Windows Presentation Foundation (WPF) alapismeretek

Jánosi-Rancz Katalin Tünde

Sapientia EMTE
tsuto@ms.sapientia.ro

WPF tulajdonságai

- okos kliens alkalmazások vizuális felületének létrehozására használjuk
- a `Windows Presentation Foundation (WPF)` a .NET környezet vektoros alapú grafikus felületi rendszere
- .NET 3.0, 2006-tól
- a Windows Forms utódának tekinthető
  - legfontosabb különbség, hogy az alkalmazás megjelenéséért felelős kód elkülönül az alkalmazás funkcionalitását leíró kódtól
  - WinForm alapjául szolgáló grafikus technológia a GDI/GDI+, a WPF-é már a DirectX
    - DirectX tehermentesíti a processzort, inkább a videokártyát (GPU-t) terheli meg. Gyorsabbá válnak pl. az animációk lejátszása
    - lehetővé teszi a 3D grafikus kártyák kihasználását (Direct3D)
    - a WinForms esetében a felbontás kötötte a fejlesztőket, így a felhasználói felület nem volt skálázható. WPF-ben a grafikai elemek már nem raszteresek, hanem vektor alapúak, az egyes elemek tetszőlegesen átméretezhetőek. A vektor alapú képek kevesebb helyet foglalnak a raszteres elemekhez képest. A WPF továbbra is támogatja a raszter grafikát.

WPF tulajdonságai -2

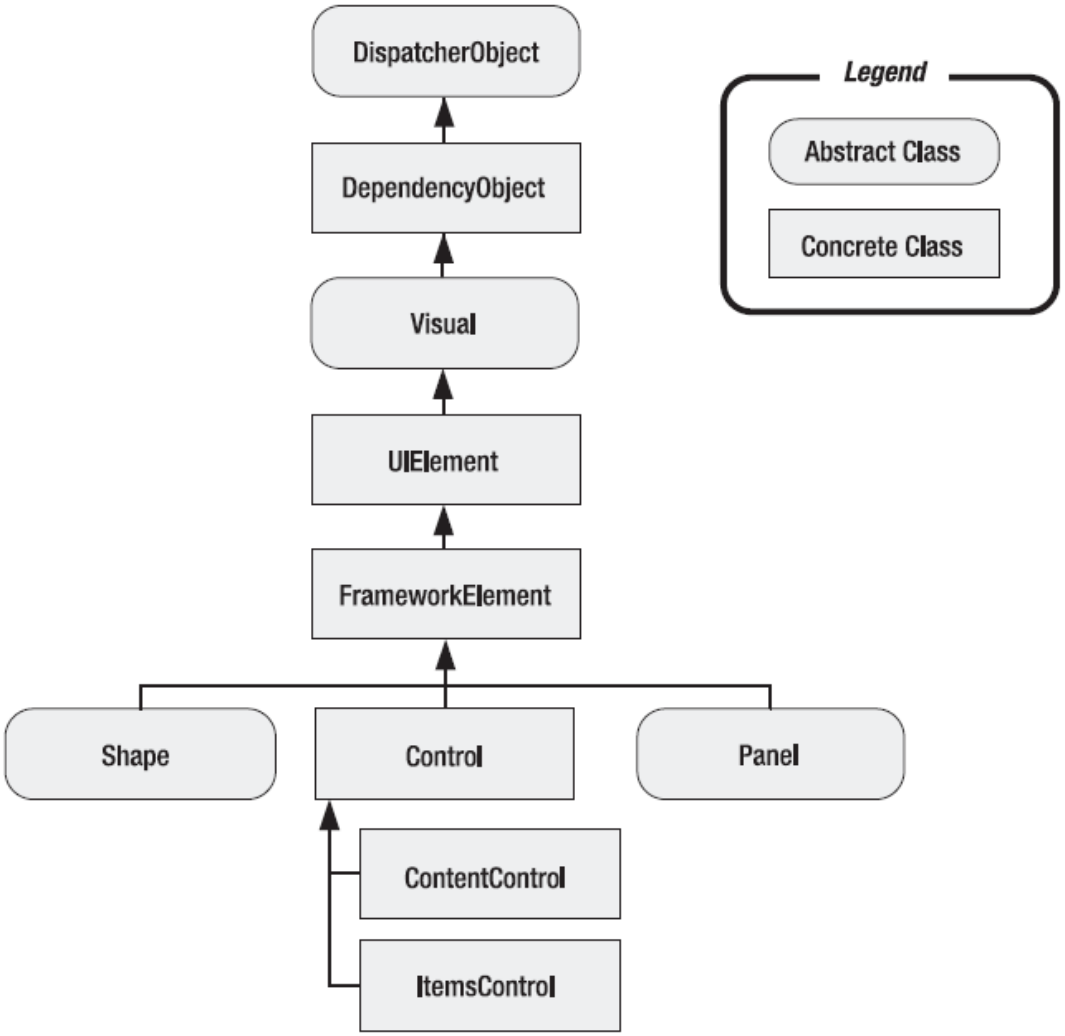
- összekombinálja UI-t, 2D, 3D grafika, dokumentumok és multimédia használatát egyetlen keretrendszerbe
- lehetővé teszi, hogy felhasználói felületet készítsünk úgy Windows alkalmazásoknak mint webes alkalmazásoknak (Silverlight / XBAP)
- lehetőséget ad a felület deklaratív leírására (XAML)
- azzal, hogy függetleníti a megjelenést és a vezérlést, jelentősen javít az alkalmazások architektúrán (MVVM)
- hátránya, hogy csak Windows rendszerekre érhető el

képek: http://www.wpftutorial.net/

WPF felépítése

- az osztályok a `System.Windows` névtérben helyezkednek el
  - WinForms osztályai a `System.Windows.Forms` névtérben (a WinForms Button osztálya a `System.Windows.Forms.Button`, míg a WPF Button osztálya a `System.Windows.Controls.Button`)
- az alkalmazások futása, és a kirajzolás folyamata jóval összetettebb
  - a képalkotást külön szál (rendering thread) végzi az elemkezeléstől (dispatcher thread)

