



Odissee
DE CO-HOGESCHOOL

Workshop – Flappy Bird



Jens Baetens



Wie ben ik?

- ▣ Burgerlijk ingenieur computerwetenschappen
- ▣ Gedoctoreerd in de computerwetenschappen
- ▣ Web developer, Consultant, C++ developer
- ▣ Nu ongeveer een jaar docent bij Odisee
 - ▣ Keuzetraject AI- en Data-Engineer



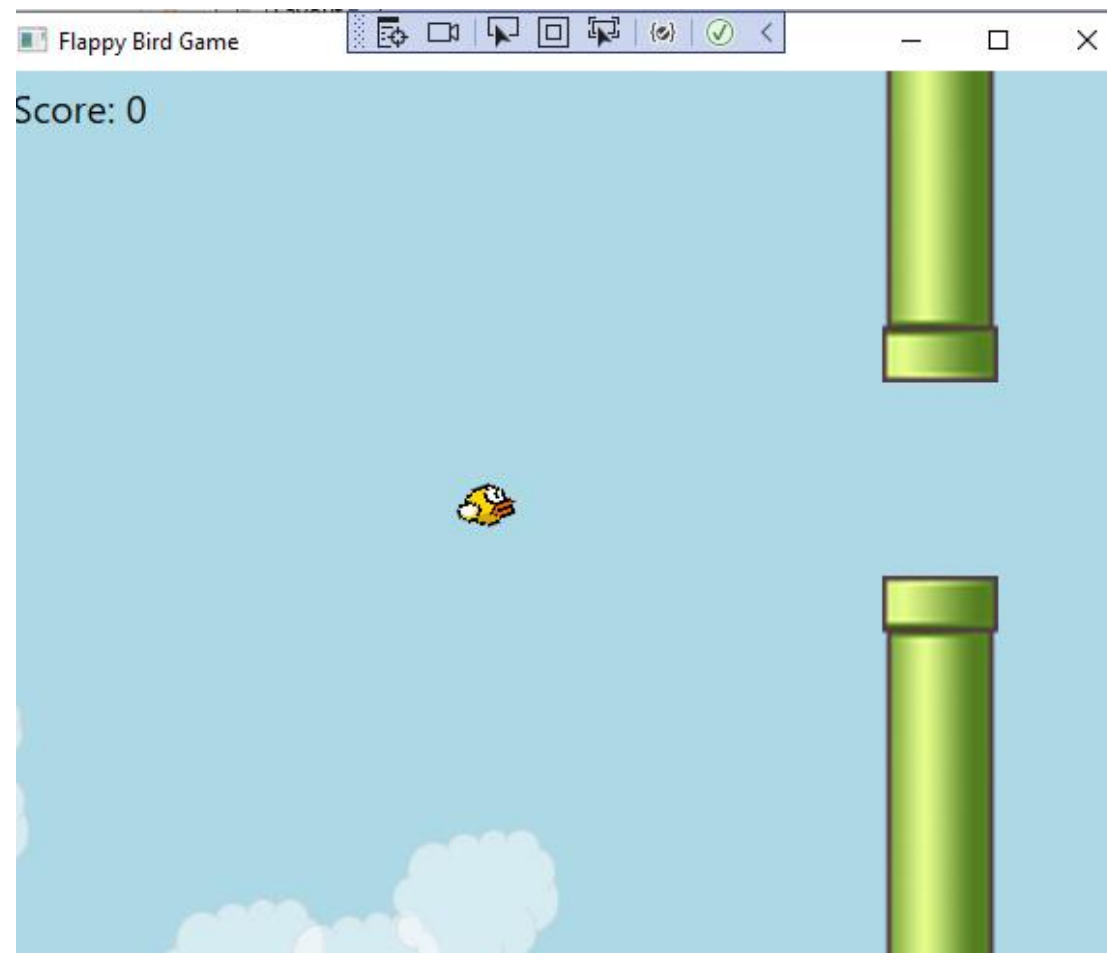
Flappy Bird



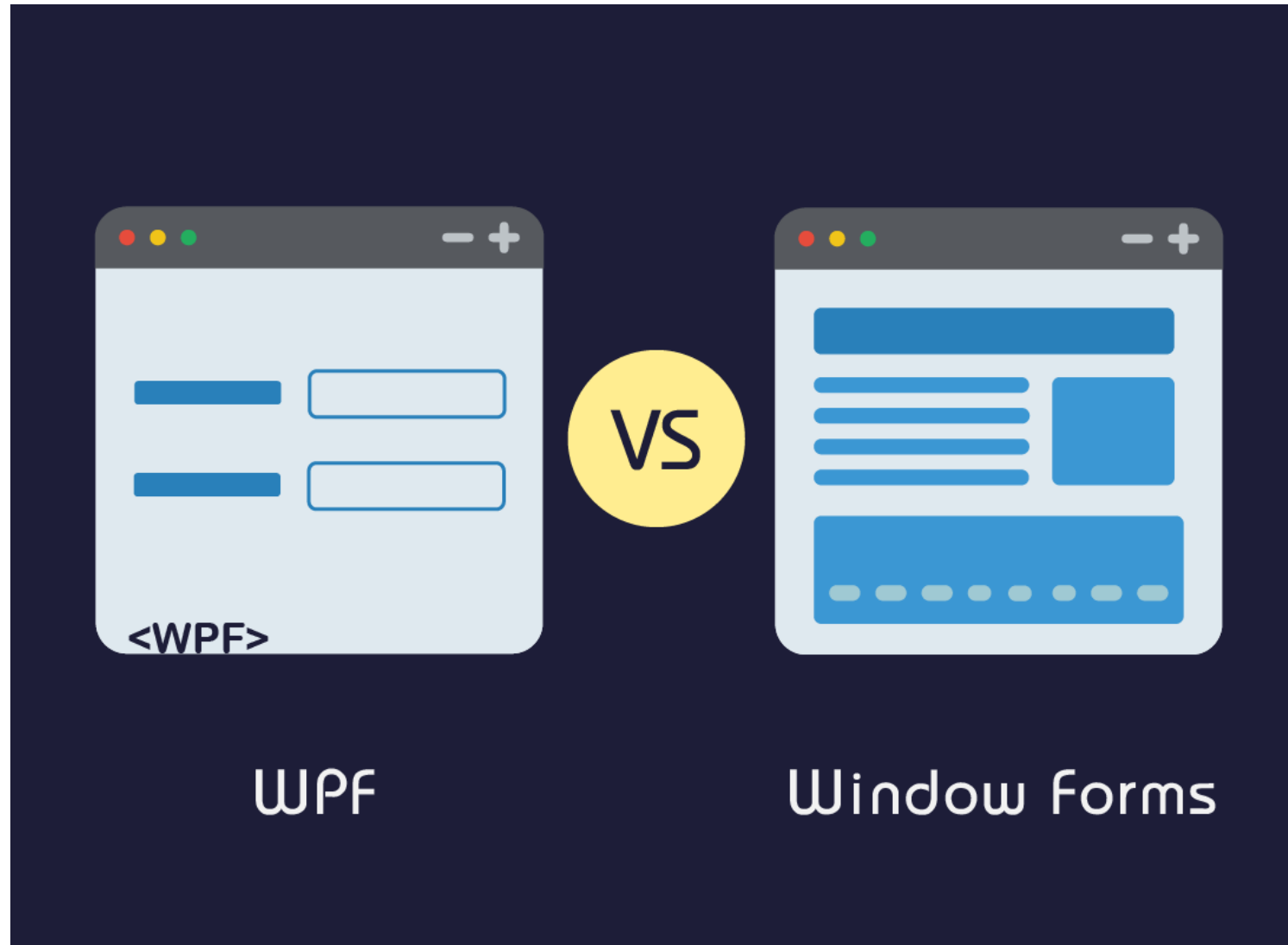
Wat zijn de verschillen tussen een videogame en een gewone applicatie?



