

Képernyőképek különböző DPI-vel



Lebegőpontos méretek

- Mindenütt DIP
- Mindenütt dupla pontos lebegőpontos számok
- XAML-ben beépített konverterek (in, cm, pt)
  - Például: `<TextBlock Width=„12 cm" />`
- **Vigyázat!** A double nem úgy viselkedik mint az egész szám!
  - Vigyázzunk a ==, !=, <=, >= operátorokkal !
  - A kerekítés és ábrázolás meglepetéseket okozhat !

WPF – XAML, vezérlők

Agenda

1. WPF architektúra, GDI, DirectX, DWM
   - Többszálúság, property system
   - Grafikus primitívek, rajzolás alacsony szinten
   - Média/kép kezelés
2. Vezérlő hierarchia, XAML, layout
3. Adatkötés, sablonok, erőforrások, stílusok, triggerek, eseménykezelés
4. MVVM, 3D, dokumentum kezelés

Agenda

- XAML
  - XAML részletesen
- Vezérlők hierarchiája
- Content controls
- HeaderedContentControls
- Items controls
- HeaderedItemsControls
- Range controls
- Text and link controls

XAML alapok

```
<Button
   xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
     Content="OK" Click="button_Click"/>
```

```
using System.Windows.Controls;
using System.Windows;
Button b = new Button();
b.Click += new RoutedEventHandler(button_Click);
b.Content = "OK";
```

XAML

- XAML: XML alapú, deklartív objektum példányosító nyelv
  - A WPF mellett a Workflow Foundation is ezt használja
  - Nincs mágia, nincs kódból el nem érhető funkció
- WPF esetén a programozók és designerek, fejlesztőeszközök és designer eszközök közös nyelve
- DE: a WPF vektorgrafikus modellje NEM a XAML-ből fakad!
  - A XAML „csak” egy eszköz, nyelv a felhasználói felület deklaratív leírására
  - Kicsit hasonlít az XHTML-re, de sokkal kifejezőbb, erőteljesebb
  - Elemek hierarchiája, layout, grafika, média, animáció, adatkötés, triggerek, stílusok, … mind leírhatók benne!

XAML alapok 1.

- A tegek osztályokat példányosítanak
  - Nem WPF specifikus !
- Az attribútumok tulajdonságokat állítanak be
- Névterek, szerelvények
  - Az alapértelmezett névtér a WPF vezérlők névtere
  - Az ‘x’ névtér az általános XAML motor
    - például az x:Name – a generált változó neve
  - Egyéb szerelvények és névtereik importálhatók
    - Bármelyik szerelvényhez attribútummal névtér rendelhető

XAML alapok 2.

- Egy teg tartalma (gyerek elemek) az „alapértelmezett” tulajdonság értékét határozza meg
- Az attribútum értelmezéséhez lehet írni konvertert

```
<TextBlock Width=„12 cm" />
```

- Az attribútum értéke lehet összetett objektum is
  - Pl: LinearGradientBrush
  - (PropertyElementSyntax)
- Eseménykezelők

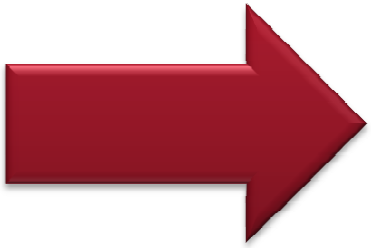
Cél:

XAML-t NE KELLJEN KÉZZEL ÍRNI !

2009: Vagy mégis???

XAML alapok

```
<Button>
        <Button.Content>
           OK
        </Button.Content>
</Button>
```



```
<Button>
        OK
</Button>
```

```
<Button>
   <Button.Content>
      <Rectangle Height="40"
         Width="40" Fill="Black"/>
    </Button.Content>
</Button>
```

```
<Button>
    <Rectangle Weight="40“
        Width="40" Fill="Black"/>
</Button>
```

A XAML feldolgozása

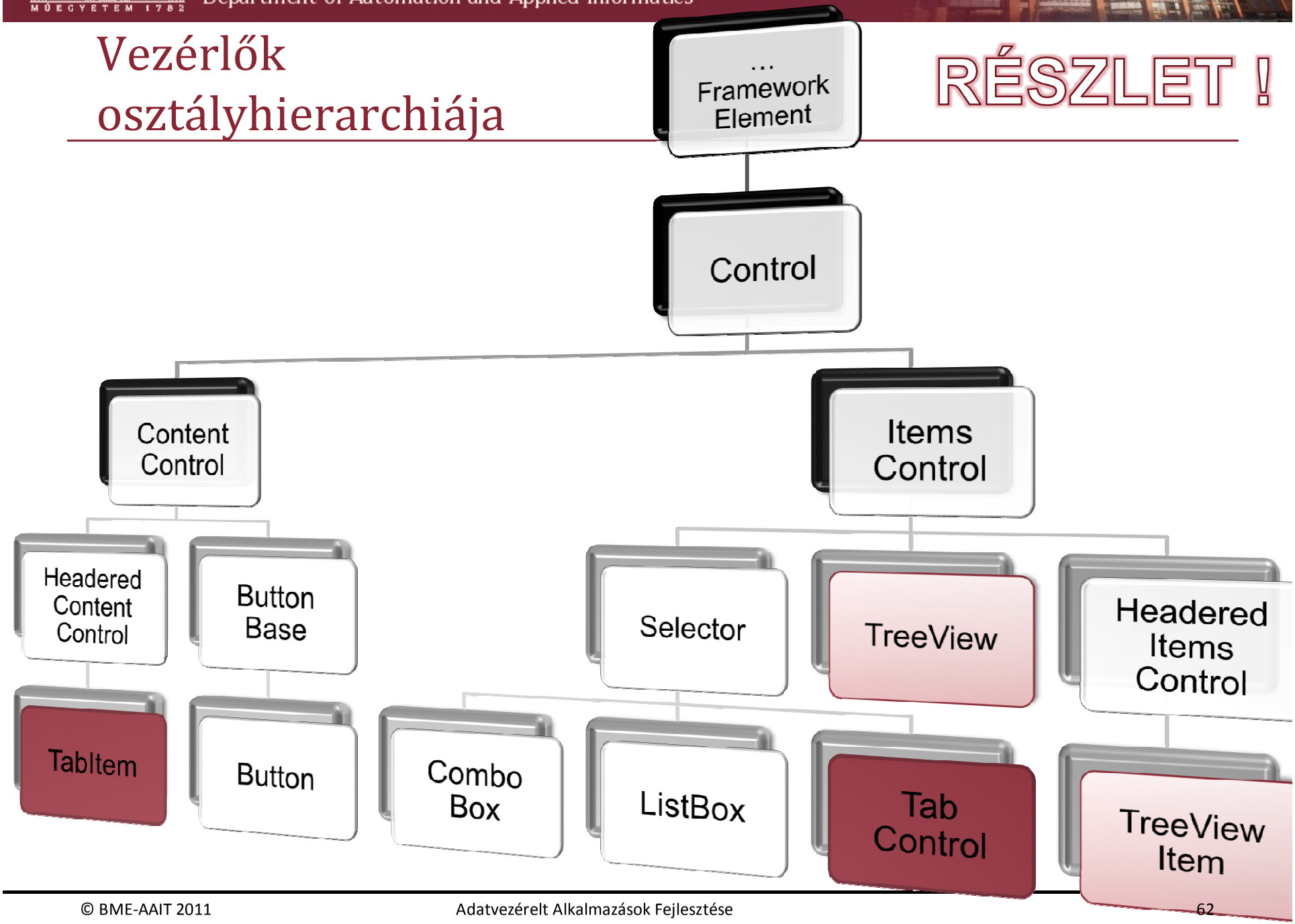
- A XAML-t a fordító tömör bináris formává alakítja: BAML
  - Előtte kétfázisú fordítás idejű ellenőrzés
  - Futásidőben már nincs XML értelmezés
- A BAML erőforrásként kerül tárolásra
- A WPF keretrendszer gondoskodik a BAML értelmezéséről és annak alapján az objektumfa és a kötések létrehozásáról
  - A XAML-ből nem generál közvetlenül kódot

Agenda

- XAML
  - XAML részletesen
- Vezérlők hierarchiája
- Content controls
- HeaderedContentControls
- Items controls
- HeaderedItemsControls
- Range controls
- Text and link controls

Vezérlők osztályhierarchiája

RÉSZLET !



A megjelenés testreszabása

- Minden vezérlő **tartalma** testreszabható
- Többféle tartalom a vezérlő típusától függően
  - Egyszerű, például gomb (: ContentControl)
  - Fejléces, például tabfül (: HeaderedContentControl)
  - Listás, például ListBox (: Selector : ItemsControl)
  - Hierarchikus, pl. TreeView (: HeaderedItemsControl)
- A tartalmat a ContentPresenter jeleníti meg
  - Alapértelmezett tartalom: ToString() – szöveg
  - De testreszabható !

Agenda

- XAML
  - XAML részletesen
- Vezérlők hierarchiája
- Content controls
- HeaderedContentControls
- Items controls
- HeaderedItemsControls
- Range controls
- Text and link controls

ContentControl : Control

- Content tulajdonság (object): a tartalom
  1. ContentTemplate adott: használja a sablont
  2. UIElement: megjeleníti azt az objektum fát
  3. Ellenőrzi a típus alapú DataTemplate-eket
  4. ToString-et használ
     - ContentStringFormat: formázó sztring
- ContentTemplate (DataTemplate)
  - A tartalom megjelenítését lehet vele testreszabni
- ContentTemplateSelector: sablonválasztó logika

ButtonBase : ContentControl

- Kezeli az egéreseményeket
- Tulajdonságok
  - IsPressed
  - ClickMode: Release, Press, Hover
- Integrálódik a Command mintával
  - Command, CommandParameter, CommandTarget

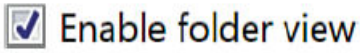
Button, RepeatButton : ButtonBase

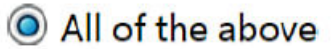
- Button

  - IsDefault, IsCancel
  - Handles OnClick
- RepeatButton
  - Delay property
  - Interval property
  - Nyomva tartva többször lövi el a Click eseményt
  - System.Windows.Controls.Primitives névtérben van

ToggleButton, CheckBox

- ToggleButton : ButtonBase
  - IsChecked: lehet null is
  - IsThreeStat
- CheckBox : ToggleButton
  - Saját megjelenése van
- RadioButton : ToggleButton
  - Saját megjelenése van
  - GroupName-el lehet csoportba foglalni őket és csak egyet lehet kiválasztani





Label : ContentControl

- Szöveget jelenít meg
- Tartget porperty
  - A focust beállítja a hivatkozott vezérlőre
  - A gyorsbillentyűt az aláhúzás karakter jelzi
    - AccessKeys : mnemonics
- Máshol is használhatók a gyorsbillentyűk
  - AccessText-et kell a Content helyére tenni
  - Állítható a megjelenése

UserControl : ContentControl

- Más vezérlőket fog össze és jelenít meg
  - Nincs template támogatás: korlátozott külső testreszabhatóság
- Nincs további saját tulajdonsága vagy viselkedése

Agenda

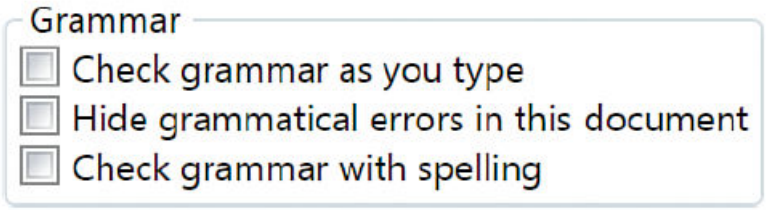
- XAML
  - XAML részletesen
- Vezérlők hierarchiája
- Content controls
- HeaderedContentControls
- Items controls
- HeaderedItemsControls
- Range controls
- Text and link controls

