Mobile Programming Lecture 11

Layout Inspector, Linkify, Gestures, and Version Control

Agenda

- Layout Inspector
- Linkify
- Gestures
- GestureDetector
- GestureOverlayView
- Version Control with Git

Layout Inspector

- You've heard it said that LinearLayouts may be inefficient. How do you know for sure?
- How long does it take for the system to draw your UI on the screen?

Layout Inspector

- The Layout Inspector tool allows you to debug and optimize your UI.
- It provides a visual representation of the Layout's View Hierarchy

Layout Inspector

To run this tool:

1. First launch the application and go to the screen you want to analyze
2. Open the Android Device Monitor tool window from ***Android Studio***
3. From the ***Tools*** menu select ***Layout Inspector***
4. Double-clicking the process of your application

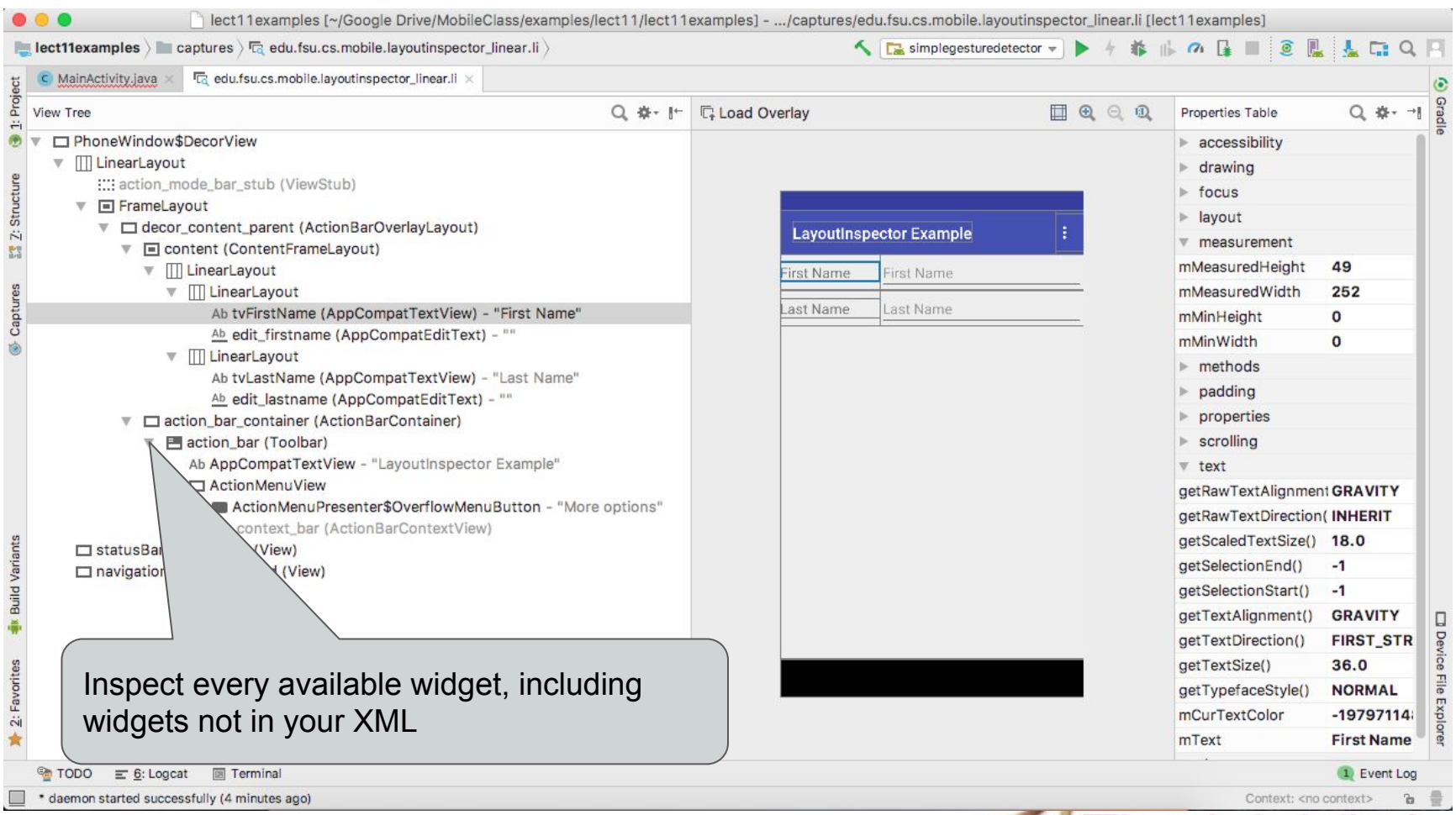
Layout Inspector

In newer version of the Android SDK you should use the monitor tool. To access the monitor tool from a terminal, execute

<sdk-directory>/tools/monitor

From the *Window* menu, choose *Open Perspective…* and select *Hierarchy View*

Layout Inspector



The Mobile Lab at Florida State University

Layout Inspector

For your Homework 2, what's the time difference between loading your LinearLayout versus your RelativeLayout?

See Layout Inspector Example

Linkify

Linkify takes a piece of text and a regular expression and turns all of the regex matches in the text into clickable links

Linkify

See Linkify Example

GestureDetector

- Android can detect various gestures and events by the user
- The `GestureDetector.OnGestureListener` callback will notify users when a particular motion event has occurred

GestureDetector

To use this class:

1. Create an instance of `GestureDetector` for your View
2. In the `Activity.onTouchEvent(MotionEvent)`

   method ensure you call

   `GestureDetector.onTouchEvent(MotionEvent)`

   The methods defined in your callback will be executed when the events occur.

