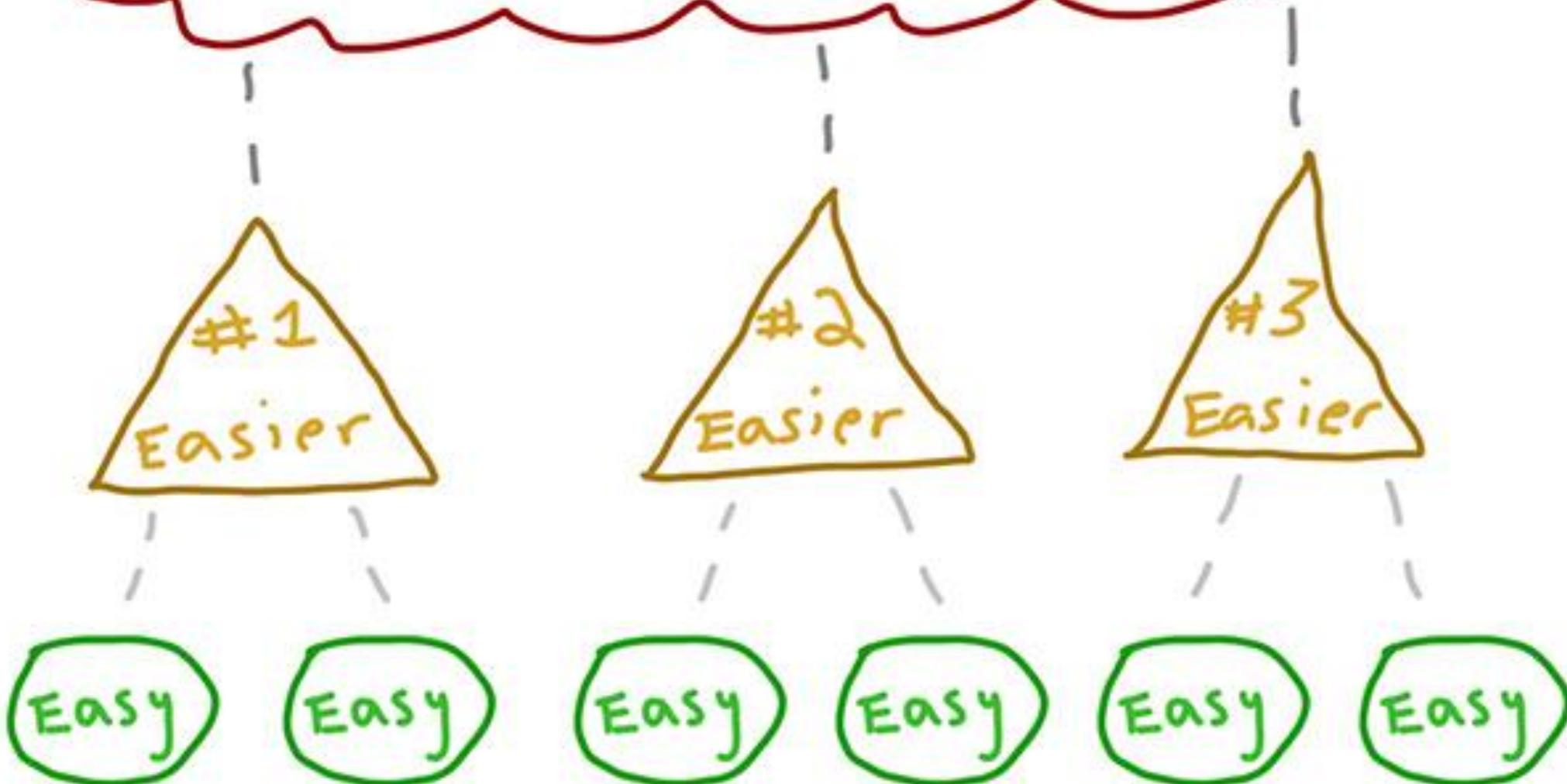


Xamarin.Forms UI's

Build **Amazing Things** from **Simple Things**



Hard Problem





[KymPhillpotts](#)



kymphillpotts.com



twitter.com/kphillpotts

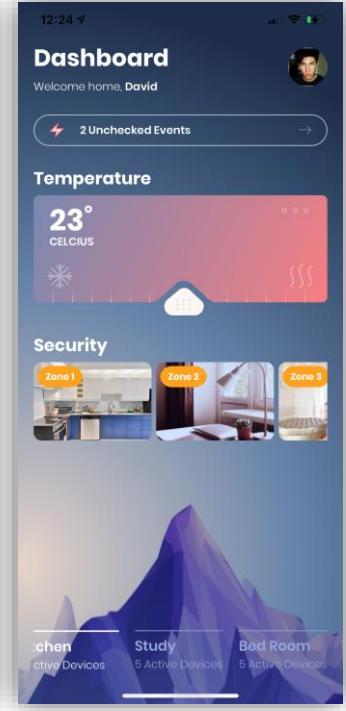
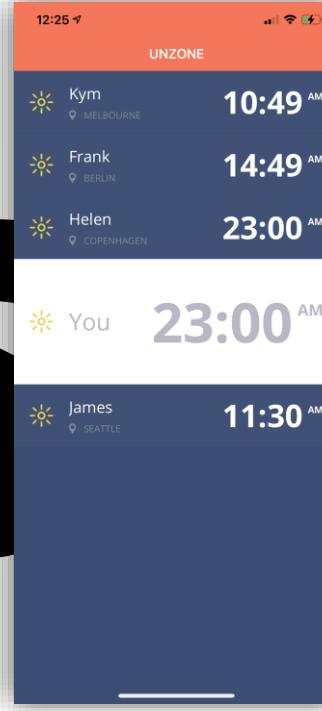
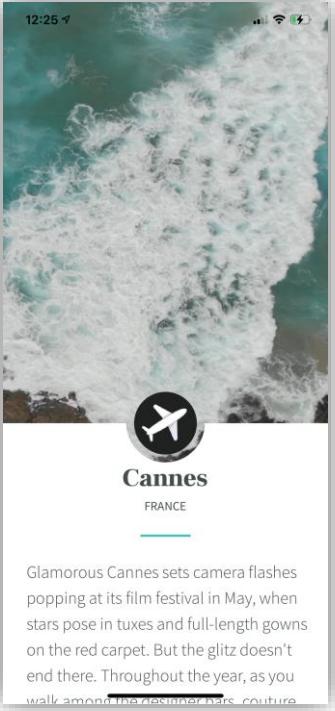


twitch.tv/kymphillpotts



youtube.com/kphillpotts

**Can you create
amazing looking apps
with Xamarin.Forms?**



I'm not a designer.

Finding Inspiration





Javier Suárez



Leomaris Reyes



Oludayo Alli
(DevCrux)



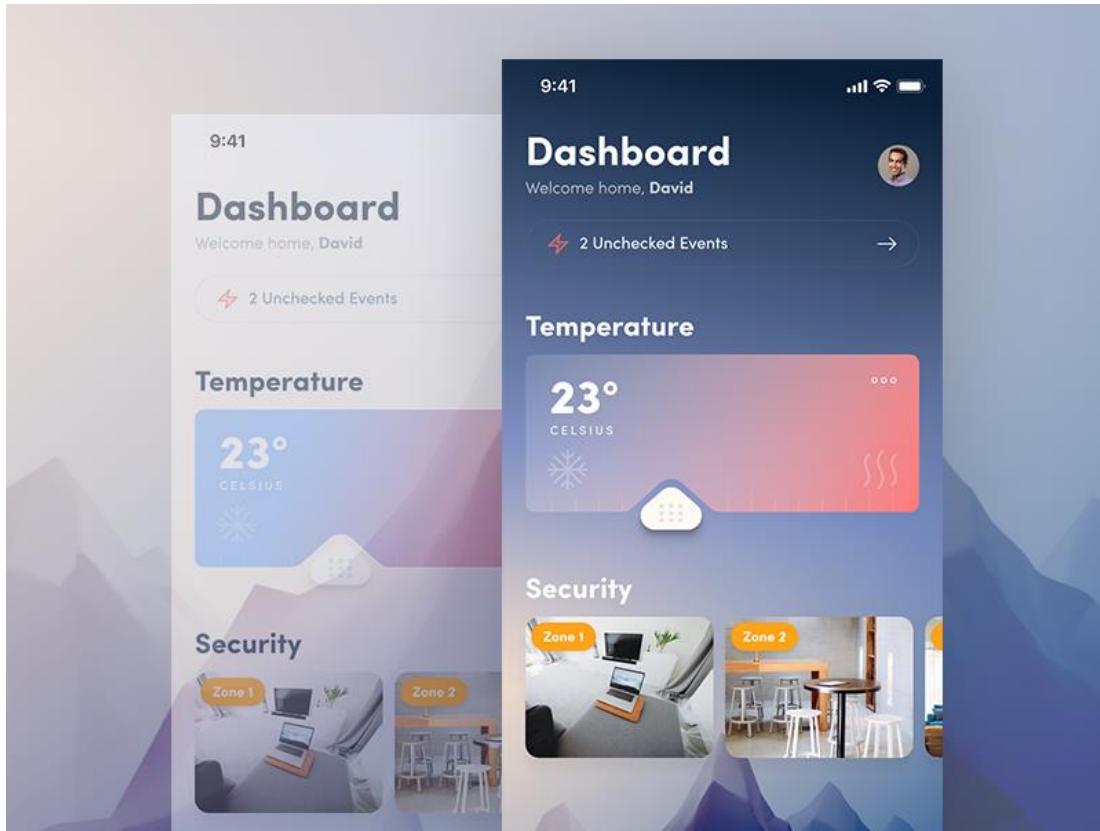
Altvir Cardoso

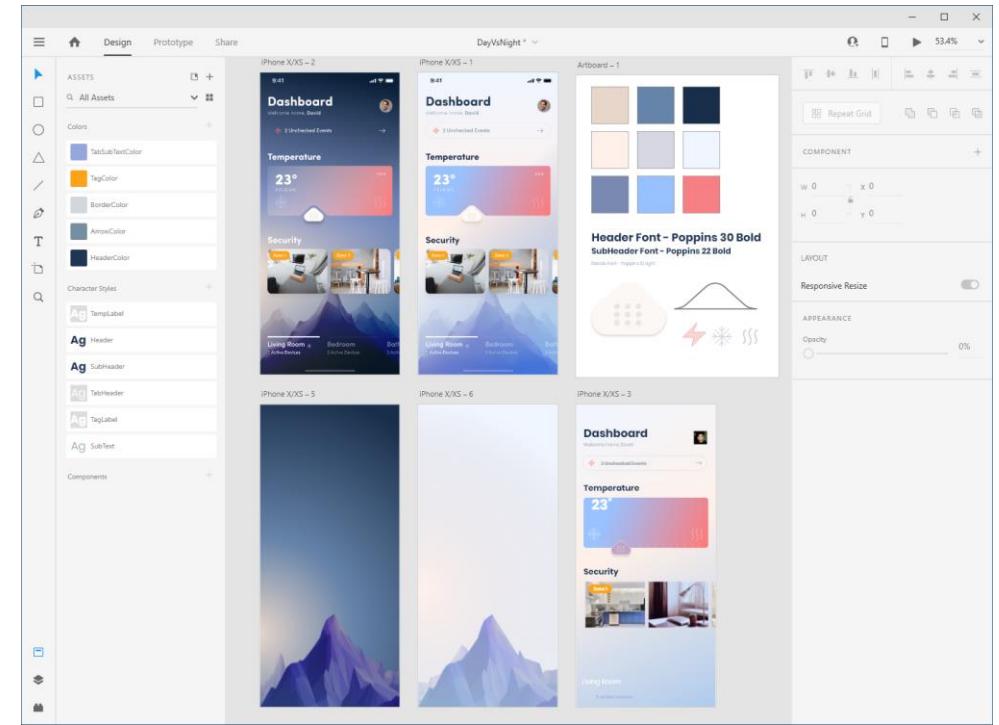
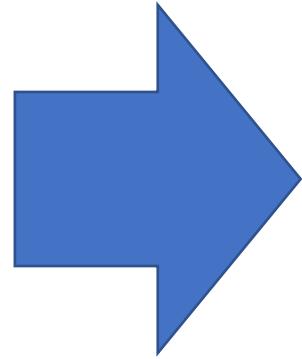
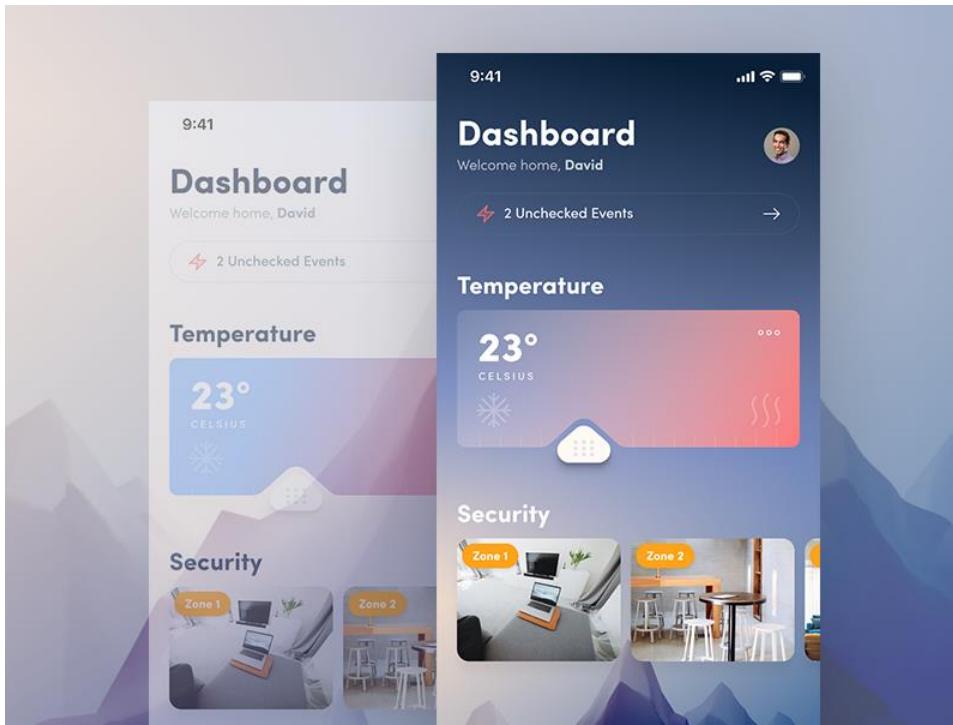


