

Make your images look great on all devices

Download class materials from
university.xamarin.com



Microsoft

Xamarin University

Information in this document is subject to change without notice. The example companies, organizations, products, people, and events depicted herein are fictitious. No association with any real company, organization, product, person or event is intended or should be inferred. Complying with all applicable copyright laws is the responsibility of the user.

Microsoft or Xamarin may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any license agreement from Microsoft or Xamarin, the furnishing of this document does not give you any license to these patents, trademarks, or other intellectual property.

© 2014-2018 Xamarin Inc., Microsoft. All rights reserved.

Xamarin, MonoTouch, MonoDroid, Xamarin.iOS, Xamarin.Android, Xamarin Studio, and Visual Studio are either registered trademarks or trademarks of Microsoft in the U.S.A. and/or other countries.

Other product and company names herein may be the trademarks of their respective owners.

Objectives

1. Include an image in your Application Bundle using Build Action
2. Display an image with UIImageView
3. Show an image at constant size across devices using points
4. Provide crisp images across all devices
5. Organize images using an asset catalog





Include an image in your Application Bundle using Build Action

Tasks

1. Add an asset to your application
2. Set an asset's Build Action



What is an asset?

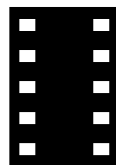
- ❖ An *asset* is a non-code file that is included in your application



Image



Audio



Video



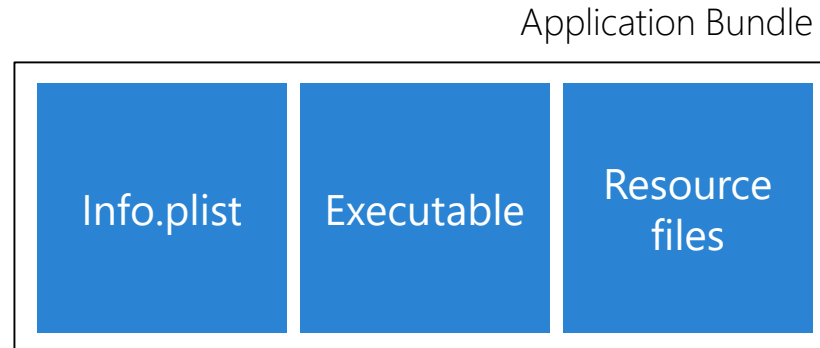
Data

A purple horizontal line with upward-pointing curly braces at each end, grouping the four asset types.

Common asset types

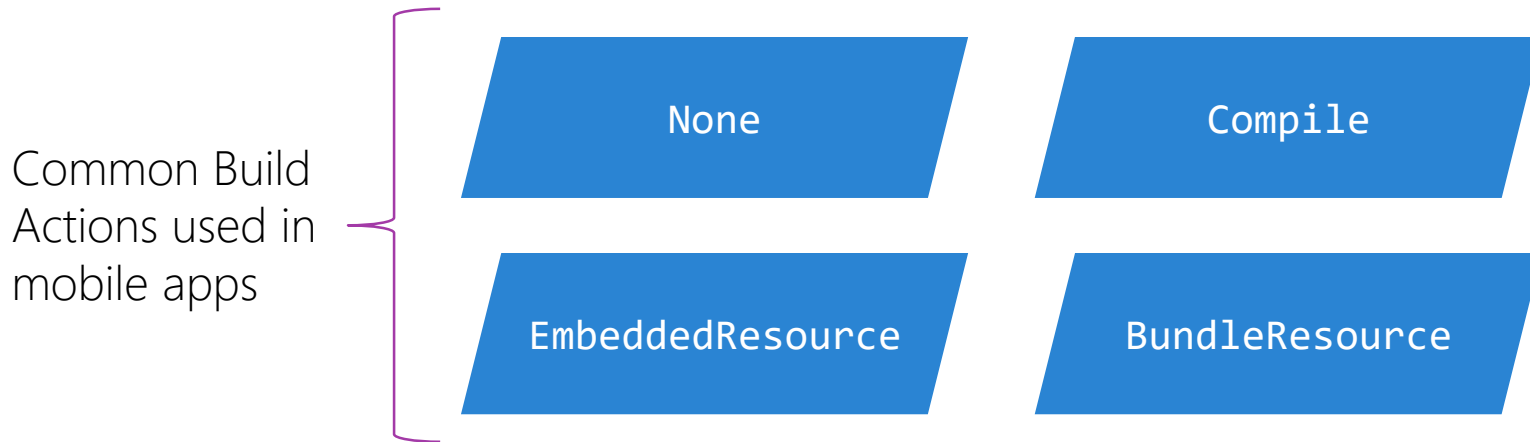
What is the Application Bundle?

- ❖ The *Application Bundle* is a folder that contains everything the application needs to run successfully



What is a Build Action?

- ❖ A *Build Action* is a piece of project metadata that controls how the build toolchain processes a file



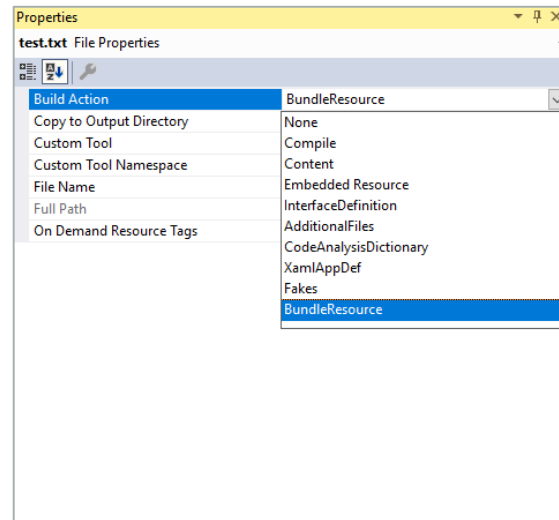
What is the BundleResource Build Action?

- ❖ The **BundleResource** Build Action instructs the build toolchain to add the file to the Application Bundle



How to set Build Action

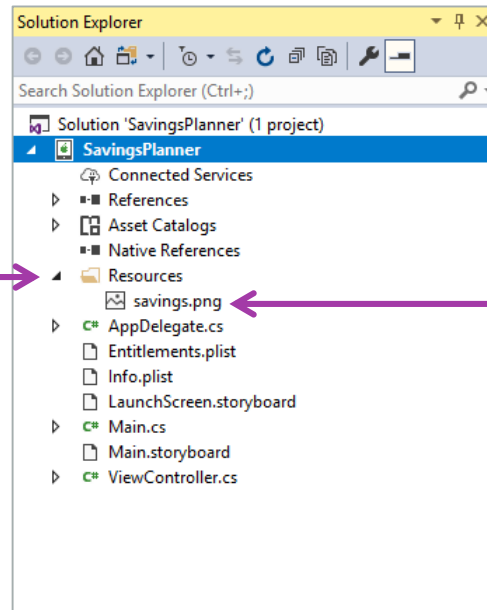
- ❖ Visual Studio assigns a default Build Action to every file added; you can also use the Visual Studio GUI to manually change



Drop-down menu lets
override the default settings
with your own choice

What is the Resources folder?

- ❖ The *Resources* folder is a standard directory to place application assets



Place application assets

Asset files added are set to
BundleResource automatically

Group Exercise

Include an image in your Application Bundle using Build Action

Summary

1. Add an asset to your application
2. Set an asset's Build Action





Display an image with UIImageView

Tasks

1. Create a **UIImage**
2. Visualize a **UIImage** with a **UIImageView**



What is UIImage?

- ❖ A **UIImage** is an object that stores and manages image data but does not have any visualization in the UI



savings.png



```
10110101011010110  
10110110110101010  
11011010110110101  
01101010110101011  
0101010010101011  
0101010010101011
```

UIImage

How to create a UIImage

- ❖ The **UIImage** class has several static methods that create a new instance

```
public class UIImage
{
    public static UIImage FromBundle(string name) { ... }
    public static UIImage FromFile(string filename) { ... }
    ...
}
```

Differ in their
caching strategy

Create a UIImage with FromBundle

- ❖ **UIImage.FromBundle** creates and caches a **UIImage**; this is efficient when the same image is added to the UI multiple times


```
public override void ViewDidLoad()  
{  
    ...  
    UIImage pigImage = UIImage.FromBundle("savings.png");  
}
```

↑
Filename is relative
to the Resources
folder

Create a UIImage with FromFile

- ❖ **UIImage.FromFile** creates a **UIImage** but the image is not cached; this is useful when the image appears in the UI once

```
public override void ViewDidLoad()
{
    ...
    UIImage pigImage = UIImage.FromFile("savings.png");
}
```

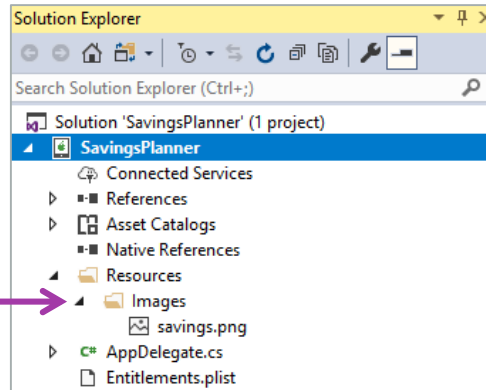


Filename is relative
to the Resources
folder

Resource subfolders

- ❖ You can create subfolders below **Resources** to organize assets

Images subfolder



```
var pigImage = UIImage.FromBundle("Images/savings.png");
```

What is a UIImageView?

