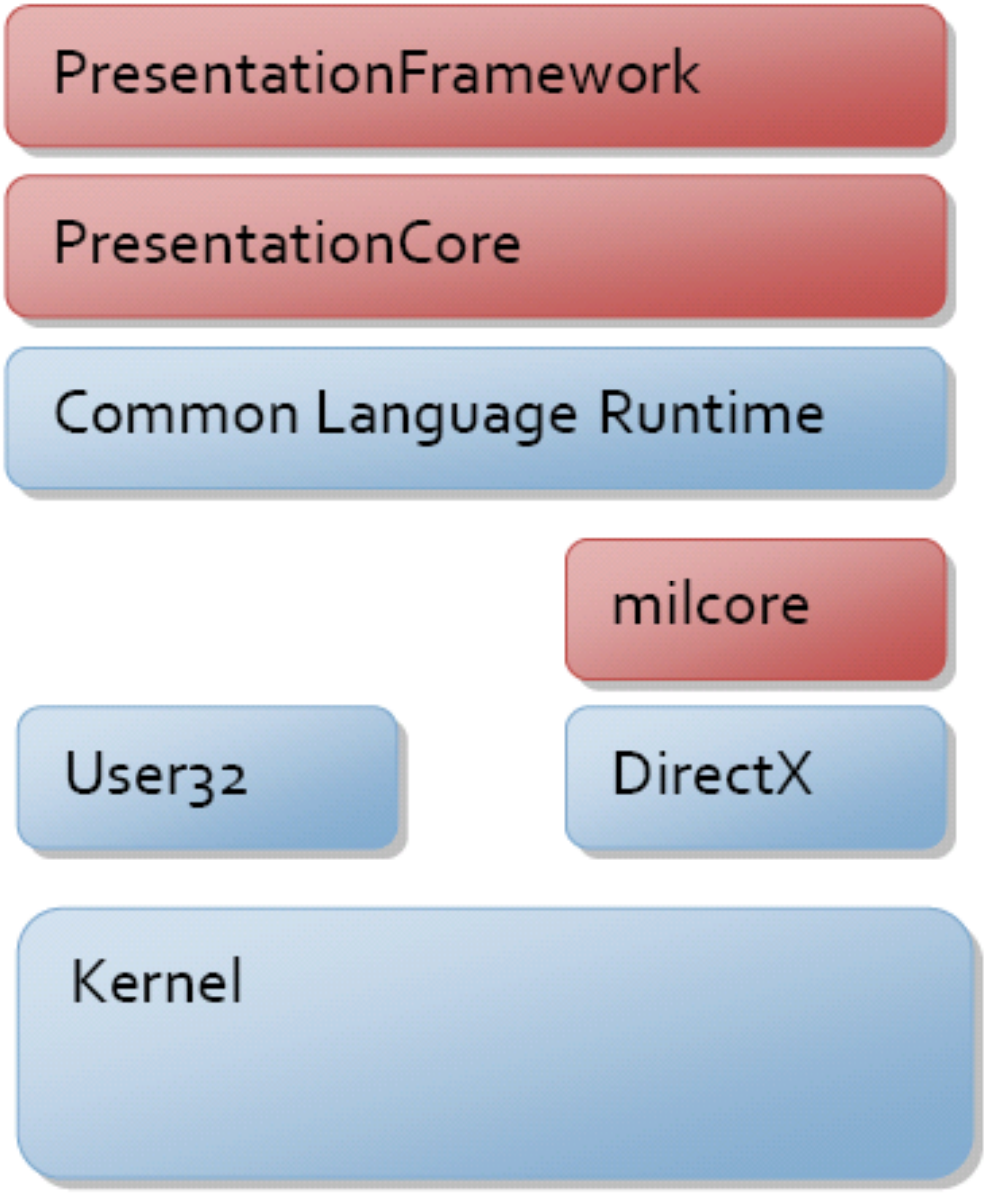
WPF (Windows Presentation Foundation)

Maja Štula

Ak. God. 2011/2012

WPF arhitektura

- Crveno označeno su dijelovi WPF-a
- milcore je neupravljivi kôd zbog povezivanja sa DirectX modulom. Sva iscrtavanja (i prozora, i kontrola i sadržaja u prozoru) idu preko DirectX-a.