WPF osztályhierarchiája



forrás: http://www.c-sharpcorner.com/

WPF osztályhierarchiája -2

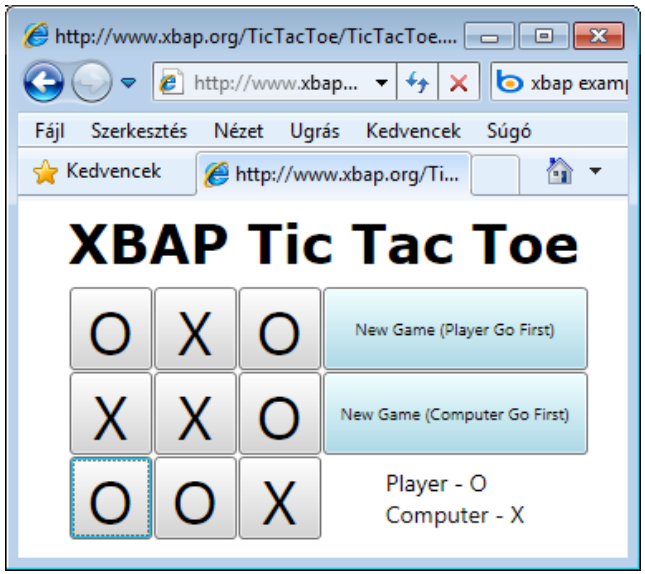
- összes osztály a `System.Object` -ből származik
- a legtöbb objektuma a `DispatcherObject` -ből származik. A WPF alkalmazások az egyszálú (single-thread affinity, STA) modellt használják. Ennek értelmében egyetlen szál vezérli és felügyeli a teljes felhasználói felületet.
- `DependencyObject` - kiszámolja a property-k értékeit és a property változásokról értesítést küld a rendszernek (SetValue, GetValue)
- a `Visual` osztály kapcsolatot biztosít menedzselt WPF könyvtárak és a milcore.dll között. (a milcore.dll a WPF magja, magasabb szintű grafikai elemeket fordítja át DirectX elemekre)
- az `UIElement` tartalmazza a WPF lényegesebb elemeit (pl. StackPanel, Grid), továbbá támogatja a fókusz és események kezelését

WPF osztályhierarchiája -3

- `FrameworkElement` - elrendezés tulajdonságait adhatjuk meg, mint például HorizontalAlignment, Width, Margin, stb.
- `Shape` - az alapvető alakzatok, mint például Rectangle, Poligon, Line származnak ebből az osztályból
- a `Control` osztályból származnak az alapvető vezérlők TextBox, Button, ListBox, ezek további beállítására is biztosít lehetőséget (pl. Font, Background)
- `ContentControl` osztály lehetővé teszi, hogy az egyes vezérlőkhöz gazdag tartalmat adhassunk (pl. előbbi button)

WPF alkalmazás típusok

- Hagyományos asztali alkalmazás
  - igazi újdonság a vektorgrafikus megjelenítés
- Navigáció alapú alkalmazás
  - előre, hátra gombok, lapok közötti lépkedés, előzmények tárolása, navigációs sáv
- XBAP (XAML Browser Application) alkalmazás



forrás: http://johanyak.hu

Felület létrehozása

- C# kódból
- XAML – Visual Studio, Expression Blend, Jegyzettömb

Az XAML nyelv

- Az eXtensible Application Markup Language (XAML) olyan XML alapú deklaratív nyelv, mely biztosítja a grafikus felület teljes leírását
- lehetőséget ad 2D/3D elemek, transzformációk, animációk, valamint további effektek leírására
- Cél: design és kód különválasztása

```
<Window x:Class="HelloWorld.MainWindow"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Title="MainWindow" Height="350" Width="525">
  <Grid>
    <TextBlock Text="Hello World!" />
  </Grid>
</Window>
```

- NB: weboldalak esetén <Window> helyett a gyökér <Page>

XAML

- Gyökérelemek
  - Window
  - Page
  - Application
- Elem
  - egy osztály
- Attribútum
  - egy tulajdonság a komponens osztályában

Az XAML nyelv

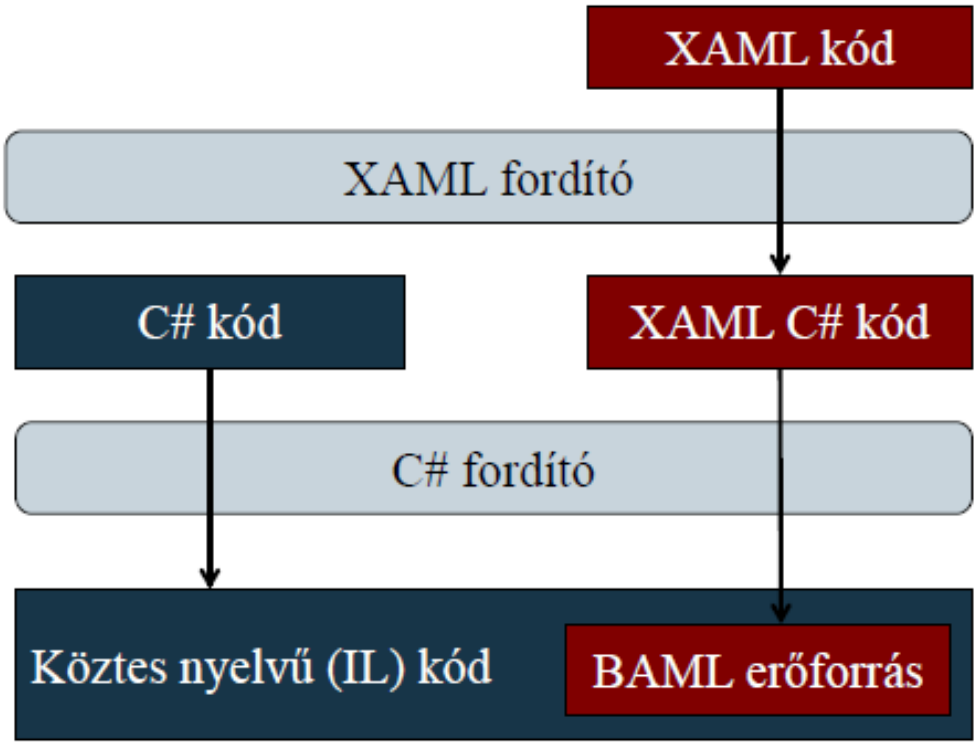
- minden XAML elemtípus megfeleltethető egy .NET osztálynak
- minden leírható kóddal is

```
<Canvas Name=”myCanvas”>  <!--vászon -->
    <Label Name=”myLabel” BorderBrush=”Red”> <!--címke a vászo
    an -->
            Hello World!
    </Label>
</Canvas>
```

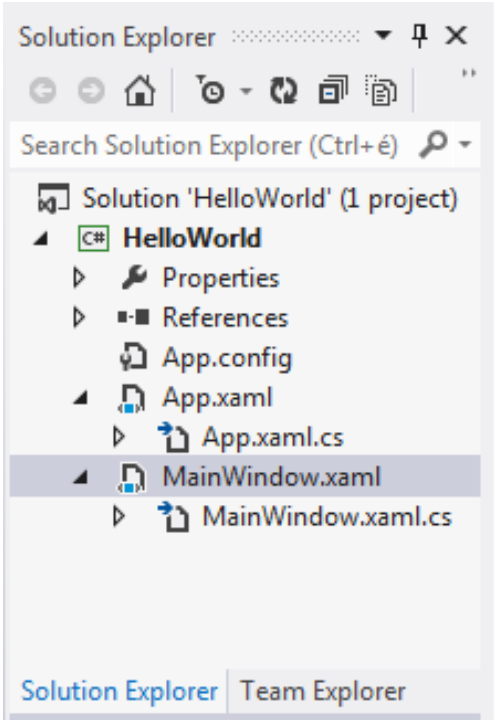
```
Canvas myCanvas= new Canvas(); // vászon
Label myLabel= new Label(); // címke
myLabel.Content= ”Hello World!”; // tartalom
myLabel.BorderBrush= Brushes.Red; // szegély
myCanvas.Children.Add(myLabel); // behelyezés
```