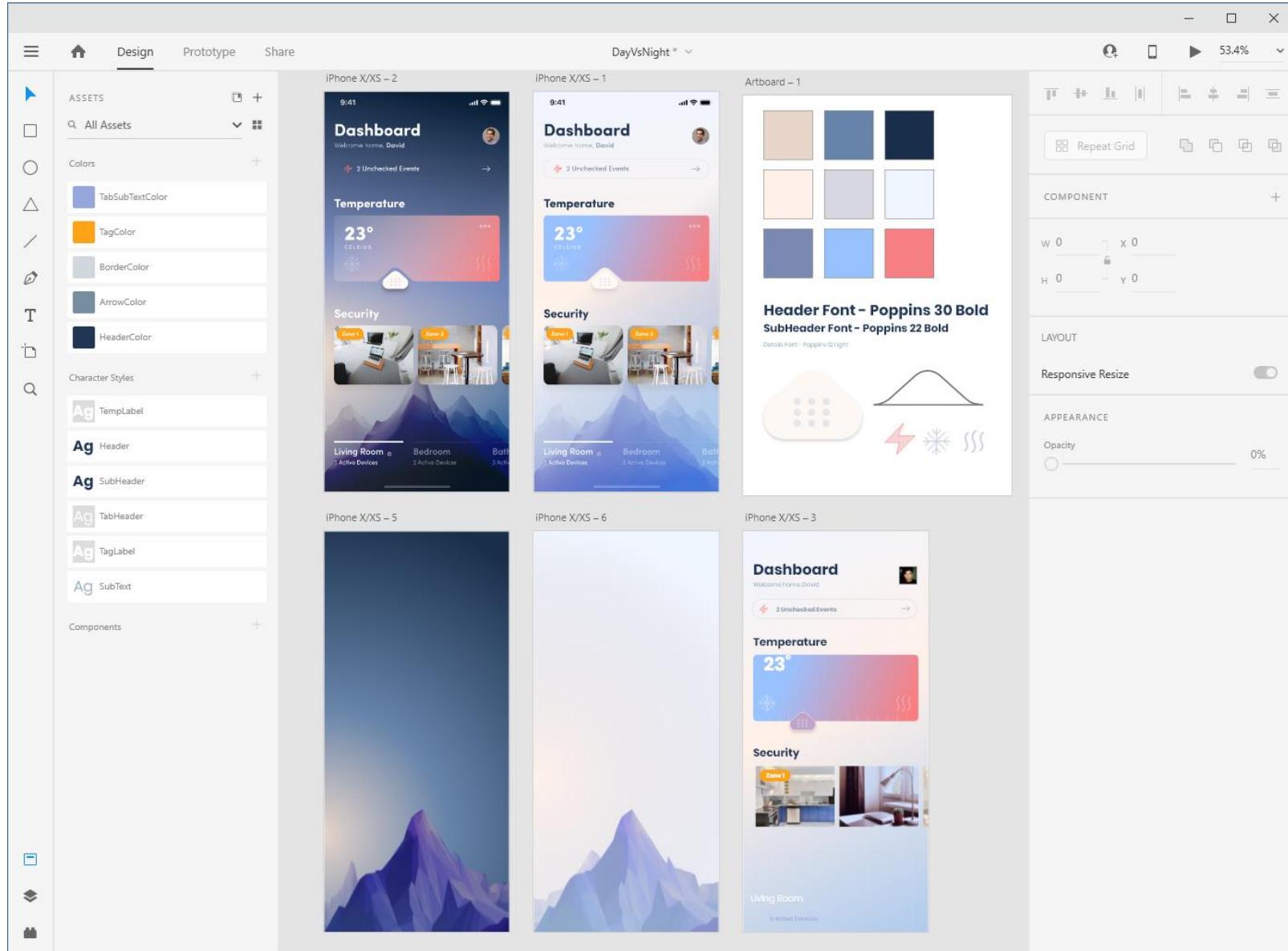
Luis Pujols

<https://github.com/jsuarezruiz/xamarin-forms-goodlooking-UI>

Breaking down design







- Fonts Family & Sizes
- Colors
- Margins & Spacing
- Graphical assets

Use Resources and Styles

```
<!-- Colors -->
<Color x:Key="BorderColor">#d2d7dd</Color>
<Color x:Key="HeaderColor">#213654</Color>
<Color x:Key="SubTextColor">#95a8b6</Color>

<!-- Sizes -->
<x:Double x:Key="HeaderFontSize">32</x:Double>

<!-- Styles -->
<Style x:Key="Header" TargetType="Label">
    <Setter Property="TextColor" Value="{StaticResource HeaderColor}" />
    <Setter Property="FontFamily" Value="HeaderFont" />
    <Setter Property="FontSize" Value="{StaticResource HeaderFontSize}" />
</Style>

<!-- Implicit Styles -->
<Style TargetType="Grid">
    <Setter Property="RowSpacing" Value="0" />
    <Setter Property="ColumnSpacing" Value="0" />
</Style>

<Style TargetType="Frame">
    <Setter Property="Padding" Value="0" />
    <Setter Property="HasShadow" Value="False" />
</Style>
```