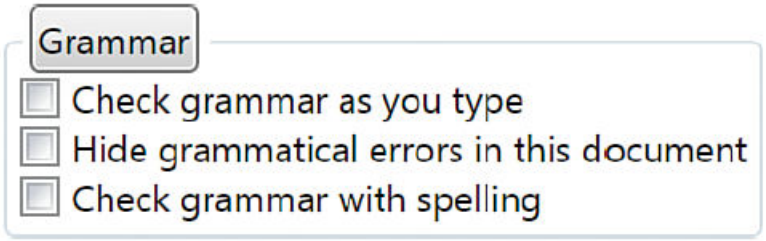
HeaderedContentControl : CC

- A tartalom mellett van egy fejléc is (Header)
  - Header és HeaderFormatString tulajdonságok
  - HeaderTemplate, HeaderTemplateSelector

GroupBox : HCC

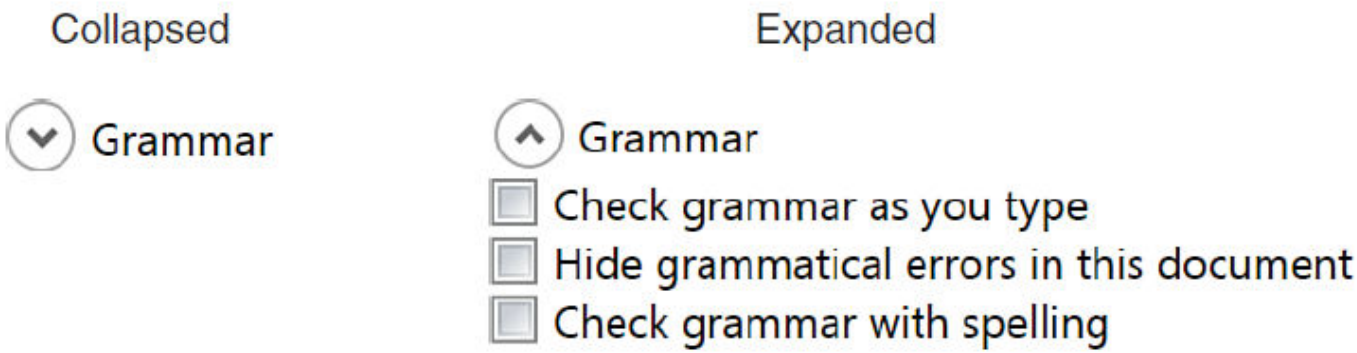
- Rajzol egy keretet a tartalom köré





Expander : HCC

- Becsukható / kinyitható tartalom
  - A fejléc mindig látszik
- IsExpanded tulajdonság
- ExpandDirection: négy irány
- Expanded / Collapsed események



Agenda

- XAML
  - XAML részletesen
- Vezérlők hierarchiája
- Content controls
- HeaderedContentControls
- Items controls
- HeaderedItemsControls
- Range controls
- Text and link controls

ItemsControl : Control

- Items (ItemCollection típus)
  - Ha ItemsSource megvan adva, akkor read-only
  - Egyébként itt lehet beállítani a tartalmat
- ItemsSource írható property
  - Ha null, akkor Items-et használ
- HasItem, IsGrouping
  - Items tulajdonság (ItemCollection (:CollectionView)) képes csoportosítani, rendezni, stb.
- DisplayMemberPath
  - Az elemek megjelenítését egy property-hez kötjük
  - Ha nincs megadva és nincs ItemTemplate, akkor a ToString-et jeleníti meg (type)

Container osztályok

- Minden megjelenített listaelemhez van egy Container osztály
  - Pl ListBoxItem, ami egy ContentControl
- Az ItemContainerGenerator-on keresztül lehet elérni a container itemeket
  - Explicit is létre lehet hozni (a forráshoz hozzáadni), akkor a logical tree része lesz, enumerálni lehet
- A container elemek felelősek a kiválasztott elem háttérszínéért, stb.
- Az ItemsPanellel változtatható az elrendező panel

Selector : ItemsControl

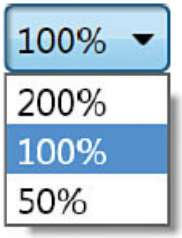
- Kiválasztható elemek
- SelectedIndex: 0 alapú index
- SelectedItem
  - A kiválasztott objektum
- SelectedValue
  - SelectedValuePath által hivatkozott tulajdonsága a kiválasztott objektumnak
- Attached property:
  - IsSelected
  - IsSelectedActive -> fókusz rajta van-e a kiválasztott elemen
  - SelectionChanged esemény

ListBox : Selector

- SelectionMode
  - Single/Multiple/Extended (Shift-et vagy a Ctrl-t le kell nyomni ha ki akarunk többet jelölni))
- ScrollViewer . CanContentScroll

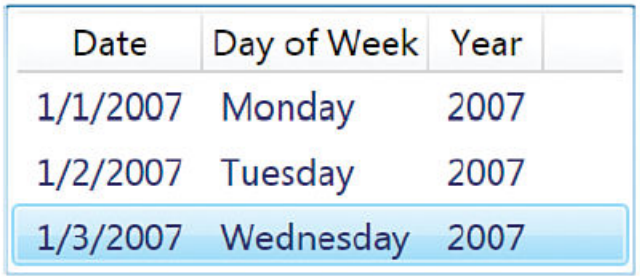
ComboBox : Selector

- DropDownOpened, DropDownClosed események
- IsDropDownOpen (read-only porp)
- IsEditable
- IsReadOnly
- TextSearch.TextPath ComboBox-ban
- TextSearch.Text -> ComboBoxItem-ben



ListView : ListBox

- View property
  - Például Grid



Agenda

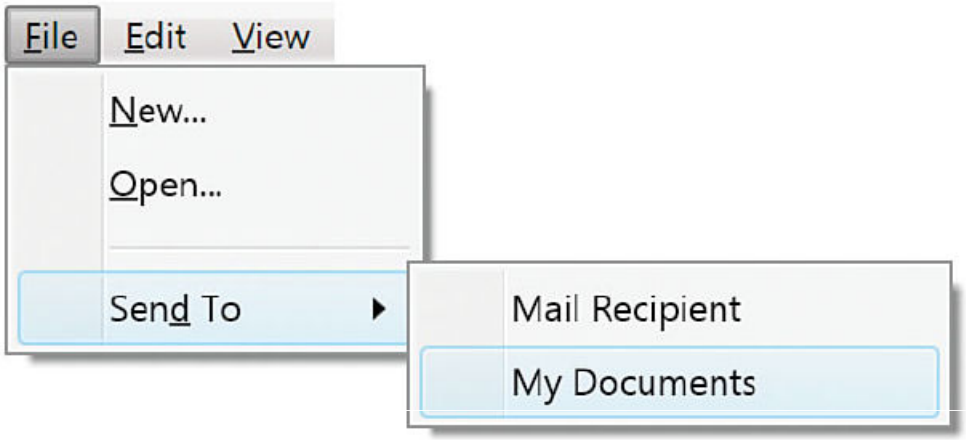
- XAML
  - XAML részletesen
- Vezérlők hierarchiája
- Content controls
- HeaderedContentControls
- Items controls
- HeaderedItemsControls
- Range controls
- Text and link controls

HeaderedItemsControl : ItemsControl

- Header tulajdonság a fejléchez

Menu : HIC

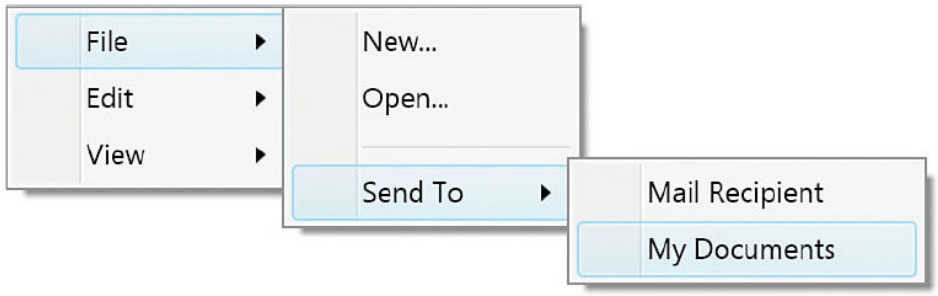
- Tulajdonságok
  - Icon
  - InputGestureText
  - IsCheckable
  - Separator



- IsMainMenu ->ha true, akkor Alt-ra vagy F10-re fokuszál

ContextMenu : HIC

- Ugyanaz jellemző rá, mint a menu-re.
- Open/Closed események
- ContextMenuService.ShowOnDisabled



Agenda

- XAML
  - XAML részletesen
- Vezérlők hierarchiája
- Content controls
- HeaderedContentControls
- Items controls
- HeaderedItemsControls
- Range controls
- Text and link controls

RangeBase

- Max, min
- SmallChange, LargeChange
- Value lekérdezés, validálás
- ValueChanged esemény

Range Controls

IsIndeterminate, ha true, akkor folyton megy a zöld csík

Range Controls

Slider:

- TickPlacement (TopLeft,ButtomRight,None)
  - A vonalkák hol legyenek
- IsSelectionRangeEnabled
- SelectionStart, SelectionEnd

```
<Slider Minimum="0" Maximum="10" Value="5"
        TickPlacement="BottomRight"
        IsSelectionRangeEnabled="True"
        SelectionEnd="8” SelectionStart="2"/>
```

Agenda

- XAML
  - XAML részletesen
- Vezérlők hierarchiája
- Content controls
- HeaderedContentControls
- Items controls
- HeaderedItemsControls
- Range controls
- Text and link controls

Text and Ink Controls

TextBox:

•Copy,Paste,Cut,Undo,Redo és spell checking alapban benne van
•SpellCheck.IsEnabled
•TextWrapping (Wrap/WrapWithOverflow) (Wrap esetén ha egy szó nem fér ki a sorban, akkor törik, WrapWithOverflow esetén nem engedi szétszedni a szót)

RichTextBox:

•FlowDocument-ben tárolja a szöveget
•Sokkal többet tud, mint a TextBox
•Lesz rá példa később!

PasswordBox:

•Password-ben tárolja a szöveget!
•PasswordChanged esemény

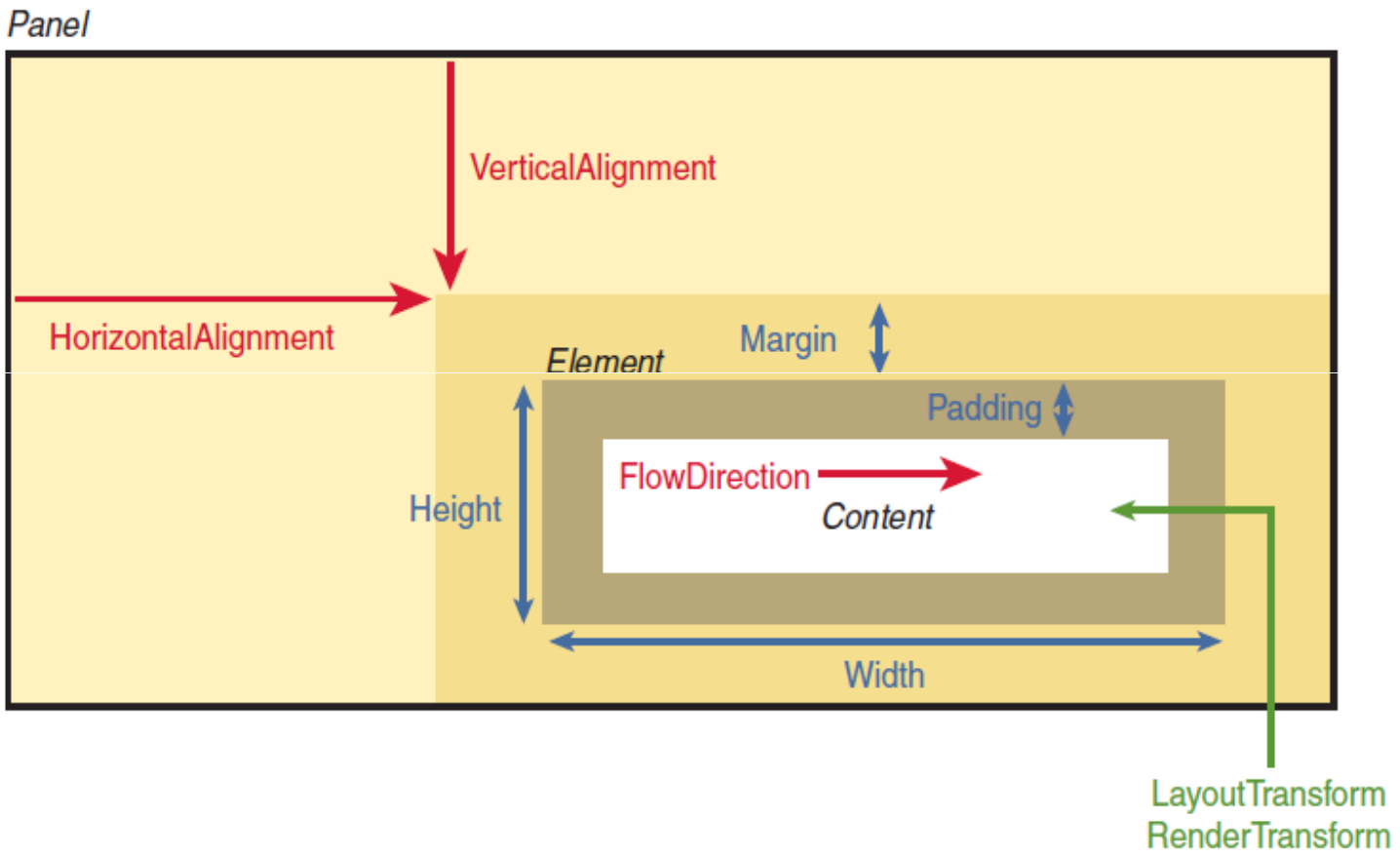
Text and Ink Controls

- EditingMode:
  - Ink
  - InkAndGesture
  - GestureOnly
  - EraseByStroke
  - EraseByPoint
  - Select
  - None
- 15 esemény
- DrawingAttributes-al testre szabhatóak a Stroke-ok
- Nagyon gazdag vezérlő