GestureDetector

```
public class SimpleGestureDetectorExample extends Activity {
	GestureDetector detector;

	class MyGestureDetector extends SimpleOnGestureListener {
		@Override
		public boolean onDoubleTap(MotionEvent e) {
			Toast.makeText(getApplicationContext(),"You double tapped", Toast.LENGTH_SHORT).show();
			return true;
		}
	}

	@Override
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.main);
		detector = new GestureDetector(this, new MyGestureDetector());
	}

	@Override
	public boolean onTouchEvent(MotionEvent event) {
		if (detector.onTouchEvent(event))
			return true;
		else
			return false;
	}
}
```

GestureDetector

1. Create an instance of GestureDetector for your View

```
publi                                    ple extends Activity {

      class MyGestureDetector extends SimpleOnGestureListener {
            @Override
            public boolean onDoubleTap(MotionEvent e) {
                  Toast.makeText(getApplicationContext(), "You double tapped",
            Toast.LENGTH_SHORT).show();
                  return true;
            }
      }

      @Override public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.main);
            detector = new GestureDetector(this, new MyGestureDetector());
      }

      @Override public boolean onTouchEvent(MotionEvent event) {
            if (detector.onTouchEvent(event))
                  return true;
            else
                  return false;
      }
}
```

GestureDetector

```
public class SimpleGestureDetectorExample extends Activity {
      GestureDetector detector;

      class MyGestureDetector extends SimpleOnGestureListener {
            @Override
            public boolean onDoubleTap(MotionEvent e) {
                  Toast.makeText(getApplicationContext(), "You double tapped",
            Toast.LENGTH_SHORT).show();
                  return true;
            }
      }

      (                                                    eState) {

                                                           stureDetector());
      }

      @Override public boolean onTouchEvent(MotionEvent event) {
            if (detector.onTouchEvent(event))
                  return true;
            else
                  return false;
      }
}
```

2. In the Activity.onTouchEvent(MotionEvent) method, ensure you call GestureDetector.onTouchEvent(MotionEvent)

Ge

We extend SimpleOnGestureListener, which is a convenience class when you don't want to implement all of the gesture methods

```
public                                                    ity {
      (

      class MyGestureDetector extends SimpleOnGestureListener {
            @Override
            public boolean onDoubleTap(MotionEvent e) {
                  Toast.makeText(getApplicationContext(), "You double tapped",
            Toast.LENGTH_SHORT).show();
                  return true;
            }
      }

      @Override public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.main);
            detector = new GestureDetector(this, new MyGestureDetector());
      }

      @Override public boolean onTouchEvent(MotionEvent event) {
            if (detector.onTouchEvent(event))
                  return true;
            else
                  return false;
      }
}
```

GestureDetector

The one we do choose to implement in this example, is onDoubleTap(), although there are many more events that we can implement

```
public clas
      Gestur

      class MyGestureDetector extends SimpleOnGestureListener {
            @Override
            public boolean onDoubleTap(MotionEvent e) {
                  Toast.makeText(getApplicationContext(), "You double tapped",
            Toast.LENGTH_SHORT).show();
                  return true;
            }
      }

      @Override public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.main);
            detector = new GestureDetector(this, new MyGestureDetector());
      }

      @Override public boolean onTouchEvent(MotionEvent event) {
            if (detector.onTouchEvent(event))
                  return true;
            else
                  return false;
      }
}
```

GestureDetector

When our detector detects a double tap, we make a Toast

```
public class Simp
      GestureDete

      class MyGes
            @Overri
            public boolean onDoubleTap(MotionEvent e) {
                  Toast.makeText(getApplicationContext(), "You double tapped",
            Toast.LENGTH_SHORT).show();
                  return true;
            }
      }

      @Override public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.main);
            detector = new GestureDetector(this, new MyGestureDetector());
      }

      @Override public boolean onTouchEvent(MotionEvent event) {
            if (detector.onTouchEvent(event))
                  return true;
            else
                  return false;
      }
}
```

GestureDetector

```
public class SimpleGestureDetectorExample extends Activity {
      GestureDetector detector;

      class MyGestureDetector extends SimpleOnGestureListener {
            @Override
            public boolean onDoubleTap(MotionEvent e) {
                  Toast.makeText(getApplicationContext(), "You double tapped",
            Toast.LENGTH_SHORT).s
                  return true;
            }
      }

      @Override public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.main);
            detector = new GestureDetector(this, new MyGestureDetector());
      }

      @Override public boolean onTouchEvent(MotionEvent event) {
            if (detector.onTouchEvent(event))
                  return true;
            else
                  return false;
      }
}
```

Return true if you don't want any further processing for this event to occur

GestureDetector

```
public class SimpleGestureDetectorExample extends Activity {
      GestureDetector detector;

      class MyGestureDetector extends SimpleOnGestureListener {
            @Override
            public boolean onDoubleTap(MotionEvent e) {
                  Toast.makeText(getApplicationContext(), "You double tapped",
            Toast.LENGTH_SHORT).show();
                  return true;
            }
      }

      @Overr                                          avedInstanceState) {
                                                      ;
            setCont      iew(R.layout.main);
            detector = new GestureDetector(this, new MyGestureDetector());
      }

      @Override public boolean onTouchEvent(MotionEvent event) {
            if (detector.onTouchEvent(event))
                  return true;
            else
                  return false;
      }
}
```