- ❖ A **UIImageView** is a **UIView** derived type that visualizes a **UIImage**



savings.png



```
10110101011010110  
10110110110101010  
11011010110110101  
01101010110101011  
0101010010101011  
0101010010101011
```

UIImage



Connecting UIImageView to UIImage

- ❖ You create a **UIImageView** and specify which **UIImage** it should display

```
public class UIImageView : UIView
{
    public UIImageView(UIImage image);
    public UIImage Image { get; set; }
    ...
}
```

← Can set at creation

← Can set after creation

How to create a UIImageView

- ❖ There are two ways to create a **UIImageView**



Can create in code



Can create using the Designer

Create UIImageView in code

- ❖ You can create a **UIImageView** in code which is useful when you need to dynamically display an image

```
public override void ViewDidLoad()
{
    ...
    var pigImageView = new UIImageView();
    pigImageView.Image = UIImage.FromBundle("savings.png");
    pigImageView.Frame = new CGRect(...);
    View.AddSubview(pigImageView);
}
```

Create the view →

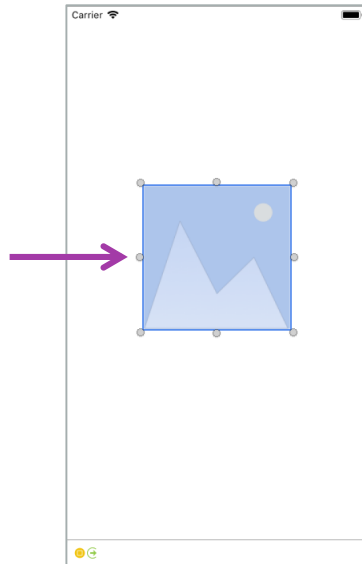
Specify the image to display →

Load it into the UI →

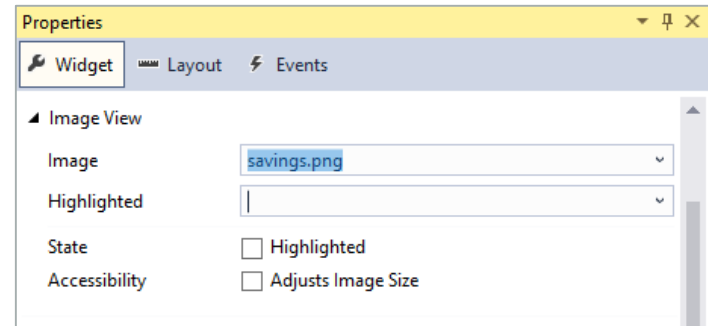
Create UIImageView in Designer

- ❖ You can create a **UIImageView** in the Designer which is common when displaying a static image

Drag **UIImageView**
from the toolbox



Set the image using the properties pane
(**UIImage** object created automatically)





Individual Exercise

Display an image with UIImageView

Summary

1. Create a **UIImage**
2. Visualize a **UIImage** with a **UIImageView**





Show an image at constant size
across devices using points

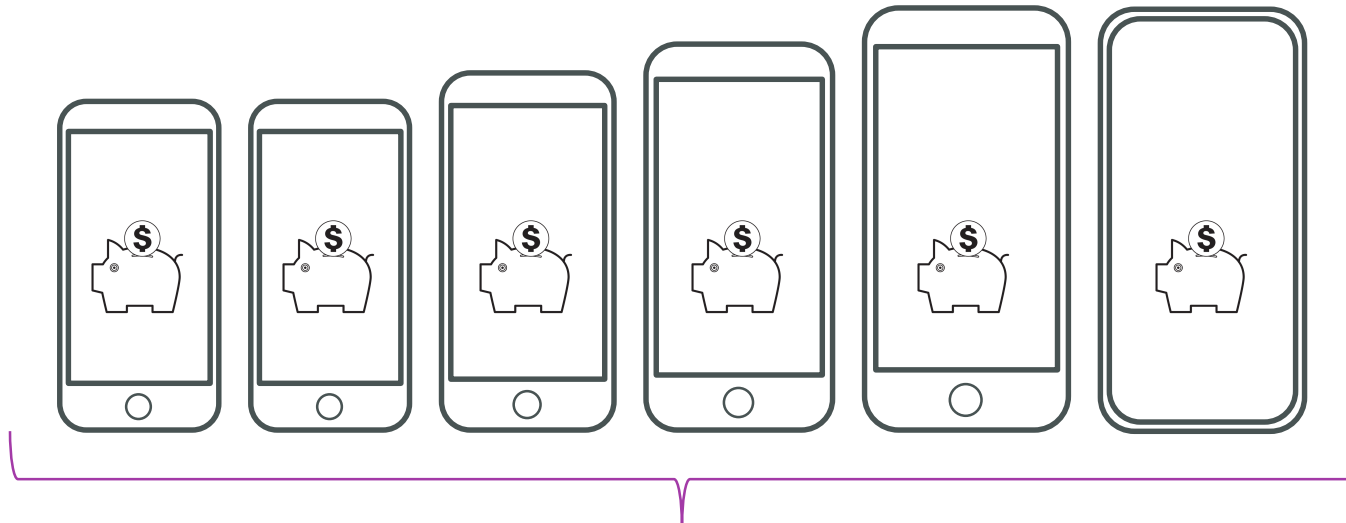
Tasks

1. Size your views using points



Motivation

- ❖ UI designers often display images at the same physical size across all devices to avoid scaling and keep images crisp

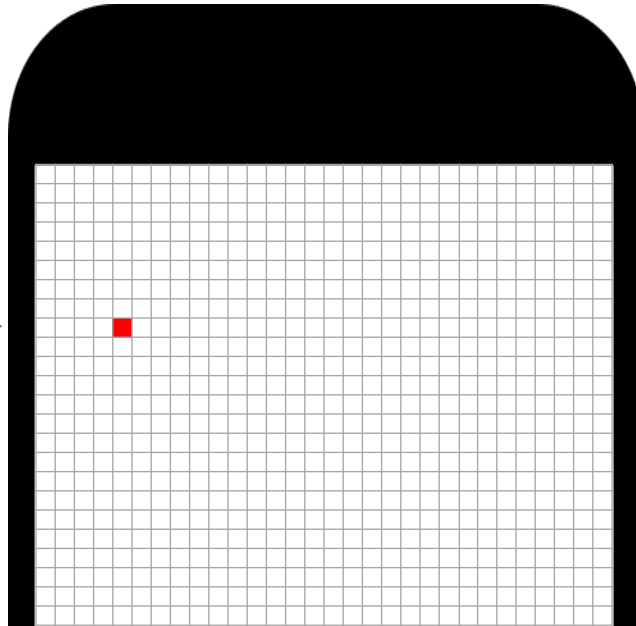


We want this image to be 1" x 1" everywhere

What is a pixel?

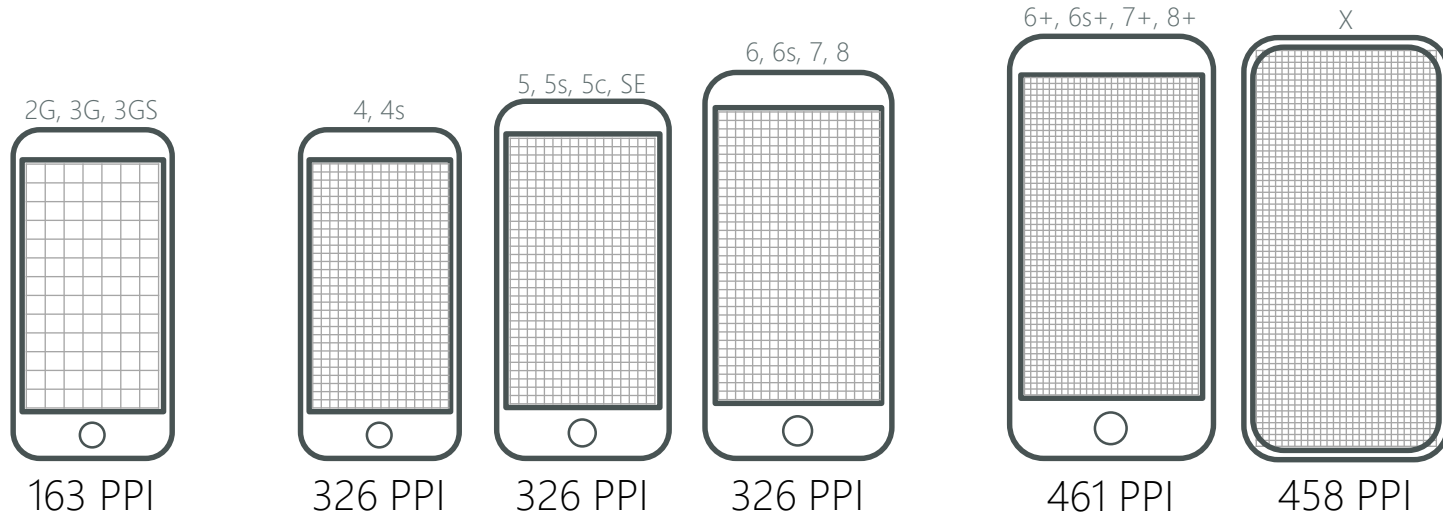
- ❖ A *pixel* (picture element) is an indivisible unit of an image or screen; the entire pixel is rendered in the same color