Agenda

- XAML
  - XAML részletesen
- Vezérlők hierarchiája
  - Content controls
  - HeaderedContentControls
  - Items controls
  - HeaderedItemsControls
  - Range controls
  - Text and link controls
- Elhelyezés
  - Méretezés, pozicionálás
  - Transzformáció
  - Layout engine

Méretezés


Panel
VerticalAlignment
HorizontalAlignment
Element
Margin
Padding
FlowDirection
Content
Height
Width
LayoutTransform
RenderTransform

Felhasználói felület, layout

- Gyerek elemeken lévő általános módosítók
  - Horizontal/Vertical Alignment
    - A szülőn belül hova kerüljön
    - Left/Top, Right/Bottom, Center, Stretch
  - Margók: Left, Right, Top, Bottom
    - Az elem széleinek távolsága a szülő vagy egyéb vezérlők megfelelő széleitől
    - Az ActualHeight/Width-en kívül értelmezettek
  - Padding: csak néhány elemen van (pl border)
    - A benne lévő elemek távolsága a panel szélétől
  - Hasonlóak a horgonyzáshoz (anchor) csak többet tudnak

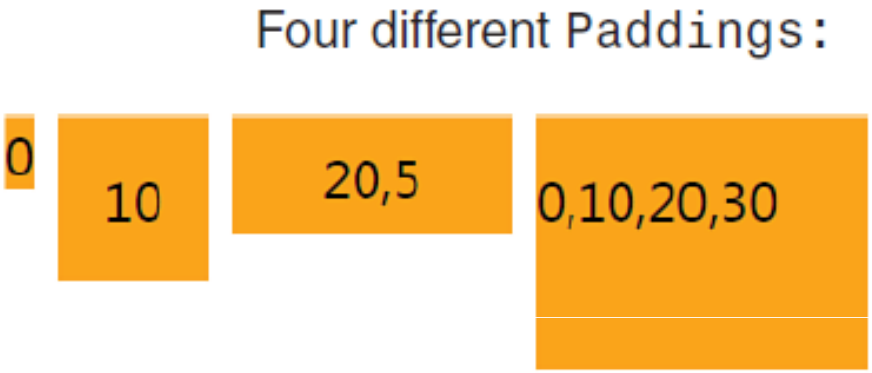
Méretezés – Width, Height

- Tulajdonságok:
  - Width, Height (alapértelmezett Auto = Double.NaN)
  - MinWidth, MinHeight (alapértelmezett 0)
  - MaxWidth, MaxHeight (alapértelmezett Double.PositiveInfinity)

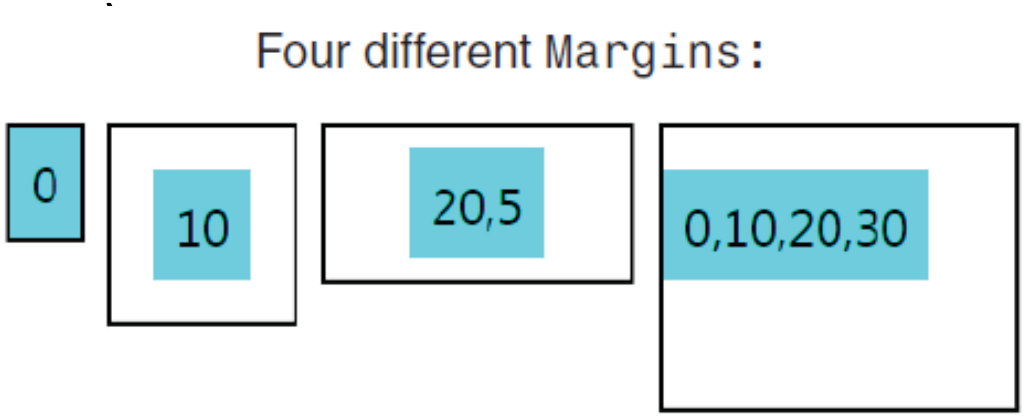
- Számold tulajdonságok:
  - DesiredSize (UIElement) -> layout készítéskor számolja
  - RenderSize (UIElement) -> renderelés után számolja
  - ActualHeight, ActualWidth -> renderelés után

Méretezés – Margin, Padding

- Padding:

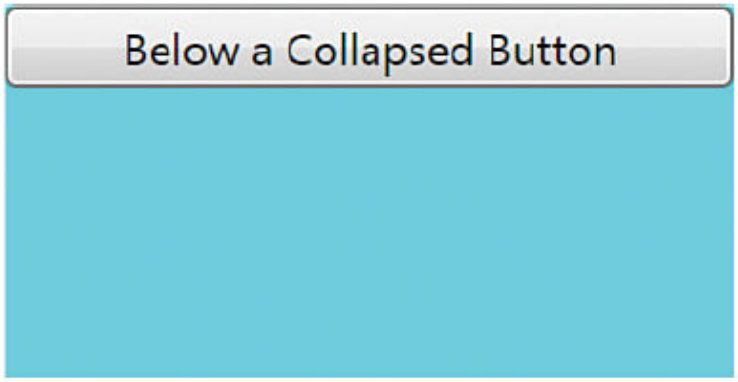


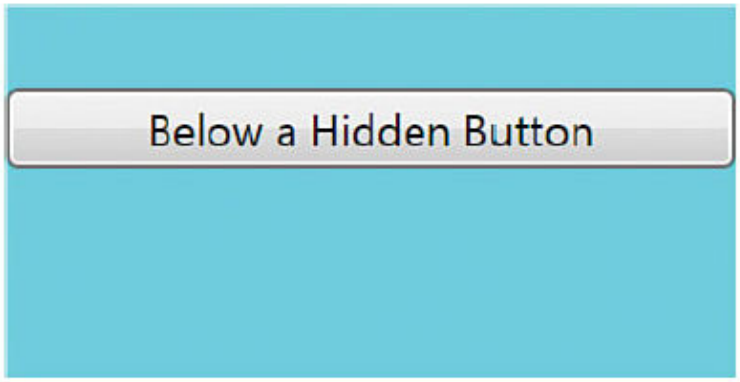
- Margin (Thick



Méretezés – UIElement . Visibility

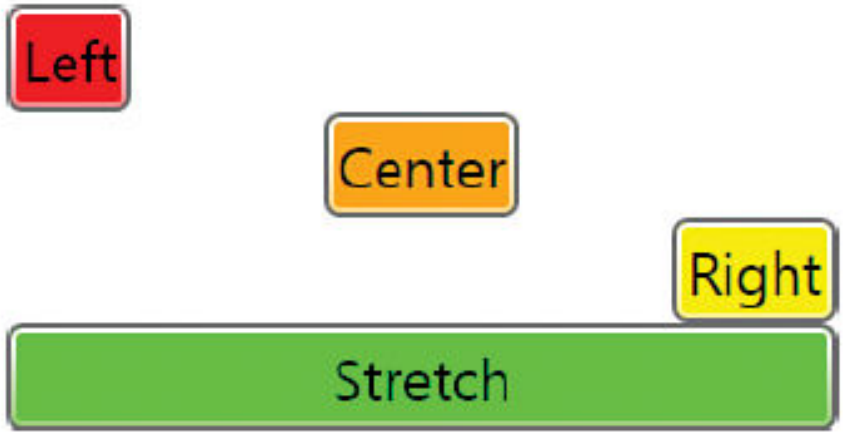
- System.Windows.Visibility:
  - Visible
  - Collapsed
  - Hidden





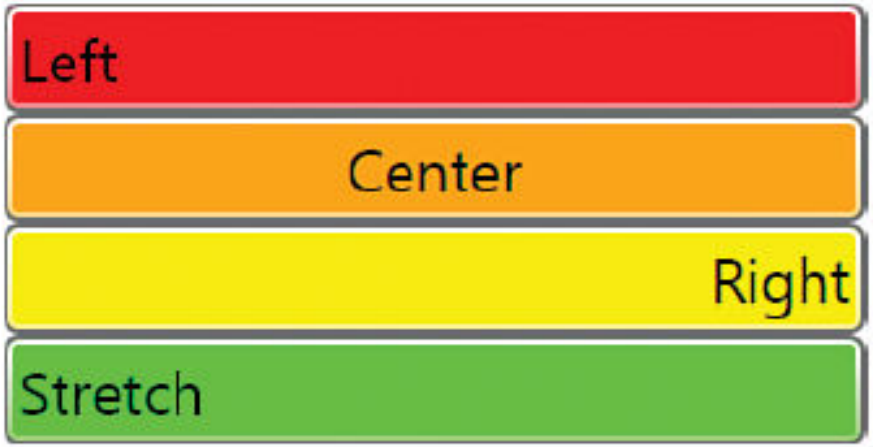
Pozícionálás

- Alignment:
  - HorizontalAlignment: Left, Center, Right, Stretch
  - VerticalAlignment: Top, Center, Buttom, Stretch



Pozícionálás

- Tartalom igazítás a Control osztályban
  - HorizontalAlignment / VerticalAlignment propertyk
  - HorizontalContentAlignment: Left, Center, Right, Stretch
  - VerticalContentAlignment: Top, Center, Buttom, Stretch



Pozícionálás

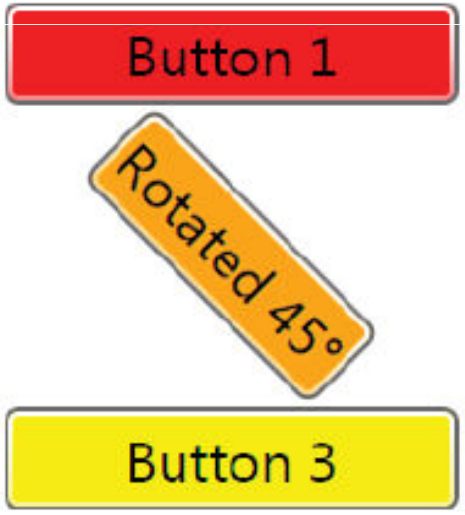
- FlowDirection:
  - RightToLeft
  - LeftToRight

Agenda

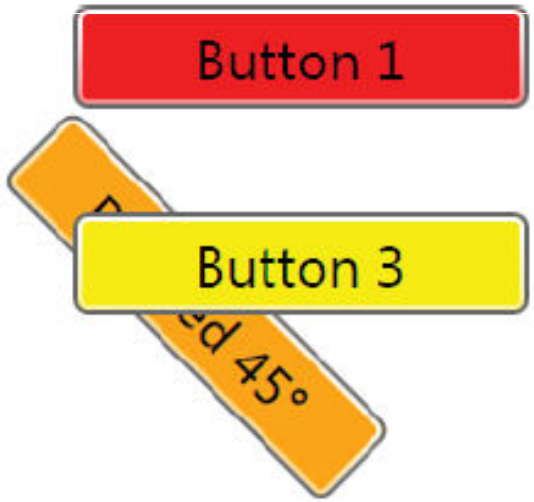
- XAML
  - XAML részletesen
- Vezérlők hierarchiája
  - Content controls
  - HeaderedContentControls
  - Items controls
  - HeaderedItemsControls
  - Range controls
  - Text and link controls
- Elhelyezés
  - Méretezés, pozicionálás
  - Transzformáció
  - Layout engine