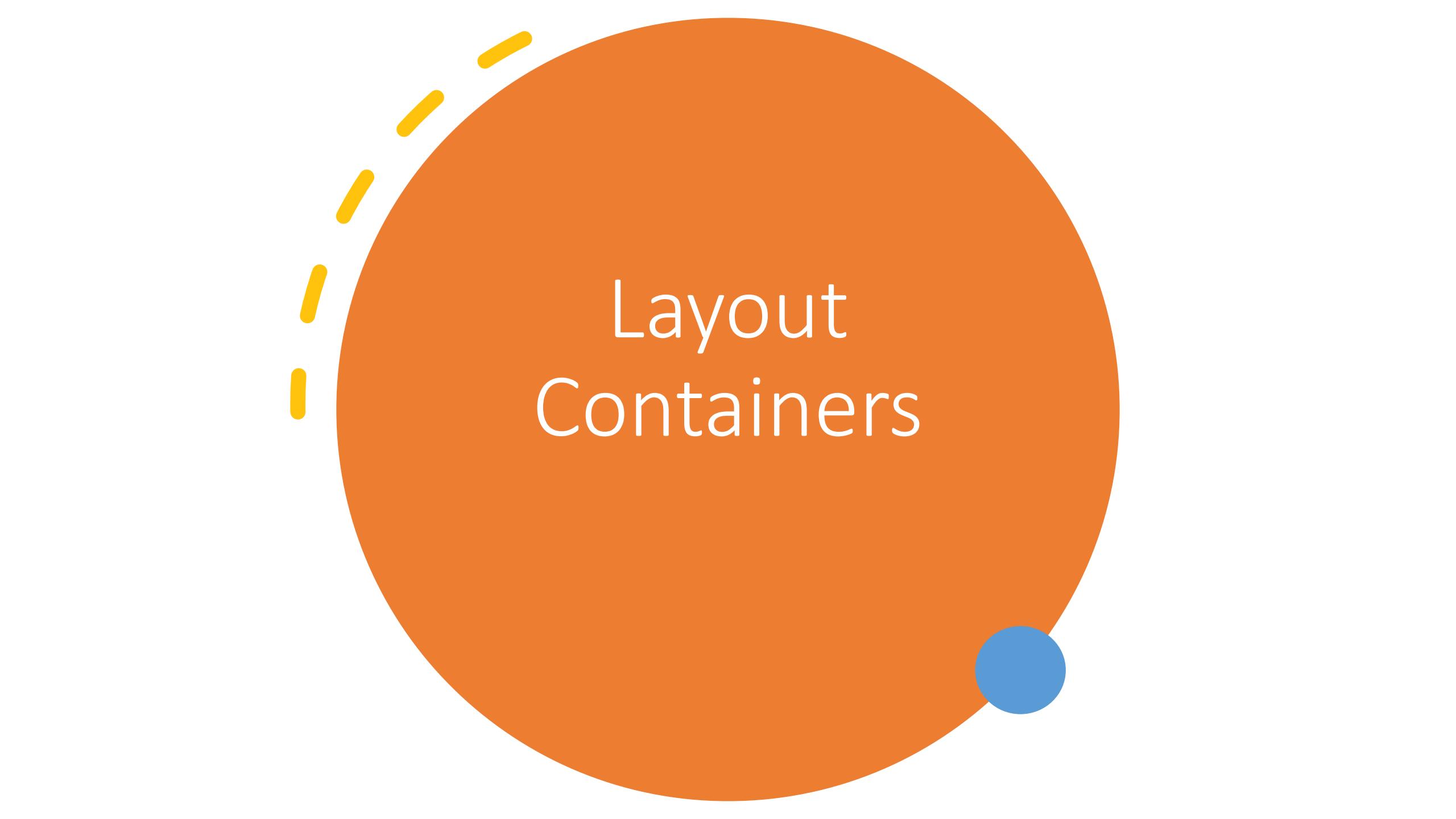
Consistency

Faster feedback
whilst building

Easier tweaking



**Learn the
Simple Things**

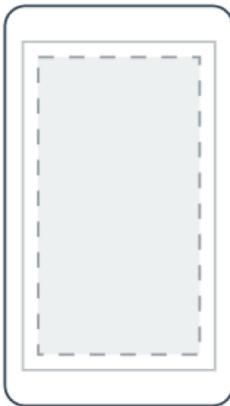


Layout
Containers

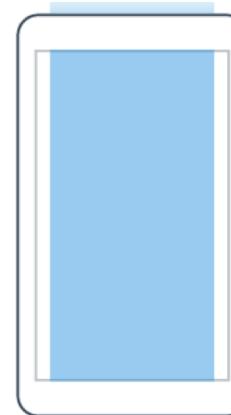
Layout Containers



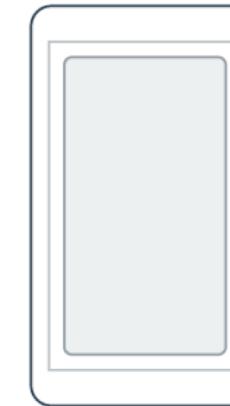
ContentPresenter



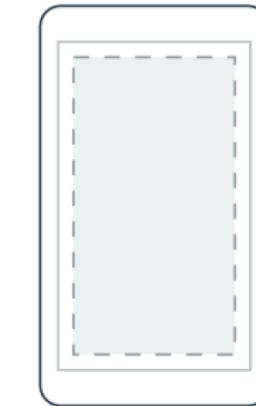
ContentView



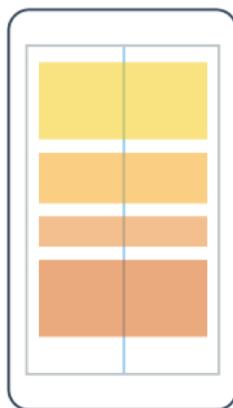
ScrollView



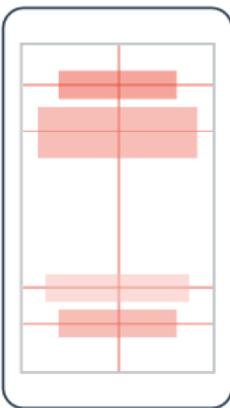
Frame



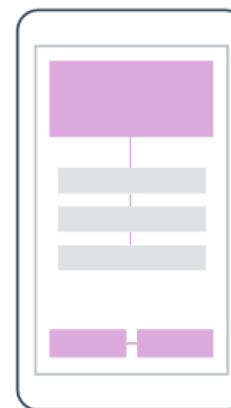
TemplatedView



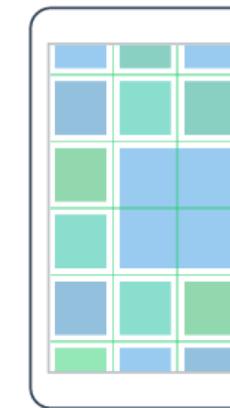
StackLayout



AbsoluteLayout



RelativeLayout

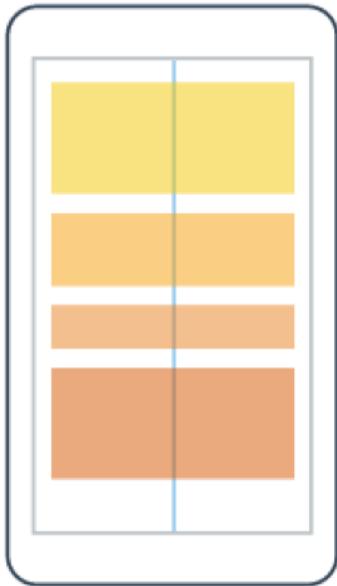


Grid

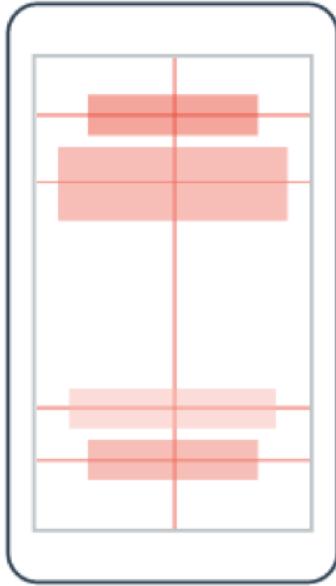


FlexLayout

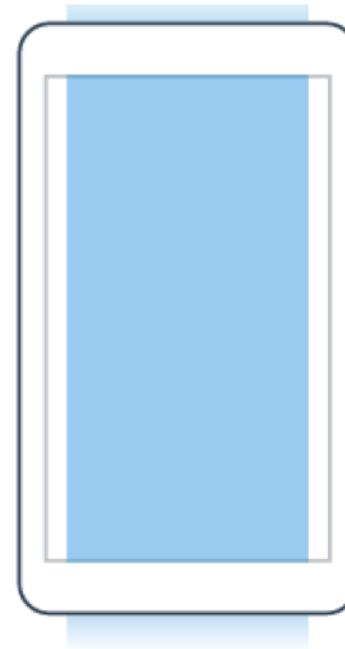
(Almost) all layouts can be created with a combination of these



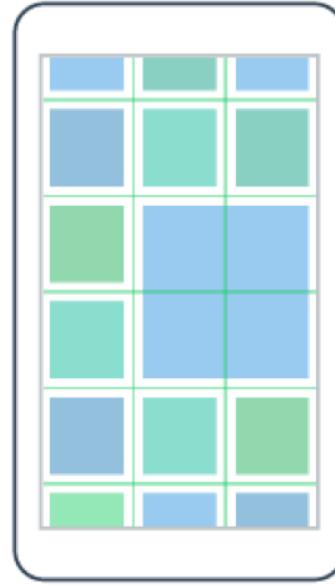
StackLayout



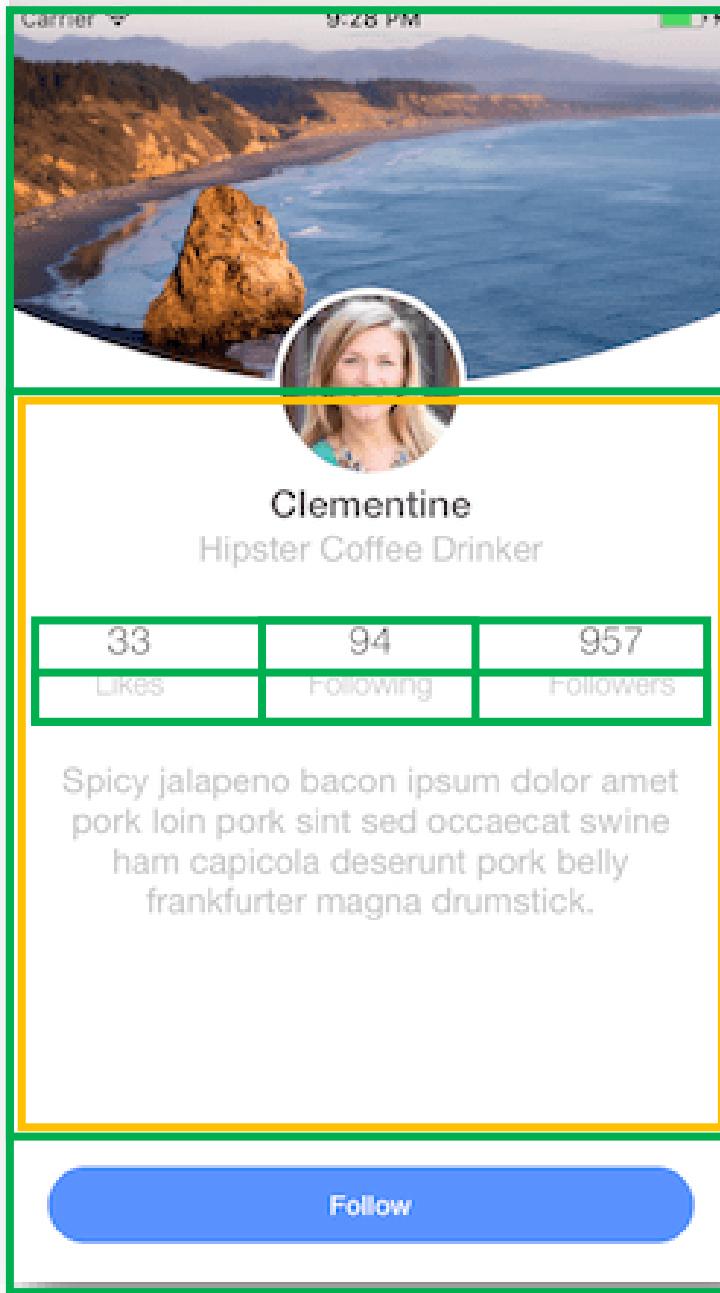
AbsoluteLayout



ScrollView



Grid



Grid (3 Rows)

Vertical StackLayout

Grid (3 cols, 2 rows)

Overlapping Elements



Positioning Views

- HorizontalOptions / VerticalOptions
- WidthRequest / HeightRequest
- Margin
- Padding
- Spacing (StackLayouts)
- RowSpacing / ColumnSpacing (Grids)
- RowSpan / ColumnSpan



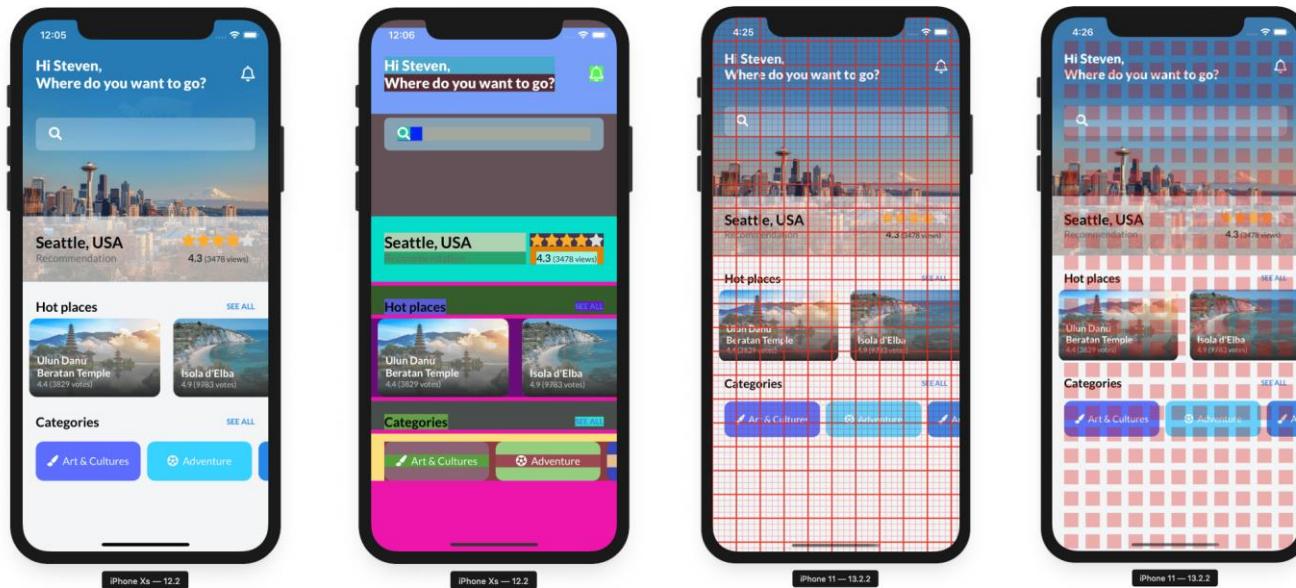
Transform Properties

- Translation X/Y
- Opacity
- Scale
- Rotation X/Y/Z



Debugging Layouts

Use `Xamarin.Forms.DebugRainbows` by Steven Thewissen



A yellow circular icon resembling a bomb or a bombshell, positioned on the left side of the slide. It has a yellow outline and a yellow fill. There are five short yellow lines radiating from the top right of the circle, suggesting it has just exploded or is about to. The background behind the bomb is orange.

Build
Amazing Things

Art Auction

Designer: Alex Pesenka

Components:

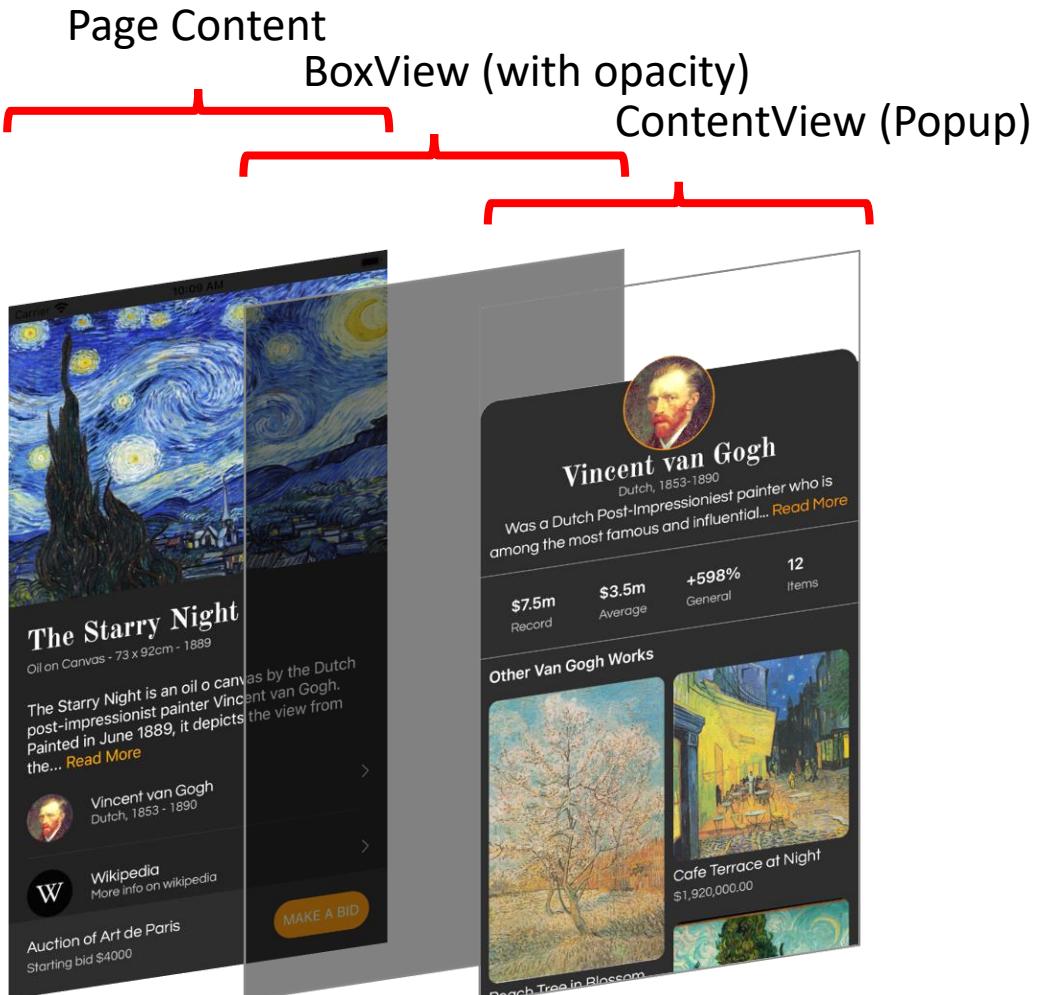
- Xamarin.Forms
- PancakeView
- ImageCircle



Github: <https://github.com/kphillpotts/artauction>

Dribbble: <https://dribbble.com/shots/6177235-Valuable-Auction-Product-Page>

How to Popup and Fade Background



How to Popup and Fade Background



Vincent van Gogh



BoxView (Opacity 0, Visible=False)

