CHAPTER 12

Sensors

GLOBAL EDITION
Starting Out with App Inventor for Android
Tony Gaddis • Rebecca Halsey
ALWAYS LEARNING
PEARSON

Topics

- The `LocationSensor`
- The `OrientationSensor`
- The `AccelerometerSensor`
- Using the `ActivityStarter` component to launch Google Maps

The `LocationSensor`

- Most smart phones have the capability to tell you the location of the device at any given time.
- The `LocationSensor` can be found in the Sensors Pallet. It is a non-visible component.
- The `LocationSensor` will only work with App Inventor applications that have been packaged and downloaded to a device.

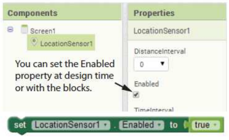
The `LocationSensor`

- There are three sources that the LocationSensor can use to obtain information: GPS, WiFi and cellular towers.
- GPS providers use satellite technology.
- If you are inside a building, your device may attempt to use location information from a WiFi router.
- Your device can also obtain location information from cellular towers.

The `LocationSensor`

`LocationSensor` Component Properties

The `LocationSensor` has an `Enabled` property that must be set to `true` for the sensor to work.



You can set the Enabled property at design time or with the blocks.

set LocationSensor1 . Enabled to true

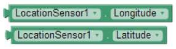
Figure 12-1 The `Enabled` Property (*Source:* MIT App Inventor 2)

The `LocationSensor`

`LocationSensor` Component Properties

The `Latitude` and `Longitude` properties hold the latitude and longitude of the device's current location.

Figure 12-2 Latitude and Longitude Property Blocks (*Source:* MIT App Inventor 2)



LocationSensor1 . Longitude

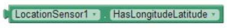
LocationSensor1 . Latitude

The `LocationSensor`

`LocationSensor` Component Properties

- The `HasLongitudeLatitude`, indicates whether or not the device can report the latitude and longitude values.
- There is not a *set* block for this property, it is read only and is determined by the device.

Figure 12-3 HasLongitudeLatitude Property Block (*Source:* MIT App Inventor 2)



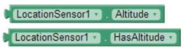
LocationSensor1 . HasLongitudeLatitude

The `LocationSensor`

`LocationSensor` Component Properties

The `Altitude` property holds the altitude of the device if your device has the capability.

Figure 12-4 The Altitude Blocks (*Source:* MIT App Inventor 2)



LocationSensor1 . Altitude

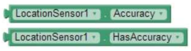
LocationSensor1 . HasAltitude

The `LocationSensor`

`LocationSensor` Component Properties

The `Accuracy` property holds the level of accuracy in meters. There is a corresponding Boolean `HasAccuracy` property to check if the device is able to report accuracy.

Figure 12-5 The Accuracy Blocks (*Source:* MIT App Inventor 2)



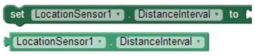
```
LocationSensor1 . Accuracy
LocationSensor1 . HasAccuracy
```

The `LocationSensor`

`LocationSensor` Component Properties

The `LocationSensor` updates location information after the device has moved a certain distance by setting the `DistanceInterval` property.

Figure 12-6 The DistanceInterval Blocks (*Source:* MIT App Inventor 2)



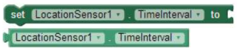
```
set LocationSensor1 . DistanceInterval to
LocationSensor1 . DistanceInterval
```

The `LocationSensor`

`LocationSensor` Component Properties

- You can set the `TimeInterval` to property to the minimum time interval between updates.
- If you set this property to 600,000 it will wait 10 minutes before another update.

Figure 12-7 The TimeInterval Blocks (*Source:* MIT App Inventor 2)



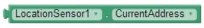
```
set LocationSensor1 . TimeInterval to
LocationSensor1 . TimeInterval
```

The `LocationSensor`

`LocationSensor` Component Properties

- The `CurrentAddress` property provides the physical street address in text format.
- This property is read-only.

Figure 12-8 The CurrentAddress Block (*Source:* MIT App Inventor 2)



```
LocationSensor1 . CurrentAddress
```

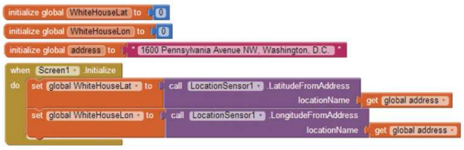
The LocationSensor

`LocationSensor` Component Methods

- The `LocationSensor` has two methods, both relate to the sensors and geocoding capabilities.
- Given an address, geocoding can determine the site's latitude and longitude values.
- For example, if you were to supply the value of "1600 Pennsylvania Ave, NW, Washington, DC" to the `LatitudeFromAddress` method, it would return the latitude of the White House. See Figure 12-9.