Screen is a
grid of pixels →



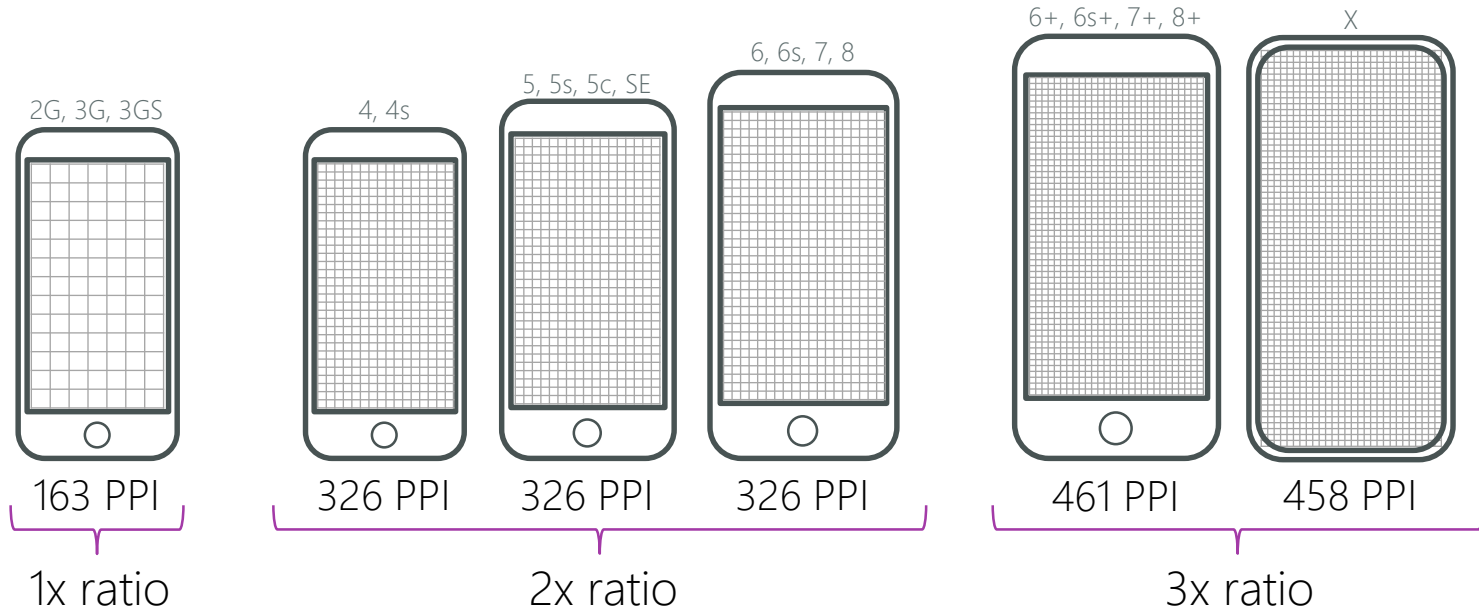
What is pixel density?

❖ *Pixel density* is the number of physical pixels per inch (PPI) on a screen



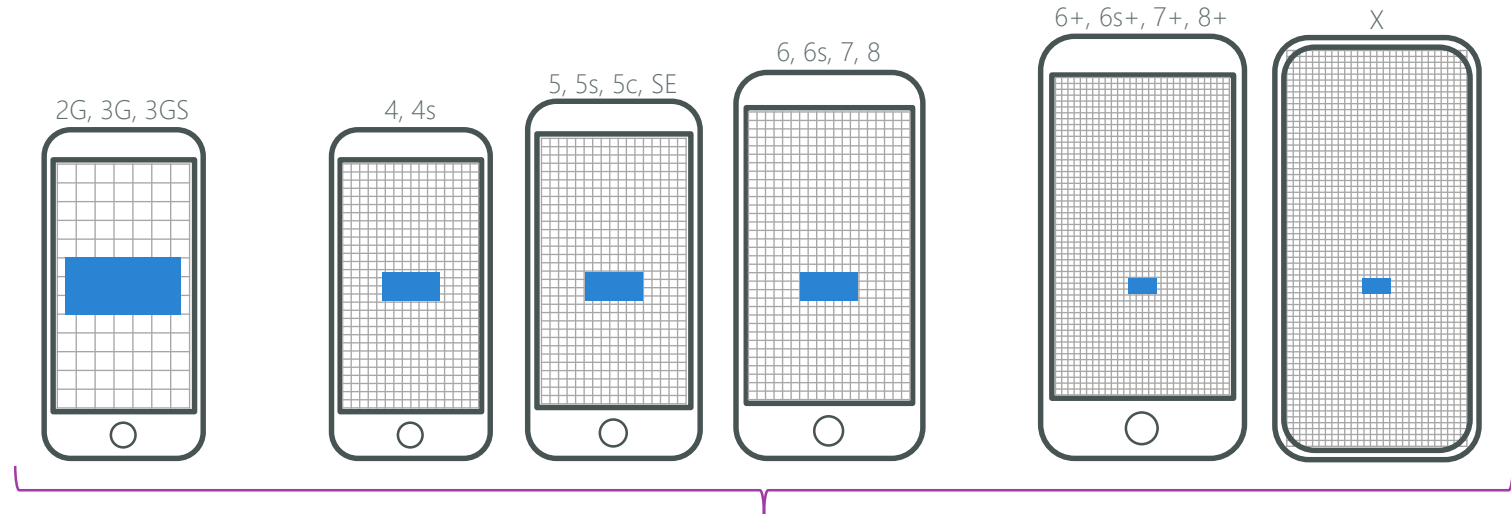
What is pixel density ratio?

- ❖ *Pixel density ratio* is the ratio of a screen's pixel density to the pixel density of the original iPhone (rounded to the nearest integer)



No pixel sizing

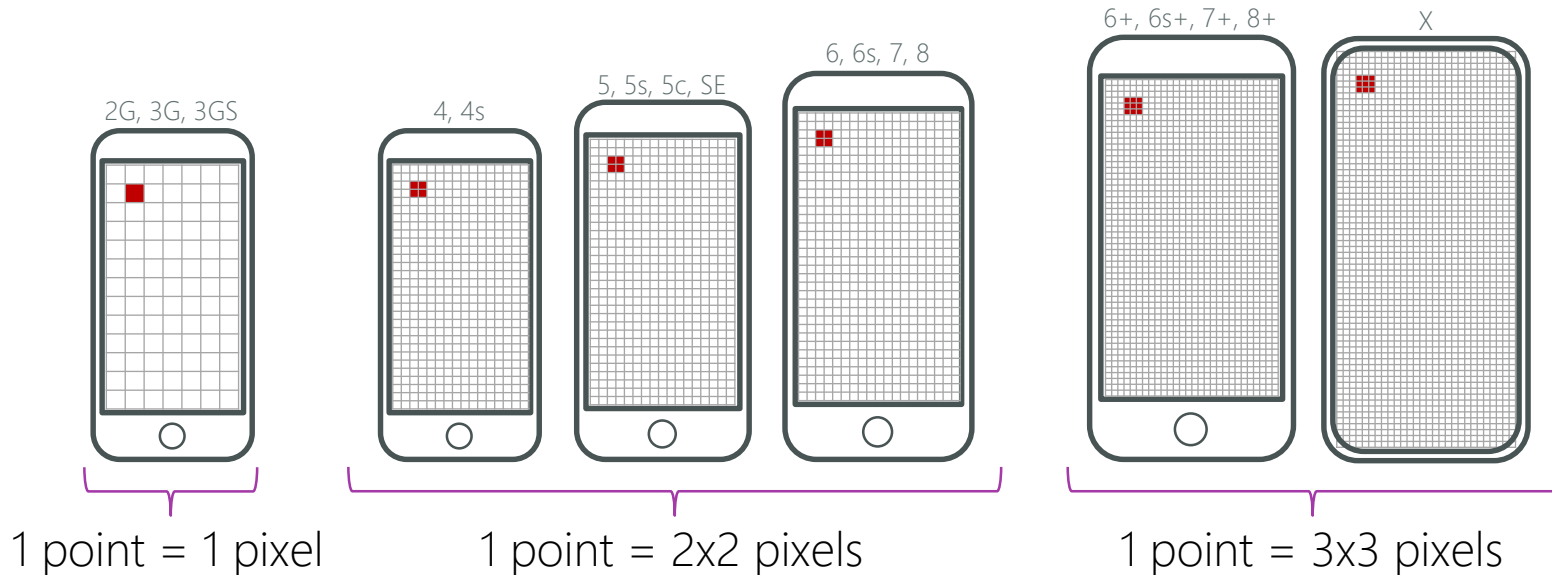
- ❖ Sizing an image in pixels would yield different sizes on different screens



