AND241

# Gestures in Android

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, trademarked, 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.

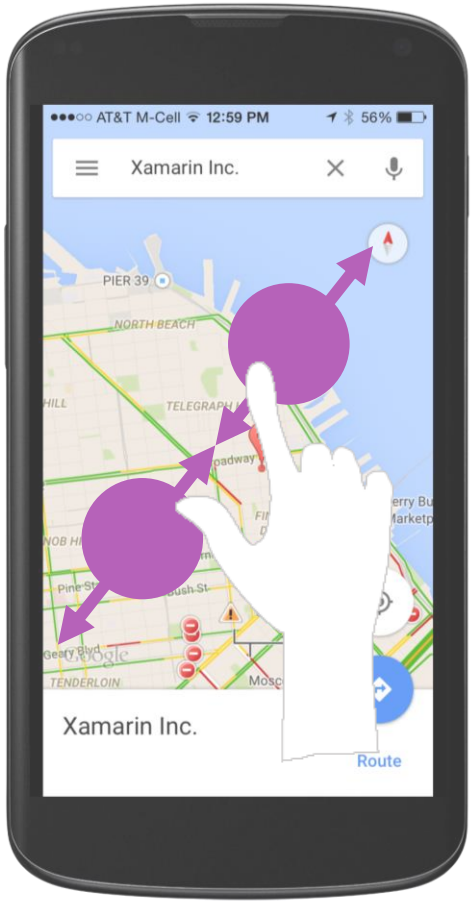


Xamarin, MonoTouch, MonoDroid, Xamarin.iOS, Xamarin.Android, Xamarin Studio, and Visual Studio are either registered trademarks or trademarks of Microsoft in the U.S.A. and/or other countries.

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

# Objectives

1. Utilize built-in gestures
2. Create custom gestures programmatically

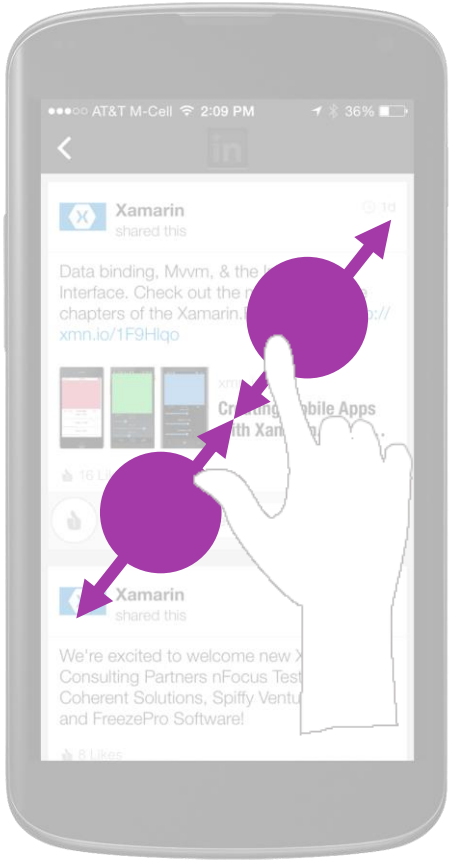


# Utilize built-in gestures

Xamarin University

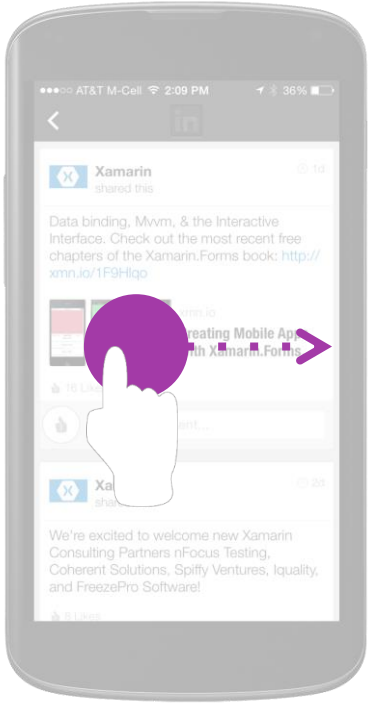
# Tasks

1. Define gestures
2. Implement the gesture detector
3. Handle gestures
4. Discuss scaling and double tap gestures



# What is a gesture?

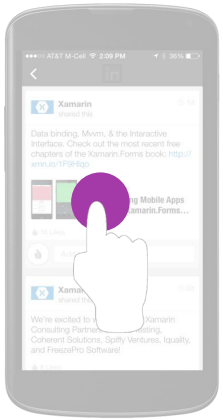
- A gesture is a hand-drawn shape on the touch screen
- Gestures are detected by an app from a sequence of points