The LocationSensor

Figure 12-9 LocationSensor Methods (*Source:* MIT App Inventor 2)



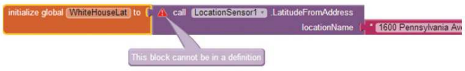
```
initialize global WhiteHouseLat to 0
initialize global WhiteHouseLon to 0
initialize global address to " 1600 Pennsylvania Avenue NW, Washington, D.C. "
when Screen1 .Initialize
do  set global WhiteHouseLat to  call LocationSensor1 .LatitudeFromAddress
                                                  locationName  get global address
    set global WhiteHouseLon to  call LocationSensor1 .LongitudeFromAddress
                                                  locationName  get global address
```

The LocationSensor

`LocationSensor` Component Methods

You cannot use the `LatitudeFromAddress` or `LongitudeFromAddress` in a declaration block.

Figure 12-10 Do Not Use in a Variable's Initialization Block (*Source:* MIT App Inventor 2)



```
initialize global WhiteHouseLat to  call LocationSensor1 .LatitudeFromAddress
                                                locationName  " 1600 Pennsylvania Av
```

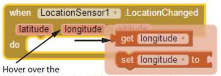
This block cannot be in a definition

The LocationSensor

Location Changed Event

- The `LocationSensor`'s `LocationChange` event fires when the application first starts and whenever the device reports a new location.
- The values from the latitude, longitude, and altitude can be found by hovering over the parameter names on the event handler block.

Figure 12-11 Finding `LocationChanged` Parameter Values (*Source:* MIT App Inventor 2)



```
when LocationSensor1 .LocationChanged
  latitude  longitude
do
                get longitude
                set longitude to
```

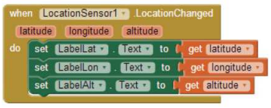
Hover over the parameter names to get the get and set blocks.

The LocationSensor

Location Changed Event

See Figure 12-12 for an example of the LocationChanged event handler which will update labels with the device’s current latitude, longitude, and altitude each time the location is updated.

Figure 12-12 LocationChanged Event Handler (*Source:* MIT App Inventor 2)



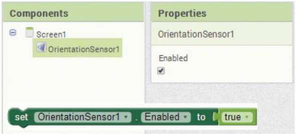
```
when LocationSensor1 .LocationChanged
  latitude  longitude  altitude
do  set LabelLat . Text to  get latitude
    set LabelLon . Text to  get longitude
    set LabelAlt . Text to  get altitude
```

The OrientationSensor

OrientationSensor Component Properties

- The OrientationSensor allows you to determine how a device is oriented.
- Similar to the LocationSensor, the OrientationSensor is a non-visible component and has an Enabled property that must be set to true for the sensor to work.

Figure 12-24 Orientation Sensor Enabled Property (*Source:* MIT App Inventor 2)



```
set OrientationSensor1 . Enabled to  true
```

The OrientationSensor

OrientationSensor Component Properties

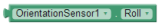
- The three basic properties of the sensor are the Role, Pitch, and Azimuth.
- From these three properties, the OrientationSensor also provide Magnitude and Angle to help determine how much the devices being tilted.

The OrientationSensor

OrientationSensor Component Properties

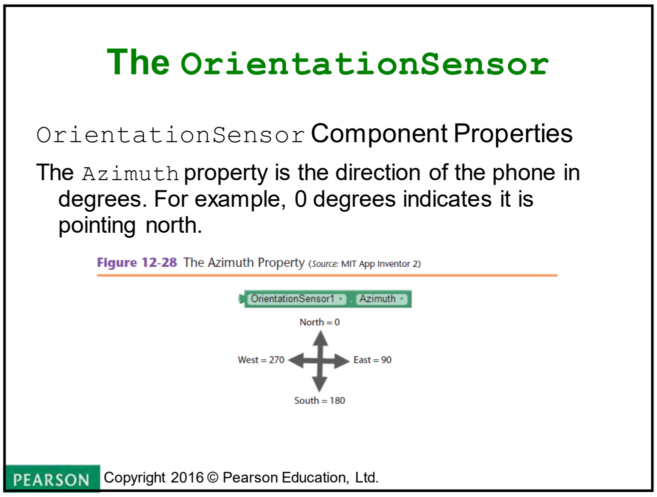
- The Role shows the amount of the tilt left to right in degrees.
- If the device is lying flat, the Role is 0 degrees.