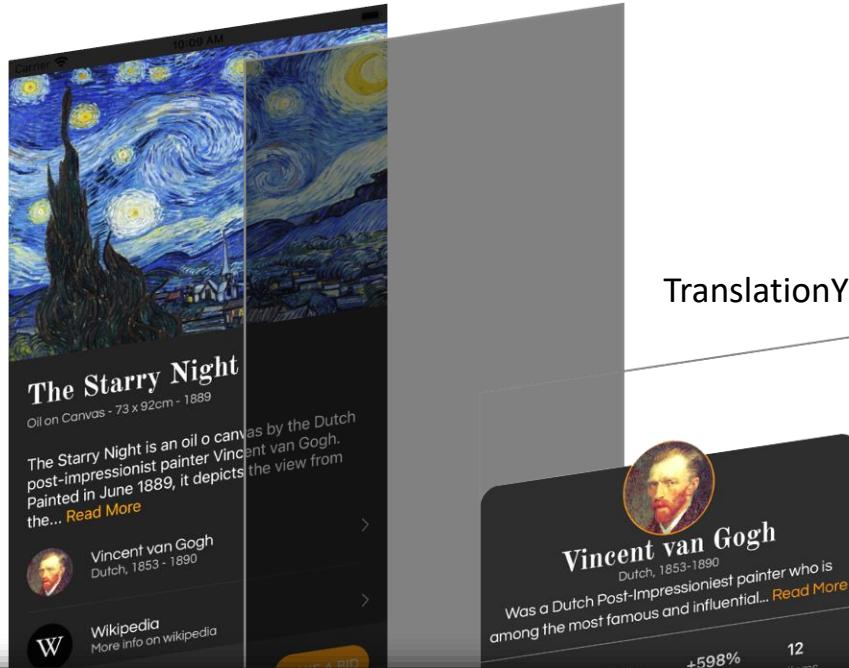
TranslationY = Page.Height



How to Popup and Fade Background



IsVisible=True, FadeTo(1)



TranslationY = Page.Height – FirstSectionHeight

```
private void Artist_Tapped(object sender, EventArgs e)
{
    var firstSection = ArtistDetailsPopup.FirstSectionHeight;
    PageFader.Visible = true;
    PageFader.FadeTo(1, AnimationSpeed, Easing.SinInOut);
    ArtistDetailsPopup.TranslateTo(0, Height - firstSection, AnimationSpeed, Easing.SinInOut);
}
```

How to Popup and Fade Background



TranslationY = 80

```
private void ExpandArtistDetails()
{
    ArtistDetailsPopup.TranslateTo(0, 80, AnimationSpeed, Easing.SinInOut);
}
```

How to Popup and Fade Background

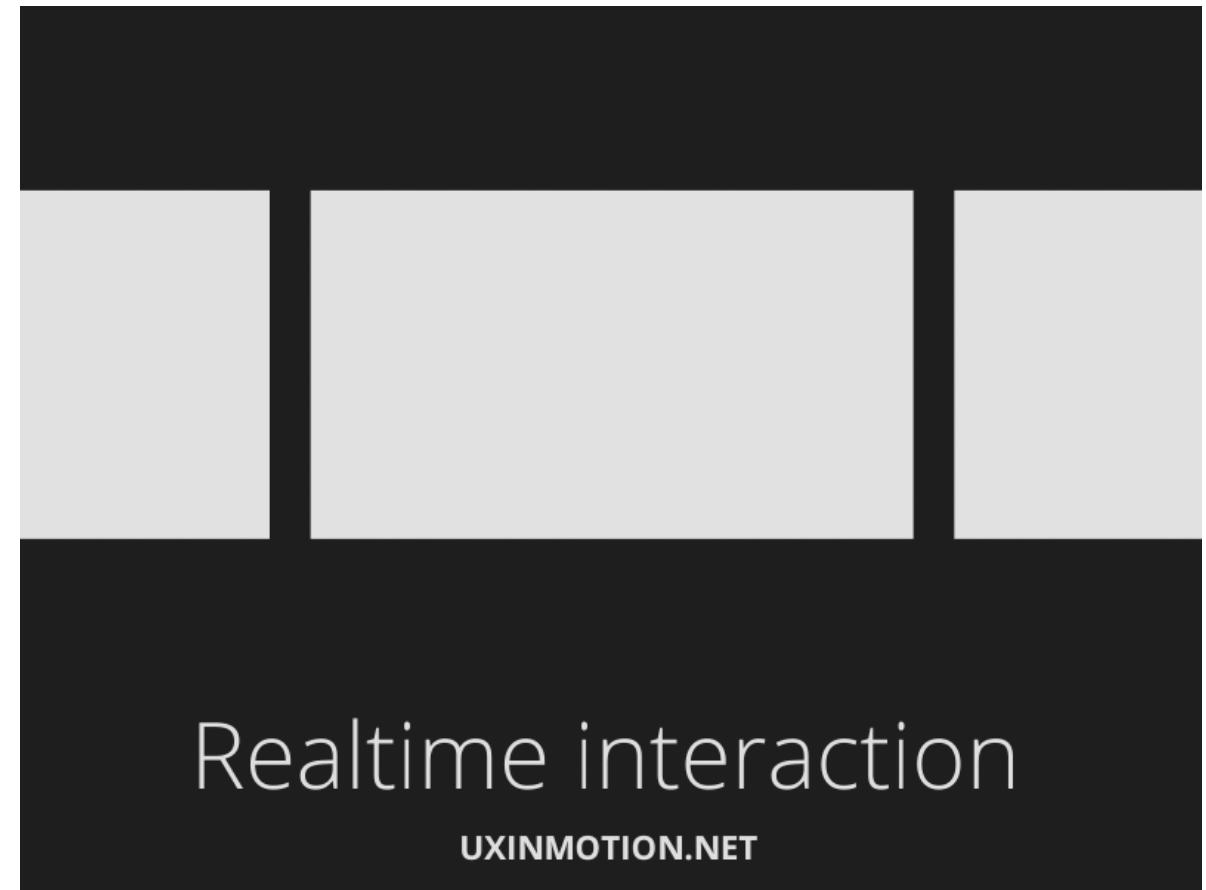
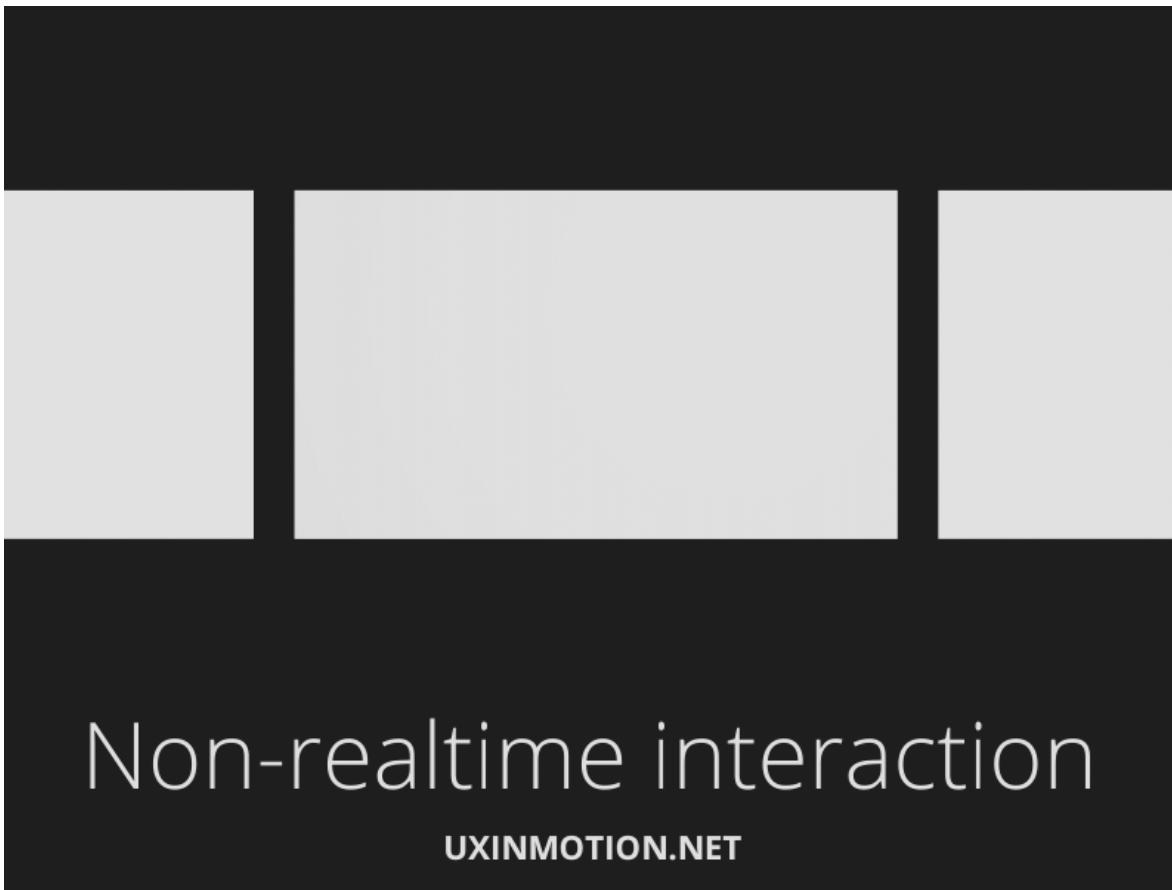


BoxView (Opacity 0, Visible=False)

```
private async void PageFader_Tapped(object sender, EventArgs e)
{
    _ = ArtistDetailsPopup.TranslateTo(0, Height, AnimationSpeed, Easing.SinInOut);
    await PageFader.FadeTo(0, AnimationSpeed, Easing.SinInOut);
    PageFader.Visibile = false;
}
```

TranslationY = Page.Height

Different Types of Interactions



UnZone

Designer: Rick Waalders

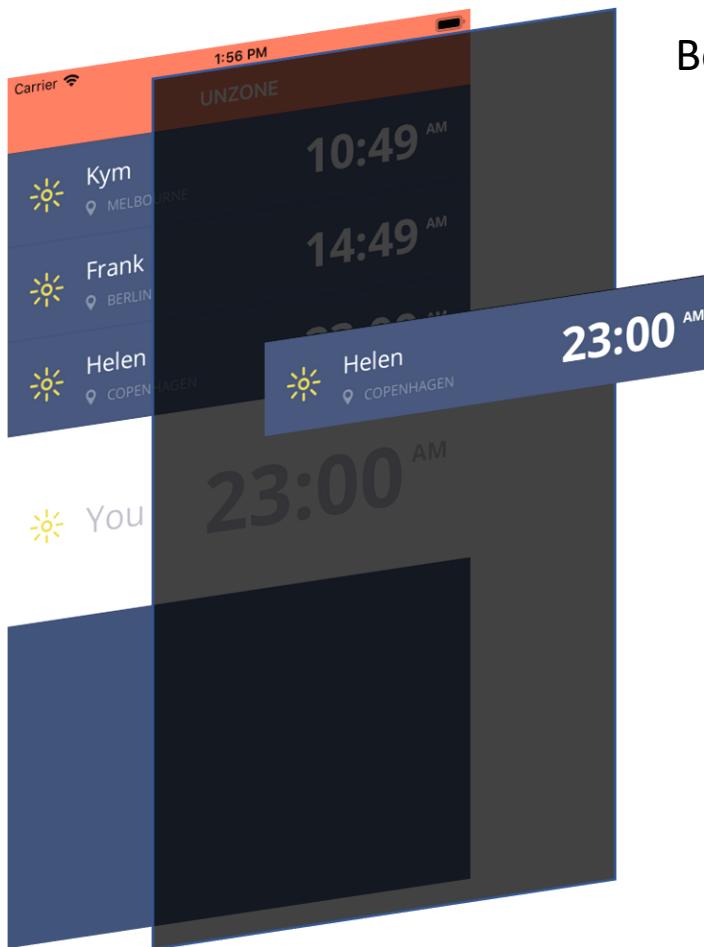
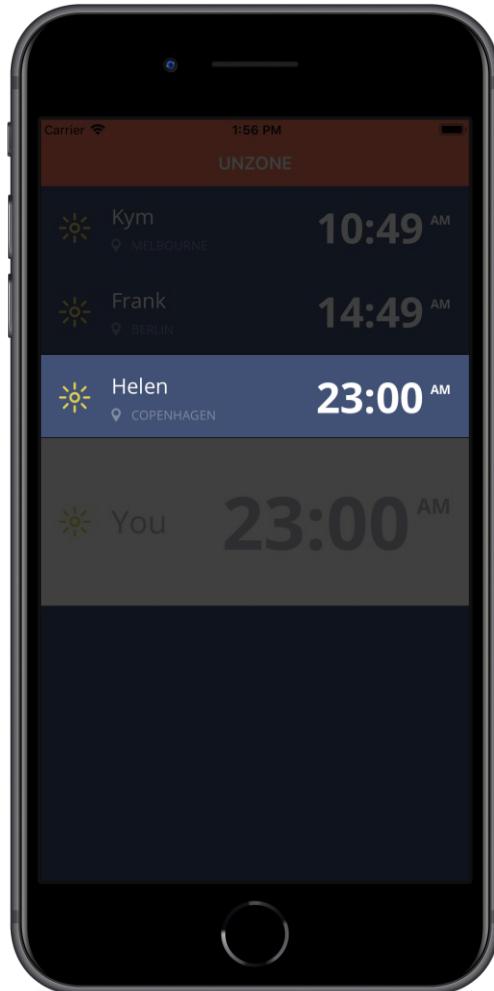
Components:

- Xamarin.Forms

Github: <https://github.com/kphillpotts/unzone>

Dribbble: <https://dribbble.com/shots/1551934-Unzone-Details-popup-animation>

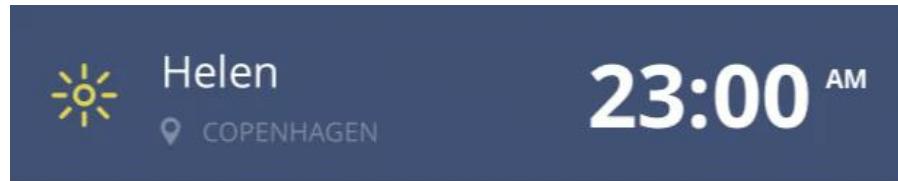
A more advanced Popup



BoxView (Opacity .5)

Copy of the Cell
Placed at the same position
As the tapped cell
(using AbsoluteLayout and Bounds)

How to Fold



```
await OpenDropDown(DeleteDropDown);  
await OpenDropDown(EditDropDown);  
await OpenDropDown(InfoDropDown);
```

```
private async Task OpenDropDown(View view)  
{  
    view.Visible = true;  
    view.RotationX = -90;  
    view.Opacity = 0;  
    _ = view.FadeTo(1, animationSpeed);  
    await view.RotateXTo(0, animationSpeed);  
}
```

The key is setting the **AnchorX** and **AnchorY** properties to determine where the rotation is based

How to Flip

from

```
private async Task Flip (VisualElement from, VisualElement to)
{
    await from.RotateYTo(-90, animationSpeed, Easing.SpringIn);
    to.RotationY = 90;
    to.Visible = true;
    from.Visible = false;
    await to.RotateYTo(0, animationSpeed, Easing.SpringOut);
}
```

How to Flip

```
private async Task Flip (VisualElement from, VisualElement to)
{
    await from.RotateYTo(-90, animationSpeed, Easing.SpringIn);
    to.RotationY = 90;
    to.Visible = true;
    from.Visible = false;
    await to.RotateYTo(0, animationSpeed, Easing.SpringOut);
}
```

How to Flip

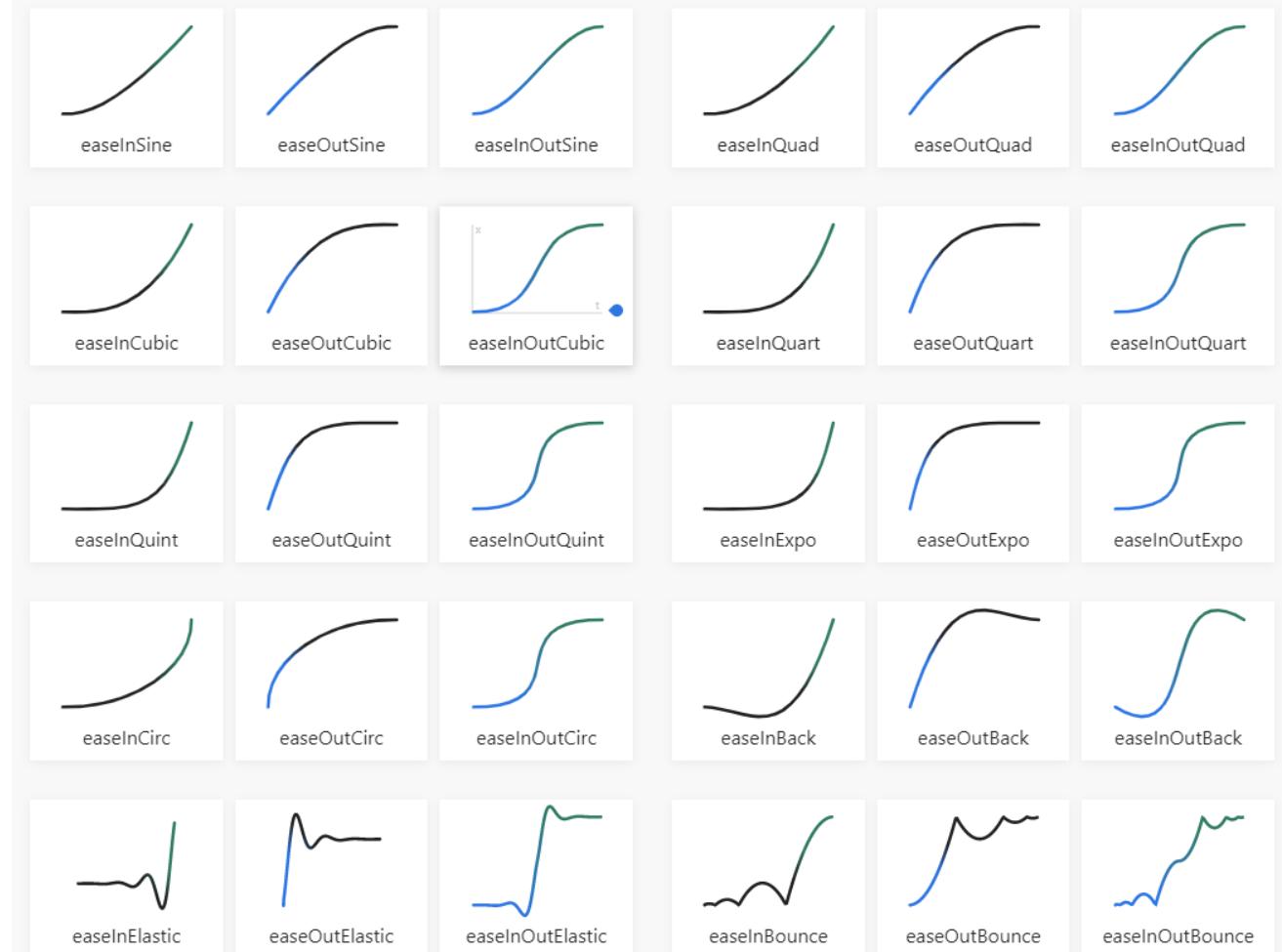
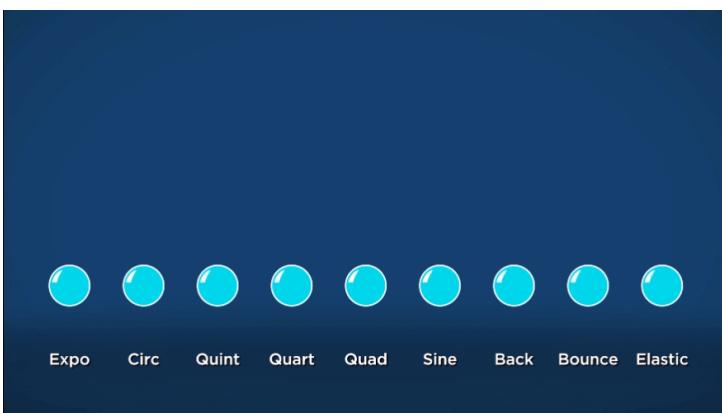
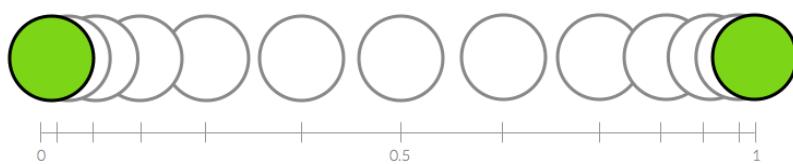
```
private async Task Flip (VisualElement from, VisualElement to)
{
    await from.RotateYTo(-90, animationSpeed, Easing.SpringIn);
    to.RotationY = 90;
    to.Visible = true;
    from.Visible = false;
    await to.RotateYTo(0, animationSpeed, Easing.SpringOut);
}
```

How to Flip

to

```
private async Task Flip (VisualElement from, VisualElement to)
{
    await from.RotateYTo(-90, animationSpeed, Easing.SpringIn);
    to.RotationY = 90;
    to.isVisible = true;
    from.isVisible = false;
    await to.RotateYTo(0, animationSpeed, Easing.SpringOut);
}
```

Easing Function



<https://easings.net>

How to float



```
private void AnimateCloseButton(VisualElement elementToTransform, bool entering)
{
    ...
    elementToTransform.FadeTo(endingOpacity, 500);
    elementToTransform.RotateTo(endingRotation, 700, Easing.SinInOut);
    elementToTransform.TranslateTo(0, endingTranslation, 600, translationEasing);
}
```

Marvel Movies Interaction

Designer: Vijay Verma

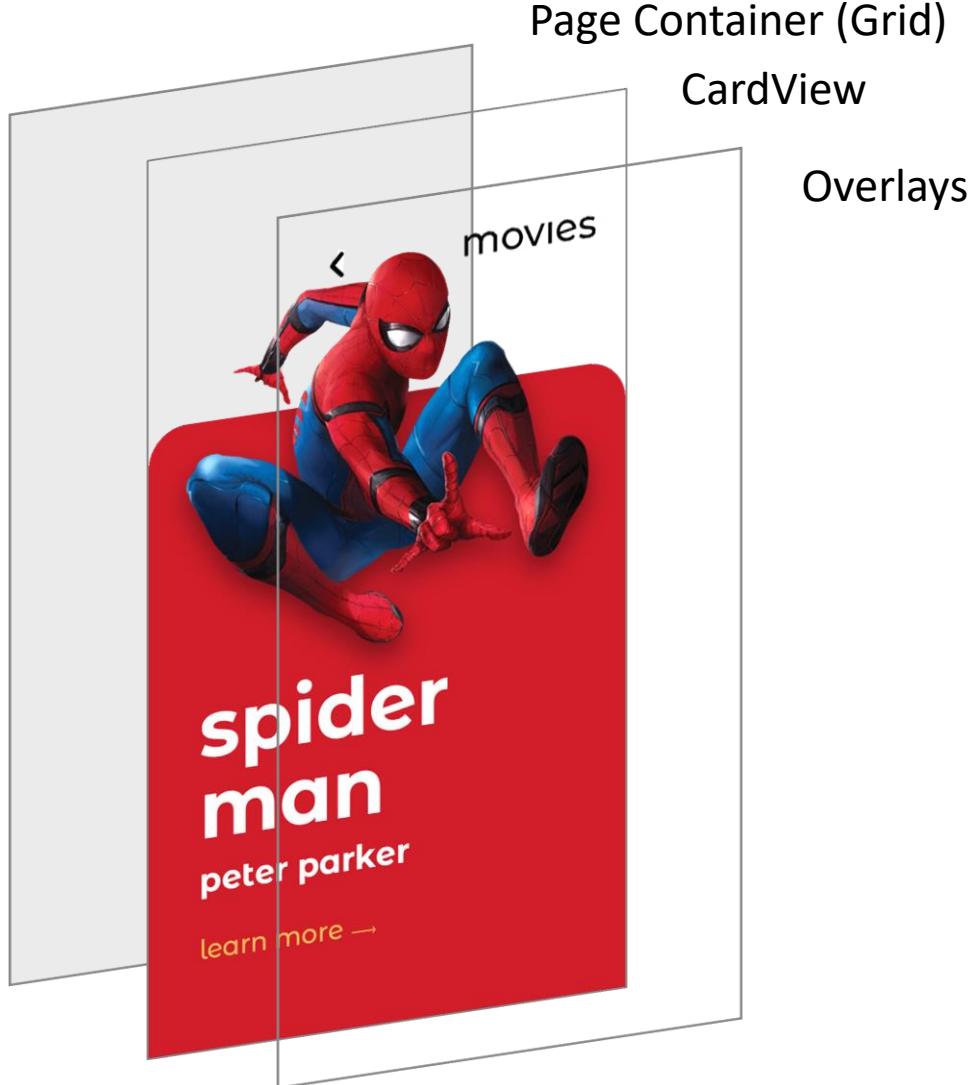
Components:

- Xamarin.Forms
- PancakeView
- SkiaSharp
- CardsView
<https://github.com/AndreiMisiukevich/CardView>