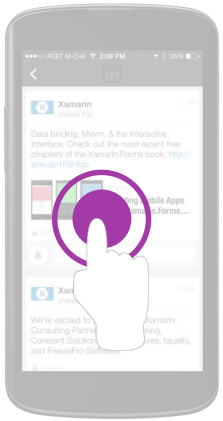

Interpreted as: **`OnFling`**

# Built-in gestures

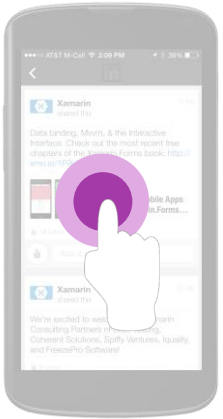
- Android supports several built-in gestures



**`OnDown`**
Down



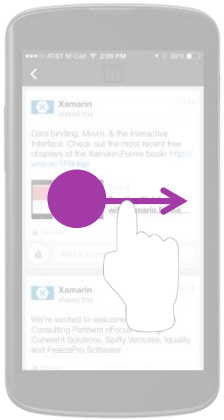
**`OnShowPress`**
Down, hold (~100ms)



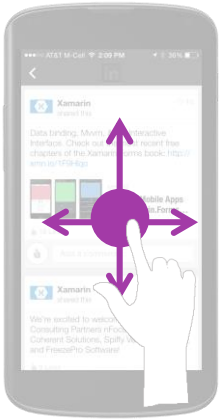
**`OnLongPress`**
Down, hold (~600ms)

# Built-in gestures

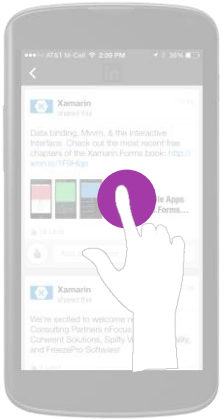
- Android supports several built-in gestures



**`OnFling`**
Down, move, up



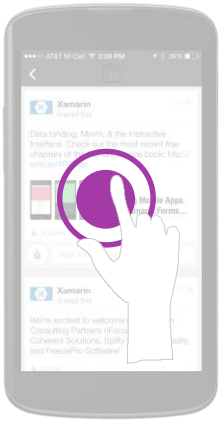
**`OnScroll`**
Move



**`OnSingleTapUp`**
Down, up

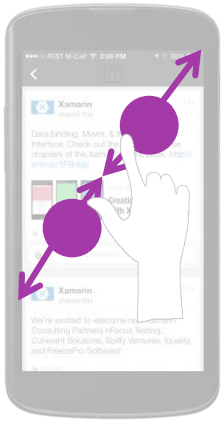
# Built-in gestures

- Android supports several built-in gestures

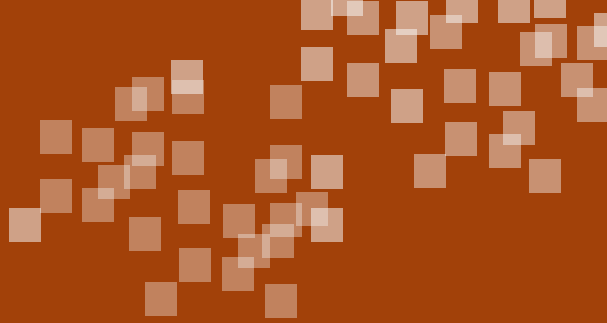


`OnDoubleTap`
Down, up, down

?