Create an instance of MyGestureDetector, that we will use later

GestureDetector

```
public class SimpleGestureDetectorExample extends Activity {
      GestureDetector detector;

      class MyGestureDetector extends SimpleOnGestureListener {
            @Override
            public boolean onDoubleTap(MotionEvent e) {
                  Toast.makeText(getApplicationContext(), "You double tapped",
            Toast.LENGTH_SHORT).show();
                  return true;
            }
      }

      @Override public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.main);
            detector = new GestureDetector(this, new MyGestureDetec
      }

      @Override public boolean onTouchEvent(MotionEvent event) {
            if (detector.onTouchEvent(event))
                  return true;
            else
                  return false;
      }
}
```

This is a callback method for the Activity class, we override onTouchEvent, which may contain a gesture

GestureDetector

```
public class SimpleGestureDetectorExample extends Activity {
      GestureDetector detector;

      class MyGestureDetector extends SimpleOnGestureListener {
            @Override
            public boolean onDoubleTap(MotionEvent e) {
                  Toast.makeText(getApplicationContext(), "You double tapped",
            Toast.LENGTH_SHORT).show();
                  return true;
            }
      }

      @Override public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.main);
            detector = new GestureDetector(this, new MyGestureDetector());
      }

      @Override public boolean onTouchEvent(Motic
            if (detector.onTouchEvent(event))
                  return true;
            else
                  return false;
      }
}
```

Our instance of MyGestureDetector will determine if this touch event contains a gesture ...

GestureDetector

```
public class SimpleGestureDetectorExample extends Activity {
      GestureDetector detector;

      class MyGestureDetector extends SimpleOnGestureListener {
            @Override
            public boolean onDoubleTap(MotionEvent e) {
                  Toast.makeText(getApplicationContext(), "You double tapped",
            Toast.LENGTH_SHORT).show();
                  return true;
            }
      }

      @Override public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.main);
            detector = new GestureDetector(this, new MyGestureDetector());
      }

      @Override public boolean onTouchEvent(Motion
            if (detector.onTouchEvent(event))
                  return true;
            else
                  return false;
      }
}
```

If it does, it checks whether you implemented the callback method for the given gesture

GestureDetector

```
public class SimpleGestureDetectorExample extends Activity {
      GestureDetector detector;

      class MyGestureDetector extends SimpleOnGestureListener {
            @Override
            public boolean onDoubleTap(MotionEvent e) {
                  Toast.makeText(getApplicationContext(), "Y
            Toast.LENGTH_SHORT).show();
                  return true;
            }
      }

      @Override public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.main);
            detector = new GestureDetector(this, new MyGestureDetector());
      }

      @Override public boolean onTouchEvent(MotionEvent event) {
            if (detector.onTouchEvent(event))
                  return true;
            else
                  return false;
      }
}
```

So if the touch event is a double tap, this code will be executed

GestureDetector

Another example, implementing the `onFling()` method of `SimpleGestureDetector`

GestureDetector

```
class MyGestureDetector extends SimpleOnGestureListener {
     @Override
     public boolean onFling(MotionEvent e1, MotionEvent e2, float velocityX,
          float velocityY) {
          if (Math.abs(e1.getY() - e2.getY()) > 250)
               return false;

          if (e1.getX() - e2.getX() > 120 && Math.abs(velocityX) > 200) {
               Toast.makeText(getApplicationContext(), "You swiped right to left",
                                                       Toast.LENGTH_SHORT).show();
          }
          else if (e2.getX() - e1.getX() > 120 && Math.abs(velocityX) > 200) {
               Toast.makeText(getApplicationContext(), "You swiped left to right",
                                                       Toast.LENGTH_SHORT).show();
          }

          return false;
     }
}
```

GestureDetector

Override onFling

```
class MyGestu                    impleOnGestureListener {

     @Override
     public boolean onFling(MotionEvent e1, MotionEvent e2, float velocityX,
          float velocityY) {

          if (Math.abs(e1.getY() - e2.getY()) > 250)
               return false;

          if (e1.getX() - e2.getX() > 120 && Math.abs(velocityX) > 200) {
               Toast.makeText(getApplicationContext(), "You swiped right to left",
                                                       Toast.LENGTH_SHORT).show();
          }
          else if (e2.getX() - e1.getX() > 120 && Math.abs(velocityX) > 200) {
               Toast.makeText(getApplicationContext(), "You swiped left to right",
                                                       Toast.LENGTH_SHORT).show();
          }

          return false;
     }
}
```

GestureDete

First 2 args are MotionEvent objects, which are used to report some kind of movement, so they store information about an event involving movement

```
class MyGestureDetector exte

     @Override
     public boolean onFling(MotionEvent e1, MotionEvent e2, float velocityX,
          float velocityY) {

          if (Math.abs(e1.getY() - e2.getY()) > 250)
               return false;

          if (e1.getX() - e2.getX() > 120 && Math.abs(velocityX) > 200) {
               Toast.makeText(getApplicationContext(), "You swiped right to left",
                                                       Toast.LENGTH_SHORT).show();
          }
          else if (e2.getX() - e1.getX() > 120 && Math.abs(velocityX) > 200) {
               Toast.makeText(getApplicationContext(), "You swiped left to right",
                                                       Toast.LENGTH_SHORT).show();
          }

          return false;
     }
}
```