Transformok

LayoutTransform

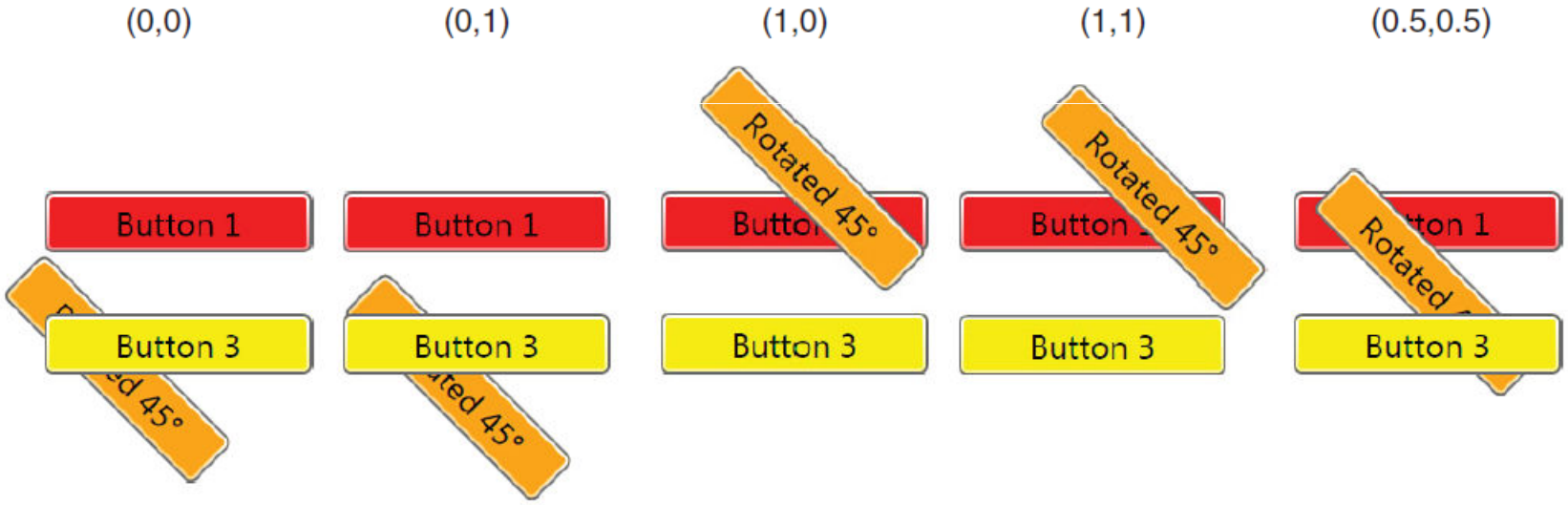


RenderTransform



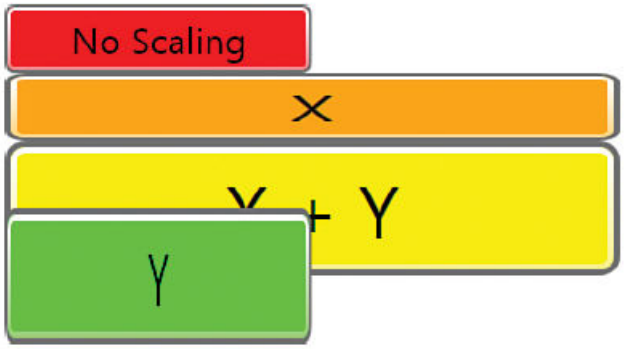
Transformok

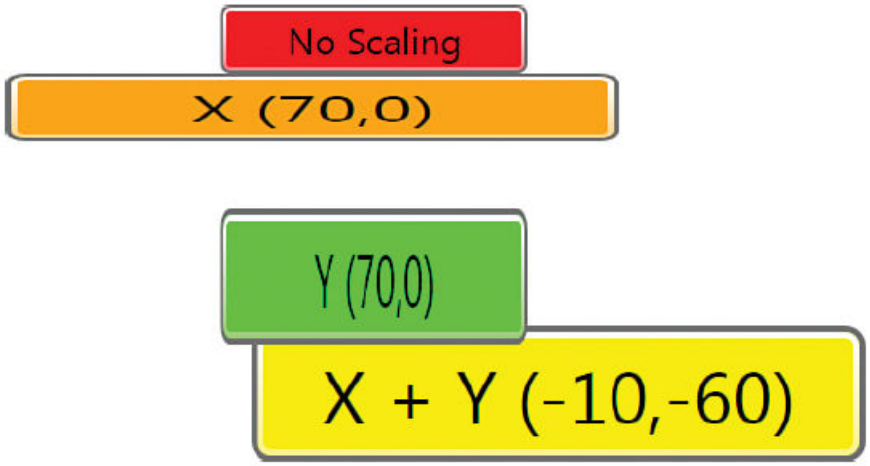
- RenderTransformOrigin



ScaleTransform

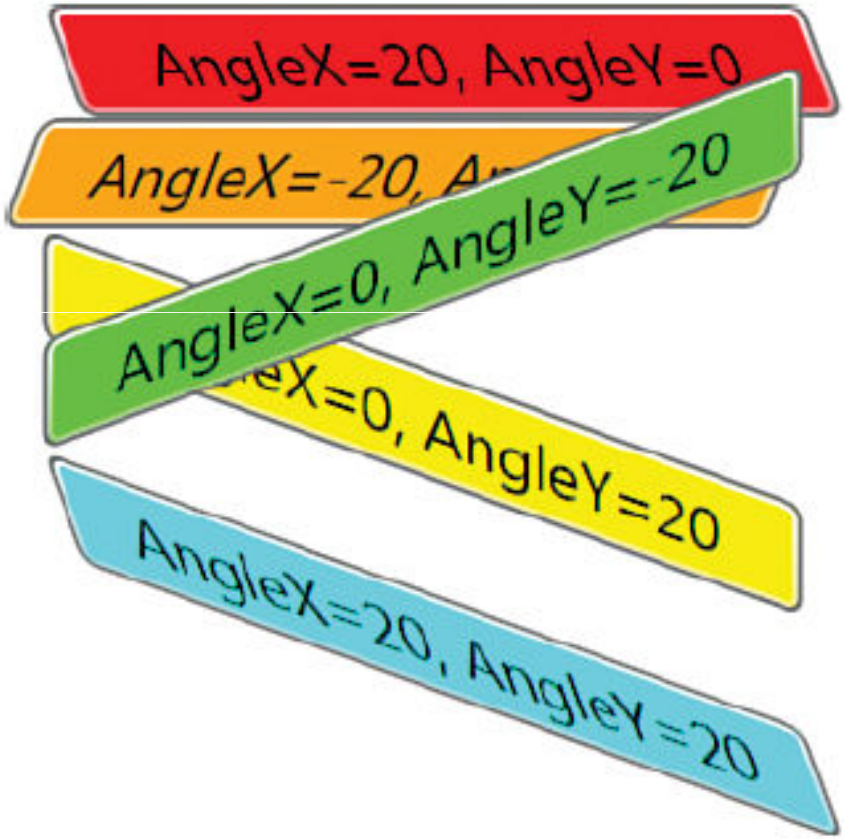
- ScaleX
- ScaleY
- CenterX
- CenterY
- Padding skálázódik, de a Margin nem!