Github: <https://github.com/kphillpotts/marvelcards>

Dribbble: <https://dribbble.com/shots/5935613-Marvel-Movies-Interaction>

Page Layout



Realtime interaction on cards

```
2 references | 0 changes | 0 authors, 0 changes
private void AnimateFrontCardDuringSwipe(HeroCard card, double percentFromCenter)
{
    // opacity of the maincard during swipe
    MainCardView.CurrentView.Opacity = LimitToRange((1 - (percentFromCenter)) * 2, 0, 1);

    // scaling on the main card during swipe
    card.MainImage.Scale = LimitToRange((1 - (percentFromCenter)) * 1.5), 0, 1);

    // y offset of image during swipe
    card.MainImage.TranslationY = _heroImageTranslationY + (_movementFactor * percentFromCenter);

    // adjust opacity of image
    card.MainImage.Opacity = LimitToRange((1 - (percentFromCenter)) * 1.5, 0, 1); ;
}

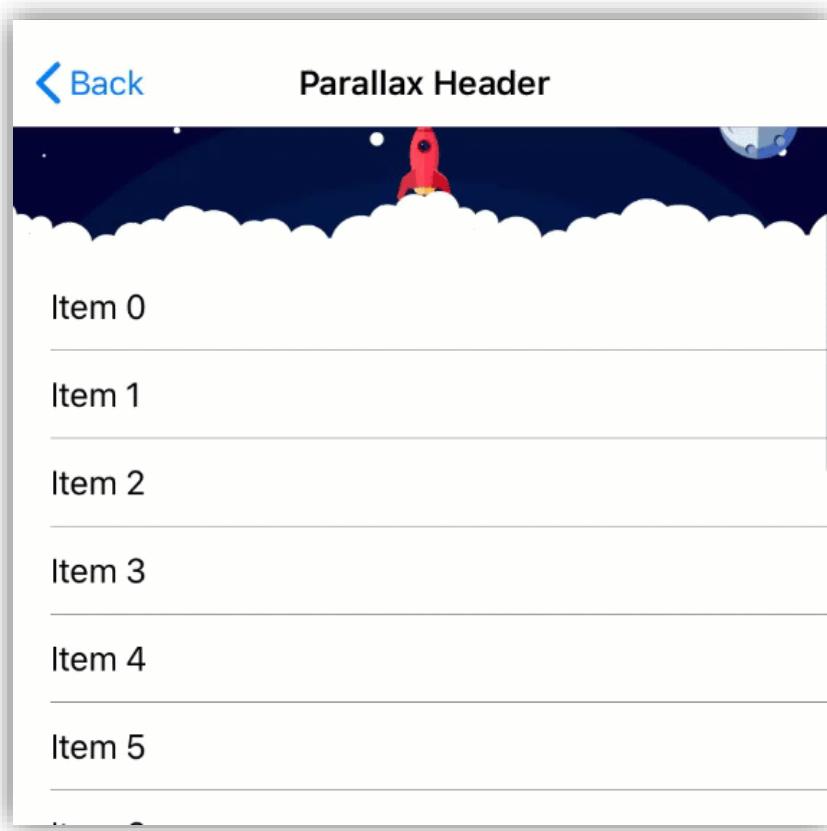
1 reference | 0 changes | 0 authors, 0 changes
private void AnimateIncomingCardDuringSwipe(HeroCard nextCard, double percentFromCenter)
{
    // opacity fading in
    nextCard.MainImage.Opacity = LimitToRange(percentFromCenter * 1.5, 0, 1);

    // scaling in
    nextCard.MainImage.Scale = LimitToRange(percentFromCenter * 1.1, 0, 1);

    var offset = _heroImageTranslationY + (_movementFactor * (1-(percentFromCenter*1.1)));
    nextCard.MainImage.TranslationY = LimitToRange(offset, _heroImageTranslationY,
        _heroImageTranslationY + _movementFactor);
}
```



Parallax



```
var scrollValue = e.ScrollY;  
Rocket.TranslationY = scrollValue / 2.5;  
Moon.TranslationY = scrollValue / 1.5;  
Clouds2.TranslationY = scrollValue / 2.2;  
Background.TranslationY = scrollValue;
```

Parallax is just translations based on a multiplier

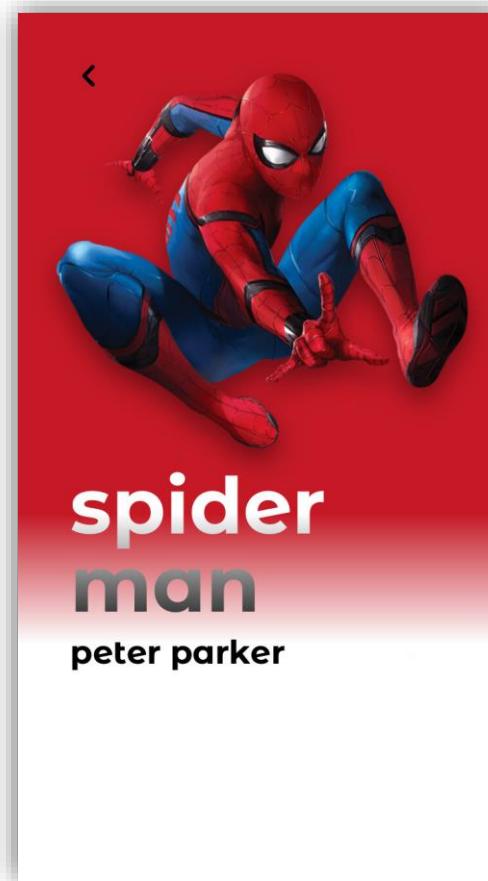
Card Transition



User selects card element



Background expands up



Gradient slides up



Other elements animate in

Pizza Shop

Designer: Bidyut Kumar Bera

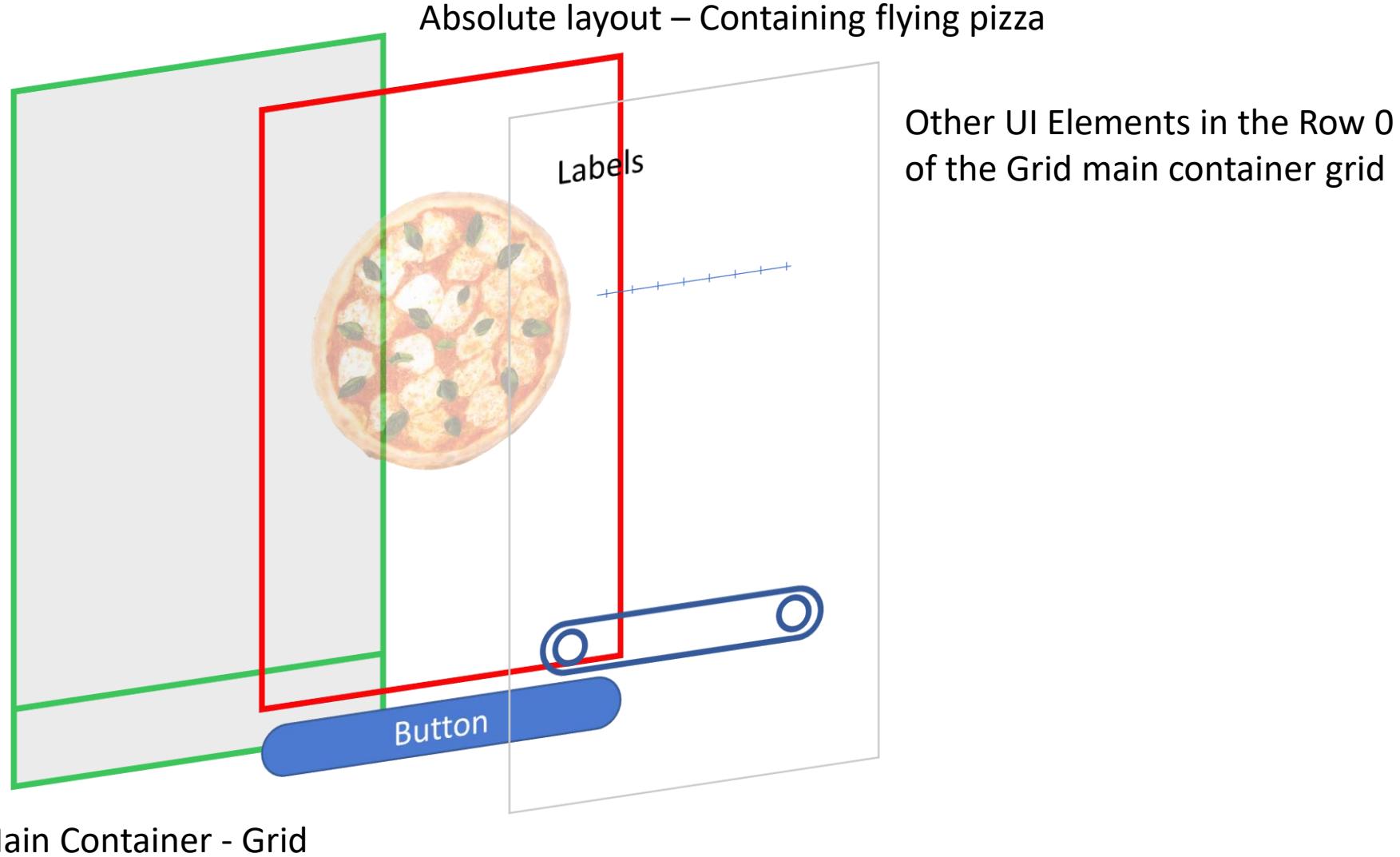
Components:

- Xamarin.Forms
- PancakeView
- SkiaSharp

Github: <https://github.com/kphillpotts/pizza>

Dribbble: <https://dribbble.com/shots/10921038-Pizza-Shop-app-Interaction>

Page Layout



State based animations

```
enum State
{
    Small,
    Medium,
    Large
}
```



```
var animState = new AnimationStateMachine();

animState.Add(State.Small, new ViewTransition[]
{
    new ViewTransition(Pizza, AnimationType.Layout, smallRect),
    new ViewTransition(Pizza, AnimationType.Rotation, 45),
    // ... etc.
});
animState.Add(State.Medium, new ViewTransition[]
{
    new ViewTransition(Pizza, AnimationType.Layout, mediumRect),
    new ViewTransition(Pizza, AnimationType.Rotation, 90),
    // ... etc.
});
animState.Add(State.Large, new ViewTransition[]
{
    new ViewTransition(Pizza, AnimationType.Layout, largeRect),
    new ViewTransition(Pizza, AnimationType.Rotation, 135),
    // ... etc.
});
```

```
private void PizzaRulerThumb_Tapped(object sender, EventArgs e)
{
    // navigate to the next state
    switch (animState.CurrentState)
    {
        case State.Small:
            animState.Go(State.Medium);
            break;
        case State.Medium:
            animState.Go(State.Large);
            break;
        case State.Large:
            animState.Go(State.Small);
            break;
    }
}
```

Create Transforms for different states



AnimationStateMachine

Trigger State Transitions



How to eat a pizza

```
private async Task PizzaFly()
{
    // check if the pizza is already flying
    if (FlyingPizza.IsVisible) return;

    // position flying pizza exactly over the display pizza
    FlyingPizza.IsVisible = true;
    AbsoluteLayout.SetLayoutBounds(FlyingPizza, Pizza.Bounds);

    // work out where it needs to fly to?
    var buttonBounds = PlaceOrderButton.Bounds;
    var size = new Size(buttonBounds.Height, buttonBounds.Height);
    var location = new Point(buttonBounds.Right - size.Width, buttonBounds.Top);
    var chompBounds = new Rectangle(location, size);

    // animate the pizza down
    _ = FlyingPizza.LayoutTo(chompBounds, 500, Easing.SinInOut);
    _ = FlyingPizza.RelRotateTo(90, 500, Easing.SinInOut);

    // do the button chomp
    var buttonChompBounds = new Rectangle(PlaceOrderButton.Bounds.Location,
        new Size(PlaceOrderButton.Width - buttonBounds.Height, buttonBounds.Height));
    await PlaceOrderButton.LayoutTo(buttonChompBounds, 500, Easing.SinInOut);

    // rotate the pizza as it's being eaten
    _ = FlyingPizza.RelRotateTo(-90, 500, Easing.SinInOut);

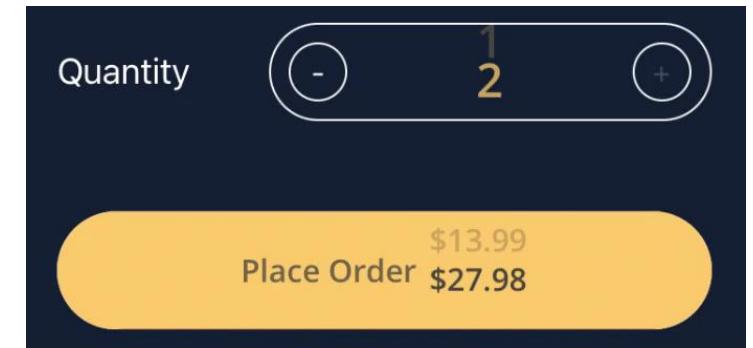
    // close the button chomp
    await PlaceOrderButton.LayoutTo(buttonBounds, 500, Easing.SinInOut);

    // hide the pizza
    FlyingPizza.IsVisible = false;
}
```