GestureDete

e1 stores information about the beginning of the movement, e2 stores the same for the end of the movement

```
class MyGestureDetector exte

     @Override
     public boolean onFling(MotionEvent e1, MotionEvent e2, float velocityX,
          float velocityY) {

          if (Math.abs(e1.getY() - e2.getY()) > 250)
               return false;

          if (e1.getX() - e2.getX() > 120 && Math.abs(velocityX) > 200) {
               Toast.makeText(getApplicationContext(), "You swiped right to left",
                                                       Toast.LENGTH_SHORT).show();
          }
          else if (e2.getX() - e1.getX() > 120 && Math.abs(velocityX) > 200) {
               Toast.makeText(getApplicationContext(), "You swiped left to right",
                                                       Toast.LENGTH_SHORT).show();
          }

          return false;
     }
}
```

GestureDetector

velocityX is the speed in the X direction, velocityY is the speed in the Y direction

```
class MyGestureDetector extends SimpleOnGestureListener {

     @Override
     public boolean onFling(MotionEvent e1, MotionEvent e2, float velocityX,
          float velocityY) {

          if (Math.abs(e1.getY() - e2.getY()) > 250)
               return false;

          if (e1.getX() - e2.getX() > 120 && Math.abs(velocityX) > 200) {
               Toast.makeText(getApplicationContext(), "You swiped right to left",
                                                       Toast.LENGTH_SHORT).show();
          }
          else if (e2.getX() - e1.getX() > 120 && Math.abs(velocityX) > 200) {
               Toast.makeText(getApplicationContext(), "You swiped left to right",
                                                       Toast.LENGTH_SHORT).show();
          }

          return false;
     }
}
```

GestureDetector

```
class MyGestureDetector extends SimpleOnGestureListener {

     @Override
     public boolean onFling(MotionEvent e1, MotionEvent e2, float velocityX,
          float velocityY) {

          if (Math.abs(e1.getY() - e2.getY()) > 250)
               return false;

          if (e1.getX() - e2.getX() > 120 && Math.abs(velo
               Toast.makeText(getApplicationContext(), "You swiped right to left",
                                                       Toast.LENGTH_SHORT).show();
          }
          else if (e2.getX() - e1.getX() > 120 && Math.abs(velocityX) > 200) {
               Toast.makeText(getApplicationContext(), "You swiped left to right",
                                                       Toast.LENGTH_SHORT).show();
          }

          return false;
     }
}
```

We try to filter out swipes that may be vertical swipes. It's up to you how you want to filter this out

GestureDetector

```
class MyGestureDetector extends SimpleOnGestureListener {

     @Override
     public boolean onFling(MotionEvent e1, MotionEvent e2, float velocityX,
          flo
          if                                                  > 250)

          if (e1.getX() - e2.getX() > 120 && Math.abs(velocityX) > 200) {
               Toast.makeText(getApplicationContext(), "You swiped right to left",
                                                       Toast.LENGTH_SHORT).show();
          }
          else if (e2.getX() - e1.getX() > 120 && Math.abs(velocityX) > 200) {
               Toast.makeText(getApplicationContext(), "You swiped left to right",
                                                       Toast.LENGTH_SHORT).show();
          }

          return false;
     }
}
```

Checking for right-to-left swipe. Did the user swipe "long enough"? "Long enough" is arbitrary

GestureDetector

```
class MyGestureDetector extends SimpleOnGestureListener {

     @Override
     public boolean onFling(MotionEvent e1, MotionEvent e2, float velocityX,
          float velocityY) {

          if (Math.abs(e1.getY() - e2.getY())
               return false;

          if (e1.getX() - e2.getX() > 120 && Math.abs(velocityX) > 200) {
               Toast.makeText(getApplicationContext(), "You swiped right to left",
                                                       Toast.LENGTH_SHORT).show();
          }
          else if (e2.getX() - e1.getX() > 120 && Math.abs(velocityX) > 200) {
               Toast.makeText(getApplicationContext(), "You swiped left to right",
                                                       Toast.LENGTH_SHORT).show();
          }

          return false;
     }
}
```

Checking for right-to-left swipe. Did the user swipe "fast enough"? "Fast enough" is also arbitrary

GestureDetector

```
class MyGestureDetector extends SimpleOnGestureListener {

     @Override
     public boolean onFling(MotionEvent e1, MotionEvent e2, float velocityX,
          float velocityY) {

          if (Math.abs(e1.getY() - e2.getY()) > 250)
               return false;

          i                                    & Math.abs(velocityX) > 200) {
                                               nContext(), "You swiped right to left",
                                                    Toast.LENGTH_SHORT).show();
          }
          else if (e2.getX() - e1.getX() > 120 && Math.abs(velocityX) > 200) {
               Toast.makeText(getApplicationContext(), "You swiped left to right",
                                                    Toast.LENGTH_SHORT).show();
          }

          return false;
     }
}
```

Check for left-to-right swipe

GestureDetector

See Simple GestureDetector Example

GestureDetector

- These are gestures anywhere on the Activity itself
- You may want to check for gestures on a single View or several Views
- You would need to setup the `onTouch()` events for the individual Views instead

GestureDetector

```
public class GestureDetectorOnViewExampleActivityextends Activity {
	GestureDetector detector;

	class MyGestureDetector extends SimpleOnGestureListener {
		@Override
		public boolean onDoubleTap(MotionEvent e) {
			Toast.makeText(getApplicationContext(),"You double tapped", Toast.LENGTH_SHORT).show();
			return true;
		}
	}

	@Override
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.main);

		final TextView tv = (TextView) findViewById(R.id.textView1);
		tv.setOnTouchListener(new OnTouchListener() {
			@Override
			public boolean onTouch(View v, MotionEvent event) {
				detector.onTouchEvent(event);
				return true;
			}
		});

		detector = new GestureDetector(this, new MyGestureDetector());
	}
}
```