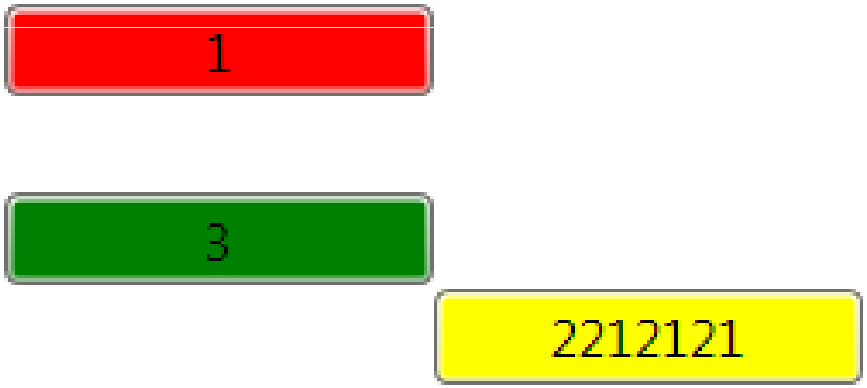
SkewTransform

- AngleX
- AngleY
- CenterX
- CenterY



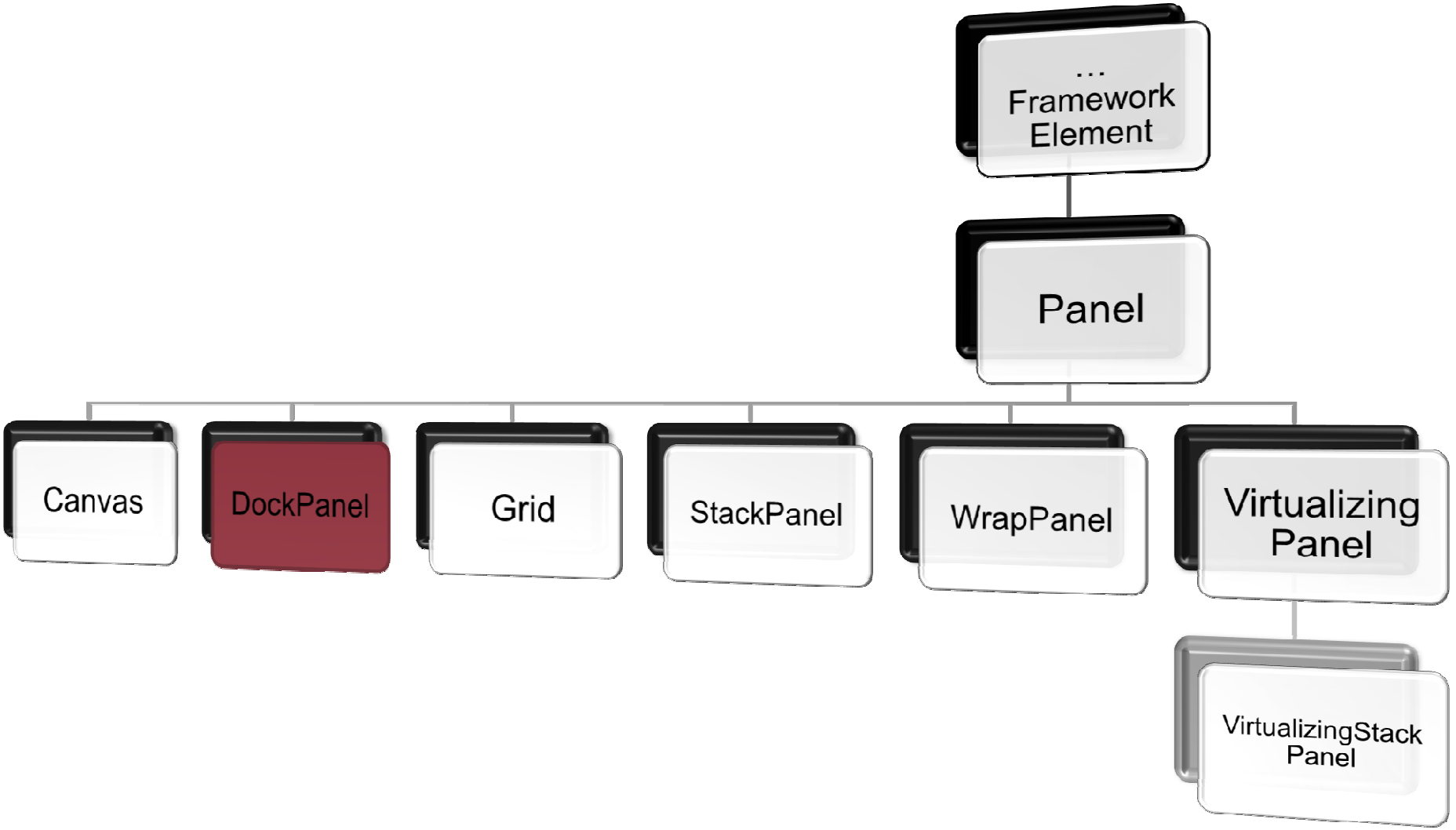
TranslateTransform

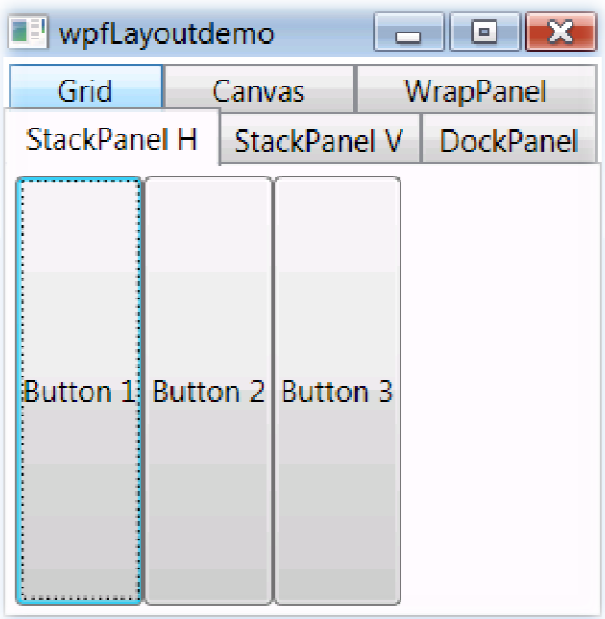
- X
- Y
- Csak a RenderTransform van rá hatással!!!

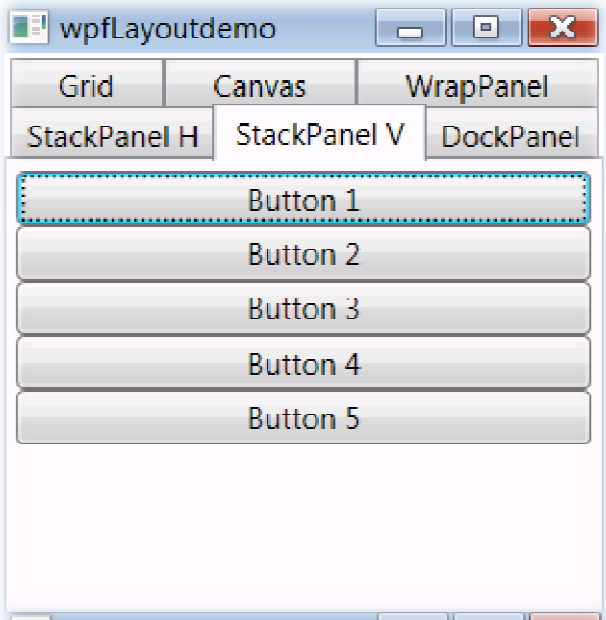


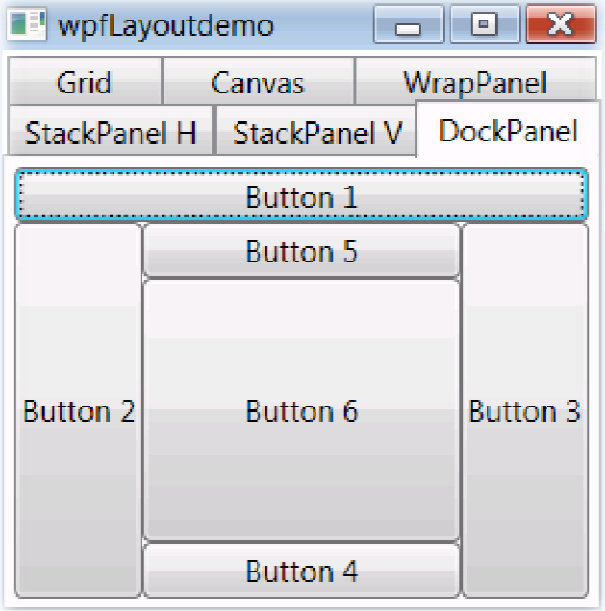
Agenda

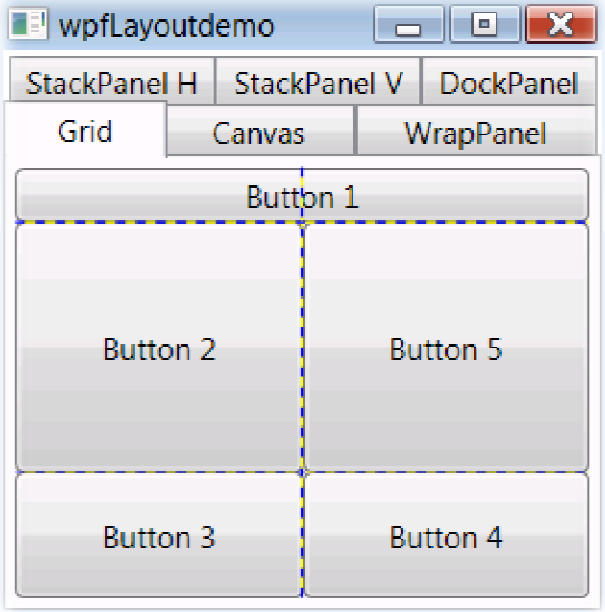
- XAML
  - XAML részletesen
- Vezérlők hierarchiája
  - Content controls
  - HeaderedContentControls
  - Items controls
  - HeaderedItemsControls
  - Range controls
  - Text and link controls
- Elhelyezés
  - Méretezés, pozicionálás
  - Transzformáció
  - Layout engine

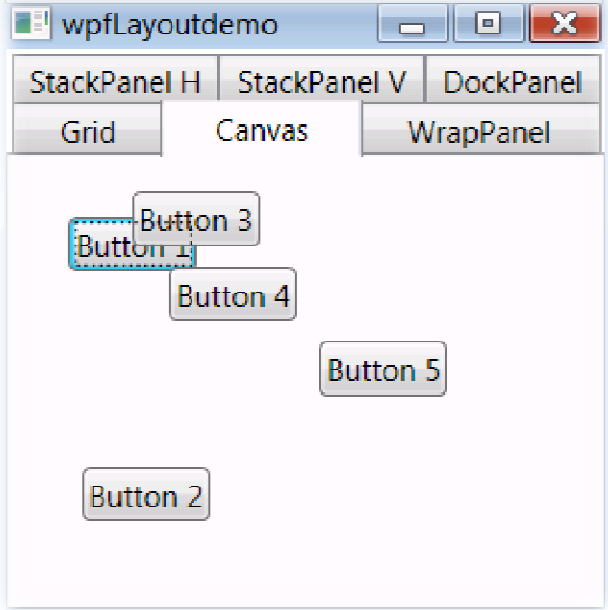

...
Framework Element
Panel
Canvas
DockPanel
Grid
StackPanel
WrapPanel
Virtualizing Panel
VirtualizingStack Panel

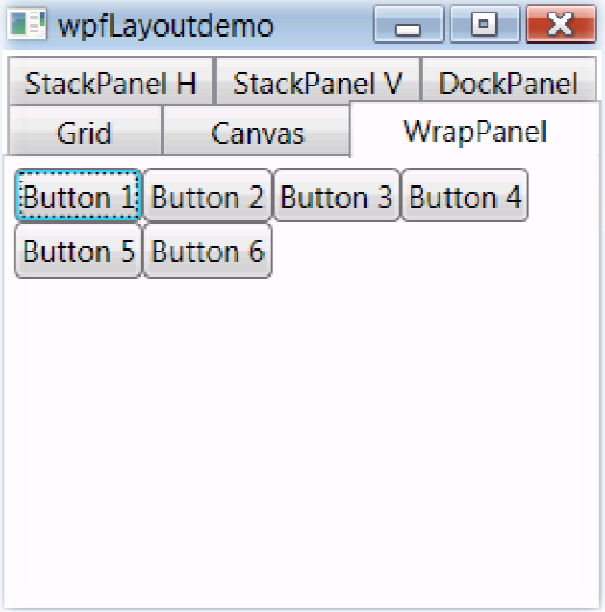

wpfLayoutdemo
Grid
Canvas
WrapPanel
StackPanel H
StackPanel V
DockPanel
Button 1
Button 2
Button 3


wpfLayoutdemo
Grid
Canvas
WrapPanel
StackPanel H
StackPanel V
DockPanel
Button 1
Button 2
Button 3
Button 4
Button 5


wpfLayoutdemo
Grid
Canvas
WrapPanel
StackPanel H
StackPanel V
DockPanel
Button 1
Button 2
Button 5
Button 6
Button 3
Button 4


wpfLayoutdemo
StackPanel H
StackPanel V
DockPanel
Grid
Canvas
WrapPanel
Button 1
Button 2
Button 5
Button 3
Button 4

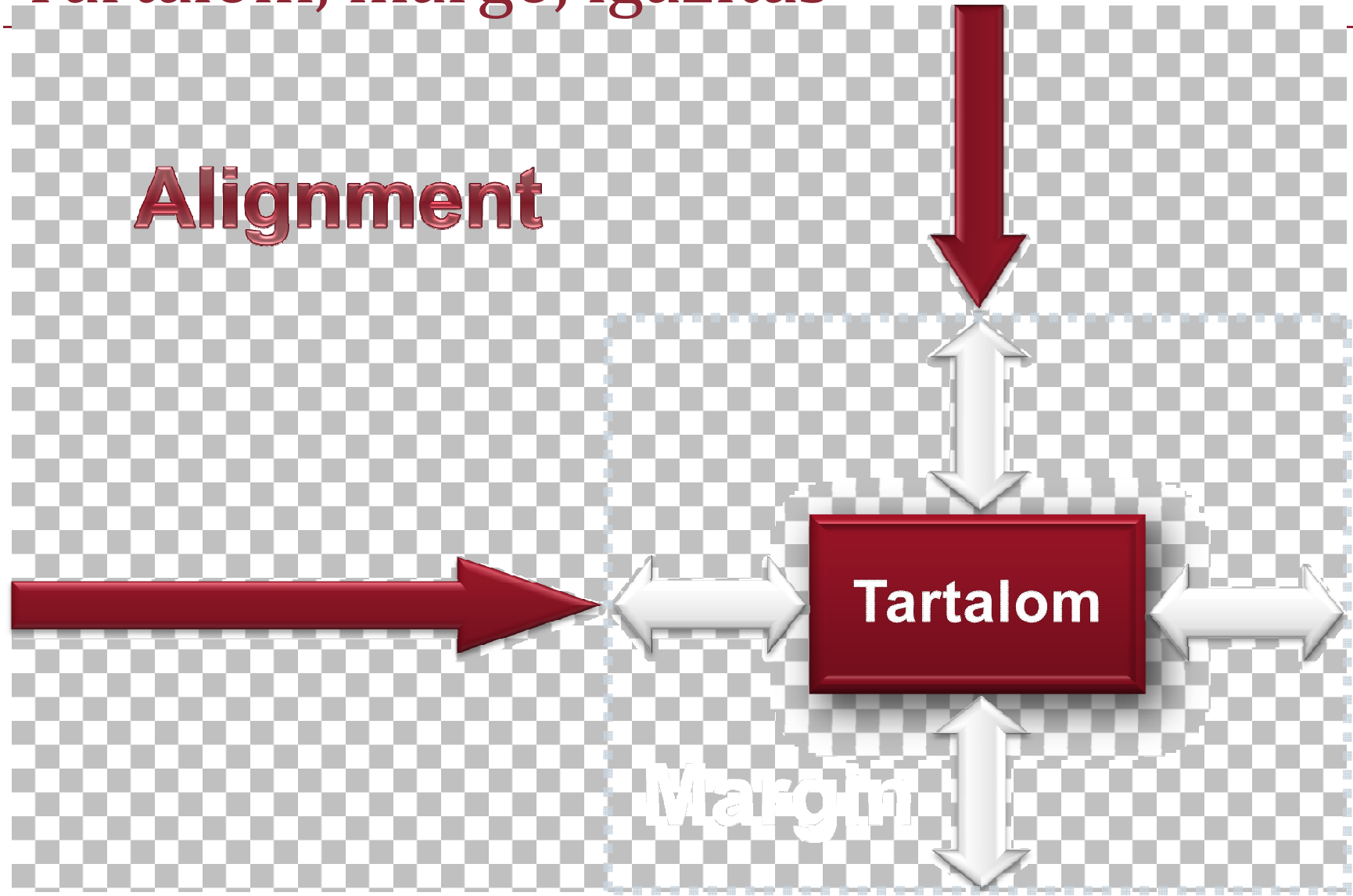

wpfLayoutdemo
StackPanel H
StackPanel V
DockPanel
Grid
Canvas
WrapPanel
Button 3
Button 4
Button 5
Button 2


wpfLayoutdemo
StackPanel H
StackPanel V
DockPanel
Grid
Canvas
WrapPanel
Button 1
Button 2
Button 3
Button 4
Button 5
Button 6