A 150x100 pixel shape shown on all screens

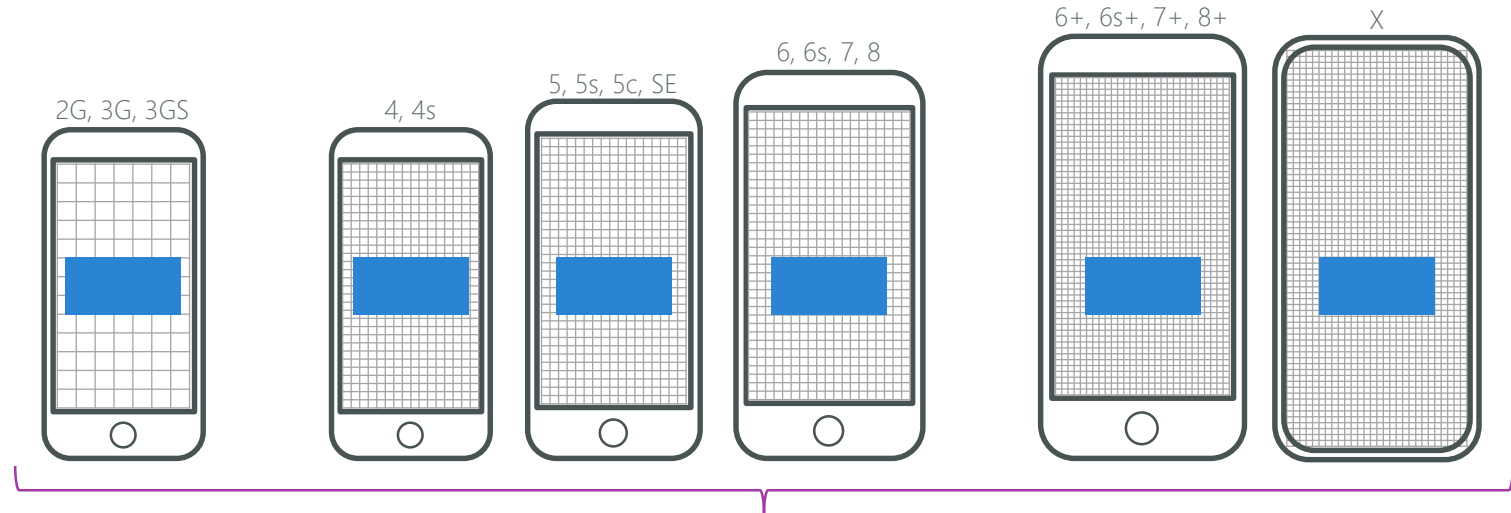
What is a point?

- ❖ A *point* is an abstract unit of measure that is converted to pixels at runtime by multiplying by the pixel-density ratio



Use point sizing

- ❖ Specifying sizes in points ensures the image occupies the same physical amount of space on all screens



A 150x100 point shape shown on all screens



Individual Exercise

Show an image at constant size across all devices using points

Summary

1. Size your views using points





Provide crisp images
across all devices



Xamarin
University

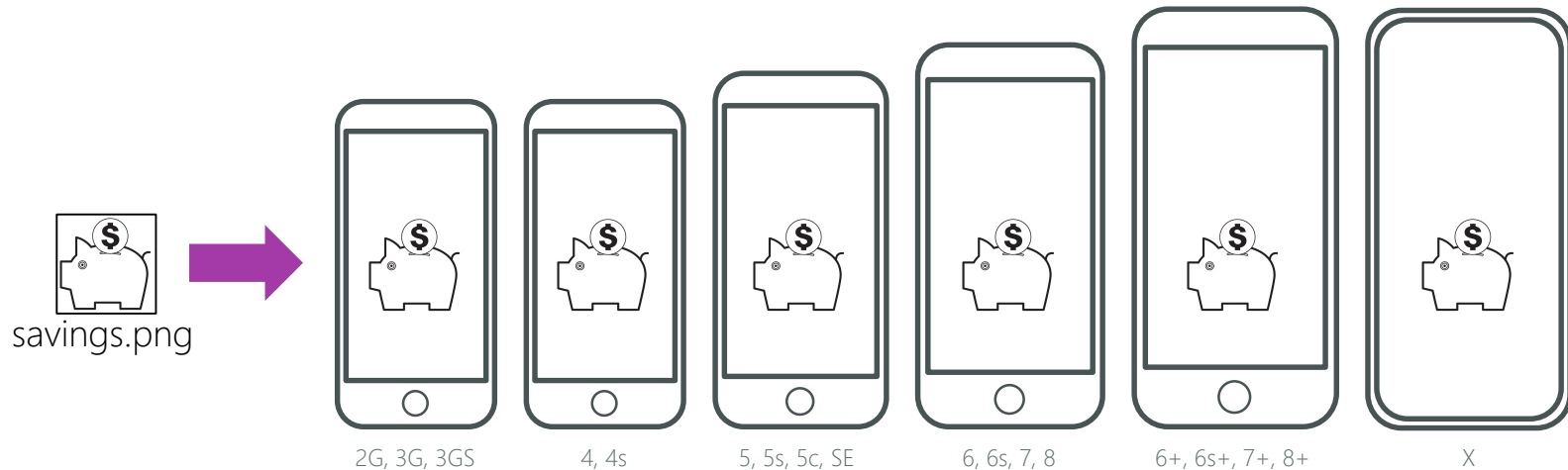
Tasks

1. Provide a multi-resolution image
2. Display a multi-resolution image in a **UIImageView**



Motivation

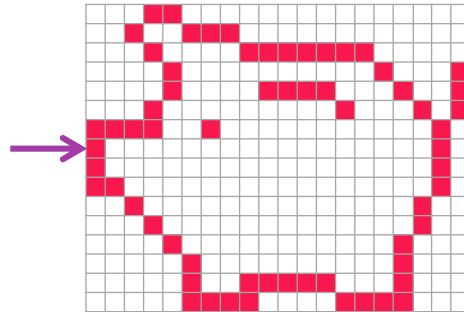
- ❖ It is difficult to make your images look good everywhere because iOS devices have different screen sizes and resolutions



What is a raster image?

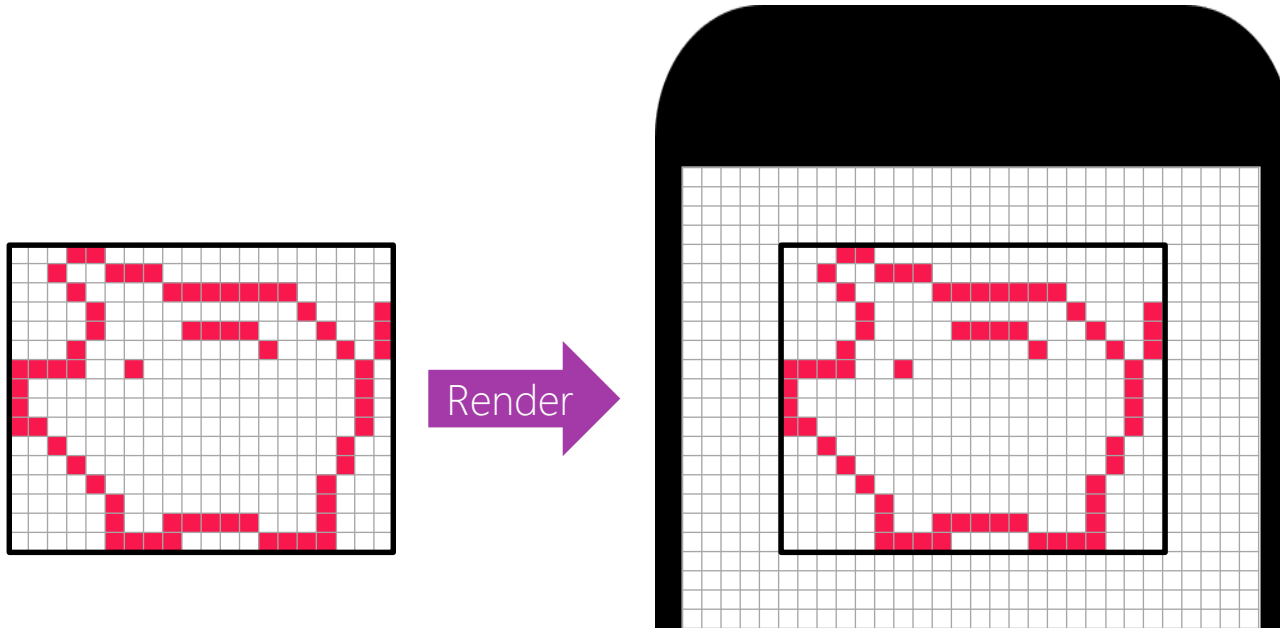
- ❖ *Raster* is an image format that stores the color of every pixel; raster file formats include Bitmap, JPEG, PNG, etc.