WPF

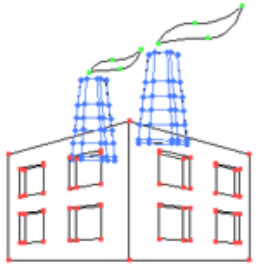
- Grafički sustav WPF-a:
  - Svi grafički elementi se oslanjaju na Direct3D, a ovo uključuje i elemente poput prozora, izbornika, …
  - Ideja je da se na ovaj način postavi jedinstven i jasan način za prikazivanje grafike. Dodatna je prednost što se time koristi GPU (*Graphics Processing Unit*) koji se nalazi na grafičkim karticama i time se CPU (*Central Processing Unit*) na računalu ostavlja više vremena za druge zadatke – dakle, rasterećuje se procesor, a grafički se zadaci dodjeljuju procesoru na grafičkoj kartici.
  - Cijeli je grafički sustav temeljen na vektorima, a to konkretno znači da će botuni, ikone i ostali UI elementi imati mnogo bolju prilagodbu na promjene dimenzija i razlučivosti nego dosadašnji elementi temeljeni na rasterima.
  - Upravo zbog boljeg skaliranja, Vista i na zaslonima visokih razlučivosti izgleda odlično, a to također možemo zahvaliti WPF-u koji se brine da se PPI (pixels per inch) na zaslonima povećava u skladu s potrebama.
  - Još je jedna važna prednost činjenice da se sustav sada temelji na vektorskoj grafici – skaliranje dokumenata je sada značajno elegantnije (to se još dodatno povezuje s XML Paper Specification svojstvom – XPS, Microsoft page description language).

Vektorska grafika ↔ Rasterska grafika

- Računalna grafika (koja se koristi na računalu i s računalom povezanim uređajima poput printera i sl.) se dijeli na dva osnovna tipa: vektorsku i rastersku grafiku.
- Rasterska slika (grafika) je podatkovna struktura koja opisuje (obično) pravokutnu mrežu piksela (***pic****ture* ***el****ement*). Ukupni broj opisanih piksela je rezolucija slike. Informacija po svakom pikselu se obično naziva dubina boje (*color depth*). Ukupni broj piksela zajedno sa informacijom po pikselu određuju kvalitetu slike. Rasterska slika se ne može skalirati (mijenjati dimenziju) bez gubitka kvalitete prikaza.
- BMP, TIF, GIF, JPG, PNG formati su tipični primjeri rasterske grafike.

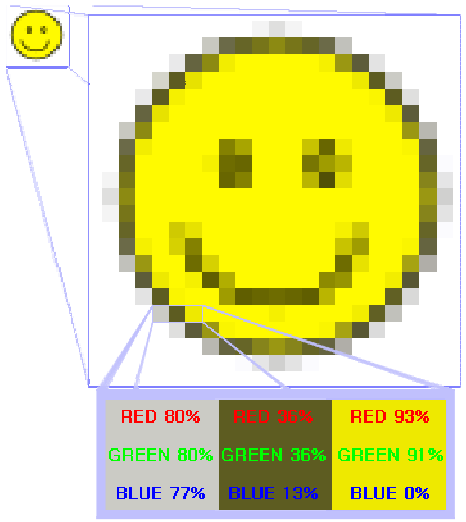
Vektorska grafika ↔ Rasterska grafika

- Vektorska grafika koristi tzv. geometrijske primitive (točke, linije, poligone,..) da bi opisala sliku. U konačnici se grafička informacija u vektorskom formatu mora pretvoriti u rastersku da bi se prikazala na ekranu.
- WMF (Windows Metafile) je Microsoft Windows format za vektorsku grafiku, dozvoljava i uključivanje rasterske grafike.
- SVG (Scalable Vector Graphics) je također vektorski format.