`OnScale`
Two fingers down,
move together/apart

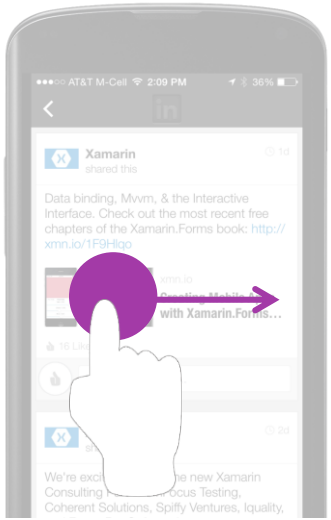


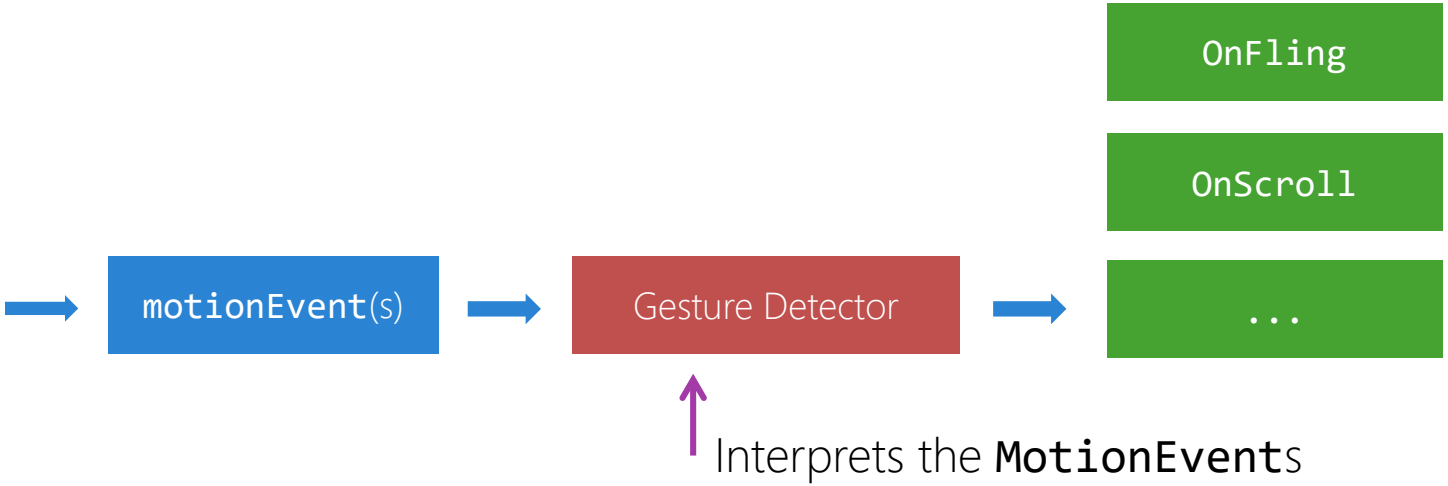
# Demonstration

GestureListener

# Interpreting a gesture

- Android *gesture detectors* interpret a sequence of **`MotionEvent`**s as a gesture



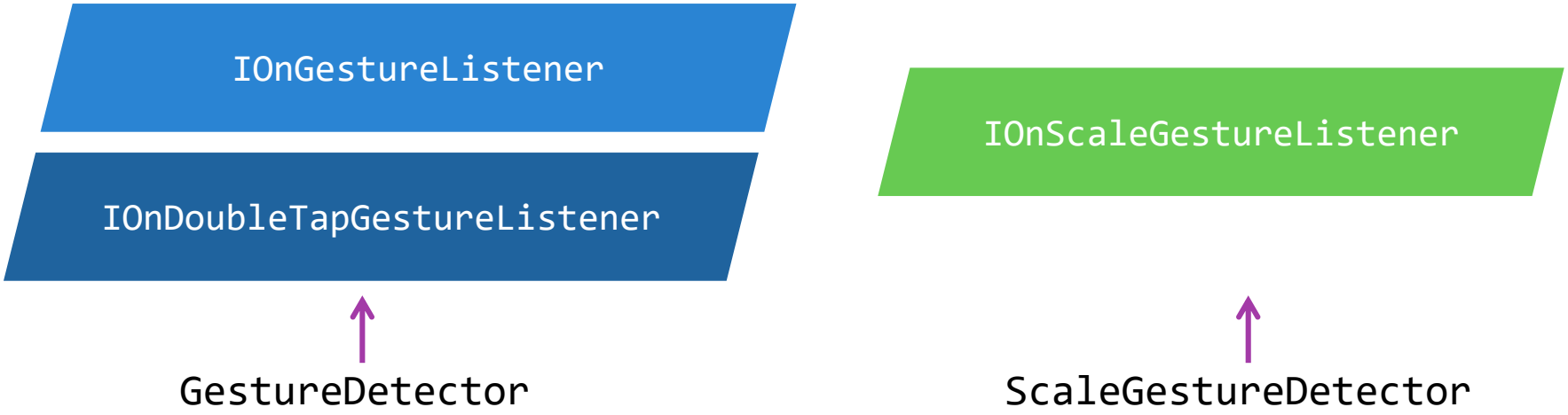


# Two types of Gesture Detectors

- Android provides two built-in Gesture Detectors

# Listener interface pattern

- The gesture detectors use a listener interface pattern to report when gestures have been detected




Depending on the gesture you are checking for, you will use one of three gesture listener interfaces with their corresponding **GestureDetector**

# IOnGestureListener

- The **`IOnGestureListener`** defines the callback methods for six of the most common gestures

# IOnDoubleTapGestureListener

- Implement the **`IOnDoubleTapGestureListener`** for double tap gestures

# IOnScaleGestureListener

- The **`IOnScaleGestureListener`** is used to listen for scaling/pinch and zoom gestures


For scaling gestures implement the **`ScaleGestureDetector`** with the **`IOnScaleGestureListener`**

# Flash Quiz

Xamarin University

# Flash Quiz

① Which gesture detector object is used with `IOnDoubleTapGestureListener`?

a) DoubleTapGestureDetector

b) GestureDetector

c) ScaleGestureDetector

# Flash Quiz

① Which gesture detector object is used with `IOnDoubleTapGestureListener`?