Raster image formats
would record that this
pixel is R248 G24 B77



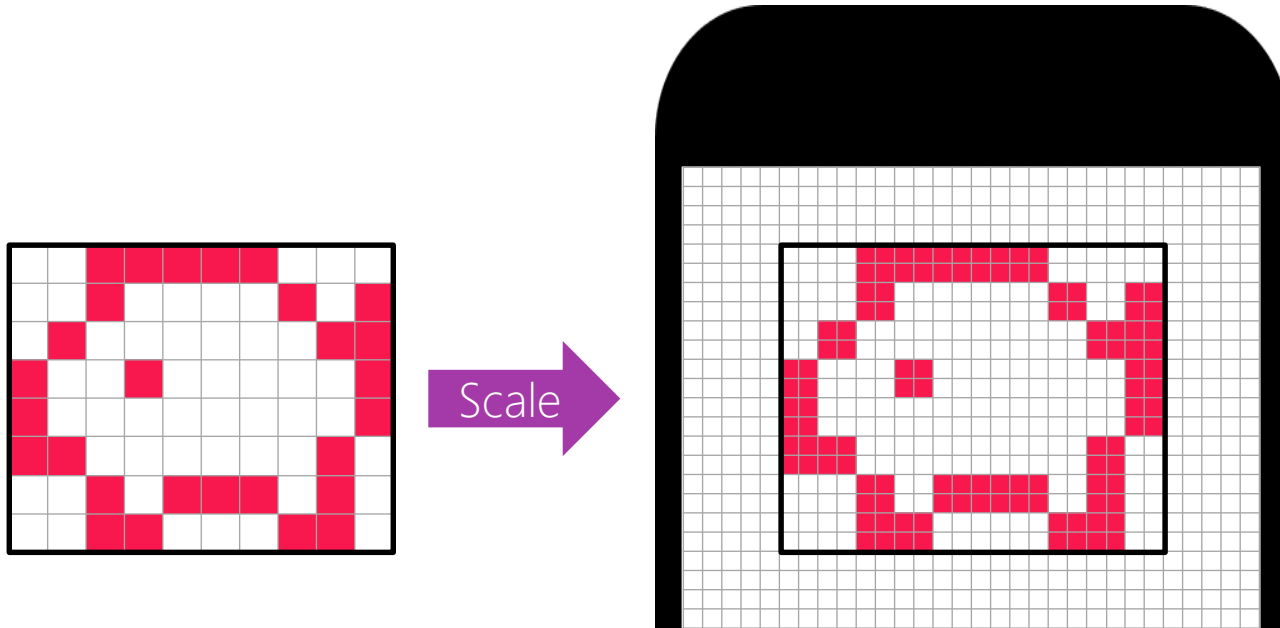
Simple raster image display

- ❖ When the number of pixels in the raster image matches the display area, then the display algorithm is one-to-one pixel mapping



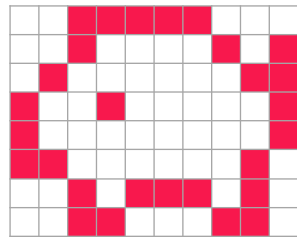
Raster image scaling

- ❖ When the number of pixels in the raster image does not match the number of pixels in the display area then the image must be scaled

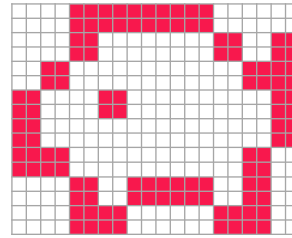


What is upscaling?

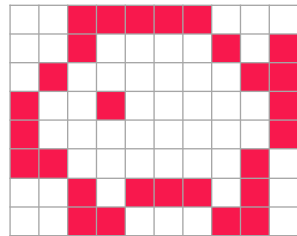
- ❖ *Upscaling* is the process of converting an image to a higher resolution (a larger number of pixels) using one of many possible algorithms



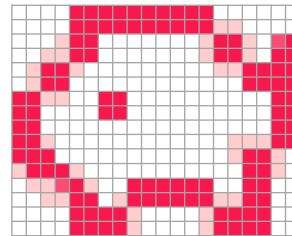
Upscale



Some algorithms do pixel doubling which can look jagged



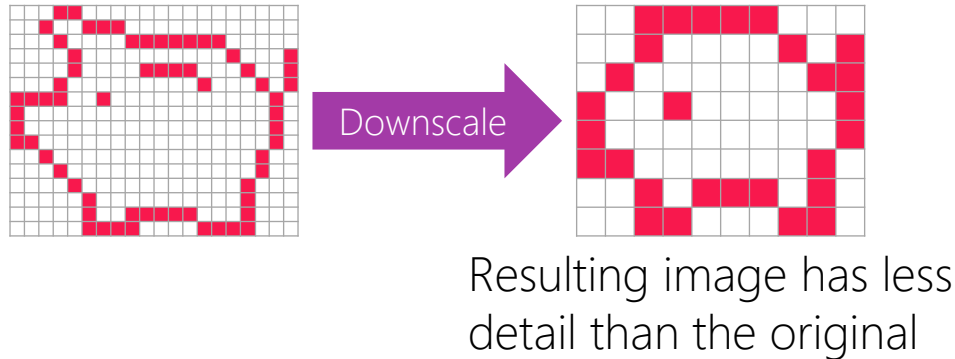
Upscale



Some algorithms interpolate to smooth jagged elements which can look less crisp

What is downscaling?

- ❖ *Downscaling* is the process of converting an image to a lower resolution (a smaller number of pixels) using one of many possible algorithms



How to generate crisp images [steps]