- Az XAML kód átalakul BAML (BinaryXAML) formátumra, amely erőforrásként csatolható a felügyelt kódhoz

Az XAML fordítása



WPF projekt alapvető állományai



WPF projekt alapvető állományai

- App.xaml - az alkalmazást vezérli (erőforrásait és indítási beállításait tartalmazza)

```
<Application x:Class="HelloWorld.App"
 xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
 xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
 StartupUri="MainWindow.xaml"> <!--megadjuk a kezdoablakot -->
    <Application.Resources>

    </Application.Resources>
</Application>
```

- Háttérkód osztály (MainWindow.xaml.cs) - megjelenést el lehessen választani a funkcionalitásától

```
using ...
namespace HelloWorld {
 public partial class MainWindow : Window
 {
    public MainWindow() //felületi kódot a konstruktor fogja lefuttatni
    {
       InitializeComponent();
    }
```

Application osztály

- System.Windows.Application
- Tulajdonságok:
  - Current – a futó alkalmazás objektum – pl. alkalmazás szintű változók, leállítás
  - MainWindow – a fő ablak
  - Windows – az alkalmazást létrehozó szálból létrehozott ablakok gyűjteménye
  - StartupUri – az alkalmazás indításakor automatikusan megnyitott ablak/lap
  - Properties – az alkalmazásból bárhonnan elérhető adatok tárolása

```
Application.Current.Properties[”Név”]=Érték;
```

- Események:
  - Startup
  - Exit

XAML névterek

- a névterek adják meg a XAML-dokumentumunkban (az adott elemen belül) használható kulcsszavak körét

```
<Window x:Class="WpfApplication.MainWindow"
      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
       Title="MainWindow" Height="150" Width="150">
    <Grid> </Grid>
</Window>
```

- 1. alapértelmezett WPF névtér - a felhasználói felület építéséhez szükséges WPF osztályokat (vezérlőket)

```
< Window  x:Class="WpfApplication.MainWindow"
       xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
       Title ="MainWindow"  Height ="150"  Width ="150">
      <Grid></Grid>
</ Window >
```

- az ebből érkező kulcsszavakat nem kell névtérnévvel ellátni

XAML névterek -2

- 2. a XAML-hez tartozó névtér - a XAML dokumentumok értelmezéséhez szükséges általános definíciókat
  - `Class, Null, Static, Array, ClassModifier, FieldModifier, DynamicResource, StaticResource, Key, Name, Code, …`

```
<Window x:Class="WpfApplication.MainWindow" x:ClassModifier="internal"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="MainWindow" Height="150" Width="150">
    <Button Content="Ok" Name="buttonOk" x:FieldModifier="public"/>
</Window>
```

- ennek a névtérnek a neve hagyományosan `x`, az ebből érkező kulcsszavakat `x:` prefix-el kell ellátni
- xmlns feladata: egy álnevet rendel az URI (Uniform Resource Locator) formájában megadott névtérhez
- létrehozhatunk saját névtereket is

Ablakok - Windows

- Window osztály leszármazottai, amelyek parciális osztályként rendelkeznek felületi kóddal (.xaml), valamint háttérkóddal (.xaml.cs)
- a felületi kódban adjuk meg a deklaratív leírást, pl.:

```
<Window x:Class="Face.MainWindow"…
Title="My Window" Height="200" Width="200">-
<!-megadjuk címét és méreteit -->
<Canvas> … </Canvas>
<!--vászon a további elemeknek -->
</Window>
```

- az x:Class attribútum megmondja a XAML parser-nek, hogy hozzon létre egy új osztályt a megadott néven (MainWindow.xaml.cs)
- meg kell adnunk a felhasznált sémákat és névtereket

Ablakok - Windows -2

- az ablakba csak egy elem helyezhető (ez általában rács, vagy vászon, amely további elemeket tartalmaz)
- a háttérkódban írhatjuk meg a további tevékenységeket, pl. eseménykezelők
- az eseménykezelő társítás történhet a háttérkódban, illetve a felületi kódban is, pl.:

```
// MyWindow.xaml:
<Button Name=”myButton” Click=”myButton_Click”>
<!--gomb eseménykezelo társítással -->
// MyWindow.xaml.cs:
void myButton_Click…() { myButton.Background = Brushes.Red; … }
```

- window tulajdonságok: Icon, ResizeMode, SizeToContent, TopMost, WindowState,...
- (megjegyzést CTRL+KC vagy menüből)

```
<!--gomb eseménykezelo társítással -->
```

Tulajdonságok

```
<Grid>
    <Button x:Name=”Gomb”
     Content=”Gomb” Width=”150” Height=”30” HorizontalAlignment=”Center”
          VerticalAlignment=”Top” Background=”Azure” Foreground=”Blue”
          FontFamily=”Times New Roman” FontSize=”20” FontStyle=”Italic”
          FontWeight=”Heavy” Opacity=”0.5”/>
</Grid>
```

- `x:Name` **nem** tulajdonság, hanem egyedi azonosítót rendel az objektumpéldányhoz
- Összetett tulajdonságok <osztálynév.tulajdonságnév>, pl: háttér színátmenetes kitöltéssel

```
<Button.Background>
    <LinearGradientBrush EndPoint=”1,0.5” StartPoint=”0,0.5”>  <
         GradientStop Color=”Black” Offset=”0” />
    <GradientStop Color=”White” Offset=”1” />
  </LinearGradientBrush>
</Button.Background>
```

- a Content tulajdonság objektum típusú, ezért nyomógombon akár egy színátmenetes kitöltésű ellipszis is elhelyezhető