Figure 12-26 The Roll Property (*Source:* MIT App Inventor 2)



```
OrientationSensor1 . Roll
```

The value increases from 0 to 90 as device rolls from flat to up on the left edge. It decreases from 0 to –90 when rolls from flat to the right edge.



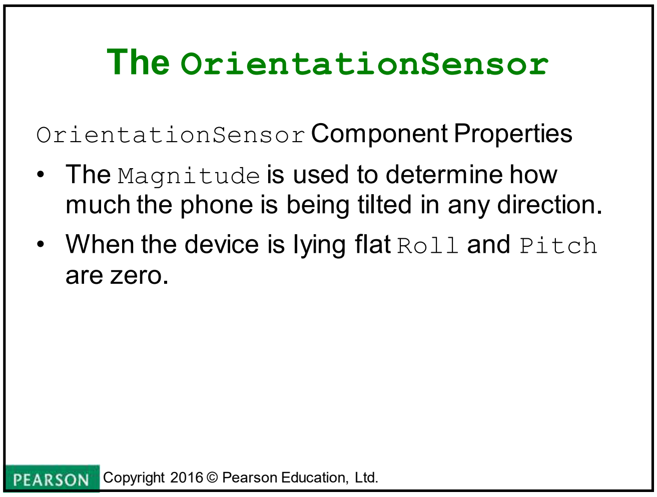
The OrientationSensor

`OrientationSensor` Component Properties

The `Azimuth` property is the direction of the phone in degrees. For example, 0 degrees indicates it is pointing north.

Figure 12-28 The Azimuth Property (*Source:* MIT App Inventor 2)

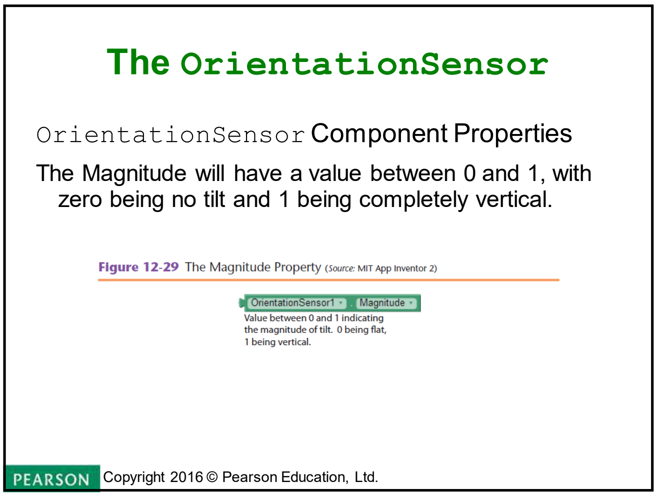
OrientationSensor1 . Azimuth
North = 0
West = 270
East = 90
South = 180



The OrientationSensor

`OrientationSensor` Component Properties

- The `Magnitude` is used to determine how much the phone is being tilted in any direction.
- When the device is lying flat `Roll` and `Pitch` are zero.



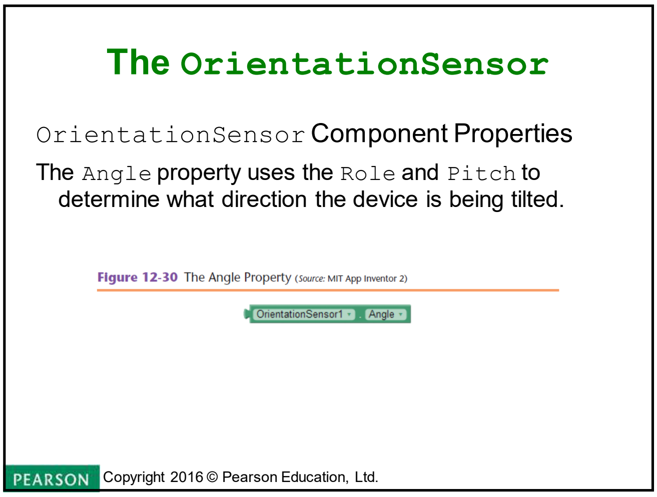
The OrientationSensor

`OrientationSensor` Component Properties

The Magnitude will have a value between 0 and 1, with zero being no tilt and 1 being completely vertical.

Figure 12-29 The Magnitude Property (*Source:* MIT App Inventor 2)

OrientationSensor1 . Magnitude
Value between 0 and 1 indicating the magnitude of tilt. 0 being flat, 1 being vertical.