a) DoubleTapGestureDetector

b) GestureDetector

c) ScaleGestureDetector

# Flash Quiz

② The gesture detector is responsible for _____

a) forwarding the motion events

b) generating the motion events

c) interpreting the motion events

# Flash Quiz

② The gesture detector is responsible for _____

a) forwarding the motion events

b) generating the motion events

c) interpreting the motion events

# Listen for gestures

- To create a gesture detector you must implement one of the gesture listener interfaces

```
public class MainActivity : Activity, IOnGestureListener
{
    ...
}
```

The interface is often implemented on an Activity

# IOnGestureListener methods

- **`IOnGestureListener`** requires 6 methods; 1 for each of it's supported gestures

```
bool OnDown (MotionEvent e) { ... }
bool OnFling (MotionEvent e) { ... }
void OnLongPress (MotionEvent e) { ... }
bool OnScroll (MotionEvent e) { ... }
void OnShowPress (MotionEvent e) { ... }
bool OnSingleTapUp (MotionEvent e) { ... }
```

# Instantiate the Gesture Detector

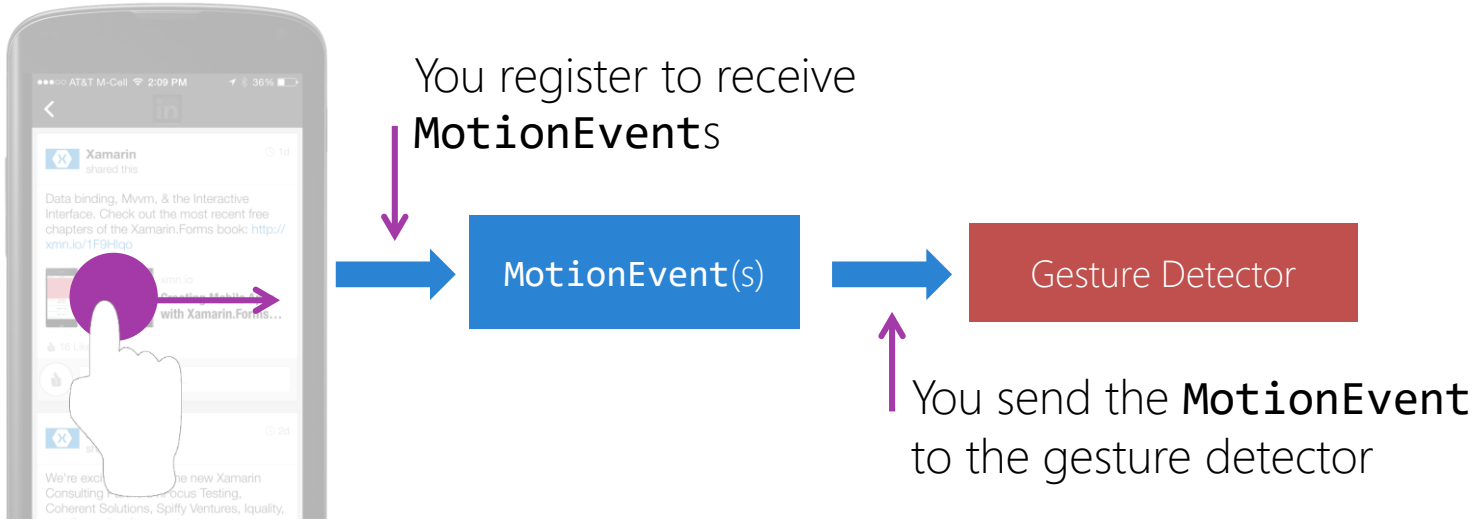
- Instantiate the Gesture Detector and pass the gesture listener as a parameter in the constructor

```
protected override void OnCreate (Bundle bundle)
{
    ...
    gestureDetector = new GestureDetector (context: this, listener: this);
}
```

A context and a listener are required by the Gesture Detector

# Forward the MotionEvent

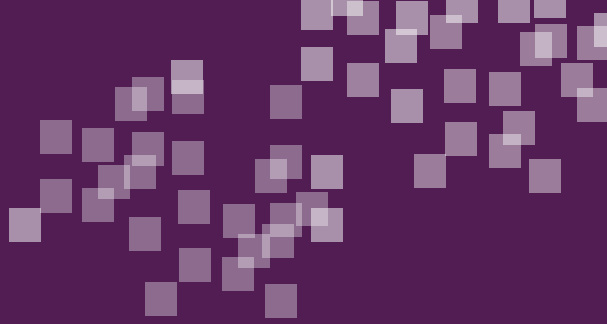
- It is your responsibility to send the **`MotionEvent`**s to the gesture detector



# Example: Tap to select

- When **`OnTouchEvent`** called - pass the **`MotionEvent`** to the gesture detector

```
public override OnTouchEvent (MotionEvent e)
{
    gestureDetector.OnTouchEvent(e);

    var x = e.GetX();
    var y = e.GetY();
    ...
}
```