Alakzatok - Shapes - 2D grafika

- Shape osztály leszármazottai; Rectangle, Line, Path (görbe), Polygon (sokszög)...

```
<Canvas>
       <!--fej:-->
       <Ellipse Canvas.Left=”10” Canvas.Top=”10” Width=”100” Height=”100” Stroke=”Blue”
            StrokeThickness=”4” Fill=”Yellow”/>
       <!--haj:-->
       <Ellipse Canvas.Left=”30” Canvas.Top=”12” Width=”60” Height=”30”>
           <Ellipse.Fill>
               <LinearGradientBrush StartPoint=”0.5,0” EndPoint=”0.5, 1”>
                   <GradientStop Offset=”0.1” Color=”DarkGreen” />
                   <GradientStop Offset=”0.7” Color=”Transparent” />
               </LinearGradientBrush>
           </Ellipse.Fill>
       </Ellipse>
       <!--bal szem:-->
       <Ellipse Canvas.Left=”30” Canvas.Top=”35” Width=”25” Height=”20” Stroke=”Green”
            StrokeThickness=”3” Fill=”White” />
       <!--szemgolyó:-->
       <Ellipse Canvas.Left=”40” Canvas.Top=”43” Width=”6” Height=”5” Fill=”Black” />
       <!--jobb szem:-->
       <Ellipse Canvas.Left=”65” Canvas.Top=”35” Width=”25” Height=”20” Stroke=”Green”
            StrokeThickness=”3” Fill=”White”/>
         <!--szemgolyó:-->
       <Ellipse Canvas.Left=”75” Canvas.Top=”43” Width=”6” Height=”5” Fill=”Black” />
       <!--száj:-->
       <Path Name=”mouth” Stroke=”Red” StrokeThickness=”4” Data=”M 40,74 Q 57,95 80,74 ” />
   </Canvas>
```

```
<Path Name=”mouth” Stroke=”Blue” StrokeThickness=”4”
Data=”M 40,74 Q 57,95 80,74 ” />
//száj - négyzetes Bézier görbe
```

- WPF elemek elérhetőek a .cs kódból is

```
public MainWindow()
{
InitializeComponent();
mouth.Data = Geometry.Parse(”M 40,92 Q 57,75 80,92”);
}
```

Vezérlők - Controls

- nem kell új (származtatott) vezérlőt írni ahhoz, hogy egy vezérlő megjelenését megváltoztassuk (pl. ellipszis alakú nyomógomb) Sablonokkal (Template) megoldva
- csak akkor kell új vezérlőt írni, ha a viselkedést akarjuk megváltoztatni
- alapvető vezérlők
  - Button, RadioButton, ComboBox, CheckBox, Calendar, DatePicker, Expander, DataGrid, ListBox, ListView, Slider, ToggleButton, TreeView, ContextMenu, ScrollBar, Slider, TabControl, TextBlock, TextBox, RepeatButton, RichTextBox, Label
- window és control dekoráció
  - Menu, ToolBar, StatusBar, ToolTip, ProgressBar
- média vezérlők
  - Image, MediaElement, SoundPlayerAction

Vezérlők -2

- Fontosabb tulajdonságok:
  - erőforrások (Resources)
  - sablon (Template), amellyel több vezérlő tulajdonságait tudjuk közösen állítani
  - kinézet (Background, Foreground, BorderBrush, BorderThickness, …)
  - betűkezelés (FontFamily, FontSize, FontStretch, …)
  - pozícionálás és méretezés (Width, ActualWidth, MaxWidth, Padding, Margin, VerticalAlignment, VerticalContentAlignment, RenderTransform, …)
  - engedélyezettség (IsEnabled), láthatóság (IsVisible), fókuszáltság (IsFocused)
  - tabulátorkezelés (TabIndex, IsTabStop)

Vezérlők -3

- vezérlők tartalmazhatnak további vezérlő(ke)t, illetve grafikus elemeket
  - a ContentControl leszármazottai tartalmazhatnak egy másik elemet a Content mezőjükben, pl.:

```
<…Button >
    <Image Source=”…”/>
    <!--a vezérlo Content értékét töltjük fel egy képpel -->
</Button>
```

  - az ItemsControl leszármazottai tetszőlegesen sok elemet tartalmazhatnak (pl. ListBox, ListView, ComboBox)

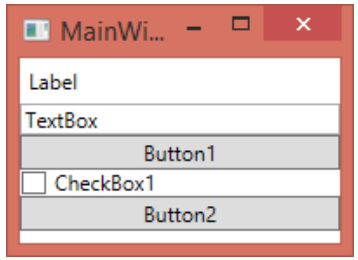
Elhelyezés: tárolók - Panels

- A ContentControl-ból származtatott osztályok (ilyen pl. a Window) **csak egy** objektumot tárolhatnak a Content tulajdonságukban. Ha több komponenst akarunk rátenni, akkor egy tárolót teszünk rá először, majd a tárolóra helyezzük el a további komponenseket.
- a tárolók egymásba ágyazhatóak
- fontosabb tárolók
  - Grid – sorok és oszlopok alakíthatók ki
  - UniformGrid – minden cella azonos méretű
  - StackPanel – vízszintesen vagy függőlegesen elhelyezett elemek
  - DockPanel – dokkolás támogatása
  - Canvas – koordináta alapú tárolás
  - WrapPanel – az elemek egymás mellett helyezkednek el

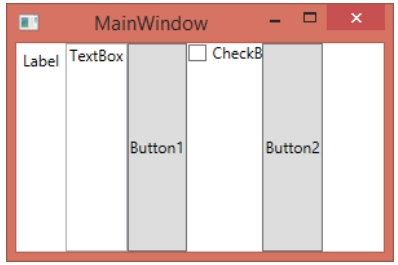
StackPanel

Vízszintesen vagy függőlegesen helyezi el az elemeket, kinyújtva őket, hogy csak egy fér el egy sorban/oszlopban. (Térköz nélkül - használj Margin-t)

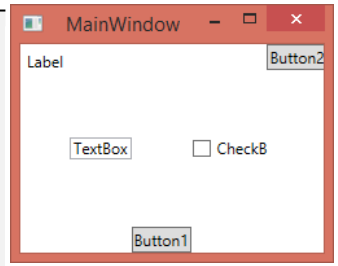
```
<StackPanel Orientation=”Vertical”>
        <Label>Label</Label>
        <TextBox>TextBox</TextBox>
        <Button>Button</Button>
        <CheckBox>CheckB</CheckBox>
</StackPanel>
```




```
<StackPanel Orientation=”Horizontal”>
   <Label>Label</Label>
   <TextBox>TextBox</TextBox>
   <Button>Button1</Button>
   <CheckBox>CheckB</CheckBox>
   <Button>Button2</Button>
</StackPanel>
```



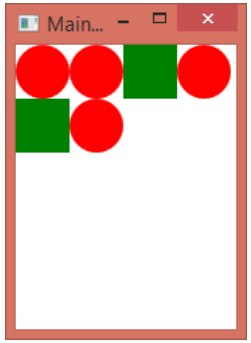

```
<StackPanel Orientation=”Horizontal”>
    <Label VerticalAlignment=”Top”>Label</Label>
    <TextBox VerticalAlignment=”Center”>TextBox</TextBox>
    <Button VerticalAlignment=”Bottom”>Button1</Button>
    <CheckBox VerticalAlignment=”Center”>CheckB</CheckBox>
    <Button VerticalAlignment=”Top”>Button2</Button>
</StackPanel>
```



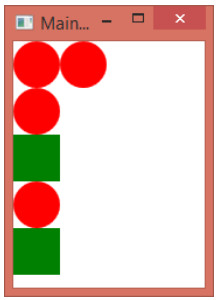
WrapPanel

Egymás mellett vagy alatt helyezi el az elemeket. Amennyiben egy elem nem fér ki a sorba, akkor az automatikusan a következőbe kerül.

