C# 2 – Les 8 – Shapes, Canvas en Animation Loop

In deze les maken we een *animation loop*. Een animation loop is een methode die meerdere keren per seconden vormen verplaatst om zo een animatie te creëren. Idealiter wordt de methode 60 keer per seconden uitgevoerd om zo een vloeiende animatie van 60 frames per seconden te verkrijgen.

Ellipse op Canvas plaatsen

Vormen zoals een Ellipse of Rectangle kan je in een Grid of StackPanel plaatsen, maar we gaan in dit document een ander soort paneel voor gebruiken: het **Canvas**-element. Dit element is een wat "dommer" layout-element waar je objecten op exacte pixelwaarden kan plaatsen van de bovenkant en linkerkant. Het Canvas-element behoort, net zoals het Grid- en StackPanel-element tot de Panels. Hier vind je een overzicht van al de Panels: https://docs.microsoft.com/en-us/dotnet/api/system.windows.controls.panel?view=net-5.0

In onze XAML plaatsen we in ons Window-element een Canvas-element en geven het een zelfgekozen naam ("cnvs"):

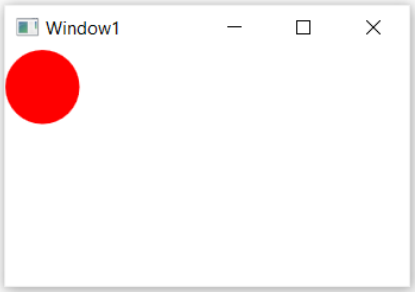
```
<Canvas x:Name="cnvs"></Canvas>
```

In de *code-behind* maken we dynamisch een Ellipse-object aan en voegen dat toe aan ons Canvas. De cnvs.Add.Children(…) regel levert hetzelfde op alsof je een Ellipse in XAML zou plaatsen binnen de <Canvas>…</Canvas>.

```
public partial class Window1 : Window
{
    public Ellipse bol { get; set; }
    public Window1()
    {
        InitializeComponent();

        bol = new Ellipse();
        bol.Width = 50;
        bol.Height = 50;
        bol.Fill = (SolidColorBrush)new BrushConverter()
                                    .ConvertFromString("#ff0000");
        cnvs.Children.Add(bol);
    }
}
```

Als alles goed gaat heb je nu een mooie ronde bol op je scherm:



De bol animeren

We maken gebruik van een speciaal event om de animatie aan te sturen: *CompositionTarget.Rendering*. Deze event is speciaal gemaakt voor animaties in WPF. We koppelen een methode aan deze event, onze *animation loop.* In het voorbeeld hieronder noemen we de methode *Loop*. (Andere vaak gebruikte namen die je kan tegenkomen zijn *update*, *tick* of *draw*.) *Canvas.SetLeft* hebben we nodig om de bol te verplaatsen. De horizontale positie houden we bij in een property *X*. De code tot dusver:

```
public partial class Window1 : Window
{
    public Ellipse bol { get; set; }
    public int X { get; set; } = 0;

    public Window1()
    {
        InitializeComponent();

        bol = new Ellipse();
        bol.Width = 50;
        bol.Height = 50;
        bol.Fill = (SolidColorBrush)new BrushConverter()
                                    .ConvertFromString("#ff0000");
        cnvs.Children.Add(bol);

        CompositionTarget.Rendering += Loop;
    }

    private void Loop(object sender, EventArgs e)
    {
        Canvas.SetLeft(bol, X++);
    }
}
```

Meer info over CompositionTarget.Rendering vind je hier:

https://docs.microsoft.com/en-us/dotnet/desktop/wpf/graphics-multimedia/how-to-render-on-a-per-frame-interval-using-compositiontarget?view=netframeworkdesktop-4.8

Wrap around

Onze bol verdwijnt na enige tijd rechts uit beeld. Je kan testen of de bol uit beeld verdwijnt en als dat het geval is deze terug aan de linkerkant plaatsen. Zo bekom je een *wrap around*. Dit werd al gebruikt in de eerste computerspelletjes.