The OrientationSensor

`OrientationSensor` Component Properties

The `Angle` property uses the `Role` and `Pitch` to determine what direction the device is being tilted.

Figure 12-30 The Angle Property (*Source:* MIT App Inventor 2)

OrientationSensor1 . Angle

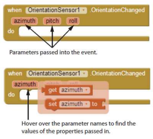
The `OrientationSensor`

Orientation Changed Event

•The `OrientationSensor`’s `OrientationChangedEvent` is called each time the device’s orientation changes.

•The device will send it the `Azimuth`, `Pitch`, and `Roll` as arguments.

Figure 12-31 OrientationSensor’s `OrientationChanged` Event Handler (*Source:* MIT App Inventor 2)



Parameters passed into the event.

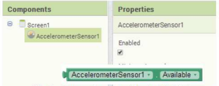
Hover over the parameter names to find the values of the properties passed in.

The Accelerometer

`AccelerometerSonsor` Properties

- The `AccelerometerSensor` is used to perform actions when the device is being shaken.
- It is a non-visible component.
- it has an `Available` property, that is set to true if the device has an accelerometer on it.

Figure 12-41 Enabled and Available (*Source:* MIT App Inventor 2)



The AccelerometerSensor should be both Enabled and Available in order to work.

The Accelerometer

`AccelerometerSonsor` Properties

- The `AccelerometerSensor` has properties that return the acceleration values `XAccel`, `YAccel`, and `ZAccel`.
- `XAccel –` property has a positive value when the device is tilted to the right and negative when tilted to the left.

The Accelerometer

`AccelerometerSonsor` Properties

`Yaccel –` property value is positive when the bottom of the phone is raised and negative when the top is raised.

Figure 12-43 YAccel Property (*Source:* MIT App Inventor 2)

Has a positive value when bottom is tilted up, negative when top is tilted up.

The Accelerometer

`AccelerometerSonsor` Properties

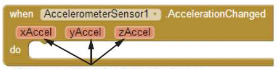
- `ZAccel –` property is positive when the device is lying on its back facing upwards and negative when it is facing downwards.
- Use the `MinimumInterval` property to set the minimum time between shakes in milliseconds.

The Accelerometer

`AccelerometerSensor` Events

- There are two events associated with the `AccelerometerSensor`: the `AccelerationChanged` event and the `Shaking` event.
- The `AccelerationChanged` event will be triggered whenever there is a change in the device’s acceleration.

Figure 12-45 `AccelerationChanged` Event Handler (*Source:* MIT App Inventor 2)



```
when AccelerometerSensor1 .AccelerationChanged
  xAccel  yAccel  zAccel
do
```

This event is triggered each time the device’s acceleration changes, and the parameters are updated with each change.

The Accelerometer

`AccelerometerSensor` Events

The `Shaking` event is triggered when there is a quick shake of the device.

Figures 12-46 and 12-47 demonstrate using the `AccelerometerSensor’s Shaking` event handler to play music.

The Accelerometer

Figure 12-46 Shaking App User Interface (*Source:* MIT App Inventor 2)

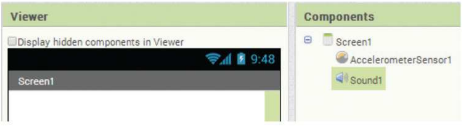
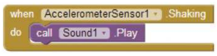


Figure 12-47 Shaking Event Handler (*Source:* MIT App Inventor 2)



```
when AccelerometerSensor1 .Shaking
do  call Sound1 .Play
```

The Accelerometer

AccelerometerSensor Events

In Figure 12-46 the AccelerometerSensor and a Sound component are added to the project. To try this example download the books companion website, at www.pearsonglobaleditions.com/gaddis.

Using the ActivityStarter Component to launch GoogleMaps

- The ActivityStarter component allows you to open up other apps.
- We are going to demonstrate the ActivityStarter by using it to open up Google Maps.
- If you know the latitude or longitude of the location, you can open up Google Maps, or a more general location by knowing the ZIP code.

Using the ActivityStarter Component to launch GoogleMaps

ActivityStarter Properties

To open up Google Maps from your application you will need to set just a few properties of the ActivityStarter.

Using the ActivityStarter Component to launch GoogleMaps

ActivityStarter Properties

- ActivityClass – This value for Google Maps is com.google.android.maps.MapsActivity.
- The ActivityPackage is com.google.android.apps.maps.
- DataUri — Is where we use the information we know about the address, either ZIP code or latitude and longitude values.