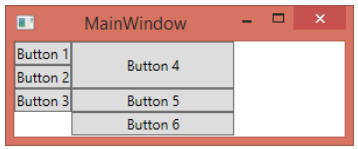
```
<WrapPanel>
    <Ellipse Fill="Red" Height="40" Width="40"/>
    <Ellipse Fill="Red" Height="40" Width="40"/>
    <Rectangle Fill="Green" Height="40" Width="40"/>
    <Ellipse Fill="Red" Height="40" Width="40"/>
    <Rectangle Fill="Green" Height="40" Width="40"/>
    <Ellipse Fill="Red" Height="40" Width="40"/>
</WrapPanel>
```




```
<WrapPanel Orientation="Vertical">
   <Ellipse Fill="Red" Height="40" Width="40"/>
   <Ellipse Fill="Red" Height="40" Width="40"/>
   <Rectangle Fill="Green" Height="40" Width="40"/>
   <Ellipse Fill="Red" Height="40" Width="40"/>
   <Rectangle Fill="Green" Height="40" Width="40"/>
   <Ellipse Fill="Red" Height="40" Width="40"/>
</WrapPanel>
```



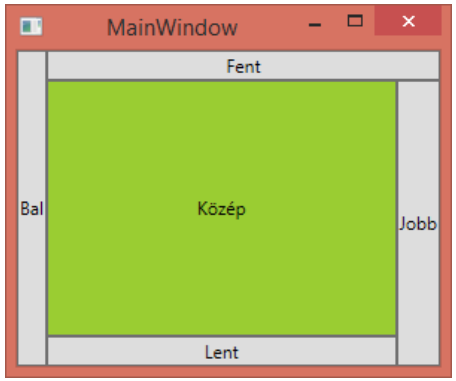

```
 <WrapPanel Orientation="Vertical">
     <Button>Button 1</Button>
     <Button>Button 2</Button>
     <Button>Button 3</Button>
     <Button Width="140" Height="40">Button 4</Button>
     <Button>Button 5</Button>
     <Button>Button 6</Button>
```



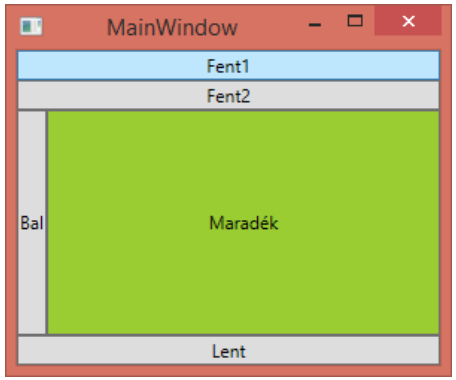
DockPanel

Lehetővé teszi a tartalom mind a négy irányban való elhelyezését. A DockPanel.Dock - milyen irányba akarod pozicionálni. **NB**: nem rendelünk dokkoló pozíciót az utolsó gyereknek, az kitölti a maradék teret, LastChildFill

```
<DockPanel>
    <Button DockPanel.Dock=”Left”>Bal</Button>
    <Button DockPanel.Dock=”Top”>Fent</Button>
    <Button DockPanel.Dock=”Right”>Jobb</Button>
    <Button DockPanel.Dock=”Bottom”>Lent</Button>
    <Button Background=”YellowGreen”>Közép</Button>
</DockPanel>
```



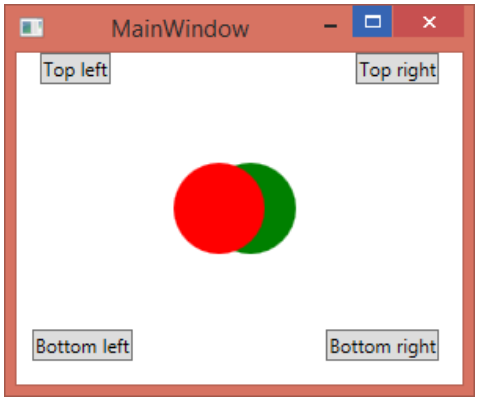

```
<DockPanel>
    <Button DockPanel.Dock=”Top”>Fent1</Button>
    <Button DockPanel.Dock=”Top”>Fent2</Button>
    <Button DockPanel.Dock=”Bottom”>Lent</Button>
    <Button DockPanel.Dock=”Left”>Bal</Button>
    <Button Background=”YellowGreen”>Maradék</
        Button>
</DockPanel>
```



Canvas - vászon

- pixel pontosan megadott elrendezést tesz lehetővé
- bal felső sarokhoz viszonyított koordinátarendszer
- ablak méretezésnél nem alakul
- ZIndex - a magasabb indexű elemek felül

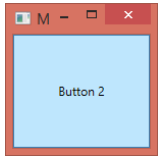
```
<Canvas>
    <Button Canvas.Left=”15”>Top left</Button>
    <Button Canvas.Right=”15”>Top right</Button>
    <Button Canvas.Left=”10” Canvas.Bottom=”15”>Bottom left</Button>
    <Button Canvas.Right=”15” Canvas.Bottom=”15”>Bottom right</Button>
    <Ellipse Canvas.Left=”100” Canvas.Top=”70” Width=”58” Height=”58” Fill=”Red”
        Canvas.ZIndex=”1” />
    <Ellipse Canvas.Left=”120” Canvas.Top=”70” Width=”58” Height=”58” Fill=”Green”  />
</Canvas>
```



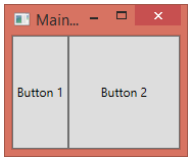
Grid - rács

Az eddig tárgyalt panelek szinte mindegyike megvalósítható vele. Kihasználja az egész teret, szabályozható a sorok és oszlopok mérete(Fixed, Auto, (*))

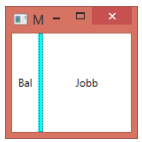
```
<Grid>
   <Button>Button 1</Button>
   <Button>Button 2</Button>
</Grid>
```




```
<Grid>
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width=”*” />
        <ColumnDefinition Width=”2*” />
    </Grid.ColumnDefinitions>
    <Button>Button 1</Button>
    <Button Grid.Column=”1”>Button 2</
        Button>
</Grid>
```



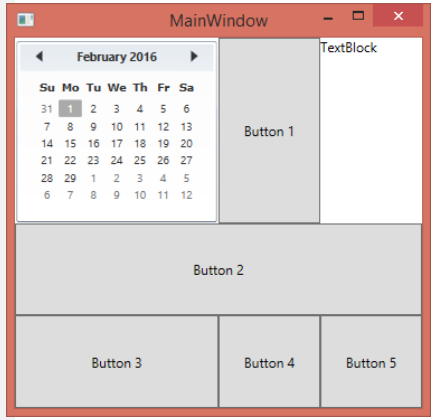

```
<Grid>
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width=”*” />
        <ColumnDefinition Width=”5” />
        <ColumnDefinition Width=”3*” />
    </Grid.ColumnDefinitions>
    <TextBlock HorizontalAlignment=”Center” VerticalAlignment=”Center”>Bal</TextBlock>
    <GridSplitter Grid.Column=”1” Width=”5” HorizontalAlignment=”Stretch” Background=”Aqua” />
    <TextBlock Grid.Column=”2”  HorizontalAlignment=”Center” VerticalAlignment=”Center” >Jobb</
        TextBlock>
</Grid>
```