GestureDetector

```
public class GestureDetectorOnViewExampleActivity extends Activity {
	GestureDetector detector;

	class MyGestureDetector extends SimpleOnGestureListener {
		@Override public boolean onDoubleTap(MotionEvent e) {
			Toast.makeText(getApplicationContext(), "You double tapped", Toast.LENGTH_SHORT).show();
			return true;
		}
	}

	@Overr                    e(Bundle savedInstanceState) {
	                          tanceState);
	                          .main);

		final TextView tv = (TextView) findViewById(R.id.textView1);
		tv.setOnTouchListener(new OnTouchListener() {

			@Override public boolean onTouch(View v, MotionEvent event) {
				detector.onTouchEvent(event);
				return true;
			}
		});

		detector = new GestureDetector(this, new MyGestureDetector());
	}
}
```

We will setup a double tap event for this TextView

The Mobile Lab at Florida State University

GestureDetector

```
public class GestureDetectorOnViewExampleActivity extends Activity {
      GestureDetector detector;

      class MyGestureDetector extends SimpleOnGestureListener {
            @Override public boolean onDoubleTap(MotionEvent e) {
                  Toast.makeText(getApplicationContext(), "You double tapped", Toast.LENGTH_SHORT).show();
                  return true;
            }
      }

      @Overr                              e(Bundle savedInstanceState) {
                                          tanceState);
                                          .main);

                                    xtView) findViewById(R.id.textView1);
            tv.setOnTouchListener(new OnTouchListener() {

                  @Override public boolean onTouch(View v, MotionEvent event) {
                        detector.onTouchEvent(event);
                        return true;
                  }
            });

            detector = new GestureDetector(this, new MyGestureDetector());
      }
}
```

To do that, we need to know when it is touched

GestureDetector

```
public class GestureDetectorOnViewExampleActivity extends Activity {
	GestureDetector detector;

	class MyGestureDetector extends SimpleOnGestureListener {
		@Override public boolean onDoubleTap(MotionEvent e) {
			Toast.makeText(getApplicationContext(), "You double tapped", Toast.LENGTH_SHORT).show();
			return true;
		}
	}

	@Override public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.main);

		final TextView tv = (TextView) findViewById(R.id.textView1);
		tv.setOnTouchListener(new OnTouchListener() {

			@Override public boolean onTouch(View v, MotionEvent event) {
				detector.onTouchEvent(event);
				return true;
			}
		});

		detector = new GestureDetector(this, new MyGestureDetector());
	}
}
```

Similar to the previous example, we call onTouchEvent on the GestureDetector

GestureDetector

```
public class GestureDetectorOnViewExampleActivity extends Activity {
	GestureDetector detector;

	class MyGestureDetector extends SimpleOnGestureListener {
		@Override public boolean onDoubleTap(MotionEvent e) {
			Toast.makeText(getApplicationContext(), "You double tapped", Toast.LENGTH_SHORT).show();
			return true;
		}
	}

	@Override public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.main);

		final TextView tv = (TextView) findViewById(R.id.textView1);
		tv.setOnTouchListener(new OnTouchListener() {

			@Override public boolean onTouch(View v, MotionEvent event) {
				detector.onTouchEvent(event);
				return true;
			}
		});

		detector = new GestureDetector
	}
}
```

If you don't return true here, the double tap will not work. We return true to indicate that you will handle these touch events yourself

Custom Gestures

- How do you recognize a more complex Gesture?
- `GestureDetector` won't be able to recognize those for you.

Custom Gestures

You can build your own custom gestures using the Gestures Builder app, which is pre-installed on your emulator

Custom Gestures

Custom Gestures

- You may want to add multiple gestures with the same name in order to get it right
- After adding gestures to the Gestures Builder app, you will need to copy the gestures file over to your `res/raw` directory

Custom Gestures

- Open the Android Device Monitor window from Android Studio
  - `Tools > Android > Android Device Monitor`
- Select the File Explorer tab
- Select your emulator in the Devices view
- Browse `mnt/sdcard/`
- Select the gestures file
- Select the pull a file from device icon
- Save the file locally
- Place the file into your `res/raw` directory

Custom Gestures

Now you need to add a `GestureOverlayView` item onto your canvas

- Open your Layout XML file
- In the Palette, open the ***Expert*** Tab
- Drag a `GestureOverlayView` onto the canvas and give it an id

You should have something like the XML file on the following slide

Custom Gestures

```
<?xml version="1.0" encoding="utf-8"?>
<android.gesture.GestureOverlayView
    xmlns:android="http://schemas.android.com/apk/res/android"
        android:id="@+id/gestures"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent">
    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent">
    </RelativeLayout>
</android.gesture.GestureOverlayView>
```

Custom Gestures

The Java code follows

Custom Gestures

```
public class GestureExampleActivity extends Activity implements
      OnGesturePerformedListener{
      GestureOverlayView mOverlay;
      GestureLibrary gestureLib;

      @Override
      public void onCreate(Bundle savedInstanceState) { }

      @Override
      public void onGesturePerformed(GestureOverlayView overlay, Gesture gesture) { }
}
```

Custom Gestures

```
public class GestureExampleActivity extends Activity implements
      OnGesturePerformedListener {

      erlay;
      Lib;

      @Override public void onCreate(Bundle savedInstanceState) { }

      @Override public void onGesturePerformed(GestureOverlayView overlay, Gesture
gesture) { }

}
```