How to create a spinning counter

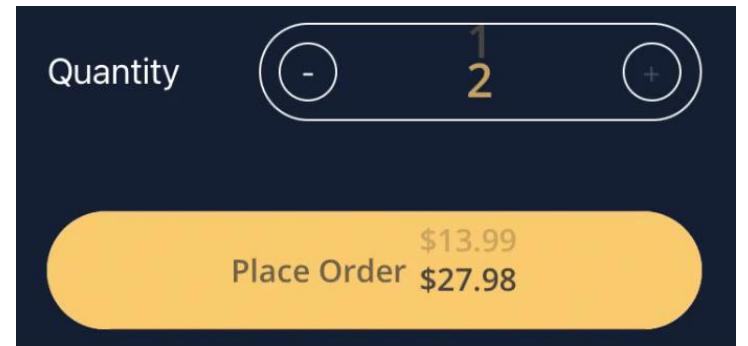
```
<Grid  
    x:Class="Pizza.Controls.FerrisLabel"  
    xmlns="http://xamarin.com/schemas/2014/forms"  
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"  
    xmlns:controls="clr-namespace:Pizza.Controls"  
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006">  
    <Label  
        x:Name="CurrentLabel"  
        HorizontalOptions="Center"  
        VerticalOptions="Center" />  
    <Label  
        x:Name="NextLabel"  
        HorizontalOptions="Center"  
        Opacity="0"  
        VerticalOptions="Center" />  
</Grid>
```



How to create a spinning counter

```
public static readonly BindableProperty TextProperty :
```

```
    ...  
    ...  
  
    async void ApplyText(string oldValue, string newValue)  
    {  
        // update the current to be the old value  
        Current.Text = oldValue;  
        Current.TranslationY = Current.TranslationX = 0;  
        Current.Opacity = 1;  
  
        // animate out the current label  
        _ = Current.TranslateTo(-AnimationOffset.X, -AnimationOffset.Y);  
        _ = Current.FadeTo(0);  
  
        // animate in the next label  
        Next.Text = newValue;  
        Next.TranslationY = AnimationOffset.Y;  
        Next.TranslationX = AnimationOffset.X;  
        Next.Opacity = 0;  
        _ = Next.TranslateTo(0, 0);  
        await Next.FadeTo(1);  
  
        // recycle the views  
        Current = NextLabel;  
        Next = CurrentLabel;  
    }
```



Day vs Night

Designer: Ionut Zamfir

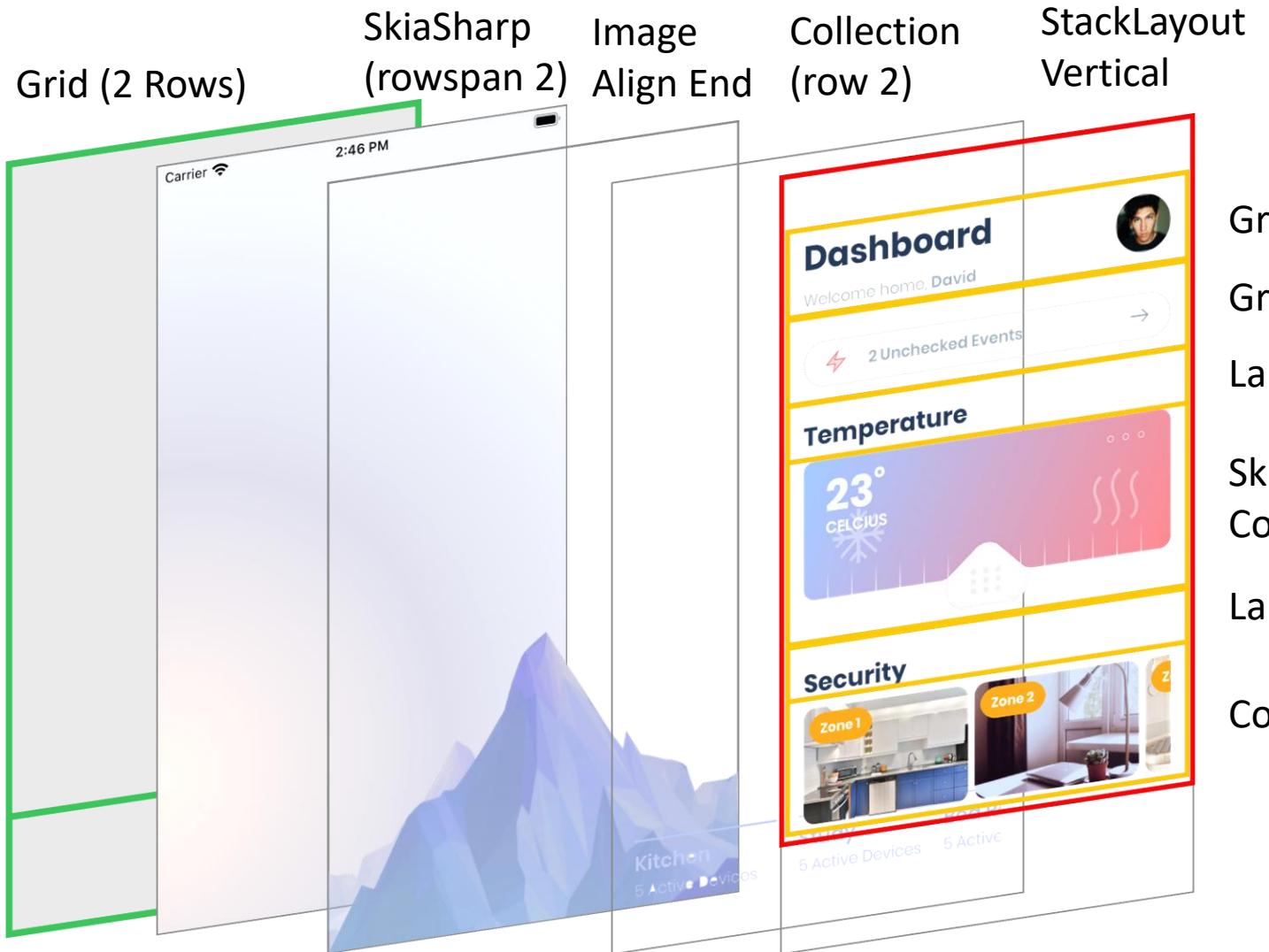
Components:

- Xamarin.Forms
- PancakeView
- SkiaSharp
- ImageCircle

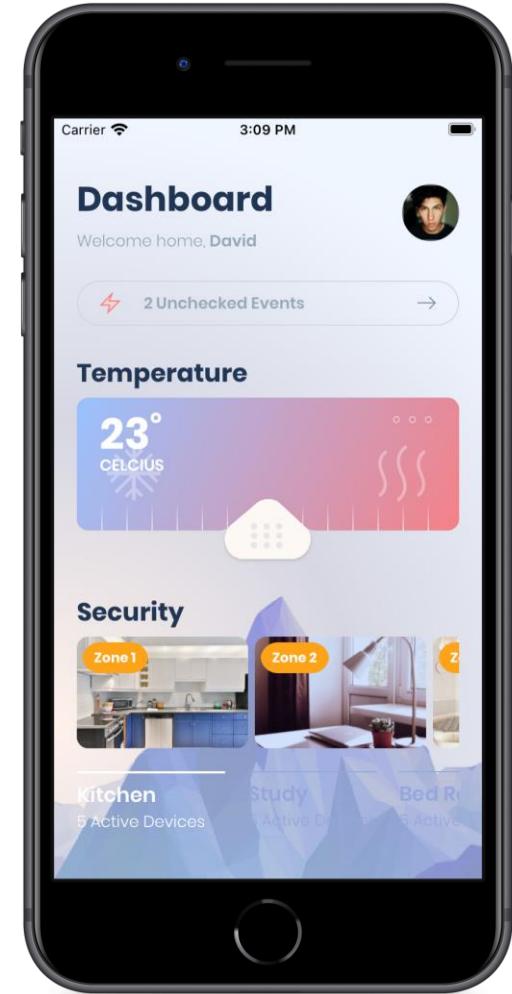
Github: <https://github.com/kphillpotts/dayvsnight>

Dribbble: <https://dribbble.com/shots/4840427-Dashboard-Day-vs-Night>

Page Layout

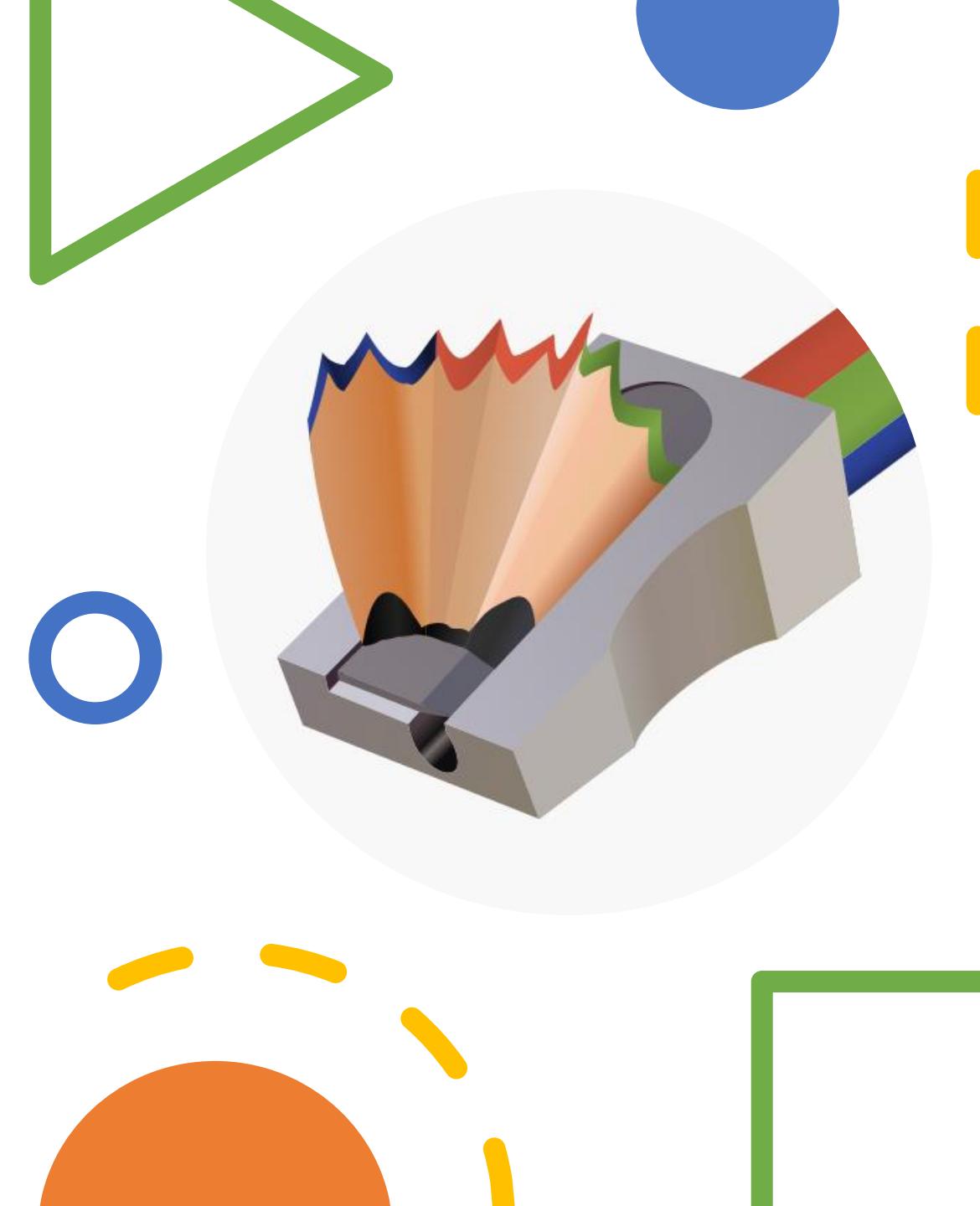


Grid
Grid
Label
SkiaSharp Control
Label
Collection



SkiaSharp

- Cross Platform 2D Graphics
- Amazing for creating custom drawing
- Super Fast and Feature Rich
- Pretty complex
- Docs and Samples are a great start

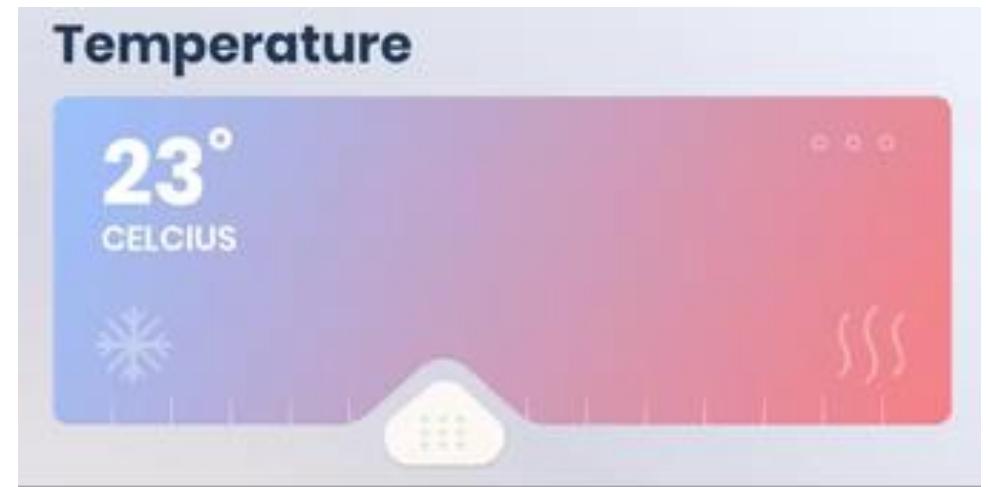


SkiaSharp Controls

```
private void TouchEffect_TouchAction(object sender, TouchEffect.TouchEventArgs args)
{
    Percent = (args.Location.X / TempGaugeCanvas.Width) * 100;
}

public double Percent
{
    get => percent;
    set
    {
        percent = value;
        TempGaugeCanvas.InvalidateSurface();
    }
}
```

```
// do some drawing stuff here - it won't be clipped
...
using (new SKAutoCanvasRestore(canvas))
{
    ...
    canvas.ClipPath(scaledClipPath, SKclipOperation.Difference, true);
    ...
    // everything here will have the clipping applied
    ...
}
// after the USING the canvas is restored to it's previous state so no clipping will happen
...
```



Animate
Amazing Things

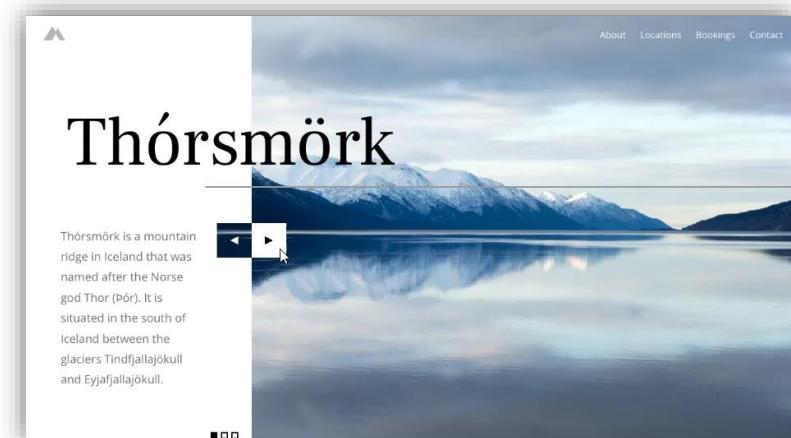
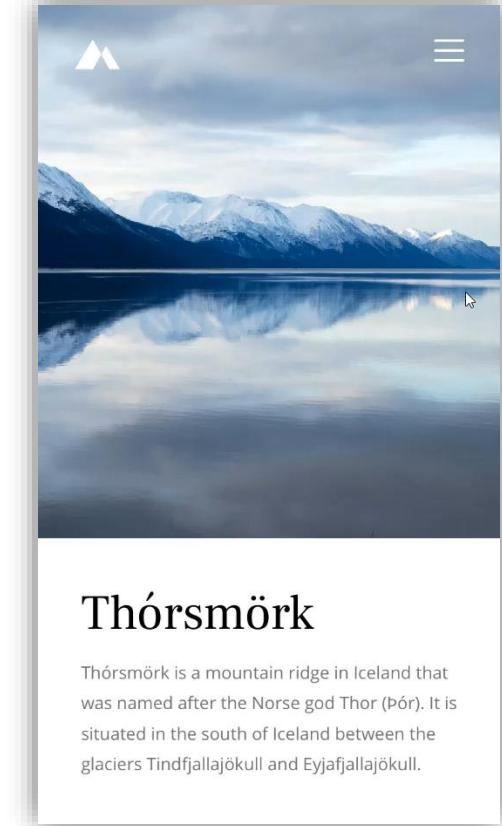


Mountain Design

Designer: Henry Price

Components:

- Xamarin.Forms
- SkiaSharp

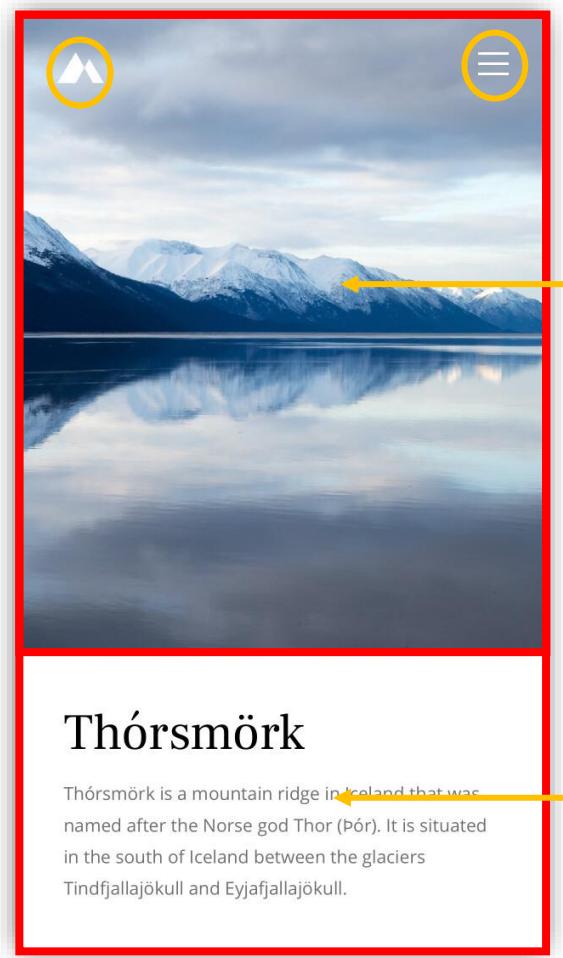


Github: <https://github.com/kphillpotts/mountainmobile>

Dribbble: <https://dribbble.com/shots/6727910-Mountain-Mobile-Animation>

Portrait Orientation Layout

Grid (2 rows)



Xamarin.Forms Images

SkiaSharp Surface

Xamarin.Forms Labels

Thórsmörk

Thórsmörk is a mountain ridge in Iceland that was named after the Norse god Thor (Þór). It is situated in the south of Iceland between the glaciers Tindfjallajökull and Eyjafjallajökull.



Animating SkiaSharp

- Create a custom animation

```
private void AnimateTransition()
{
    var startValue = 0;
    var endValue = ImageSkiaCanvas.CanvasSize.Width;

    var parentAnimation = new Animation();

    // animate values
    parentAnimation.Add(0.20, 0.70, new Animation(v => bandTranslationValues[0] = v, startValue, endValue, Easing.SinInOut));
    parentAnimation.Add(0.15, 0.65, new Animation(v => bandTranslationValues[1] = v, startValue, endValue, Easing.SinInOut));
    parentAnimation.Add(0.10, 0.60, new Animation(v => bandTranslationValues[2] = v, startValue, endValue, Easing.SinInOut));
    parentAnimation.Add(0.05, 0.55, new Animation(v => bandTranslationValues[3] = v, startValue, endValue, Easing.SinInOut));
    parentAnimation.Add(0.00, 0.50, new Animation(v => bandTranslationValues[4] = v, startValue, endValue, Easing.SinInOut));

    // perform refresh cycle
    parentAnimation.Add(0.00, 1.00, new Animation(v => ImageSkiaCanvas.InvalidateSurface(), 0, 1, Easing.Linear));

    // launch the animation
    parentAnimation.Commit(this, "TransitionAnimation", 16, 2000);
}
```

Animating SkiaSharp

- Handle the PaintSurface Event

```
private void SKCanvasView_PaintSurface(object sender, SkiaSharp.Views.Forms.SKPaintSurfaceEventArgs args)
{
    SKImageInfo info = args.Info;
    SKSurface surface = args.Surface;
    SKCanvas canvas = surface.Canvas;

    canvas.Clear();

    SKRect imageRect = new SKRect(0, 0, info.Width, info.Height);

    if (!isAnimating)...
    else
    {
        // work out our heights and widths of our rectangles
        var bandHeight = currentBitmap.Height / bandTranslationValues.Length;
        var bandWidth = currentBitmap.Width;

        for (int i = 0; i < bandTranslationValues.Length; i++)
        {
            var bandyOffset = i * bandHeight;

            DrawBitmapSliceAtOffset(canvas, currentBitmap,
                bandyOffset, (float)bandTranslationValues[i], bandWidth, bandHeight);

            DrawBitmapSliceAtOffset(canvas, offscreenBitmap,
                bandyOffset, (float)incomingBandTranslationValues[i], bandWidth, bandHeight);
        }
    }
}
```

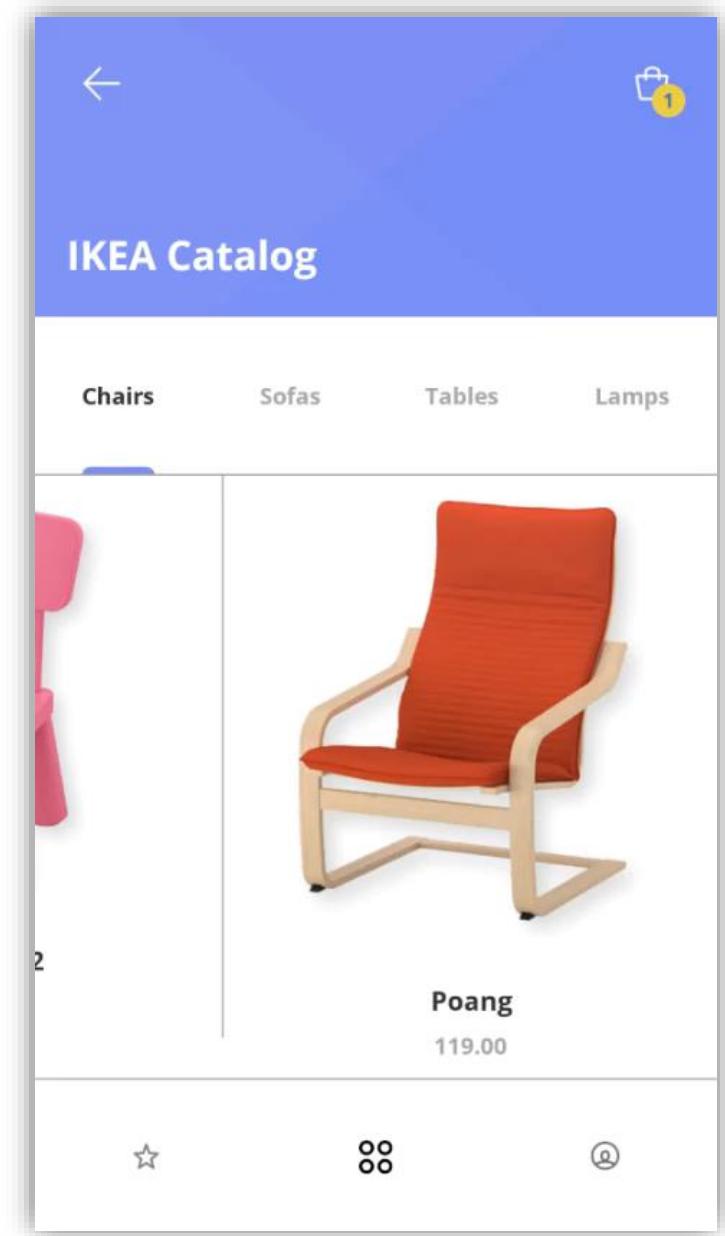
Multi Page Animations

Xamarin.Plugin.SharedTransitions

```
<Image  
    sharedTransition:Transition.Group="{Binding Name}"  
    sharedTransition:Transition.Name="SelectedItemImage"  
    Source="{Binding Image}" />
```

```
SharedTransitionNavigationPage.SetTransitionSelectedGroup(this, selectedItem.Name);  
Navigation.PushAsync(new ProductDetailsPage(selectedItem));
```

```
<Image  
    sharedTransition:Transition.Name="SelectedItemImage"  
    HorizontalOptions="End"  
    Source="{Binding Image}"  
    TranslationY="50"  
    VerticalOptions="End" />
```



Advanced animation with *Lottie*

For complex animations provided by designers

```
<lottie:AnimationView  
    x:Name="AnimationView"  
    Animation="funky_chicken.json"  
    AutoPlay="True"  
    HeightRequest="200"  
    HorizontalOptions="Center"  
    Loop="True"  
    Opacity="0"  
    TranslationX="-300"  
    VerticalOptions="Center"  
    WidthRequest="200" />
```

1 reference | Kym Phillipotts, 341 days ago | 1 author, 1 change

```
private async Task ValidLoginAnimation()  
{  
    await LoginContainer.FadeTo(0, 250);  
    AnimationView.Opacity = 1;  
    await AnimationView.TranslateTo(0, 0, 2000, Easing.SinInOut);  
    ConfettiView.Opacity = 1;  
    ConfettiView.Play();  
    ConfettiView.OnFinish += ConfettiView_OnFinish;  
    await Task.Delay(1000);  
    await AnimationView.TranslateTo(300, 0, 2000, Easing.SinInOut);  
    AnimationView.Opacity = 0;  
}  
}
```

Com.Airbnb.Xamarin.Forms.Lottie

- <https://useanimations.com/>
- <https://lottiefiles.com>



Thank You

LINKS AND SLIDES FROM THIS SESSION

github.com/kphillpotts/MonkeyFest2020



kymphillpotts.com



twitter.com/kphillpotts



twitch.tv/kymphillpotts



youtube.com/kphillpotts