Demo



Aanmaken GUI





Waarom WPF?

- ▣ Nieuwer en volgt beter de nieuwe standaarden
- ▣ Visual Studio gebruikt het ook
- ▣ Flexibeler
- ▣ Eenvoudiger om GUI en code te splitsen
- ▣ Performanter/sneller



Onderdelen Flappy Bird code

▣ De GUI

- Mainwindow.xaml
- Bevat de grafische elementen (view)

▣ De code-behind

- Mainwindow.xaml.cs
- Bevat code voor de grafische elementen aan te passen

▣ De logica

- Gamestate.cs
- De focus van de workshop



1.

De GUI

De GUI

■ Open de MainWindow.xaml

■ Window

- Titel
- Dimensies venster

■ Canvas

- Plaats elementen op basis van x,y coördinaten

■ Image

- Element voor het tonen van een figuur

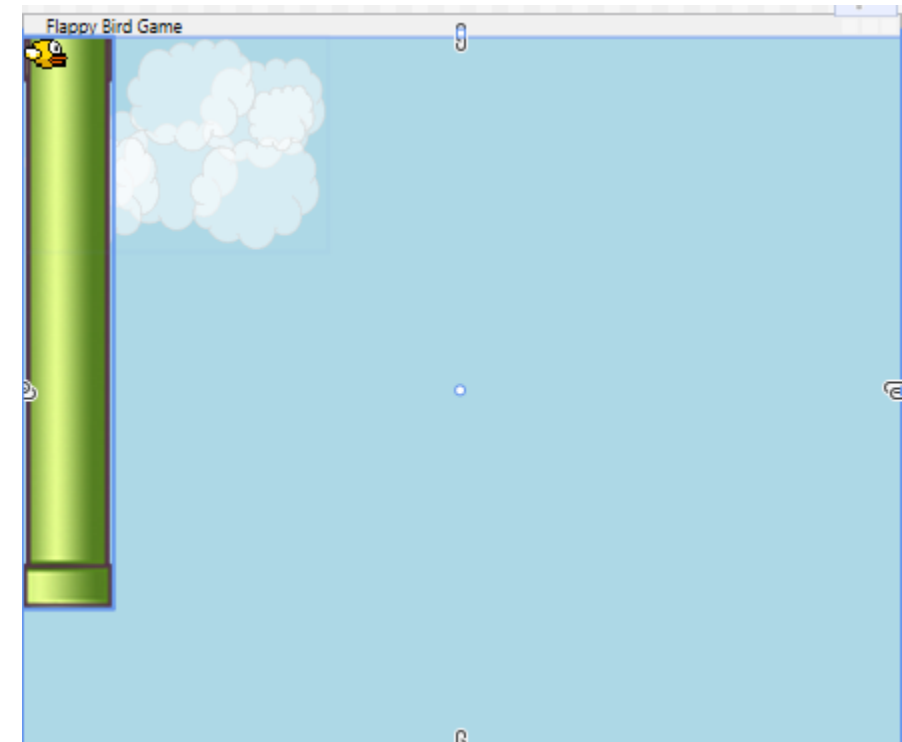
■ Staat reeds klaar

```
Window
<Window x:Class="FlappyBird.MainWindow"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
  xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
  xmlns:local="clr-namespace:FlappyBird"
  mc:Ignorable="d"
  Title="Flappy Bird Game" Height="500" Width="600"
  FocusManager.FocusedElement="{Binding ElementName=GameWindow}"
>
  <Canvas Name="GameWindow" Focusable="True" Background="LightBlue">
    ...
  </Canvas>
</Window>
```