Vector Logo Outline

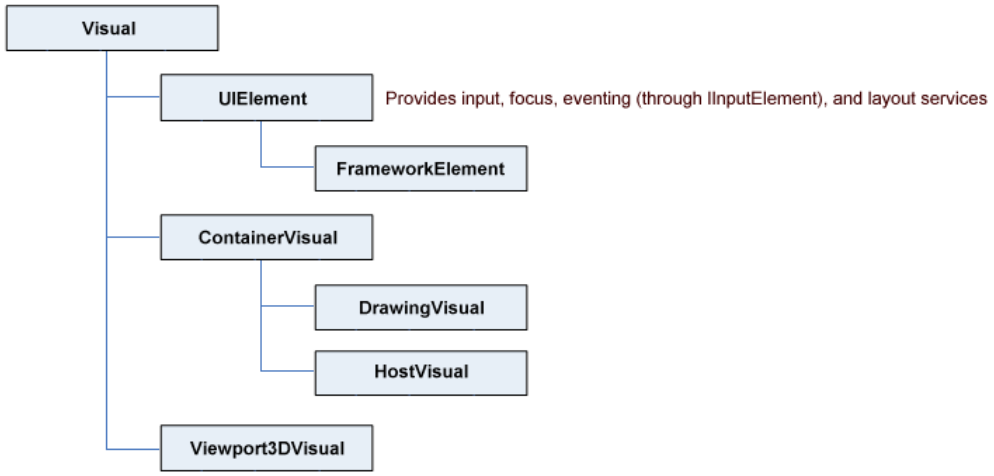
Vector Logo Filled In

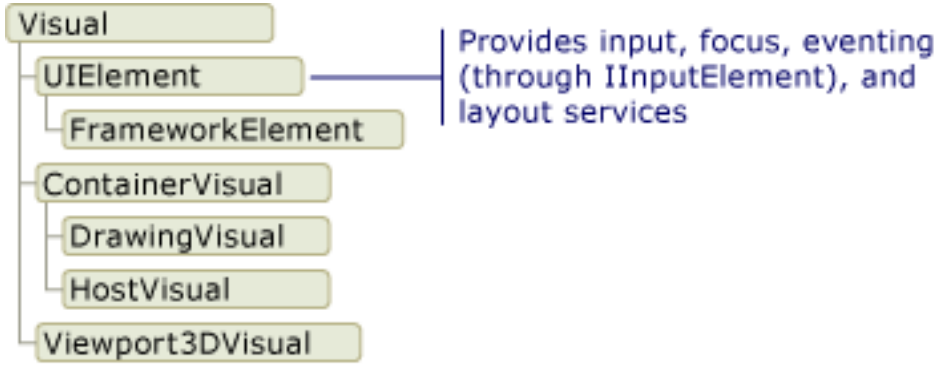


WPF

- Visual je temeljni WPF objekt, osnovna funkcionalnost je pružanje podrške za iscrtavanje (rendering support.).
- Visual klasa pruža podršku za:

1. Output display: Iscrtavanje vizualnog sadržaja. (Rendering the persisted, serialized drawing content of a visual.)
2. Transformations: Izvođenje transformacija na vizualnom sadržaju. (Performing transformations on a visual.)
3. Clipping: Podrška za isijecanje područja vizualnog sadržaja. (Providing clipping region support for a visual.)
4. Hit testing: Određivanje da li su koordinate ili određena geometrija sadržani u granicama vidljivog. (Determining whether a coordinate or geometry is contained within the bounds of a visual.)
5. Bounding box calculations: Određivanje pravokutnika koji omeđuje vidljivo područje. (Determining the bounding rectangle of a visual.)





Logičko i vizualizacijsko stablo

- Poredak elemenata sučelja zove se logičko stablo. Stablo vizualno prikazuje hijerarhiju svih elemenata sučelja i u biti samo stvara sliku onoga sto smo napisali u XAML kodu.
- Vizualizacijsko stablo je proširena inačica logičkog stabla. Ono razdvaja elemente u manje dijelove. Drugim riječima, umjesto da vidimo "crne kutije" kao npr. *Button* kontrola, vidjeti ćemo vizualne komponente te kontrole, kao što je obrub kontrole.
- Vizualizacijsko stablo je akumulacija svih vizualnih elemenata kreiranih direktno u kôdu ili u markup XAML kôdu. Također sadrži i vizualne elemente kreirane definiranjem novih predložaka elemenata.
- Vizualizacijsko stablo (*visual tree*) sadrži sve vizualne elemente iz grafičkog sučelja aplikacije.
- Vizualni elementi imaju informacije o vlastitom iscrtavanju, a vizualizacijsko stablo se može promatrati kao grafička scena koja sadrži informacije o iscrtavanju koje su potrebne da bi se svi vizualni elementi složili u jedinstveni prikaz na uređaju na kojem se vrši iscrtavanje.