❖ Recommended steps for generating crisp images on all devices

1

Determine physical size in points

2

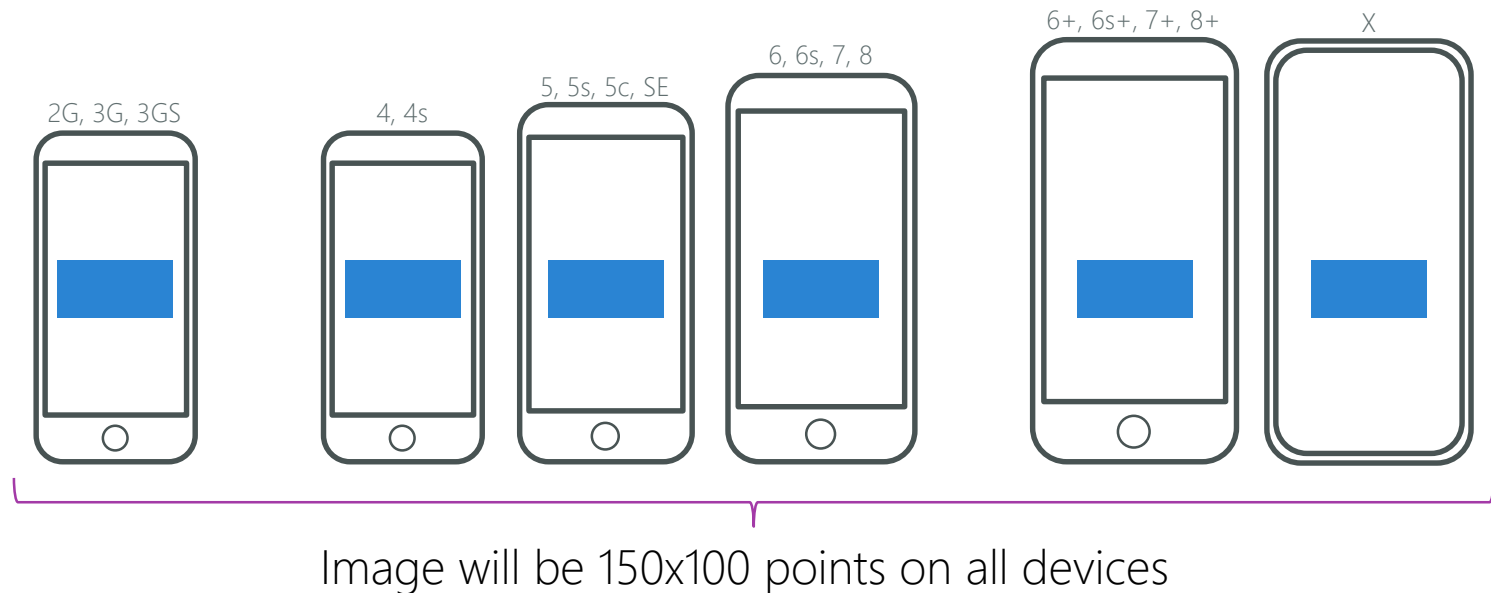
Calculate number of pixels

3

Generate images with exact pixel counts

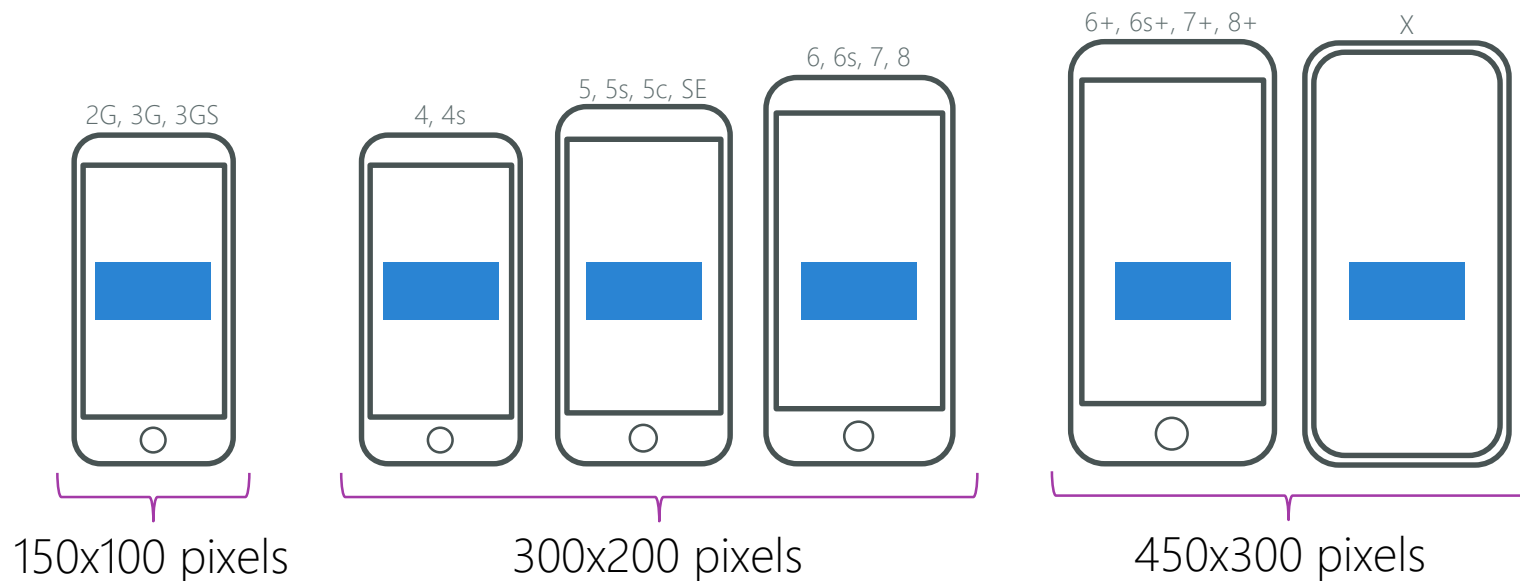
How to generate crisp images [step 1]

- ❖ Determine the desired physical size in points



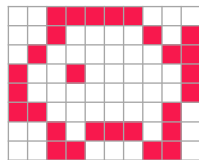
How to generate crisp images [step 2]

- ❖ Calculate the number of pixels for each pixel density ratio

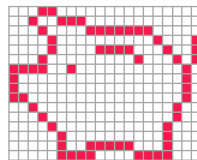


How to generate crisp images [step 3]

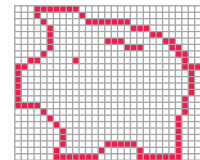
- ❖ Generate images with the exact pixel counts



150x100 pixels



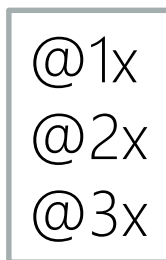
300x200 pixels



450x300 pixels

How to provide multiple images

- ❖ iOS supports two ways to provide multiple resolution images



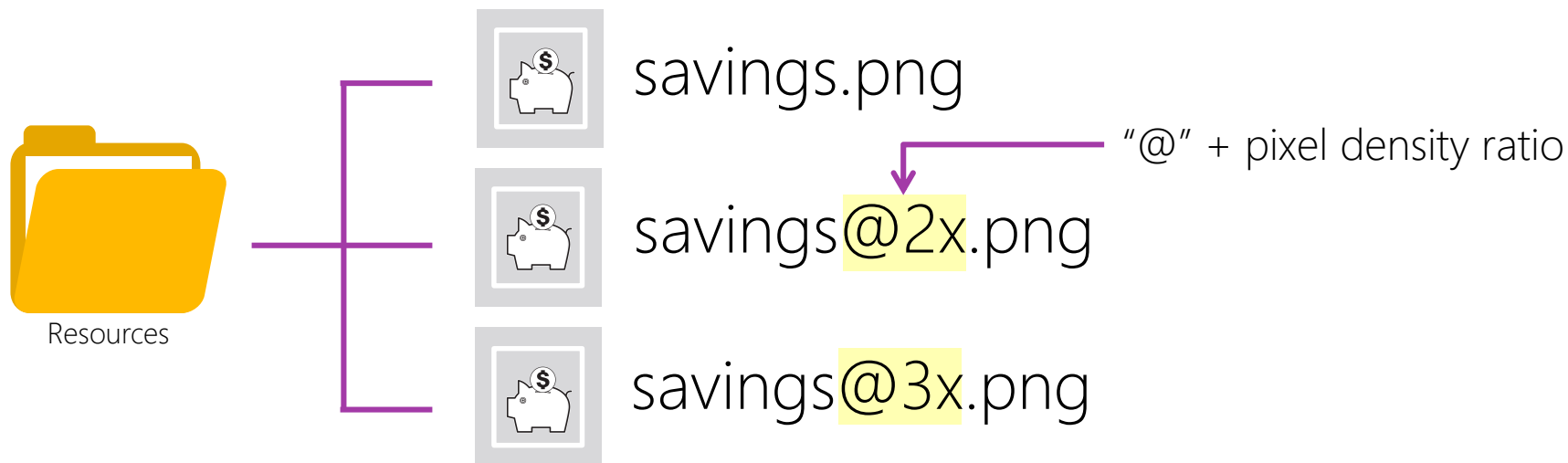
File naming convention
(covered in this section)



Image Set with Asset Catalog
(covered later in the course)

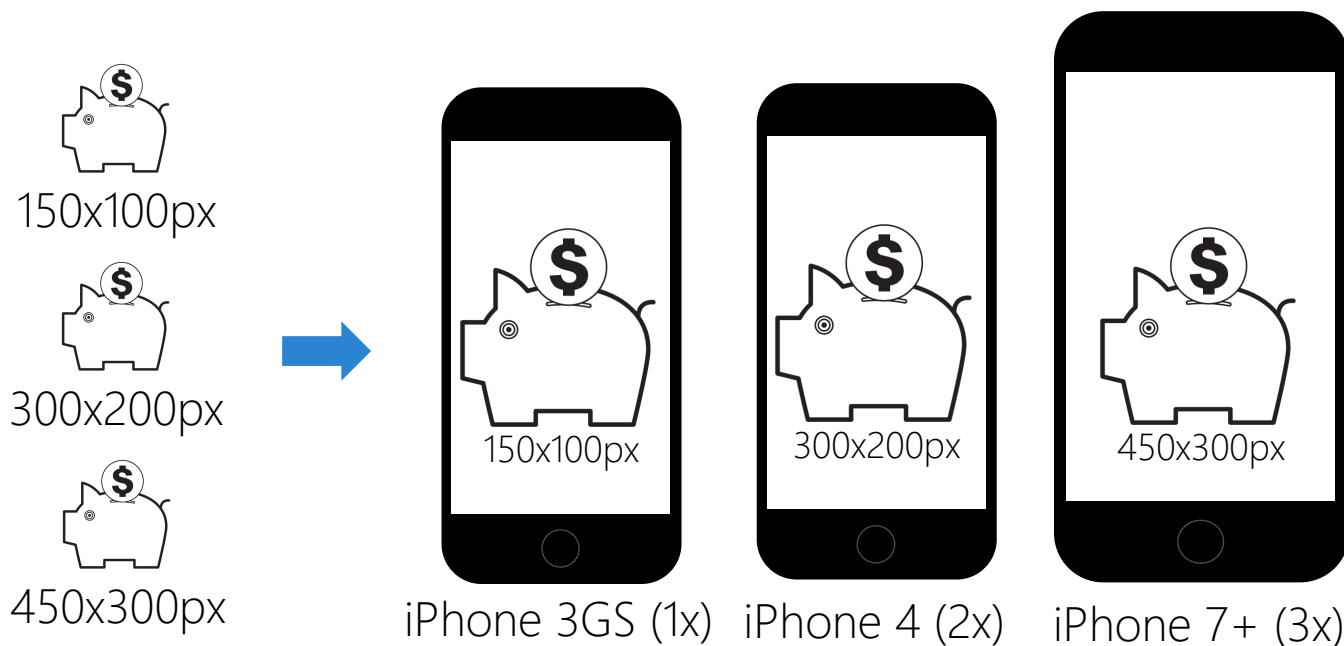
Adding multiple images

- ❖ Place multiple versions of an image into the resources folder and append the filename with the pixel density ratio