# Individual Exercise

Create an app that utilizes the scrolling gesture

Xamarin University

# Double tap listener

- The **IOnDoubleTapListener** is implemented similarly to **IOnGestureListener** and is used with the **GestureDetector** class

```
bool OnDoubleTap (MotionEvent e) { ... }

bool OnDoubleTapEvent (MotionEvent e) { ... }

bool OnSingleTapConfirmed (MotionEvent e) { ... }
```

# SetOnDoubleTapListener method

- **`GestureDetector`** reports double tap gestures to a separate listener object

```
protected override void OnCreate (Bundle bundle)
{
    ...
    gestureDetector = new GestureDetector (context: this, listener: this);

    gestureDetector.SetOnDoubleTapListener (listener: this);
}
```



**`DoubleTapListener`** is registered using a set method, *not* the constructor

# Three steps to implement scaling

- Scaling is handled similarly to other gestures but uses the **`IOnScaleListener`** and **`ScaleGestureDetector`**

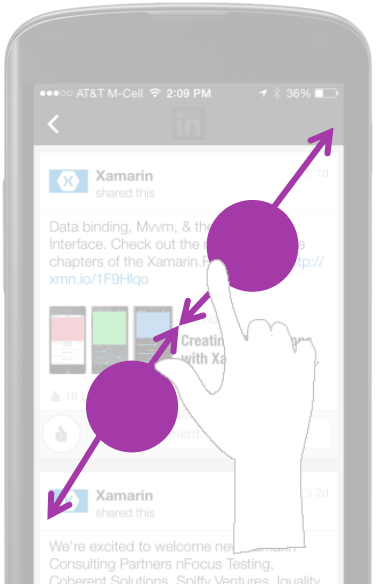
Implement **`IOnScaleListener`**

Instantiate the **`ScaleGestureDetector`**

Forward **`MotionEvents`** to the **`ScaleGestureDetector`**

# Implement the scale gesture listener

- The **`IOnScaleGestureListener`** receives scaling events




```
public class MainActivity : Activity, IOnScaleGestureListener
{
    ...
}
```

```
protected override void OnCreate (Bundle bundle)
{
   ...
   scaleDetector = new ScaleGestureDetector (this, this);
}
```

# Implement IOnScaleGestureListener

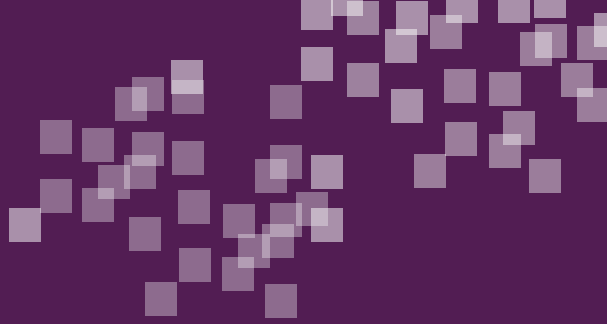
- The methods for the scaling interface are then applied

```
bool OnScale (MotionEvent e) { ... }

bool OnScaleBegin (MotionEvent e) { ... }

void OnScaleEnd (MotionEvent e) { ... }
```

# Forward the motion event

- When listening for scaling gestures you will pass the **`MotionEvent`** to the **`ScaleGestureDetector`**

```
public override bool OnTouchEvent (MotionEvent e)
{
    scaleGestureDetector.OnTouchEvent (e);

    return true;
}
```