Let op de "Name" tag

- Kan in de code-behind file gebruikt worden om de locatie aan te passen.

```
>  
<Canvas Name="GameWindow" Focusable="True" Background="LightBlue">  
  
  <Image Height="145" Width="207" Source="images/clouds.png" Name="clouds1"/>  
  <Image Height="145" Width="207" Source="images/clouds2.png" Name="clouds2"/>  
  
  <Image Height="389" Width="60" Stretch="Fill" Source="images/pipeBottom.png" Name="obst1_bottom"/>  
  <Image Height="389" Width="60" Stretch="Fill" Source="images/pipeTop.png" Name="obst1_top"/>  
  
  <Image Height="389" Width="60" Stretch="Fill" Source="images/pipeBottom.png" Name="obst2_bottom"/>  
  <Image Height="389" Width="60" Stretch="Fill" Source="images/pipeTop.png" Name="obst2_top"/>  
  
  <Image Height="389" Width="60" Stretch="Fill" Source="images/pipeBottom.png" Name="obst3_bottom"/>  
  <Image Height="389" Width="60" Stretch="Fill" Source="images/pipeTop.png" Name="obst3_top"/>  
  
  <Label Name="scoreText" FontSize="20" />  
  
  <Image Stretch="Fill" Name="flappyBird" Height="21" Width="30" Source="images/flappyBird.png" />  
  
</Canvas>
```





2.

De code-behind



MainWindow.xaml.cs

- ▣ Bij elke xaml file hoort een code-file
 - Deze wordt de code-behind genoemd en heeft als extensie .xaml.cs
- ▣ Implementeert een klasse Window
- ▣ Kan de GUI aansturen en de elementen bevragen/aanpassen

MainWindow.xaml.cs

- Reeds voor jullie gecodeerd
- 2 grote onderdelen:
 - Reageer op inputs van het toetsenbord

0 references

```
private void Canvas_KeyIsUp(object sender, KeyEventArgs e)
{
    state.SpaceReleased();
}
```

0 references

```
private void Canvas_KeyIsDown(object sender, KeyEventArgs e)
{
    // if the space key is pressed
    if (e.Key == Key.Space)
    {
        state.SpacePressed();
    }
    if (e.Key == Key.R)
    {
        // if the r key is pressed AND game over boolean is se
        // run the start game function
        state.StartGame();
    }
}
```

MainWindow.xaml.cs

- ▣ Reeds voor jullie gecodeerd
- ▣ 2 grote onderdelen:
 - Reageer op inputs van het toetsenbord
 - Pas de locatie van de figuren aan

```
2 references
public void SetObject(string name, double x_center, double y_center, double rotation)
{
    foreach (var img in GameWindow.Children.OfType<Image>())
    {
        if (img.Name == name)
        {
            img.RenderTransform = new RotateTransform(rotation, img.Width / 2, img.Height / 2);
            Canvas.SetLeft(img, x_center - img.Width / 2);
            Canvas.SetTop(img, y_center - img.Height / 2);
        }
    }
}
```


3. Gamestate

GameState.cs

- ▣ Deze functionaliteit is reeds gedeeltelijk geïmplementeerd
 - Constructor
 - Collision detectie
- ▣ De game-loop is nog te implementeren
 - Deze moet een aantal keer per seconde getriggerd worden om animatie te voorzien
 - Instellen van een begin-state in `StartGame()`
 - Het uitvoeren van een update in `UpdateFrame()`
 - Het `verwerken van inputs` van de gebruiker



Stap 1: Beginscherm

- ▣ Kies in `StartGame()` een beginpositie voor flappy-bird en de buizen
 - Vul deze coördinaten in in de overeenkomstige variabelen
 - Roep de `UpdatePositions()` functie op
- ▣ Update de posities van de images via de `UpdatePositions()` functie
 - Gebruik hiervoor de `SetObstacle()` en `SetObject()` functies van de variabele “view”

```
2 references
public void StartGame()
{
    ...
}

0 references
private void UpdatePositions()
{
    ...
}
```