De code in de Loop-methode kan je veranderen in deze code:

```
X++; // verhoog X

if(X > cnvs.ActualWidth)
{
  // als X groter is dan de breedte van cnvs (bol verwijnt uit beeld)
  // verzet dan X naar links buiten beeld:
   X = -(int)bol.Width;
}

Canvas.SetLeft(bol, X);
```

De Ellipse shape in een object verpakken

Telkens Canvas.SetLeft(…) typen is wat vermoeiend. Het zou leuk zijn als we onze cirkel kunnen verplaatsen door bol.Move() te typen. Als we onze Ellipse in een object stoppen (= composition) en we geven dit “omwikkelend” object X en Y velden kunnen we zoiets verkrijgen. We maken een Circle object met daarin de Ellipse verpakt. Om gemakkelijk aan de parent van de Circle te komen houden we ook het Canvas-element bij waaraan de Ellipse is toegevoegd.

```
public class Circle
{
    public double X { get; set; }
    public double Y { get; set; }
    public string Color { get; set; }
    private Ellipse Ellps { get; set; }
    private Canvas Cnvs;

    public Circle(double x, double y, string color)
    {
        Ellps = new Ellipse();
```

```
        Ellps.Width = 50;
        Ellps.Height = 50;
        X = x;
        Y = y;
        Ellps.Fill = (SolidColorBrush)new BrushConverter().ConvertFromString(
color);
    }
    public void Move()
    {
        X += 2;
        Canvas.SetLeft(Ellps, X);
    }

    public void AddToCanvas(Canvas c)
    {
        c.Children.Add(Ellps);
        Cnvs = c;
    }
}
```

In de aanroepende class heb je nu geen X property meer nodig, deze zit ook verpakt in de Circle. Je kan een Circle zo gebruiken:

```
public partial class MainWindow : Window
{
    public Circle C { get; set; }
    public CircleGame()
    {
        InitializeComponent();

        C = new Circle(200, 200, "#000000");
        C.AddToCanvas(cnvs);
        CompositionTarget.Rendering += Loop;
    }

    private void Loop(object sender, EventArgs e)
    {
        C.Move();
    }
}
```

Oefeningen

1. Uitbreiding Constructors

Maak 2 extra *convienience* constructors (handige constructors die je wat tipwerk kunnen besparen).

Circle(double x, double y). Deze creëert een zwarte cirkel op locatie x, y.

Circle(). Deze creëert een zwarte cirkel met willekeurige X en Y waarden.

Circle(string color). Deze creëert een gekleurde cirkel met willekeurige X en Y waarden.

Als je de constructors hebt toegevoegd bekijk je code dan eens. Zie je dubbele code staan? Kan je deze afsplitsen in een extra methode zodat je minder herhaling in je code hebt?

2. Wrap Around

Voeg de “wrap around” code toe zodat de Circle terug naar links springt als hij buiten beeld verdwijnt.

3. Y-beweging

Kan je ervoor zorgen dat er ook een Y-beweging plaatsvindt? Ook met wrap around natuurlijk 😊.

4. Snelheid als veld/property

Als we verschillende cirkels zouden aanmaken gaan ze allemaal even snel… Het zou beter zijn om de Circle objecten nog 2 extra velden te geven: speedX en speedY. Breidt het Circle object verder uit met deze velden, je kan de cirkels bij instantiatie (in de constructor dus) willekeurige snelheden laten genereren. Kan je 3 cirkels tevoorschijn toveren met verschillende snelheden?

Kan je de snelheden ook negatief maken zodat een Circle ook naar links of naar boven kan bewegen?

5. Een List van Circles?

Kan je een List van 20 Circles maken en deze laten bewegen?

Kan je de Circles klikbaar maken? En een teller tonen met het aantal resterende cirkels? Hoe zou je hier een spelletje van kunnen maken met een score?