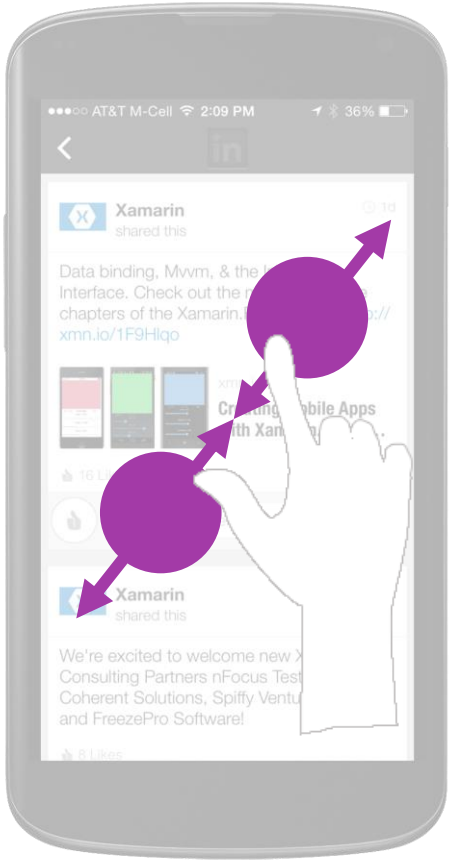
# Individual Exercise

Utilize the scaling gesture

Xamarin University

# Summary

1. Define gestures
2. Implement the gesture detector
3. Handle gestures
4. Discuss scaling and double tap gestures

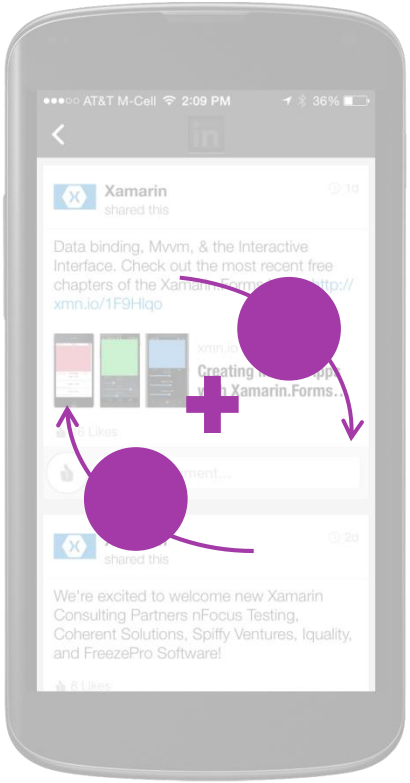


# Create custom gestures programmatically

Xamarin University

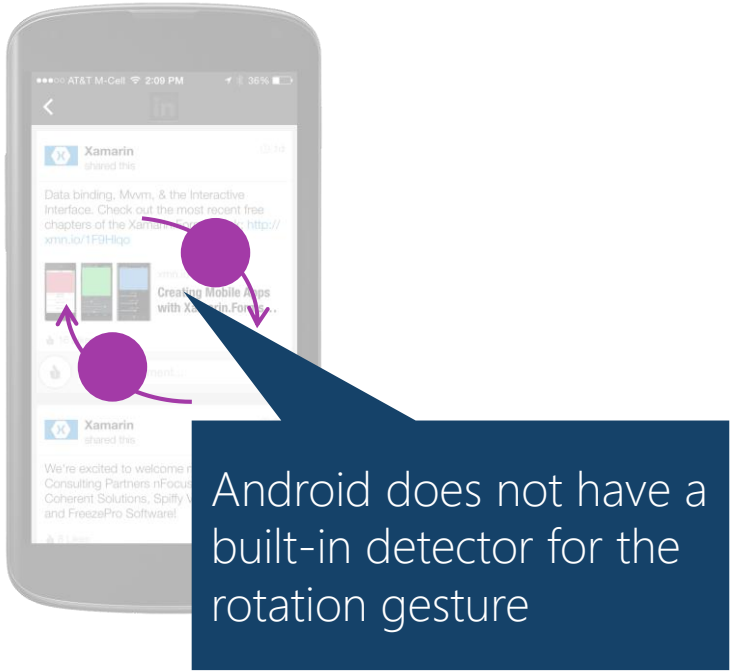
# Tasks

1. Categorize gestures
2. Build a detector for a custom gesture
3. Define a gesture listener interface



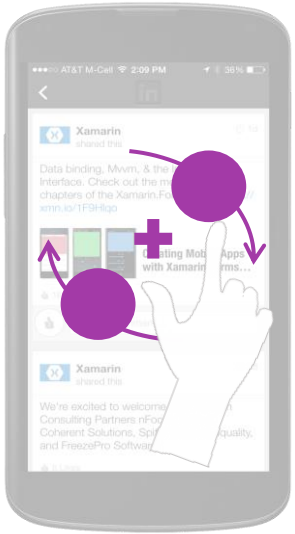
# What is a custom gesture?

- A *custom gesture* is a gesture that is not directly supported by Android

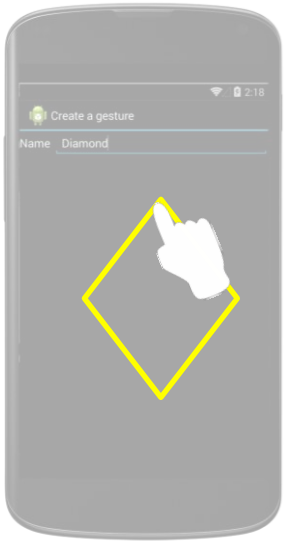



# Gesture types

- Gestures are split into two categories based on whether they have a finite path



*Continuous* gestures can be performed indefinitely




*Discrete* gestures have a fixed path

# Creation overview

- There are two steps to creating a custom gesture programmatically

Code a gesture detector

Define a listener interface



Creating a custom gesture programmatically lets you support both continuous and discrete gestures

# Define a listener interface

- You define a listener interface that clients implement to be notified when the gesture is detected

```
public interface IOnRotationGestureListener
{
  void OnRotateStarted (MotionEvent event);
  void OnRotate (MotionEvent event, float angle);
  void OnRotateCompleted (MotionEvent event);
}
```

# Code a gesture detector

- Custom gestures typically follow the standard **GestureDetector** pattern

```
public class RotationGestureDetector
{ ...
    public RotationGestureDetector (IOnRotationGestureListener listener)
    {
        ...
    }
    public bool OnTouchEvent (MotionEvent e)
    {
        ...
    }
}
```