Layout System – Framework szint

- A Panel gyerekeinek elrendezése
- Az elemek poziciójának, méretének változása új layout számítást indít – lassú lehet
  - Aszinkron módon: a méret változás nem indukál szinkron layout számítást (2 queue: measure, arrange)
- Két fázisú rekurzív layout algoritmus
  - Measure – felmérés
  - Arrange – elrendezés
- Framework: Minden UIElement egy befoglaló téglalapban van

Tartalom, margó, igazítás



Panel

- Gyűjtő vezérlő
  - Children tulajdonság
  - ZIndex csatolt tulajdonság

Canvas

- Fix X és Y koordináták
  - Attached properties: Left, Right, Top, Bottom
  - ClipToBounds = false by default
  - Átrajzolhat más vezérlőket a canvason kívül

DockPanel

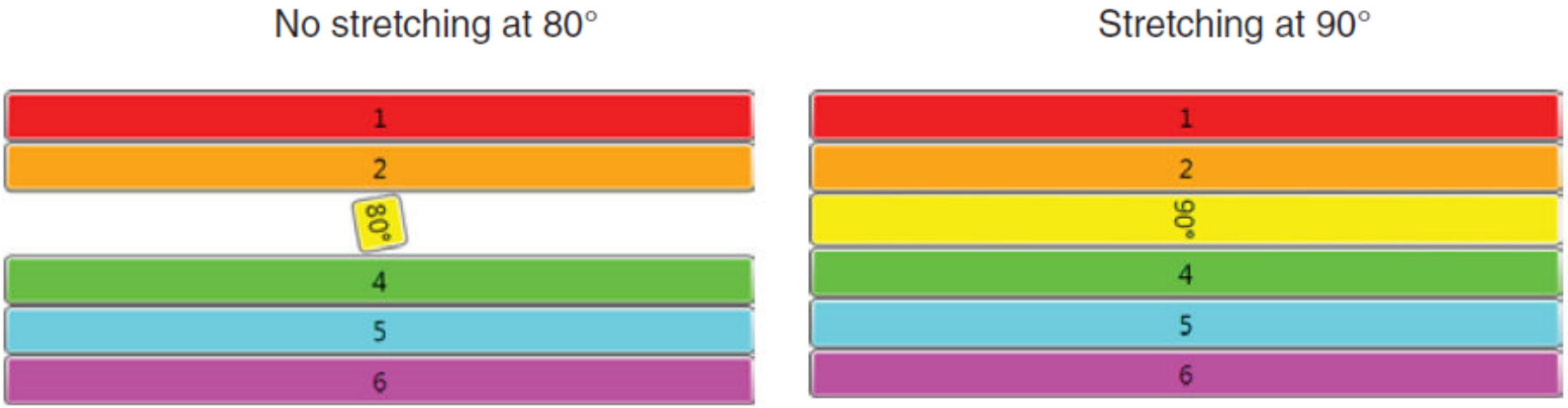
- A vezérlő szélei mentén helyezi el a gyerek elemeket
  - Az elemek sorrendjében
- Dock attached property: Top, Bottom, Left, Right
- LastChildFill property

StackPanel

- Horizontális és Vertikális elrendezés
  - Orientation property

StackPanel

- Layout Transform esetén, ha 90 fokkal forgatjuk el, akkor kitölti a teret!

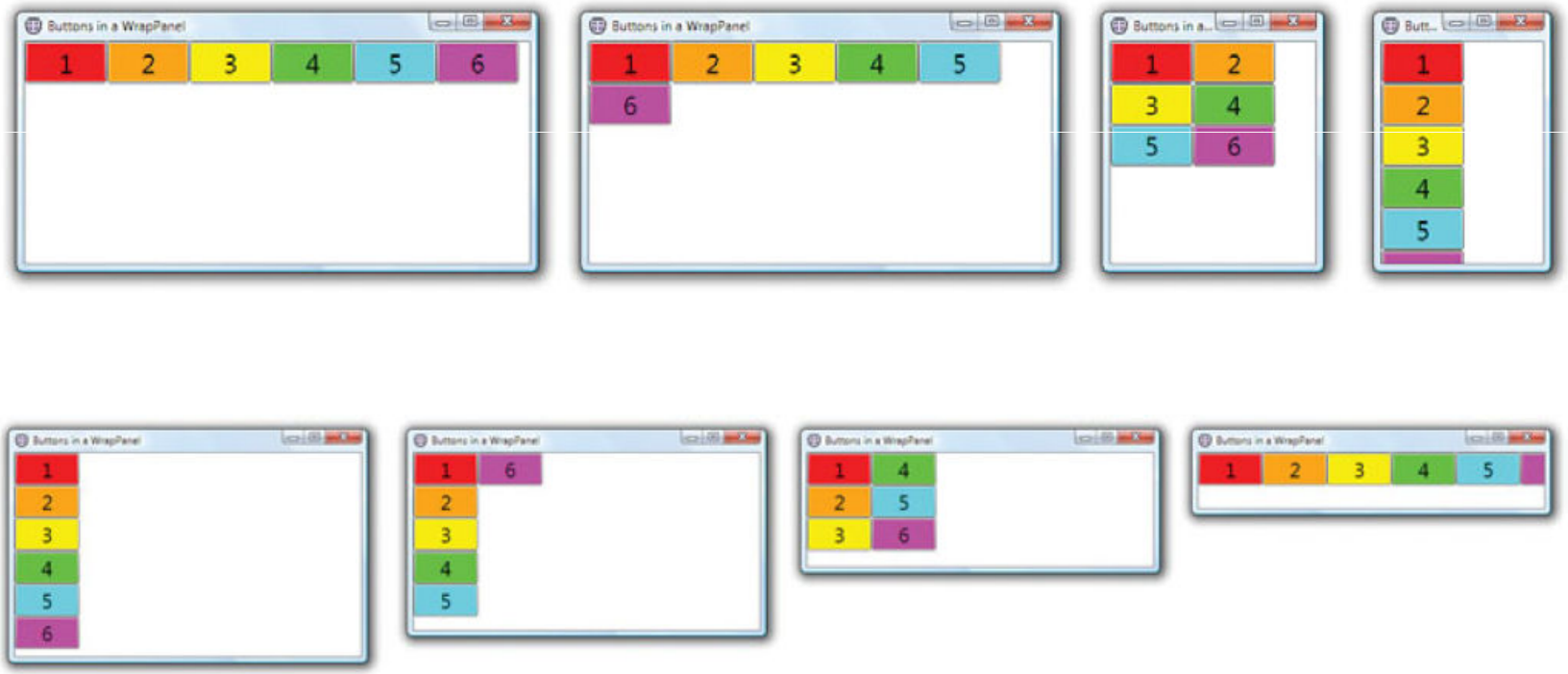


WrapPanel

- Szekvenciális elrendezés (FlowLayout)

WrapPanel

- Orientation
- ItemHeight
- ItemWidth

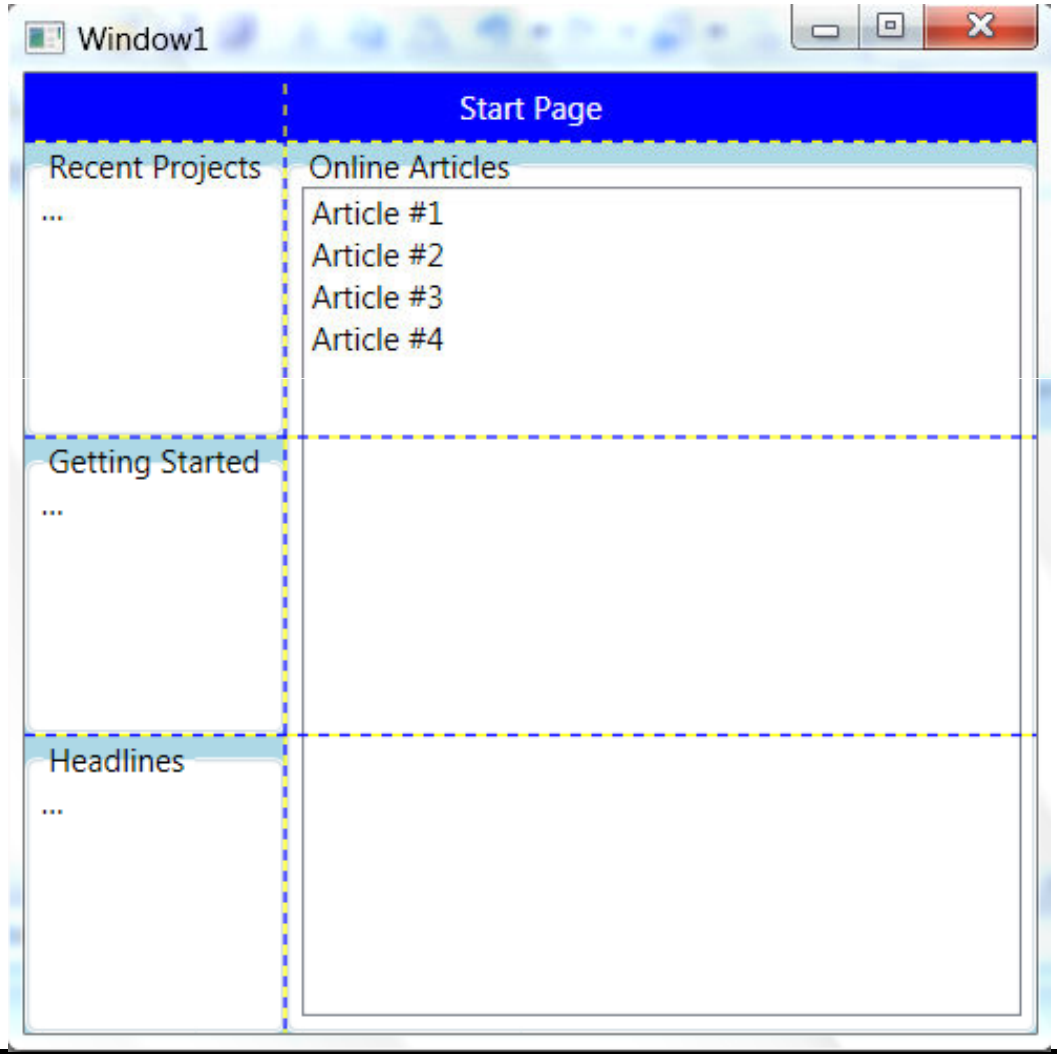


Grid

- Táblázat
- Cellák összevonása
- Automatikus méretezés
  - A benne lévő vezérlők alapján
- Elfoglalja a lehetséges helyet