Runtime image selection

- ❖ iOS selects an image based on the pixel density ratio of the runtime device



Using a multi-resolution image [code]

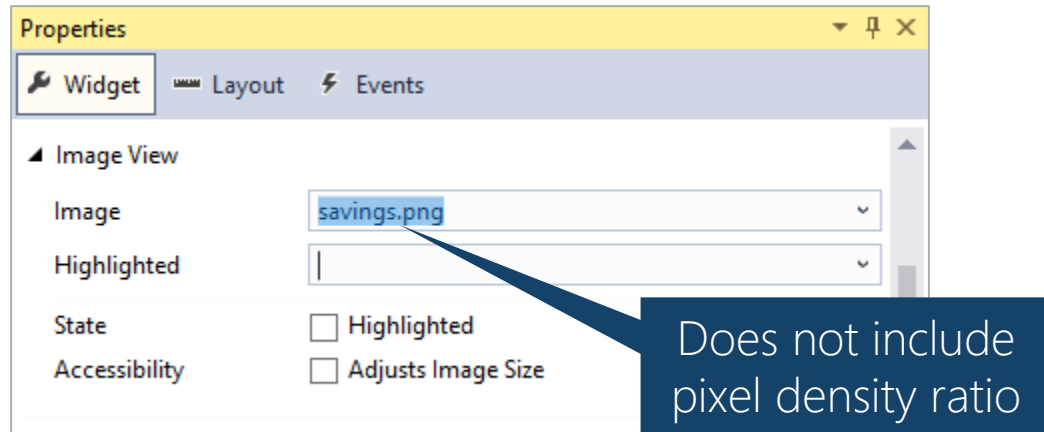
- ❖ A multi-resolution image is accessed by its name without the pixel density ratio

```
public override void ViewDidLoad()  
{  
    ...  
    UIImage pigImage = UIImage.FromBundle("savings.png");  
  
    pigImageView.Image = pigImage;  
}
```

Does not include
pixel density ratio

Using a multi-resolution image [designer]

- ❖ The designer will automatically detect multi-resolution images and display a single element representing the set





Individual Exercise

Provide crisp images across all devices



Xamarin
University

Summary

1. Provide a multi-resolution image
2. Display a multi-resolution image in a **UIImageView**



Organize images using an asset catalog

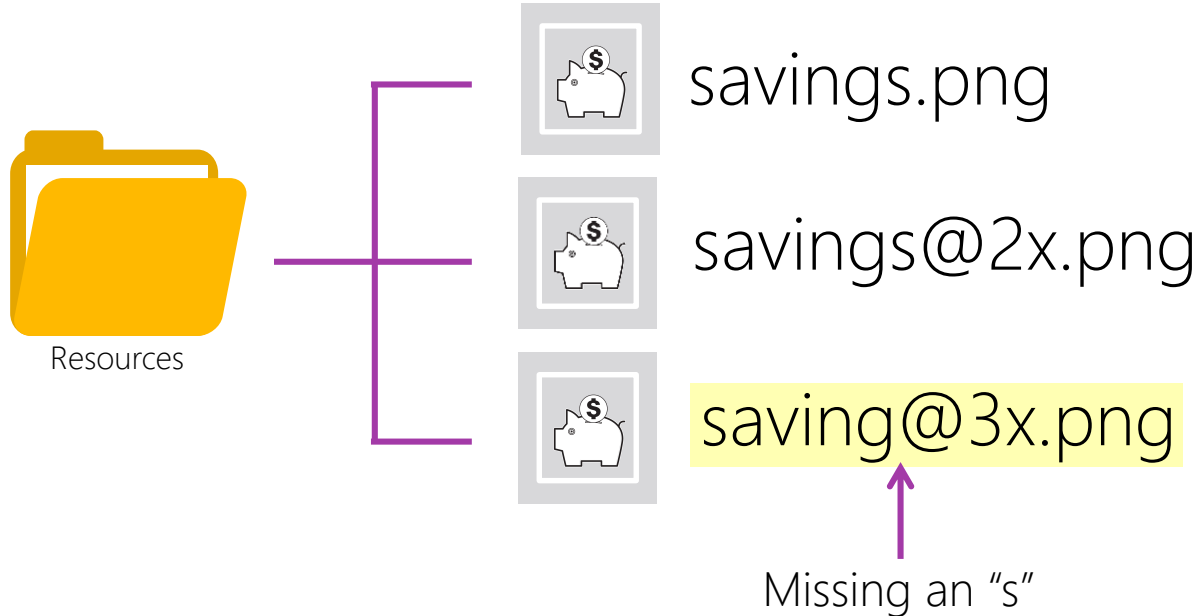
Tasks

1. Create an Asset Catalog
2. Create an Image Set
3. Load images into an Image Set
4. Display an Image Set in a **UIImageView**



Motivation

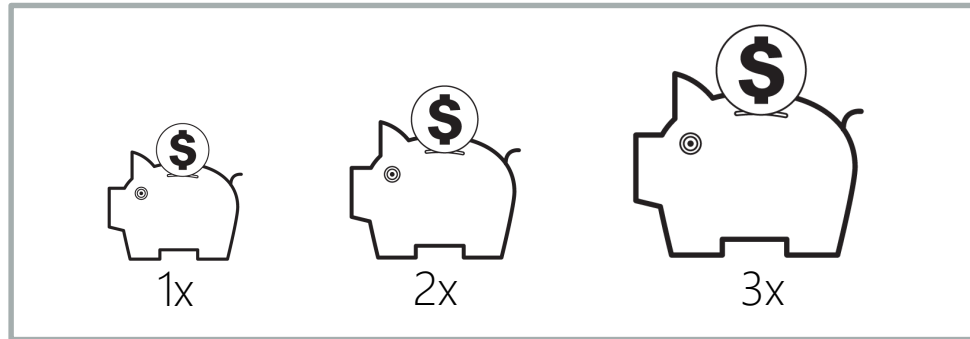
- ❖ Specifying a set of images by name is error prone



What is an image set?

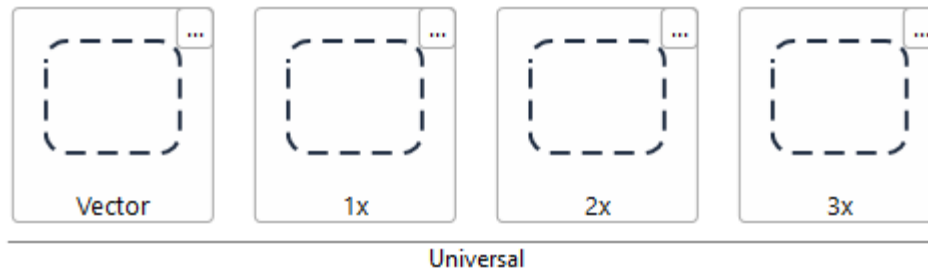
- ❖ An *image set* is a container that groups together all the versions of an image needed to support various devices and scale factors.