Client registers a listener

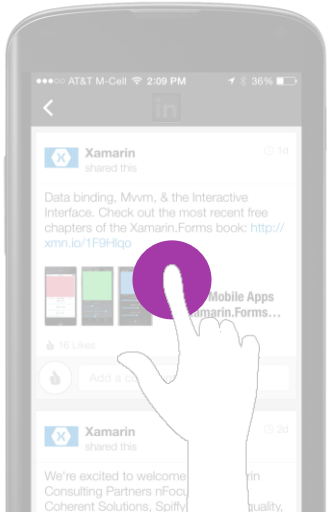
The method is named **OnTouchEvent** as a matter of convention

Client code forwards the **MotionEvent**

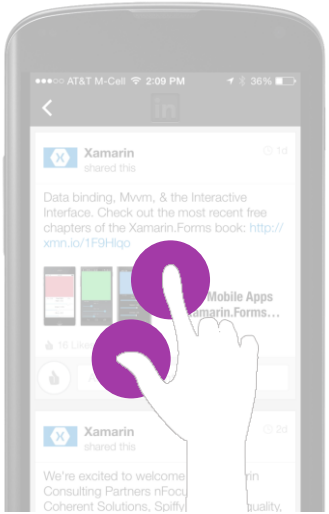
# Implementation [concept]

- The gesture detector *analyzes* the motion events that it receives, *determines* when the gesture has occurred, and *notifies* the listener

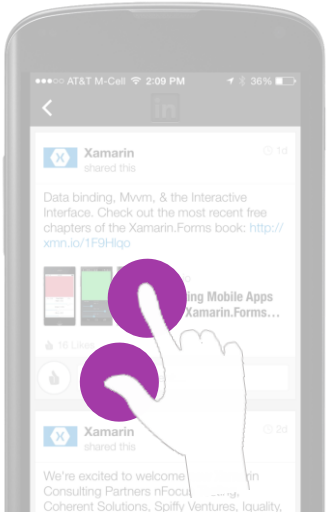
Finger 1 down



Finger 2 down

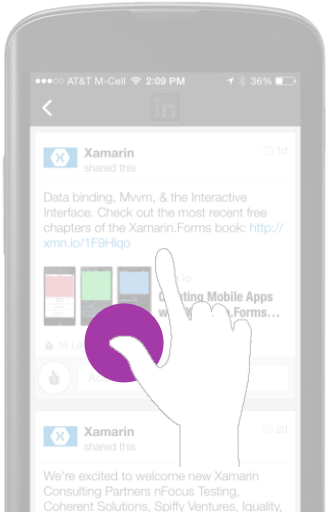


Move



Finger up

Both fingers up



# Implementation [code]

- Gesture detection and reporting typically done in **`OnTouchEvent`**

```
public bool OnTouchEvent (MotionEvent e)
{
    switch (e.ActionMasked)
    {
    case MotionEventActions.Down:        /* first finger down  */ break;
    case MotionEventActions.PointerDown: /* second finger down */ break;
    case MotionEventActions.Move:        /* rotation occurring */ break;
    case MotionEventActions.PointerUp:   /* one finger up      */ break;
    case MotionEventActions.Up:          /* second finger up   */ break;
    }
}
```

# Forward the MotionEvents

- The client is responsible for forwarding **`MotionEvent`**s to the custom gesture detector

```
public class MyActivity : Activity
{
    public override bool OnTouchEvent (MotionEvent e)
    {
        rotationGestureDetector.OnTouchEvent (e);
        return true;
    }
}
```

# Flash Quiz

Xamarin University

# Flash Quiz

① Discrete gestures
   a) have a defined beginning and end
   b) have no set end point
   c) interpret motion events

# Flash Quiz

① Discrete gestures

a) have a defined beginning and end

b) have no set end point

c) interpret motion events

# Flash Quiz

② When creating a custom gesture you must follow the patterns the APIs used for `GestureDetector`

a) True

b) False

# Flash Quiz

② When creating a custom gesture you must follow the patterns the APIs used for **`GestureDetector`**