We implement OnGesturePerformedListener

Custom Gestures

```
public class GestureExampleActivity extends Activity implements
      OnGesturePerformedListener {

      GestureOverlayView mOverlay;
      GestureLibrary gestureLib;

      @Override public void onCreate(Bundle savedInstanceState) { }

      @Override public void onGesturePerformed(GestureOverlayView overlay,
            Gesture gesture) { }

}
```

Which forces us to add the onGesturePerformed() callback method, which we show on the next slides

Custom Gestures

```
public class GestureExampleActivity extends Activity implements
      OnGesturePerformedListener {

      GestureOverlayView mOverlay;
      GestureLibrary gestureLib;

      @Override public void onCreate(Bundle savedInstanceState) { }

      @Override public void onGesturePerformed(GestureOverlayView overlay,
            Gesture gesture) { }

}
```

Note these fields, which we will use later

Custom Gestures

```
GestureOverlayView mOverlay;
GestureLibrary gestureLib;
@Override
public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);
      mOverlay = (GestureOverlayView) findViewById(R.id.gestures);
      mOverlay.addOnGesturePerformedListener(this);
      gestureLib = GestureLibraries.fromRawResource(this, R.raw.gestures);

      if (!gestureLib.load())
            Toast.makeText(this, "Couldn't load Gesture Library", Toast.LENGTH_SHORT).show();
}
@Override
public void onGesturePerformed(GestureOverlayView overlay, Gesture gesture) {
      ArrayList<Prediction> predictions = gestureLib.recognize(gesture);
      for(Prediction prediction : predictions) {
            if (prediction.score > 1.0)
                  Toast.makeText(getApplicationContext(), prediction.name,
            Toast.LENGTH_SHORT).show();
      }
}
```

Custom Gestures

```
GestureOverlayView mOverlay;
GestureLibrary gestureLib;

@Override public void onCreate(Bundle savedInstance
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);
      mOverlay = (GestureOverlayView) findViewById(R.id.gestures);
      mOverlay.addOnGesturePerformedListener(this);
      gestureLib = GestureLibraries.fromRawResource(this, R.raw.gestures);

      if(!gestureLib.load())
            Toast.makeText(this, "Couldn't load Gesture Library", Toast.LENGTH_SHORT).show();
}

@Override public void onGesturePerformed(GestureOverlayView overlay, Gesture gesture) {
      ArrayList<Prediction> predictions = gestureLib.recognize(gesture);
      for(Prediction prediction : predictions) {
            if(prediction.score > 1.0)
                  Toast.makeText(getApplicationContext(), prediction.name,
            Toast.LENGTH_SHORT).show();
      }
}
```

The GestureOverlay is a View, so we get a handle on it the usual way

Custom Gestures

```
GestureOverlayView mOverlay;
GestureLibrary gestureLib;

@Override public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);
      mOverlay = (GestureOverlayView) findViewById(R.id.ge
      mOverlay.addOnGesturePerformedListener(this);
      gestureLib = GestureLibraries.fromRawResource(this,

      if(!gestureLib.load())
            Toast.makeText(this, "Couldn't load Gesture Library", Toast.LENGTH_SHORT).show();
}

@Override public void onGesturePerformed(GestureOverlayView overlay, Gesture gesture) {
      ArrayList<Prediction> predictions = gestureLib.recognize(gesture);
      for(Prediction prediction : predictions) {
            if(prediction.score > 1.0)
                  Toast.makeText(getApplicationContext(), prediction.name,
            Toast.LENGTH_SHORT).show();
      }
}
```

Since our Activity implements onGesturePerformedListener, we set this Activity as the OnGesturePerformedListener

Custom Gestures

```
GestureOverlayView mOverlay;
GestureLibrary gestureLib;

@Override public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);
      mOverlay = (GestureOverlayView) findViewById(R.id.gestures);
      mOverlay.addOnGesturePerformedListener(this);
      gestureLib = GestureLibraries.fromRawResource(this, R.raw.gestures);
```

Alternatively, you can create an Anonymous inner class and say

mOverlay.addOnGesturePerformedListener(new OnGesturePerformedListener() {...});

```
@Override public void onGesturePerformed(GestureOverlayView overlay, Gesture gesture) {
      ArrayList<Prediction> predictions = gestureLib.recognize(gesture);
      for(Prediction prediction : predictions) {
            if(prediction.score > 1.0)
                  Toast.makeText(getApplicationContext(), prediction.name,
            Toast.LENGTH_SHORT).show();
      }
}
```

Custom Gestures

```
GestureOverlayView mOverlay;
GestureLibrary gestureLib;

@Override public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);
      mOverlay = (GestureOverlayView) findViewById(R.id.gestures);
      mOverlay.addOnGesturePerformedListener(this);
      gestureLib = GestureLibraries.fromRawResource(this, R.raw.gestures);

      if(!gestureLi
            Toast.m                                    ture Library", Toast.LENGTH_SHORT).show();
}

@Override public void onGesturePerformed(GestureOverlayView overlay, Gesture gesture) {
      ArrayList<Prediction> predictions = gestureLib.recognize(gesture);
      for(Prediction prediction : predictions) {
            if(prediction.score > 1.0)
                  Toast.makeText(getApplicationContext(), prediction.name,
            Toast.LENGTH_SHORT).show();
      }
}
```