Hijerarhija iscrtavanja vizualnih elemenata u WPF-u

- Vizualizacijsko stablo određuje redoslijed iscrtavanja WPF vizualnih objekata. Redoslijed počinje od početnog root čvora, a iscrtavanje child čvorova ide od lijeva prema desno.
- Ako jedan čvor ima djecu najprije se prelazi preko djece tog čvora, a tek se onda ide na čvorove susjede (*sibling*) na istoj hijerarhijskoj razini. Sadržaj vizualnog elementa čvora djeteta se uvijek iscrtava iznad sadržaja čvora roditelja.
- Početni root čvora je najviši element u hijerarhiji vizualizacijskog stabla. To je najčešće Window ili NavigationWindows. Ako je root element XAML datoteke Page i onda će root čvor biti Window jer se Page objekt mora smjestiti na prozor tj. ne može se iscrtati samostalno.

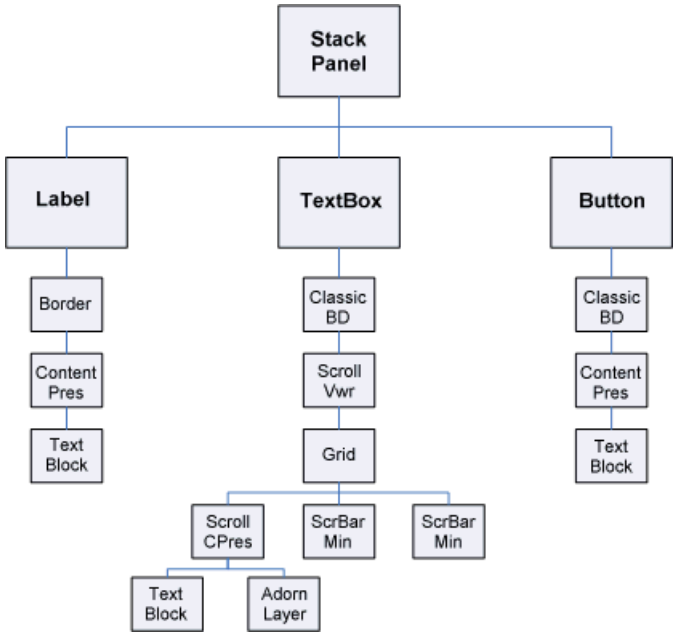
Hijerarhija iscrtavanja vizualnih elemenata u WPF-u

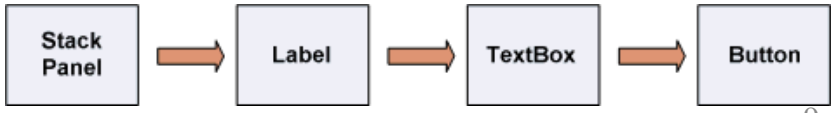
XAML kod

```
<StackPanel>
 <Label>User name:
</Label>
<TextBox />
<Button Click="OnClick">OK
</Button>
</StackPanel>
```

Primjer

Hijerarhijski dijagram vizualnog stabla





Dodavanje elementa u stablo

- Ellipse nova = new Ellipse();
- nova.Width = 100;
- nova.Height = 200;
- nova.Stroke = Brushes.Black;
- nova.HorizontalAlignment = HorizontalAlignment.Left;
- nova.VerticalAlignment = VerticalAlignment.Top;
- podloga.Children.Add(nova);