Stap 2: Animatie toevoegen

- Gebruik de `UpdateFrame()` functie om de posities up te daten van de flappy-bird en obstakel figuren
 - ▬ Deze functie wordt elke 20ms opgeroepen
 - ▬ Beweeg de buizen naar links
 - doe dit aan de hand van de speed variabele, stel deze zelf in (bvb 5)
 - ▬ Beweeg flappy naar onder
 - Gebruik hiervoor de gravity variabele, kies hier ook zelf een waarde voor (bvb 8)

```
1 reference
private void UpdateFrame(object sender, EventArgs e)
{
    ...
}
```

Stap 3: Laat Flappy ook omhoog gaan

■ Implementeer de functies

■ SpacePressed()

- Zorg ervoor dat FlappyBird omhoog gaat, gebruik hiervoor de gravity parameter

■ SpaceReleased()

- Zorg ervoor dat FlappyBird terug valt, gebruik hiervoor de gravity parameter

```
1 reference  
public void SpacePressed()  
{  
    ...  
}
```

```
1 reference  
public void SpaceReleased()  
{  
    ...  
}
```

Stap 4: Objecten komen aan de rand van het scherm

- Wanneer Flappy-bird de rand van het scherm raakt is het gameover
 - ▬ Voeg deze controle toe aan `UpdateFrame()`
 - ▬ Zorg ervoor dat animatie stopt in de `SetGameOver()` functie
- Wanneer de buizen de linkerkant van het scherm raken
 - ▬ Plaats de buizen opnieuw rechts zodat er steeds nieuwe buizen komen
 - ▬ Doe dit in de `UpdateFrame()` functie
 - ▬ Verhoog ook de score met 1 elke keer dat dit gebeurt

Stap 5: Toon de score + restart na gameover

- Update de tekst in het label om steeds de huidige code te tonen
 - ▬ Pas ook de tekst aan zodat het duidelijk is op welke knop er moet gedrukt worden om opnieuw te starten (R)
 - ▬ Hiervoor kan je de `SetScore()` functie gebruiken van het view-object
- Zorg ervoor dat je enkel het spel kan herstarten als het reeds over is

Stap 6: GameOver wanneer Flappy tegen de buizen botst

- ▣ In de `UpdateFrame()` functie
 - Gebruik de `DoesFlappyCollidesWithPipe()`
 - Indien er een collision is, is het gameover



4.

Hoe je spel verspreiden?

Compile stap

- Momenteel steeds alles uitgevoerd via visual studio
 - ▬ Build / Compile stap
 - Debug mode: extra details om debugging mogelijk te maken
 - Release mode: optimalisaties om een snellere applicatie te bekomen
 - ▬ Maakt een .exe aan
 - ▬ Deze file is het programma en kan uitgevoerd worden op elke windows computer
 - Kan gevonden worden in bin/Release
 - Let op dat virus-scanners het starten en delen van onbekende .exe bestanden kunnen blokkeren

5. Uitbreidingen

Extra's

- ▣ Animeer ook de wolken zodat ze ook naar links bewegen
 - Vergeet ze ook niet opnieuw rechts te plaatsen wanneer ze links verdwenen zijn
- ▣ Voeg onderstaande code toe op het einde van de `UpdateFrame()` functie
 - Schrijf een AI in de `ShouldJump()` functie die zelf kiest of Flappy naar boven gaat of naar beneden

```
if (ShouldJump())
{
    SpacePressed();
}
else
{
    SpaceReleased();
}
```





Extra's

- ▣ Voeg randomness toe bij het terugplaatsen van de buizen nadat ze links verdwenen zijn
- ▣ Verhoog de moeilijkheid door de snelheid te verhogen elke 10 punten
- ▣ Voeg een object toe dat Flappy een schild geeft waardoor hij 3 seconden onkwetsbaar is.
 - ▬ Laat dit object willekeurig over het scherm bewegen