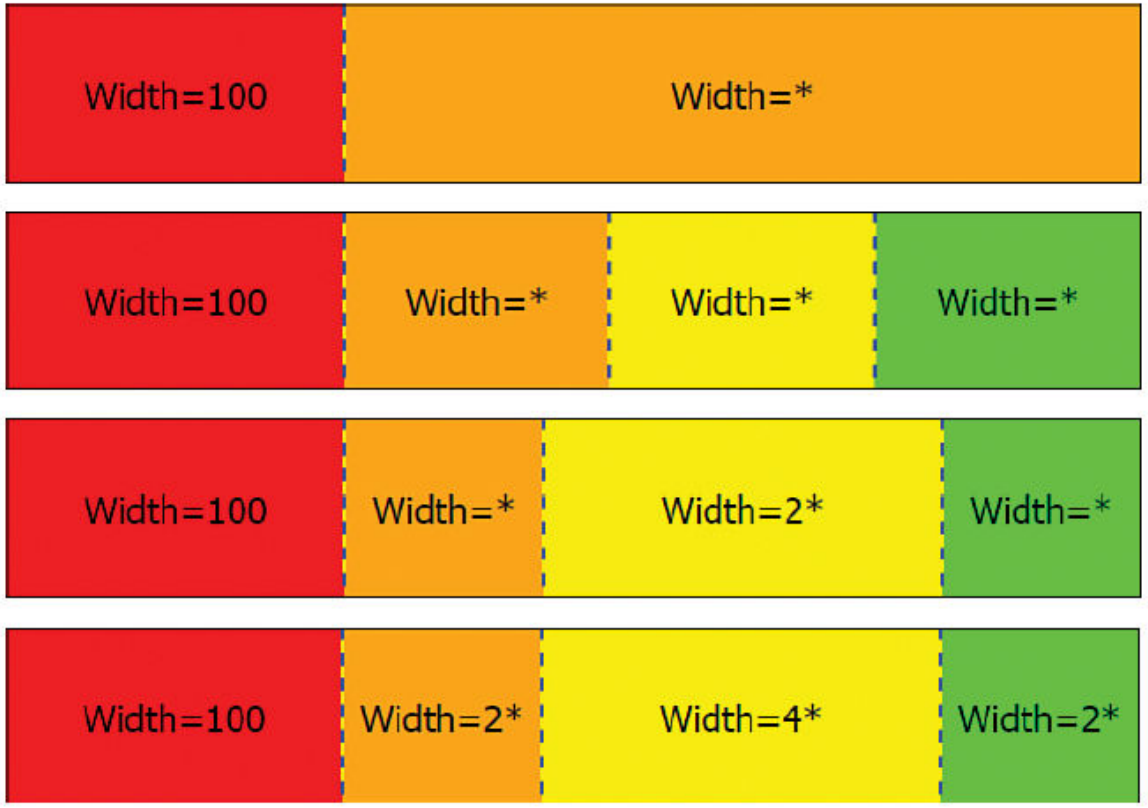
Grid

- ShowGridLines="True"



Grid

- Cellák méretezése (GridLength):
  - Abszolút méretezés: Height="100"
  - Auto méretezés: Height="Auto"
  - Star méretezés: Height="2*"



Grid

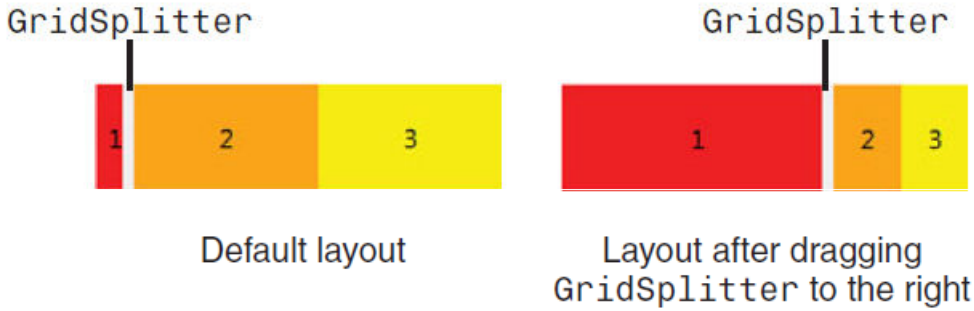
- Interaktív méretezés (GridSplitter)
- ResizeDirection, ResieBehavior (BasedOnAlignment / PreviousAndCurrent / CurrentAndNext / PreviousAndNext)
- HorizontalAlignment/VerticalAlignment:

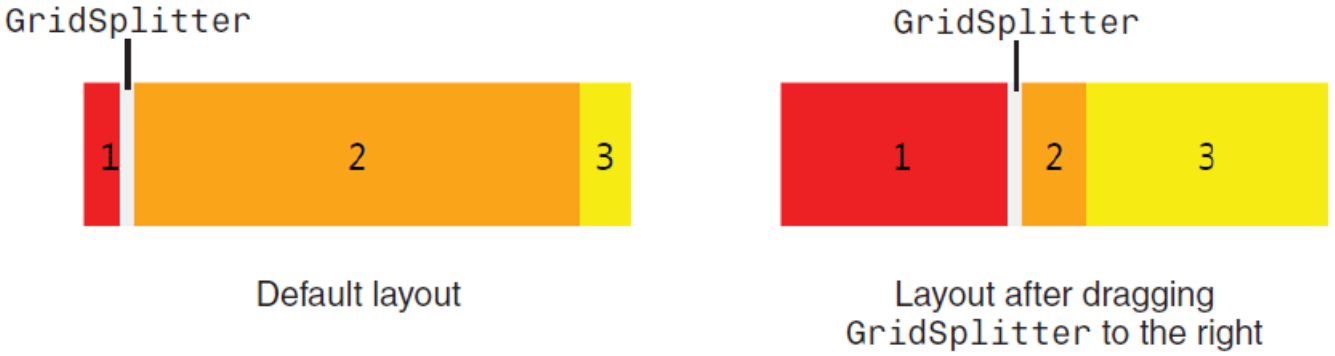
HorizontalAlignment

VerticalAlignment	Left	Right	Center	Stretch
Top	Current cell and cell to the left	Current cell and cell to the right	Cells to the left and right	Current cell and cell above
Bottom	Current cell and cell to the left	Current cell and cell to the right	Cells to the left and right	Current cell and cell above
Center	Current cell and cell to the left	Current cell and cell to the right	Cells to the left and right	**Cells above and below**
Stretch	Current cell and cell to the left	Current cell and cell to the right	Cells to the left and right	**Cells to the left and right if `GridSplitter` is taller than it is wide, *or* cells to the top and bottom if `GridSplitter` is wider than it is tall**

Grid

- Interaktív méretezés (GridSplitter)
- SharedSizedGroup:





WPF – adatkezelés

Agenda

1. WPF architektúra, GDI, DirectX, DWM
   - Többszálúság, property system
   - Grafikus primitívek, rajzolás alacsony szinten
   - Média/kép kezelés
2. Vezérlő hierarchia, XAML, layout
3. Adatkötés, sablonok, erőforrások, stílusok, triggerek, eseménykezelés
4. MVVM, 3D, dokumentum kezelés

Agenda

- Erőforrások
- Adatkötés
- Sablonok
  - Adatsablonok, triggerek
  - Vezérlősablonok, stílusok
- Animáció
- Adat validáció
- Eseménykezelés

Erőforrások

- Nem csak bináris erőforrások: szöveg, kép, stb.
- Hanem XAML-ben deklarált objektumhierarchiák
  - Alkalmazáshoz vagy valamelyik vezérlőhöz tartozik
  - Kulcs alapú azonosítás:‘x:Key’ attribútum,előldeklarálás

```
<s:Double x:Key=“betűméret”> 14 </…>
```

  - Statikus hivatkozás: csak egyszer kérdezi le

```
<TextBlock FontSize="{StaticResource betűméret}">alma</…>
```

  - Dinamikus hivatkozás: minden alkalommal lekérdezi
  - Dekomponálás, erőforrásfájlok összefűzése

```
<ResourceDictionary.MergedDictionaries> …
```

Bináris erőforrás tárolása

- „Hagyományos” erőforrások (pl. bitmap)
  - XAML is így tárolódik (BAML)
- Fajtái
  - **Resource**: szerelvénybe ágyazott
    - Build Action = Resource
  - **Content**: szerelvényen kívül álló fájl de hozzá tartozó fájl
    - Build Action = Content
    - Relatív útvonallal lehet hivatkozni rá
  - **Site of Origin**: a szerelvénytől független fájl

Bináris erőforrások elérése

- Resource:
  - Egyszerű URI hivatkozással
  - Kódból stream alapján: GetResourceStream
- Content:
  - Egyszerű URI hivatkozással
  - Kódból stream alapján: GetContentStream
- Site of Origin:
  - Hagyományos URL hivatkozással
  - Kódból stream alapján: GetRemoteStream

Logikai erőforrások deklarálása

- *FrameworkElement.Resource - ResourceDictionary*
  - Kulcs-objektum párok
- Alkalmazás vagy vezérlő szinten
- XAML erőforrás: tetszőleges objektum-fa
- Példa:

```
<Window xmlns="…" xmlns:x="…" Title="Simple
Window">
  <Window.Resources>
    <SolidColorBrush x:Key="backgroundBrush">
      Yellow</SolidColorBrush>
    <SolidColorBrush x:Key="borderBrush">
      Red</SolidColorBrush>
  </Window.Resources>
```

Logikai erőforrások használata

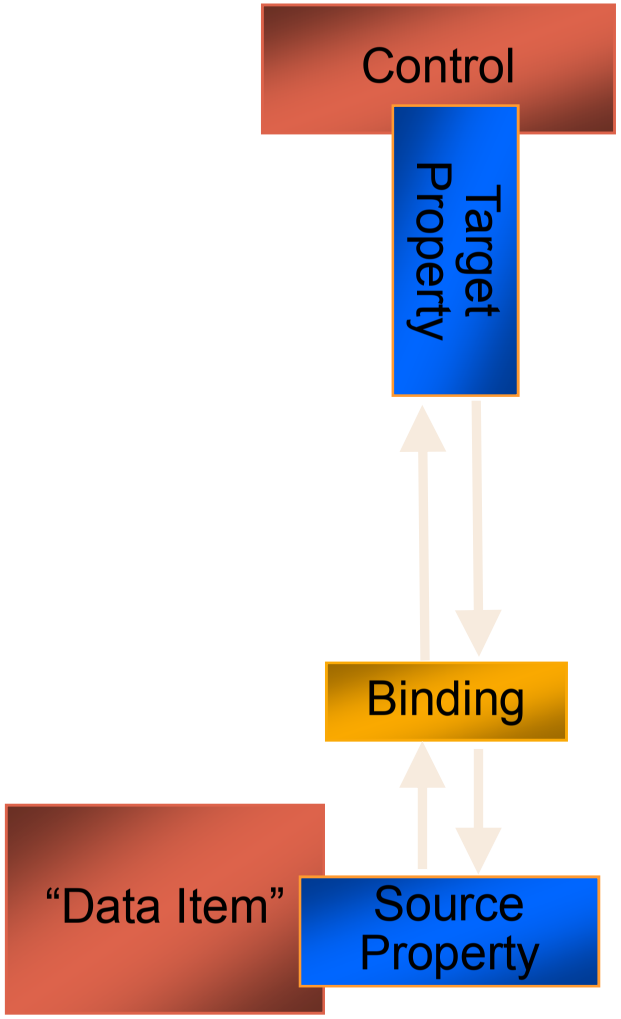
- Erőforrás elérése
  - `<StaticResource ResourceKey="backgroundBrush"/>`
  - `<Button Background="{StaticResource backgroundBrush}" … />`
- A logical treeben felfelé haladva valamennyi erőforrás elemeit látjuk (alkalmazás szintűt is)
- *Dinamikus erőforrások*
  - Előre mutató referencia hivatkozások
  - Változásértesítés – téma csere, stb.