Kod za dohvaćanje logičkog i vizualizacijskog stabla

```
private void Window_Loaded(object sender,
        RoutedEventArgs e)
{
TreeView treeViewMain = new TreeView();
TreeViewItem tviVisualTree = new
        TreeViewItem();
 tviVisualTree.Header = "Visual Element Tree";
 tviVisualTree.IsExpanded = true;
 treeViewMain.Items.Add(tviVisualTree);
 this.CreateVisualTree(this, tviVisualTree);
 TreeViewItem tviLogicalTree = new
        TreeViewItem();
tviLogicalTree.Header = "Logical Element Tree";
tviLogicalTree.IsExpanded = true;
treeViewMain.Items.Add(tviLogicalTree);
this.CreateLogicalTree(this, tviLogicalTree);
this.stackPanel.Children.Add(treeViewMain);
 }
```

```
private void CreateVisualTree(DependencyObject dependencyObject, TreeViewItem
        tviVisualTree)
{
if (dependencyObject is FrameworkElement)
elementName = ((FrameworkElement)dependencyObject).Name;
TreeViewItem tviNewElement = new TreeViewItem();
tviNewElement.Header = dependencyObject.GetType().ToString() + ": " +
        elementName;
tviVisualTree.Items.Add(tviNewElement);
tviNewElement.IsExpanded = true;
for (int treeElem = 0; treeElem <
        VisualTreeHelper.GetChildrenCount(dependencyObject); treeElem++)
this.CreateVisualTree(VisualTreeHelper.GetChild(dependencyObject, treeElem),
        tviNewElement);
}
private void CreateLogicalTree(FrameworkElement dependencyObject,
        TreeViewItem tviLogicalTree)
{
 TreeViewItem tviNewElement = new TreeViewItem();
 tviNewElement.Header = dependencyObject.GetType().ToString() + ": " +
        dependencyObject.Name;
 tviLogicalTree.Items.Add(tviNewElement);
 tviNewElement.IsExpanded = true;
 IEnumerable children = LogicalTreeHelper.GetChildren(dependencyObject);
 foreach (Object currentChild in children)
 {
 if (currentChild is FrameworkElement)
 this.CreateLogicalTree((FrameworkElement)currentChild, tviNewElement);
   }
 }
```

Primjer

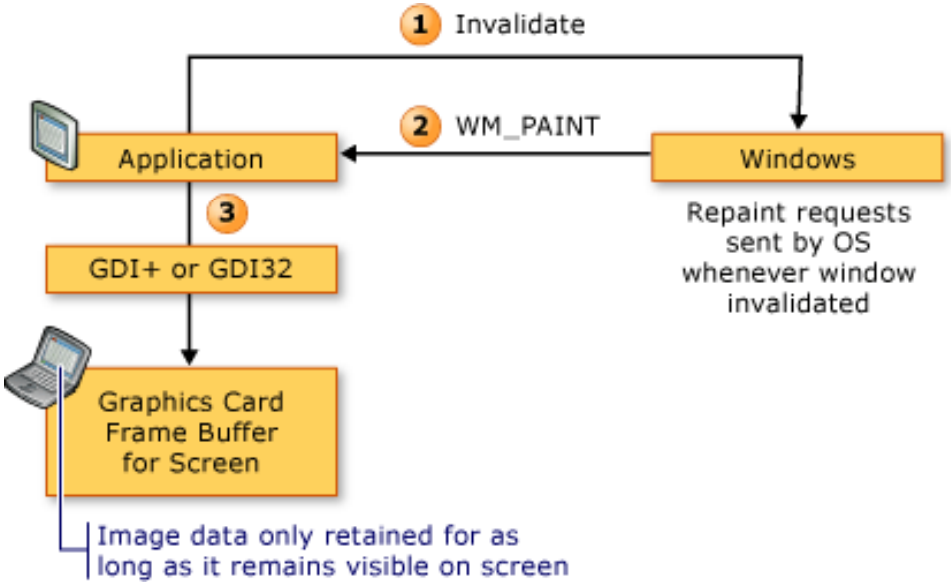
Logičko i vizualizacijsko stablo

- Svi elementi sa vizualnim sučeljem nasljeđuju klasu DependencyObject koja predstavlja objekt koji učestvuje u *dependency* svojstvima.
- VisualTreeHelper klasa sadrži metode kojima se mogu obavljati razne operacije na čvorovima u vizualizacijskom stablu.
- VisualTreeHelper.GetChild vraća vizualizacijski objekt dijete uz parametar indeksa pozicije.