GestureLibraries simply allows us to open gesture files

Custom Gestures

```
GestureOverlayView mOverlay;
GestureLibrary gestureLib;

@Override public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);
      mOverlay = (GestureOverlayView) findViewById(R.id.gestures);
      mOverlay.addOnGesturePerformedListener(this);
      gestureLib = GestureLibraries.fromRawResource(this, R.raw.gestures);

      if(!gestureLib.load())
            Toast.makeText(this, "Co                                    ast.LENGTH_SHORT).show();
}

@Override public void onGesturePerformed(GestureOverlayView overlay, Gesture gesture) {
      ArrayList<Prediction> predictions = gestureLib.recognize(gesture);
      for(Prediction prediction : predictions) {
            if(prediction.score > 1.0)
                  Toast.makeText(getApplicationContext(), prediction.name,
            Toast.LENGTH_SHORT).show();
      }
}
```

by calling fromRawResource()

Custom Gestures

```
GestureOverlayView mOverlay;
GestureLibrary gestureLib;

@Override public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);
      mOverlay = (GestureOverlayView) findViewById(R.id.gestures);
      mOverlay.addOnGesturePerformedListener(this);
      gestureLib = GestureLibraries.fromRawResource(this, R.raw.gestures);

      if(!gestureLib.load())
            Toast.makeText(this, "Couldn't loa                    _SHORT).show();
}

@Override public void onGesturePerformed(GestureOverlayView overlay, Gesture gesture) {
      ArrayList<Prediction> predictions = gestureLib.recognize(gesture);
      for(Prediction prediction : predictions) {
            if(prediction.score > 1.0)
                  Toast.makeText(getApplicationContext(), prediction.name,
            Toast.LENGTH_SHORT).show();
      }
}
```

and passing the Context and our gestures file as arguments

Custom Gestures

```
GestureOverlayView mOverlay;
GestureLibrary gestureLib;

@Override public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);
      mOverlay = (GestureOverlayView) findViewById(R.id.gestures);
      mOverlay.addOnGesturePerformedListener(this);
      gestureLib = GestureLibraries.fromRawResource(this, R.raw.gestures);

      if(!gestureLib.load())
            Toast.makeText(this, "Couldn't load Gesture Library", Toast.LENGTH_SHORT).show();
}

@Overr                                    ed(GestureOverlayView overlay, Gesture gesture) {
                                          s = gestureLib.recognize(gesture);
                                          dictions) {
            if(prediction.score > 1.0)
                  Toast.makeText(getApplicationContext(), prediction.name,
            Toast.LENGTH_SHORT).show();
      }
}
```

Try to load the gestures

Custom Gestures

```
GestureOverlayView mOverlay;
GestureLibrary gestureLib;

@Override public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);
      mOverlay = (GestureOverlayView) findViewById(R.id.gestures);
                                          tener(this);
                                          RawResource(this, R.raw.gestures);



                                          't load Gesture Library", Toast.LENGTH_SHORT).show();

@Override public void onGesturePerformed(GestureOverlayView overlay, Gesture gesture) {
      ArrayList<Prediction> predictions = gestureLib.recognize(gesture);
      for(Prediction prediction : predictions) {
            if(prediction.score > 1.0)
                  Toast.makeText(getApplicationContext(), prediction.name,
            Toast.LENGTH_SHORT).show();
      }
}
```

Now whenever the use performs some gesture, we need to determine if it's one of our predefined gestures

Custom Gestures

```
GestureOverlayView mOverlay;
GestureLibrary gestureLib;

@Override public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);
      mOverlay = (GestureOverlayView) findViewById(R.id.gestures);
      mOverlay.addOnGesturePerformedListe
      gestureLib = GestureLibraries.fromF

      if(!gestureLib.load())
            Toast.makeText(this, "Couldn                                    ORT).show();
}

@Override public void onGesturePerformed(GestureOverlayView overlay, Gesture gesture) {
      ArrayList<Prediction> predictions = gestureLib.recognize(gesture);
      for(Prediction prediction : predictions) {
            if(prediction.score > 1.0)
                  Toast.makeText(getApplicationContext(), prediction.name,
            Toast.LENGTH_SHORT).show();
      }
}
```

We are provided with our GestureOverlayView, which we defined in our XML Layout file

Custom Gestures

```
GestureOverlayView mOverlay;
GestureLibrary gestureLib;

@Override public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);
      mOverlay = (GestureOverlayView) findViewById(R.id.gestures);
      mOverlay.addOnGesturePerformedListener(
      gestureLib = GestureLibraries.fromRawRe

      if(!gestureLib.load())
            Toast.makeText(this, "Couldn't lo                    .show();
}

@Override public void onGesturePerformed(GestureOverlayView overlay,Gesture gesture) {
      ArrayList<Prediction> predictions = gestureLib.recognize(gesture);
      for(Prediction prediction : predictions) {
            if(prediction.score > 1.0)
                  Toast.makeText(getApplicationContext(), prediction.name,
            Toast.LENGTH_SHORT).show();
      }
}
```

As well as the Gesture itself. This Object contains information about the shape that was drawn on the overlay

Custom Gestures

```
GestureOverlayView mOverlay;
GestureLibrary gestureLib;

@Override public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);
      mOverlay = (GestureOverlayView) findViewById(R.id.gestures);
      mOverlay.addOnGesturePerformedListener(this);
      gestureLib = GestureLibraries.fromRa

      if(!gestureLib.load())
            Toast.makeText(this, "Couldn'                    RT).show();
}

@Override public void onGesturePerformed(GestureOve      ayView overlay, Gesture gesture) {
      ArrayList<Prediction> predictions =gestureLib.recognize(gesture);
      for(Prediction prediction : predictions) {
            if(prediction.score > 1.0)
                  Toast.makeText(getApplicationContext(), prediction.name,
            Toast.LENGTH_SHORT).show();
      }
}
```