Grid -2

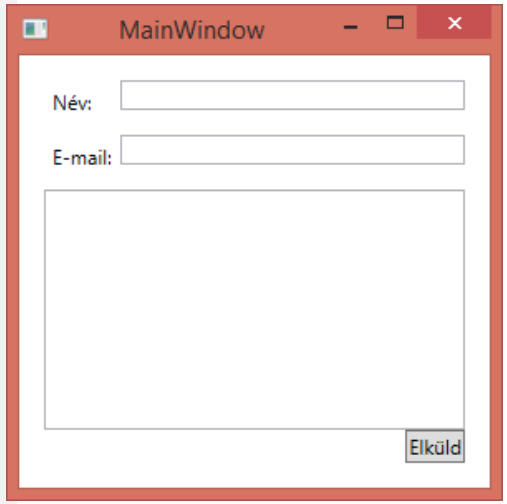
- egy cella több controlt is tartalmazhat
- egységes rács - UniformGrid: minden cella egyforma méretű
- RowSpan és ColumnSpan a sorok/oszlopok egyesítése
- GridSplitter: olyan szétválasztó, amitől tudok resize-olni
- * = a megmaradt rész, 2* = kétszerese

```
<Grid>
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width=”2*” />
        <ColumnDefinition Width=”1*” />
        <ColumnDefinition Width=”1*” />
    </Grid.ColumnDefinitions>
    <Grid.RowDefinitions>
        <RowDefinition Height=”2*” />
        <RowDefinition Height=”1*” />
        <RowDefinition Height=”1*” />
    </Grid.RowDefinitions>
    <Calendar/>
    <Button Grid.Column=”1”>Button1</Button>
    <TextBlock Grid.Column=”2”>TextBlock</TextBlock>
    <Button Grid.Row=”1”>Button 4</Button>
    <Button Grid.Row=”1”  Grid.ColumnSpan=”3”>Button2</Button>
    <Button Grid.Row=”2”>Button3</Button>
    <Button Grid.Column=”1” Grid.Row=”2”>Button4</Button>
    <Button Grid.Column=”2” Grid.Row=”2”>Button5</Button>
</Grid>
```

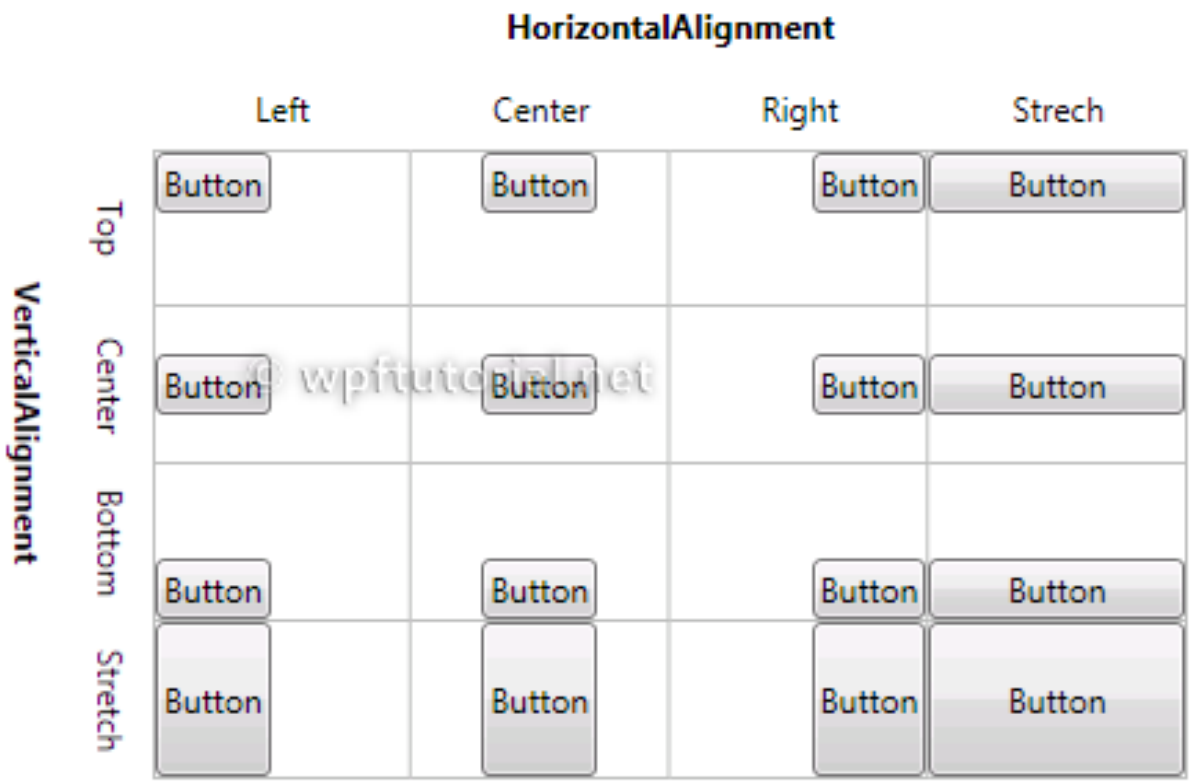


Grid -3

```
<Grid Margin="15">
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="Auto" />
        <ColumnDefinition Width="*" />
    </Grid.ColumnDefinitions>
    <Grid.RowDefinitions>
        <RowDefinition Height="Auto" />
        <RowDefinition Height="Auto" />
        <RowDefinition Height="*" />
        <RowDefinition Height="20" />
    </Grid.RowDefinitions>
    <Label>Név:</Label>
    <TextBox Grid.Column="1" Margin="0,0,0,15" />
    <Label Grid.Row="1">E-mail:</Label>
    <TextBox Grid.Row="1" Grid.Column="1" Margin="
        0,0,0,15" />
    <TextBox Grid.ColumnSpan="2" Grid.Row="2"
        AcceptsReturn="True" />
    <Button Grid.Row="3" Grid.Column="1"
        VerticalAlignment="Center"
        HorizontalAlignment="Right">Elküld</Button>
</Grid>
```