a) True

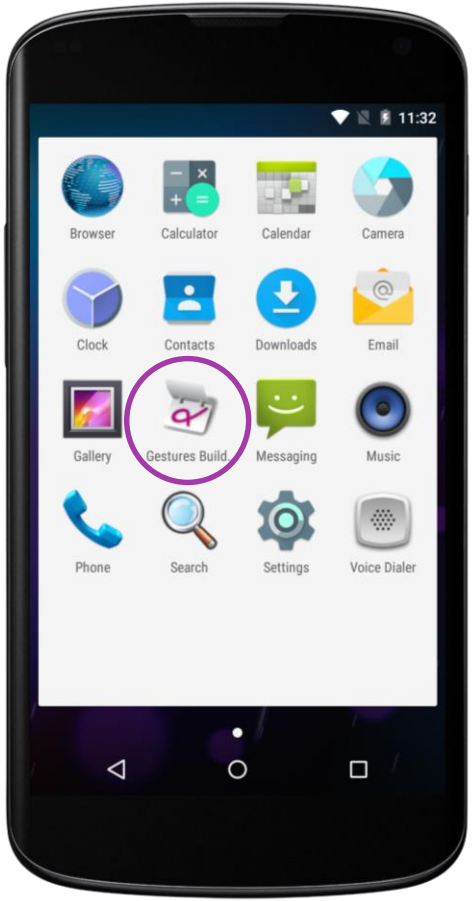
b) False

# Individual Exercise

Build a rotation gesture

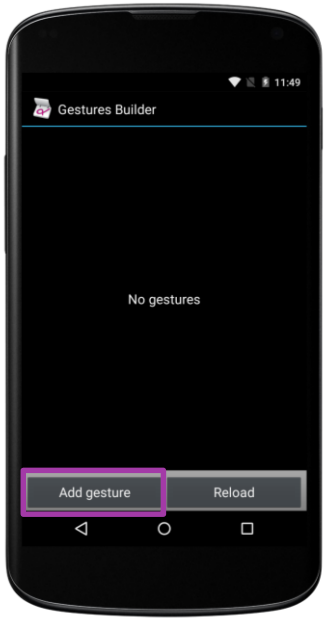
# Android Gestures Builder

- Android provides the Gestures Builder app to help you build discrete gestures

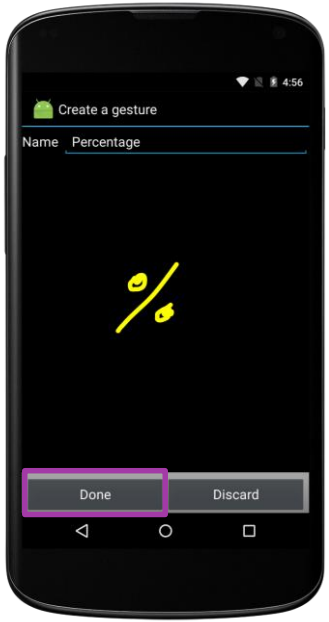


# Build a discrete gesture

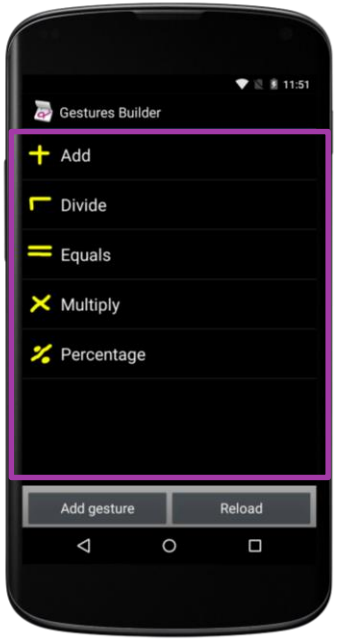
- Building a discrete gesture is as easy as tapping on “Add gesture”, naming it, drawing it on the device and tapping “Done”




Name will be used to identify the gesture in the code

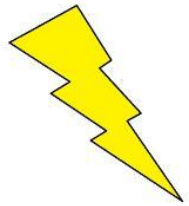



Completed gestures are stored on the device



# Use the gestures

- Using the gestures created by the Gestures Builder application requires several steps:
  - Export the gestures binary from your Android device
  - Add it to the Resources -> Raw folder
  - Load it into a **`GestureLibary`** object
  - Add a **`GestureOverlayView`** to your application
  - Use the **`GestureLibrary`** to detect gestures performed on the View

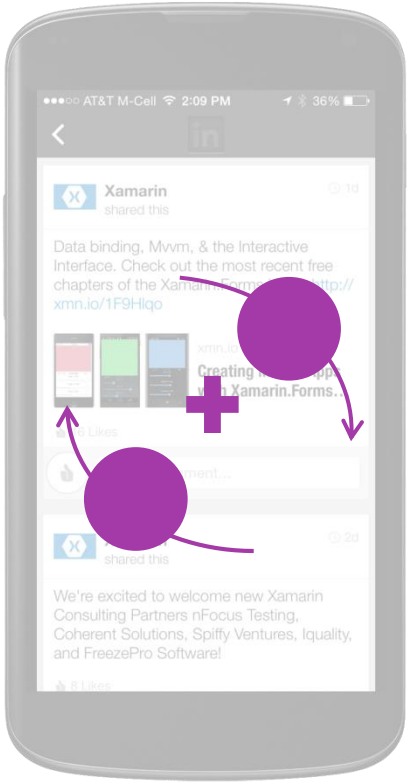




Check the lighting lectures section of Xam U website for more information on using gestures created in the Gestures Builder application

# Summary

1. Categorize gestures
2. Build a detector for a custom gesture
3. Define a gesture listener interface



# Thank You!

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


Microsoft