Remember we loaded our gestures into gestureLib. Now we use our GestureLibrary gestureLib to try and recognize the shape that was drawn on the screen

Custom Gestures

```
GestureOverlayView mOverlay;
GestureLibrary gestureLib;

@Override public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);
      mOverlay = (GestureOverlayView) findViewById(R.id.gestures);
      mOverlay.addOnGesturePerformedListener(this);
      ...esource(this, R.raw.gestures);

      ...d Gesture Library", Toast.LENGTH_SHORT).show();
}

@Override publ... oid onGesturePerformed(GestureOverlayView overlay, Gesture gesture) {
      ArrayList<Prediction> predictions = gestureLib.recognize(gesture);
      for(Prediction prediction : predictions) {
            if(prediction.score > 1.0)
                  Toast.makeText(getApplicationContext(), prediction.name,
            Toast.LENGTH_SHORT).show();
      }
}
```

Since it is possible to recognize more than one gesture, we need to store the results in a List

Custom Gestures

```
GestureOverlayView mOverlay;
GestureLibrary gestureLib;

@Override public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);
      mOverlay = (GestureOverlayView) findViewById(R.id.gestures);
      mOverlay.addOnGesturePerformedListener(this);
      ges                                                    R.raw.gestures);

      if
                                                        ibrary", Toast.LENGTH_SHORT).show();
}

@Override public voi onGesturePerformed(GestureOverlayView overlay, Gesture gesture) {
      ArrayList<Prediction> predictions = gestureLib.recognize(gesture);
      for(Prediction prediction : predictions) {
            if(prediction.score > 1.0)
                  Toast.makeText(getApplicationContext(), prediction.name,
            Toast.LENGTH_SHORT).show();
      }
}
```

We have a List of Predictions. A Prediction contains a name, which we gave to our Gesture when we used the GesturesBuilder app, and a score, which is the confidence in the prediction

Custom Gestures

```
GestureOverlayView mOverlay;
GestureLibrary gestureLib;

@Override public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);
      mOverlay = (GestureOverlayView) findViewById(R.id.gestures);
      mOverlay.addOnGesturePerformedListener(this);
      gestureLib = GestureLibraries.fromRawResource(this, R.raw.gestures);

      if(!gestureLib.load())
                                          dn't load Gesture Library", Toast.LENGTH_SHORT).show();
}

@Overr                                    d(GestureOverlayView overlay, Gesture gesture) {
      ArrayLi<Prediction> predictions = gestureLib.recognize(gesture);
      for(Prediction prediction : predictions) {
            if(prediction.score > 1.0)
                  Toast.makeText(getApplicationContext(), prediction.name,
            Toast.LENGTH_SHORT).show();
      }
}
```

This is a shortcut technique for iterating through the List

Custom Gestures

```
GestureOverlayView mOverlay;
GestureLibrary gestureLib;

@Override public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);
      mOverlay = (GestureOverlayView) findViewById(R.id.gestures);
      mOverlay.addOnGesturePerformedListener(this);
      gestureLib = GestureLibraries.fromRawResource(this, R.raw.gestures);

      if(!gestureLib.load())
            Toast.makeText(this, "Couldn't load Gesture Library", Toast.LENGTH_SHORT).show();
}

@Override pu                                    reOverlayView overlay, Gesture gesture) {
      Arrayl                                    ureLib.recognize(gesture);
      for(Predictio          ediction : predictions) {
            if(prediction.score > 1.0)
                  Toast.makeText(getApplicationContext(), prediction.name,
            Toast.LENGTH_SHORT).show();
      }
}
```

Only recognize this as a valid gesture if we have some kind of confidence in the prediction

Custom Gestures

```
GestureOverlayView mOverlay;
GestureLibrary gestureLib;

@Override public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);
      mOverlay = (GestureOverlayView) findViewById(R.id.gestures);
      mOverlay.addOnGesturePerformedListener(this);
      gestureLib = GestureLibraries.fromRawResource(this, R.raw.gestures);

      if(!gestureLib.load())
            Toast.makeText(this, "Couldn't load Gesture Library", Toast.LENGTH_SHORT).show();
}

@Override public void onGesturePerformed(GestureOverlayVie
      ArrayList<Prediction> predictions = gestureLib.recog
      for(Prediction prediction : predictions) {
            if(prediction.score > 1.0)
                  Toast.makeText(getApplicationContext(),prediction.name,
            Toast.LENGTH_SHORT).show();
      }
}
```

To find out which gesture was made, use the name field in the Prediction instance

Custom Gestures

See GestureBuilder Example

Version Control

- For your group projects, you surely don't want to be emailing source files to one another whenever you have an update
- You should use some version control software when working on these kinds of projects

Version Control

- You will be using Git, and your projects will be hosted on BitBucket
- You will be responsible for creating and managing your projects
- You and your team members will need to become familiar with Git

Git Quick Start

Cloning, committing, and pushing to a remote repo

```
$ git clone git://github.com/git/hello-world.git
$ cd hello-world
$ (edit files)
$ git add (files)
$ git commit -m 'Explain what I changed`
$ git push git://github.com/git/hello-world.git
```

Git Quick Start

Git cheat sheet 1

Git cheat sheet 2

References

- The Busy Coder's Guide to Android Development - Mark Murphy
- Android Developers
- The Mobile Lab at Florida State University