Pig image set



Universal image set

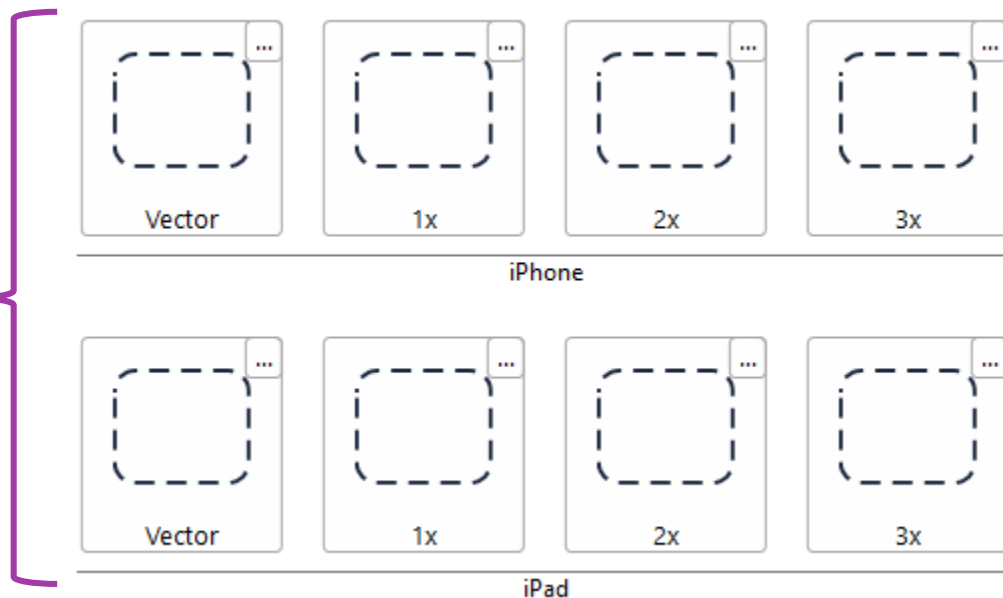
- ❖ A *universal* image set is an image set that contains one set of images that will be used on all devices



Device-specific image set

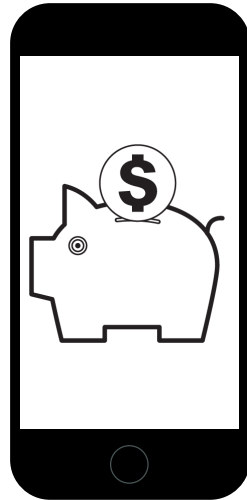
- ❖ A device-specific image set is an image set that contains one set of images for each targeted device

Could provide
different images
per device

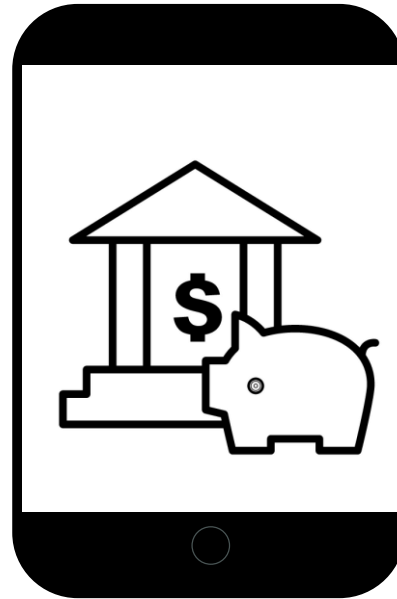


Which type of image set to choose?

- ❖ Common practice is to prefer a universal image set unless you require device specific images



iPhone



iPad

← Different image on iPad

What is an Asset Catalog?

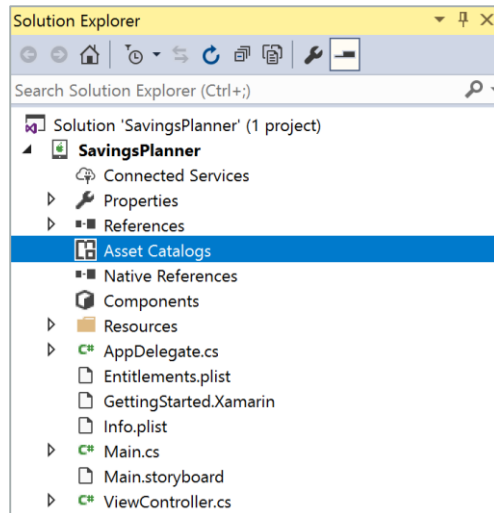
- ❖ An *Asset Catalog* is a collection of image sets (and other resource types)



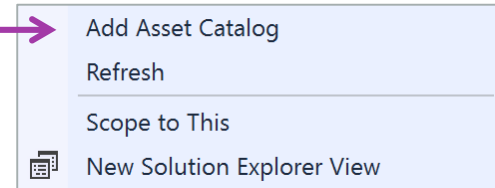
How to create an Asset Catalog

- ❖ Visual Studio provides a GUI to create an Asset Catalog from the Solution Explorer

1. Right click →



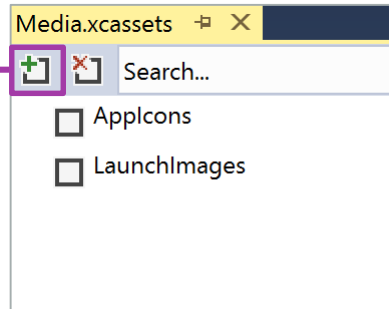
2. Select →



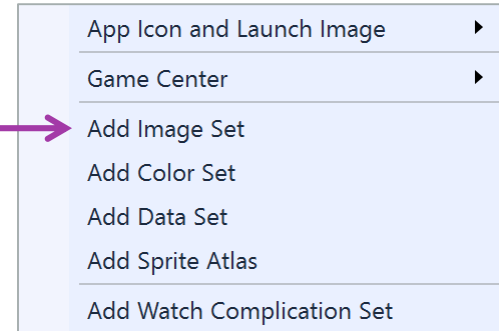
How to create an image set

- ❖ Visual Studio provides a GUI to create an image set inside an Asset Catalog

1. Click



2. Select



Adding images to an image set

- ❖ Browse and select the corresponding image for each pixel density ratio

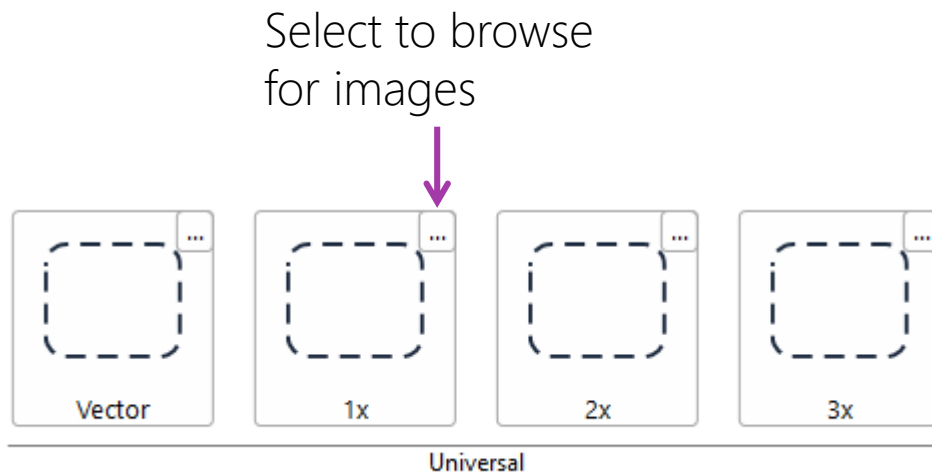


Image set options

- ❖ Image sets provide several options to customize its functionality

Set device-specific
image set



Name of image set



Set universal
image set



Properties	
Xamarin.VisualStudio.iOS.AssetCatalog.XCsetImageProperties	
1 GB False	
2 GB False	
3 GB False	
4 GB False	
AppleTV False	
Direction Fixed	
Gamut Any	
iPad False	
iPhone False	
Metal 1v2 False	
Metal 2v2 False	
Metal 3v1 False	
Metal 3v2 False	
Name savings	
On Demand Resource Tags	
Render As Default	
Universal True	

Applying an image set [code]

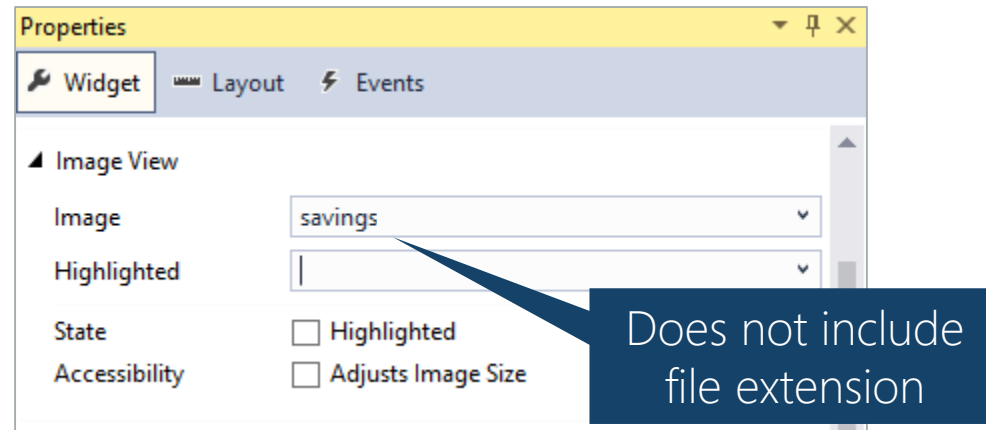
- ❖ An image set is applied by using its name without a file extension

```
public override void ViewDidLoad()  
{  
    ...  
    UIImage pigImage = UIImage.FromBundle("savings");  
  
    pigImageView.Image = pigImage;  
}
```

Image set name with
no file extension

Applying an image set [designer]

- ❖ The designer will automatically display all detected image sets by name





Individual Exercise

Organize images using an asset catalog

Summary

1. Create an Asset Catalog
2. Create an Image Set
3. Load images into an Image Set
4. Display an Image Set in a **UIImageView**



Thank You!

Please complete the class survey in your profile:
university.xamarin.com/profile