WPF

- Iscrtavanje WPF vizualnih objekata definirano je WPF svojstvima: tzv. sačuvani prikaz (retained mode graphics), vektorska grafika i grafika neovisna o uređaju (device independent graphics).
- Aplikacije temeljene na GDI ili GDI+ koriste tzv. izravan prikaz (immediate mode graphics system). Aplikacija je sama zadužena za iscrtavanje klijentskog područja prozora koje se promijenilo, bilo npr. *resajzanjem* prozora, ili na neki drugi način.

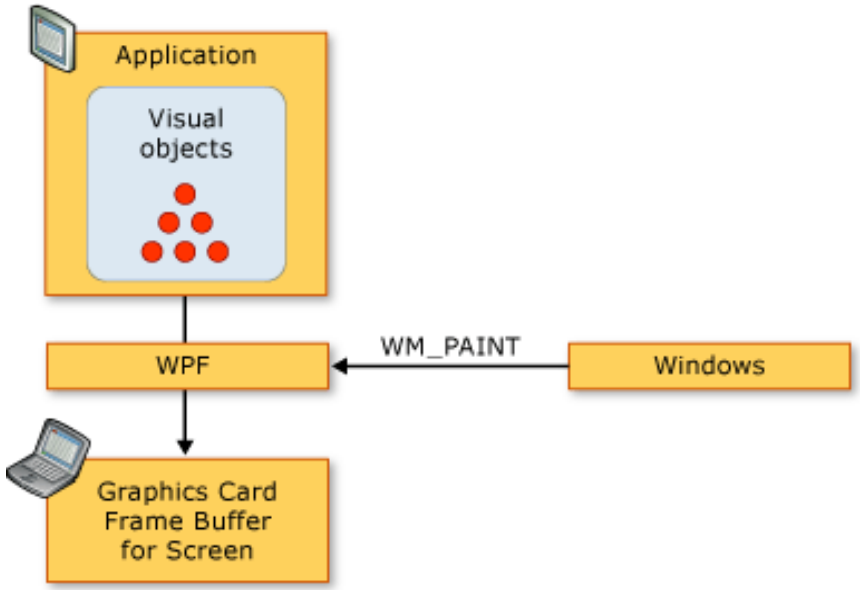
Aplikacije temeljene na GDI prikazu



WPF

- WPF koristi sačuvani prikaz.
- Objekti sa vizualnim prikazom definiraju skup serijaliziranih podataka za iscrtavanje.
- Kada su definirani podaci za iscrtavanje WPF sustav je zadužen da iscrtava svaki put kada se pokaže potreba za iscrtavanjem na temelju serijaliziranih podataka za iscrtavanje za svaki vizualni element.
- Prilikom izvođenja mogu se kreirati i novi vizualni elementi ili uništavati, a sustav ih treba ispravno iscrtavati.
- Znači podaci za iscrtavanje su trajno prisutni u aplikaciji, a WPF sustav je zadužen za ispravno iscrtavanje.

Aplikacije temeljene na WPF prikazu



WPF

- Veličina prikaza teksta ili grafike na ekranu je određena kroz rezoluciju i DPI.
- Rezolucija opisuj broj piksela koji se prikazuju na ekranu. Povećanjem rezolucije smanjuje se veličina piksela, čime se smanjuje i veličina teksta ili grafike. Npr. prikaz na rezoluciji 1024 x 768 će izgledati veći nego na 1600 x 1200.
- Druga sistemska postavaka DPI (dots per inch) opisuje veličinu uređaja za prikaz izraženu u točkama (koje ne moraju odgovarati pikselu) po inču. Najčešće se koristi za printere. Većina Windows-a ima DPI 96, što znači da po inču ima 96 točaka. Povećavanje DPI-a povećava screen inč; smanjenje DPI-a smanjuje screen inč. To znači da screen inč ne mora odgovarati stvarnom inču (2,5 cm).
- Sve aplikacije ni ne koriste DPI već se oslanjaju primarno na piksele. Neke aplikacije koriste DPI za opis veličine fonta, a za sve ostalo piksele.
- WPF podržava automatsko skaliranje korištenjem piksela neovisnih o uređaju (device independent pixel) kao mjernu jedinicu, a ne uobičajeno korištenih hardverskih piksela.