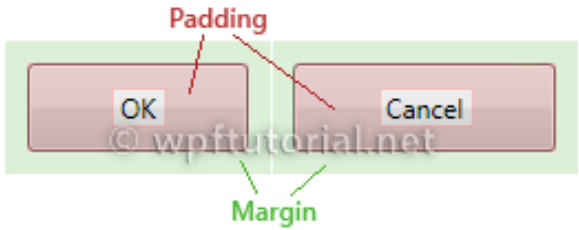


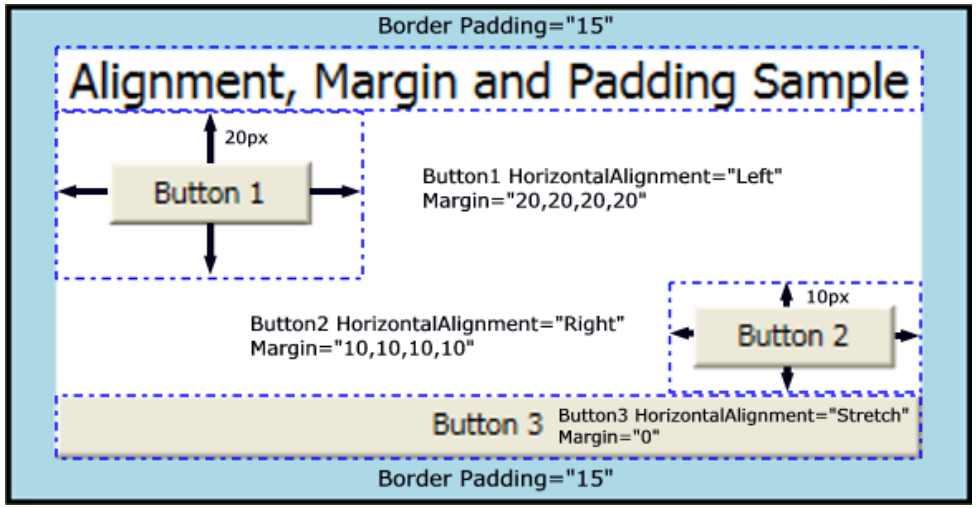
Igazítás


HorizontalAlignment
Left
Center
Right
Strech
VerticalAlignment
Top
Center
Bottom
Stretch
Button

Margin és Padding tulajdonságok

- a `Margin` és `Padding` tulajdonságokat használjuk, hogy fenntartsunk bizonyos teret a vezérlő körül vagy azon belül
  - a `Margin` fenntart egy bizonyos teret a vezérlő körül
  - a `Padding` fenntart egy bizonyos teret a vezérlőn belül





Scrolling

- ha a tartalom túl nagy ahhoz, hogy illeszkedjen a rendelkezésre álló méretben, akkor használjunk `ScrollViewer` -t.
  - függőleges és vízszintes `ScrollbarVisibility` tulajdonság

```
<ScrollViewer>
    <StackPanel>
        <Button Content=”Elso elem” />
        <Button Content=”Második elem” />
        <Button Content=”Harmadik elem” />
    </StackPanel>
</ScrollViewer>
```

Képkezelés

- alapvető képtípus a `BitmapImage`
- a kép helyét Uniform Resource Identifier (URI), illetve relatív hivatkozással (projekthez csatolt kép) adhatjuk meg
- `Strech`:
  - `None, Fill, Uniform, UniformToFill`

```
<Image Source=”E:\kepek\image001.jpg” />
<Image Source=”image002.jpg” Stretch=”Fill”/>
```

MediaElement

- lehetővé teszi különböző média file-ok lejátszását
- minden olyan típust támogat, mint amelyet a Windows Media Player 10 is

```
<MediaElement Name="Player" Grid.Row="1" LoadedBehavior="Manual" />
```

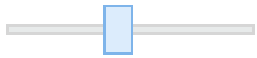
```
Player.Play(); //Média lejátszása
Player.Stop(); // Média megállítása
Player.Volume +=... //Média hangero
```

forrás: http://www.wpf-tutorial.com/

Egyéb vezérlők

- Slider - csúszka
  - egy beállítás értékének megadását teszik lehetővé, egy megadott értéktartományon belül
  - IsDirectionReserved, IsEnabled, LargeChange (PageUp, PageDown gombokhoz), Maximum, Minimum, Orientation, SmallChange (fel/le gombokhoz), Value

```
<Slider x:Name=”csuszka” Width=”100” Value=”50” Minimum=”10”
    Maximum=”100”/>
```



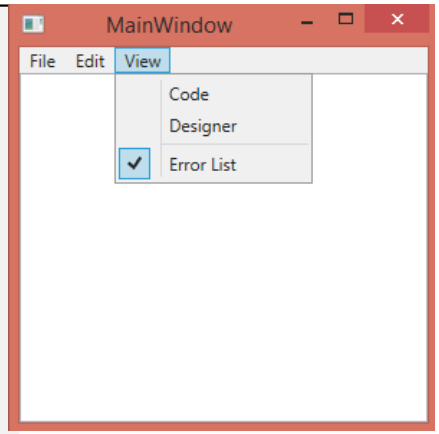
- Progressbar - folyamatjelző
  - IsEnabled, LargeChange (PageUp, PageDown gombokhoz), Maximum, Minimum, Orientation, SmallChange (fel/le gombokhoz), Value

```
<ProgressBar x:Name=”ProgressBar” Width=”200” Height=”30” Value=
    ”40”/>
```

Menu

Parancsok hierarchikus elrendezését teszi lehetővé

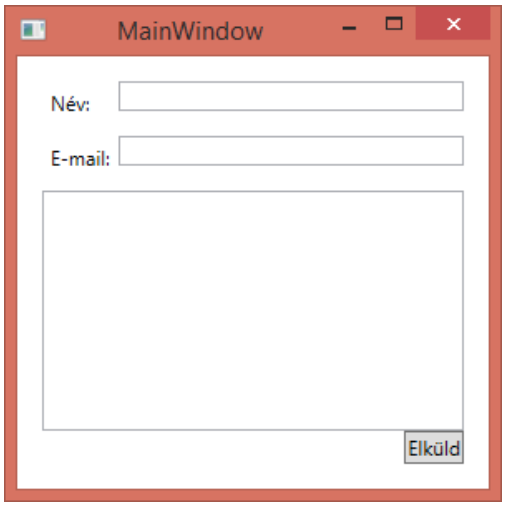
```
<DockPanel>
    <Menu DockPanel.Dock="Top">
        <MenuItem Header="_File">
            <MenuItem Header="_New" />
            <MenuItem Header="_Open" />
        </MenuItem>
        <MenuItem Header="_Edit">
            <MenuItem Header="_Undo"/>
            <MenuItem Header="_Redo"/>
        </MenuItem>
        <MenuItem Header="_View">
            <MenuItem Header="Code" Click="View_code"/>
            <MenuItem Header="Designer"/>
            <Separator />
            <MenuItem Header="Error List" IsCheckable="
                True" IsChecked="True"/>
        </MenuItem>
    </Menu>
    <TextBox AcceptsReturn="True" />
</DockPanel>
```

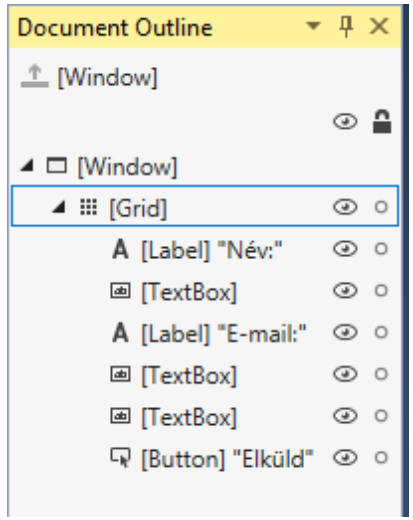


- ToolBar - eszköztárak
- StatusBar - állapotsáv

Document Outline

- felületet leíró dokumentumhierarchiát
- a logikai fa írja le az elemek közötti kapcsolatokat
  - Fontos a tulajdonságok öröklése szempontjából
  - Navigálás: GetParent, GetChildren, FindLogicalNode
- a vizuális fa írja a logikai elemek összes alkotóelemének kapcsolatát (pl. elhelyezés, áttetszőség, engedélyezettség)





Vezérlők elhelyezése és megjelenése

- A vezérlőkre különböző transzformációk alkalmazhatóak; csoportosíthatóak (TransformGroup)
  - forgatás (RotateTransform)
  - nagyítás (ScaleTransform)
  - eltolás (TranslateTransform)
  - ferdítés (SkewTransform)
- a vezérlők megjelenése számos módon testre szabható
  - a legtöbb vezérlőnél külön kezelhető a határvonal (Border/Stroke)
  - a kitöltés (Background/Fill), valamint a különböző hatások (Effect)
  - a színekhez különböző ecsetek használhatóak (pl. SolidColorBrush, LinearGradientBrush)
- a megjelenítés stílusba (Style) foglalható (később)

Színek és ecsetek

```
<Grid>
   <Grid.Resources>
       <Color x:Key=”szürke” A=”255” R=”155” G=”156” B=”159” ></Color> //saját színek
       <Color x:Key=”piros”>#FFAA2C27</Color>
       <SolidColorBrush x:Key=”szürkeKitöltes” Color=”{StaticResource szürke}”></
            SolidColorBrush>
       <SolidColorBrush x:Key=”pirosKitöltes” Color=”{StaticResource piros}”></
            SolidColorBrush>
   </Grid.Resources>
   <Button Width=”150” Height=”50” Content=”SolidColorBrush” FontSize=”18”
          Background=”{StaticResource ResourceKey=szürkeKitöltes}”
          Foreground=”{StaticResource ResourceKey=pirosKitöltes}”>
   </Button>
</Grid>
```

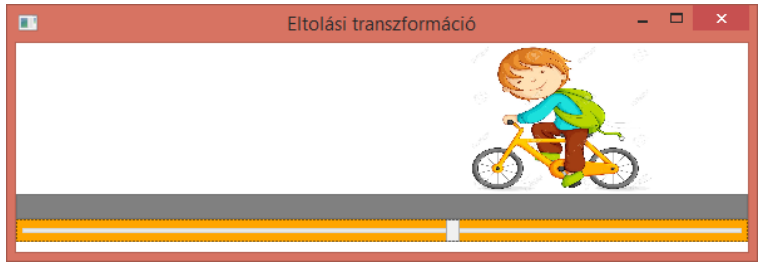
```
<LinearGradientBrush //színátmenetes kitöltést
    StartPoint=”0.5,0”  //(0,0 a bal felso sarok)
    EndPoint=”0.5,1”>  //(1,1 a jobb alsó sarok)
<GradientStop Color=”Szín1” Offset=”0.4”/>
// (0 és 1 közötti értékkel adható meg, hogy a színátmenet az átmenet tengely melyik
     pontján kezdodik)
<GradientStop Color=”Szín2” Offset=”1”/>
</LinearGradientBrush>
```

több [és komplexebb] példa: ImageBrush, DrawingBrush, VisualBrush

Transzformációk

- az összes UIelement objektum transzformálható
- System.Windows.Media.Transform
- 1. TranslateTansform (eltolási transzformáció)

```
<StackPanel>
    <Rectangle Height=”120” Width=”150” HorizontalAlignment=”Left”>
    <Rectangle.Fill>
        <ImageBrush ImageSource=”images/bicycle.jpg”/>
    </Rectangle.Fill>
    <Rectangle.RenderTransform>
        <TranslateTransform X=”{Binding ElementName=csuszka, Path=Value}”/>
    </Rectangle.RenderTransform>
    </Rectangle>
    <Button  Height=”20” Background=”Gray”/>
    <Slider x:Name=”csuszka” Minimum=”0” Maximum=”600” Background=”Orange”/>
</StackPanel>
```



Transzformációk -2

- 2. ScaleTransform (méretezési transzformáció) egy elem méretét képes megváltoztatni
- `ScaleX` X irányú megnyújtás, `CenterX` a nyújtási művelet X irányú koordinátája

```
<StackPanel>
    <Image Height=”60” Width=”60” Source=” images/bicycle.jpg” Margin=”20”>
        <Image.RenderTransform>
            <ScaleTransform CenterX=”0” CenterY=”0” ScaleX=”-3” ScaleY=”2”/>
        </Image.RenderTransform>
    </Image>
    <Image Height=”60” Width=”60” Source=” images/bicycle.jpg” Margin=”20”>
        <Image.RenderTransform>
            <ScaleTransform CenterX=”5” CenterY=”15” ScaleX=”3” ScaleY=”-4”/>
        </Image.RenderTransform>
    </Image>
</StackPanel>
```

Transzformációk -3

- 3. RotateTransform (elforgatási transzformáció)- elforgatás egy megadott szögben
- `Angle` elforgatás szöge, `CenterX` forgatás középpontja (x tengely)

```
<StackPanel>
    <Image Source="images/bicycle.jpg" Height="60" Width="70" Margin="40">
        <Image.RenderTransform>
            <RotateTransform Angle="45" CenterX="25" CenterY="-100"/>
        </Image.RenderTransform>
    </Image>
    <Image Source="images/bicycle.jpg" Height="60" Width="70" Margin="20">
        <Image.RenderTransform>
            <RotateTransform Angle="-90"/>
        </Image.RenderTransform>
    </Image>
</StackPanel>
```

több [és komplexebb] példa: SkewTransform (döntési), MatrixTransform, TransformGroup

Tanácsok

- kerüljük a fix pozíciókat - használjunk `Alignment` tulajdonságokat `Marging` -al kombinálva
- kerüljük a fix méreteket - használjunk `Auto` -t ahol csak lehet
- használjunk `StackPanel` -t párbeszéd Buttonok elrendezésére
- használjunk `Grid` -et statikus adatbeviteli formák elrendezésére. Adjunk `Auto` méretet a `Label` -nek és `*` -ot az a textbox-nak.
- `Height` és `Width` helyett használjunk inkább `MinHeight` , `MaxHeight` , `MinWidth` és `MaxWidth` tulajdonságokat