Agenda

- Erőforrások
- Adatkötés
- Sablonok
  - Adatsablonok, triggerek
  - Vezérlősablonok, stílusok
- Animáció
- Adat validáció
- Eseménykezelés

Adatkötés WPF-ben

- Adatkötés célja: egy vezérlő tulajdonságainak hozzákötése az adatforráshoz
- Target: dependency property
- Source: tetszőleges property



Binding osztály

- Az adatkötés jellemzőit írja le
- *System.Windows.Data.Binding*
  - *Source* : forrás objektum, tetszőleges típus
  - *Path* : forrás property, elmaradhat
- Cél objektum : csak **DependencyObject**
- Cél property : csak **dependency property**
  - *FrameworkElement.SetBinding vagy*
  - *BindingOperations.SetBinding*
  - *Törlés:* BindingOperations.ClearBinding

Binding . Source

- Alapértelmezett forrás a logikai fában a cél objektumhoz rendelt DataContext tulajdonság értéke
  - Ha a cél objektumnak nincs DataContextje explicit megadva, akkor a logikai fában a szülőtől örökli
  - A Source explicit beállítható másik objektumra
- Nem állítható együtt az ElementName és RelativeSource tulajdonságokkal

Binding . ElementName

- Az adatkötés forrása lehet egy másik vezérlő
- Az ElementName tulajdonság kell hivatkozzon a forrás vezérlő nevére
- Nem állítható a Source, RelativeSource tulajdonságokkal együtt

Binding . RelativeSource

- Binding . RelativeSource beállítása
  - Saját objektumon lévő propertyhez köt
  - Tipikusan sablonokban, stílusokban használják
- Típusai (RelatveSourceMode):
  - **Self**: az adat forrás megegyezik a cél objektummal
  - **TemplatedParent**: sablonoknál az adatforrás
  - **FindAncestor**: típus vagy mélység alapján keres szülő objektumot a logikai fában
  - **PreviousData**: gyűjtemény esetén az előző element jelenti, összehasonlításra jól használható

Binding . Path

- Az adatforrás pontos elemét határozza meg:
- **PropertyName**: egyszerű tulajdonság név
- **A.B**: elérési útvonal a propertyken keresztül
- **(DockPanel.Dock)**: hivatkozás csatolt tulajdonságra
- **Items[0]**: indexelt elem
- **„/”**: gyűjtemény esetén az aktuális elem kiválasztása
- **„.”**: a teljes forrás objektum kiválasztása
- A fenti lehetőségek kombinálhatók (pl „Items/s[1]”)

Binding . Mode

- Az adatkötés módját alapértelmezetten a célproperty adja meg vagy explicit beállítható
  - TextBox.Text, stb kétirányú, a többi tipikusan egyirányú
- **TwoWay**: kétirányú kötés, a forrás- és cél property változása frissíti a másikat
- **OneWay**: a cél property frissül amikor a forrás megváltozik
- **OneTime**: csak alkalmazás indulásakor vagy a DataContext változásakor kerül frissítésre
- **OneWayToSource**: a forrás property frissül a mikor a cél property megváltozik
- **Default**: alapértelmezett, a cél propertytől függ

Binding . UpdateSourceTrigger

- Kétirányú kötésnél a frissítés módját alapértelmezetten a célproperty adja meg vagy explicit beállítható
  - TextBox.Text, pl LostFocusra frissít, általában PropertyChanged
- **PropertyChanged**: a cél property változásakor frissül a forrás property
- **LostFocus**: focus vesztésnél kerül frissítésre a forrás
- **Explicit**: nincs automatikus frissítés, UpdateSource metódust kell hívni a BindingExpression-ön
  - Minden FrameworkElementnek van ilyen metódusa
- **Default**: alapértelmezett, a cél propertytől függ

Binding . Converter

- Konvertálás: *Binding.Converter*
  - Adatcsere előtt az adat transzformálható
  - *IValueConverter*-t implementáló osztály
  - **ValueConversionAttribute** a konverter osztály külön megjelölhető
  - A konvertálás bemenő argumentumot, paramétert és kultúra információt kap
    - A paraméter a kötésnél adható meg
  - Convert: az egyirányú adatkötéshez
  - ConvertBack: a kétirányú adatkötéshez

Binding egyebek

- **StringFormat**: Ha a cél property szöveg, akkor a forrás adatot ezzel a stringgel formázza
  - {0}-val kell hivatkozni az adatra
- **TargetNullValue**: Ezt az objektumot használja, amikor a forrás (illetve az elérési útban bármelyik elem) *null*.

Binding: XML forrás

- **XPath**: ez adja meg a kifejezést ami az XML-ből kiválasztja a forrás részt. Az eredmény egy attribútum, csomópont vagy csomópontok halmaza.
- A forrás az **XmlDataProvider**-rel adható meg
  - Távoli fájlra is rá lehet irányítani
  - Az névtér kezelés külön figyelmet igényel

Binding vs. BindingExpression

- **Binding**: sablon, ami az adatkötés tulajdonságait átrolja, megosztható több binding között
  - BindingOperations . Get/SetBinding(DO,DP,binding)
  - FrameworkElement . SetBinding(DP, binding)
- **BindingExpression**: egy kötés példány
  - BindingOperations . GetBindingExpression(DO, DP)
  - FrameworkElement . GetBindingExpression(DP)
  - UpdateTarget / UpdateSource
  - ValidateWithoutUpdate

Adatforrás változás jelzés

- Property Changes
  - INotifyPropertyChanged
  - PropertyChanged event pattern
    - Szöveggel hivatkozott tulajdonság megváltozott
- Collection Changes
  - INotifyCollectionChanged
    - Új elem, elem mozgatása/törlése/változása, reset
  - ObseverableCollection<T> : INCC

Listás adatkötés

- **ItemsControl . ItemsSource** propertyra kell kötni
  - A ListBox, stb. az ItemsControlból származik
- Nézetek: **CollectionView** alaposztály
  - Aktuális elem nyilvántartása
  - Adatlista rendezése, szűrése és csoportosítása (!)
  - Minél gazdagabb forrás lista, annál optimálisabb
- XAML-ben: **CollectionViewSource**, becsomagolja a CV-t, paraméterezhetjük, tipikusan erőforrásként
  - Source property: a forrás lista, tipikusan adatkötéssel
- Master-Detail: az összes vezérlőt a CV-ra kötve automatikusan megkapják az aktuális elemet

ObjectDataProvider

- XAML-ben jelenik meg, az adatforrást csomagolja be
  - Eszközökbarát (metaadat)
  - Létrehozás konstruktorral, paraméterezve
  - Létrehozás metódushívással, paraméterezve
  - Aszinkron létrehozás támogatása

Adatkötés XAML-ben

- BindingMarkupExtension
  - ItemsSource="{Binding Source={StaticResource listingDataView}}”
  - Text="{Binding Path=StartDate, Converter={StaticResource dateConverter}}„
    - <local:DateConverter x:Key=“dateConverter”/>

Agenda

- Erőforrások
- Adatkötés
- Sablonok
  - Adatsablonok, triggerek
  - Vezérlősablonok, stílusok
- Animáció
- Adat validáció
- Eseménykezelés

FrameworkTemplate ősosztály

- Logikai fa sablont tárol
- Lehetővé teszi a sablon alapú példányosítást
- A sablon határozza meg, hogy milyen vezérlőket kell a példányosítás során létrehozni és összerendelni
- Leszármazottjai:
  - DataTemplate
  - ControlTemplate
  - ItemsPanelTemplate

Adat sablon (DataTemplate)

- Meghatározza az adat vizuális megjelenését
- Sablon választás
  - Vezérlő **ItemTemplate** tulajdonsága
  - **DataTemplateSelector** beállítása
  - A DataTemplate erőforrásban ki van töltve a **DataType**
- A sablonokat a **ContentPresenter** kezeli

```
<DataTemplate DataType="{x:Type local:Task}">
 <StackPanel>
  <TextBlock Text="{Binding Path=TaskName}" />
  <TextBlock Text="{Binding Path=Description}"/>
  <TextBlock Text="{Binding Path=Priority}"/>
 </StackPanel>
</DataTemplate>
```

Adat triggerek (DataTrigger)

- Tulajdonságok megváltoztatása a forrás adatnak megfelelően
  - Vizuális *megjelenés* deklaratív leírása
  - **Binding**: a vizsgálandó adatot választja ki
  - **Value**: az érték, amire a triggel tüzel
  - **Setters**: beállítandó tulajdonságok és értékük
- **Setter** osztály: a tulajdonságok beállítását végzi
  - **Property**: a beállítandó tulajdonság
  - **Value**: az érték
  - **TargetName**: a vezérlő, amit át kell állítani
- Általában a stílus vagy adat sablon része
- **MultiDataTrigger**: több feltétel is megadható(AND)

Trigger példa

```
<DataTemplate.Triggers>
 <DataTrigger Binding="{Binding Path=TaskType}">
  <DataTrigger.Value>
    <local:TaskType>Home</local:TaskType>
  </DataTrigger.Value>
  <Setter TargetName="border"
          Property="BorderBrush" Value="Yellow"/>
 </DataTrigger>
</DataTemplate.Triggers>
```

Agenda

- Erőforrások
- Adatkötés
- Sablonok
  - Adatsablonok, triggerek
  - Vezérlősablonok, stílusok
- Adat validáció
- Eseménykezelés

Vezérlő sablonok

- Egy WPF vezérlő kódja független a megjelenéstől
- A vezérlő vizuális megjelenését és vizuális működését a ControlTemplate határozza meg
- A template-ben jelzni kell a tartalom helyét
  - ContentControlnál a ContentPresenter osztály
    - Button esetén ide kerül a tartalom (Content)
    - TabItem esetén ez jelenítette meg szöveg helyett a szűrőt
  - ItemsControl-nál az ItemsPresenter vagy a Panel (IsItemsHost=true) ad helyet az elemeknek (például ListBoxItem-ek)

Vezérlő sablon példa

```
<ItemsControl.Template>
  <ControlTemplate TargetType="ItemsControl">
    <Border BorderBrush="Aqua" BorderThickness="1"
CornerRadius="15">
      <ItemsPresenter/>
    </Border>
  </ControlTemplate>
</ItemsControl.Template>

<ItemsControl.ItemsPanel>
  <ItemsPanelTemplate>
    <WrapPanel />
  </ItemsPanelTemplate>
</ItemsControl.ItemsPanel>
```

Stílusok

- Tetszőleges tulajdonság beállítása FrameworkElement-ből származó osztályokon

```
<Style TargetType="ListBoxItem">
        <Setter Property="Opacity" Value="0.5" /> …
```

- Egy vezérlőhöz egyszerre csak egy stílus tartozhat
  - De a stílusok származhatnak egymásból
- Tipikusan erőforrásként definiáljuk a stílusokat
  - TargetType megadása esetén automatikus
  - Ha az erőforrás kulcs is meg van adva, akkor explicit
- Minden beépített vezérlőnek van saját stílusa, az határozza meg a kinézetét is a Template-tel

Agenda

- Erőforrások
- Adatkötés
- Sablonok
  - Adatsablonok, triggerek
  - Vezérlősablonok, stílusok
- Adat validáció
- Eseménykezelés

Adat validáció

- A Binding osztály ValidationRules gyűjteményébe kerülhetnek szabályok
  - **ExceptionValidationRule**: ha a forrás objektum beállítása kivételt dob, akkor az adat hibás
    - Rövidebben: **ValidatesOnExceptions** = true
  - **DataErrorValidationRule**: ha az objektum implementálja az **IDataErrorInfo** interfészt
    - Rövidebben: **ValidatesOnDataErrors** = true
  - Saját validációs szabály is készíthető
- A validációs hibákat a (**Validation.Errors**) csatolt tulajdonsággal lehet lekérdezni

Az IDataErrorInfo interfész

- **String Error**: általános hibalírás az objektumhoz
- **String Item(string propertyName)**: tulajdonság specifikusan lehet megadni a hibaszöveget

Hiba megjelenítés

- **Validation . ErrorTemplate** csatolt tulajdonsággal megadható a hibás adathoz tartozó vezérlő sablonja
  - **AdornedElementPlaceholder**: vezérlő sablon esetén a meglévő elemet lehet elhelyezni az új logikai fában

```
<ControlTemplate x:Key="validationTemplate">
 <DockPanel>
  <TextBlock Foreground="Red" FontSize="20">!</TextBlock>
  <AdornedElementPlaceholder/>
 </DockPanel>
</ControlTemplate>
```

Agenda

- Erőforrások
- Adatkötés
- Sablonok
  - Adatsablonok, triggerek
  - Vezérlősablonok, stílusok
- Adat validáció
- Eseménykezelés

Eseménykezelés

- Hasonlóan delegate alapú megközelítés
- Rugalmasabb mint a Win32+WinForms
  - Nem téglalap alakú vezérlők hierarchiája
  - A vezérlő is hasonló UIElementekből épül fel
  - Az események a teljes vezérlőfát bejárják.
    - Kétszer
    - Hasonlóan a DHTML eseménykezeléséhez!
- Alagút események (tunneling)
  - Preview, Handled tulajdonság
- Buborék események (bubbling)