72 DPI

24pt Text

200 x 25 Rectangle

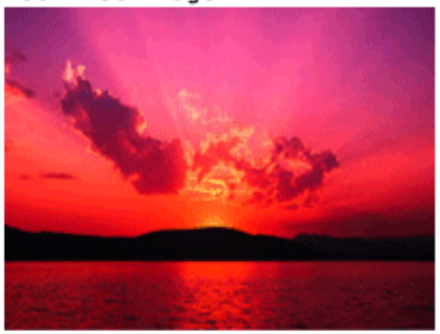
200 x 150 Image

96 DPI

24pt Text

200 x 25 Rectangle

200 x 150 Image



120 DPI

24pt Text

200 x 150 Image

200 x 25 Rectangle

Dinamičko iscrtavanje sadržaja u vizualnim objektima

- Klasa ContainerVisual se koristi kao kontejner za kolekciju vizualnih objekata.
- Klasa DrawingVisual je izvedena iz klase ContainerVisual pa može sadržavati kolekciju vizualnih objekata.
- Klasa DrawingVisual je jednostavna klasa za iscrtavanje geometrijskih oblika, slika ili teksta. Jednostavna je jer ne podržava poravnanje ili obradu događaja, ali je brza i stoga je idealna za iscrtavanja pozadine.
- Klasa Viewport3DVisual je temeljna klasa za 3D vizualne elemente. Zahtjeva definiranje Camera vrijednosti i ViewPort vrijednosti. Camera je pogled na scenu, ViewPort određuje gdje se projekcija mapira na 2D površinu.

Dinamičko iscrtavanje sadržaja u vizualnim objektima

- Visual objekt pohranjuje podatke o iscrtavanju kao instrukcijsku listu vektorske grafike. Svaki element u listi su grafički podaci niske razine i povezani resursi u serijaliziranom obliku. Četiri različita tipa grafičkih podataka mogu sadržavati grafiku.

Drawing content type	Description
Vector graphics	Represents vector graphics data, and any associated Brush and Pen information.
Image	Represents an image within a region defined by a Rect.
Glyph	Represents a drawing that renders a GlyphRun, which is a sequence of glyphs from a specified font resource. This is how text is represented.
Video	Represents a drawing that renders video.

DrawingContext u Visual objektu omogućava popunjavanje vizualnog sadržaja. Kada se koriste naredbe za crtanje na DrawingContext-u ustvari se postavljaju grafički podaci koji će se kasnije koristiti za iscrtavanje, tj. ne crta se u real-time vremenu. Kada se kreira kontrola npr. Button, kontrola implicitno generira grafičke podatke za vlastito iscrtavanje.

Dinamičko iscrtavanje sadržaja u vizualnim objektima

```
using System.Windows.Media;

// Create a DrawingVisual that contains an ellipse.
private DrawingVisual CreateDrawingVisualEllipses()
{
    DrawingVisual drawingVisual = new DrawingVisual();
    DrawingContext drawingContext = drawingVisual.RenderOpen();
    drawingContext.DrawEllipse(Brushes.Maroon, null, new Point(430, 136),
    20, 20);
    drawingContext.Close();
     return drawingVisual;
}
```

- DrawingContext se ne instancira direktno, već se dohvaća preko DrawingVisual.RenderOpen metode ili DrawingGroup.Open metode.

Dinamičko iscrtavanje sadržaja u vizualnim objektima

- DrawingVisual objekt se prikazuje unutar host kontejnera - neka klasa izvedena iz klase FrameworkElement koja pruža DrawingVisual objektu funkcionalnost poravnanja i ulaznih događaja koje ovaj nema.
- Kako bi se DrawingVisual uključio o objekt izvedene iz klase FrameworkElement, potrebno je održavati VisualCollection član koji uključuje sve DrawingVisual objekte za promatrani element. Također je potrebno overrideati svojstvo VisualChildrenCount i metodu GetVisualChild.

```
public class MyVisualHost : FrameworkElement
{
      // Create a collection of child visual objects.
      private VisualCollection _children;
      public MyVisualHost()
      {
         _children = new VisualCollection(this);
       }
       protected override int VisualChildrenCount
      {
         get { return _children.Count; }
      }
       protected override Visual GetVisualChild(int index)
      {
         if (index < 0 || index >= _children.Count)
         {
            throw new ArgumentOutOfRangeException();
         }
         return _children[index];
      }
}
```

PRIMJER: