Emerging Technologies: Mobile Development for Android Devices

Sensors

Introduction

- Sensing
- Sensor categories
- Sensor types
- Using sensors in applications
- The Sensor Manager class
- The Sensor class
- Location sensors
- GPS location sensing
- GPS coordinates
- Security considerations
- Energy Considerations
- Conclusion

Sensing

- A key point of difference between desktop computers and portable devices is that portable devices typically come equipped with a large range of sensors.
- When using a portable device, user context is changing. Sensors can be used to derive information about user context and assist the user in what he/ she is doing.
- Light-level, temperature, location, movement, orientation, proximity.
- Data from sensors can be combined to create inferences about user situation and activity eg. It is cold, dark and raining and the location is a train station. A user in this context is more likely to require a train timetable, a taxi or a hot cup of coffee.
- Applications can use such an inference to provide better or faster service.

Sensor Categories

- A sensor can be an actual piece of physical hardware containing a transducer.
- Can also be a virtual sensor (software) of some kind, eg. outside temperature data sourced over a network.
- But hardware and software (virtual) sensors can work together to combine their data.
- Several sensors of the same type can also be combined and readings averaged (Adversely affects energy requirement/battery life).
- A sensor may provide raw sensor readings or be equipped to provide readings in standard units.
- For example, an 8-bit temperature sensor gives raw readings in the range 0 – 255 but readings are required in the 0 – 100 degrees Celsius range. Raw readings map to Celsius degrees.

Sensor Types

- Temperature
- Light level
- Proximity
- Barometric
- Accelerometer
- Gyroscope
- Magnetic field sensor
- GPS location sensor (Fine location)
- Network location sensor (Coarse location)

Using Sensors in Applications

- Sensors are managed through a series of sensor classes.
- These hide individual differences in how each sensor is managed and provides a unified interface for the developer.
- Applications register an interest in a sensor with the Android OS directly.
- In order to conserve battery energy Android deliers updates at the rate required by each individual application rather than having multiple application polling a sensor.

Using Sensors in Applications

- Permissions are not required to gain access to the majority of sensors.
- Reading an accelerometer or light sensor does not particularly advantage malicious code.
- Location sensors are an exception where explicit permission is required.
- An application can track a user’s location and a user may not want to disclose this information.

Using Sensors in Applications

- Although most devices are well-equipped, there is no guarantee that a sensor will be present on a device.
- A device should be queried for the existence of a sensor before trying to use it.
- Typically, check for the default sensor of the type concerned. If a null value is returned, then the device has no sensors of this type.

The Sensor Manager Class

- The Sensor Manager class is the class through which all access to sensors is handled.
- Query this service for sensors that an application requires as well as register sensor listeners.
- The Sensor Manager takes responsibility for triggering listeners whenever a new sensor value is obtained.

The Sensor Class

- The Sensor class contains all of the information that a device has about its sensors.
- Includes all constants that differentiate between the different sensor types.
- Supports querying the specification of individual sensors.

Sensor Delay Types

- There are three main update rates:
- SENSOR_DELAY_FASTEST
- Get updates as fast as the sensor can produce them.
- SENSOR_DELAY_GAME
- At a rate appropriate for game play.
- SENSOR_DELAY_UI
- At a rate that is suitable for updating a user interface (UI)

Sensors & the Activity Life Cycle

- Generally less than optimal to register sensors in onCreate( ). Why?
- Register sensors in onStart( )
- De-register in onStop( ). Why?
- Re-register again in onStart( )
- Don’t keep the sensor active when if the application is inactive. Energy → battery supply.
- A notable exception is GPS sensing. A GPS sensor takes several seconds to get a fix after starting.
- Because of this delay, consider carefully about how GPS sensors should be registered/de-registered to ensure best performance and best service to the user.
- Balance performance/accuracy/service to user against conservation of the battery resource.

Sensor Resolution/Accuracy

- A sensor such as light-level senses a continuous value.
- This is mapped to a discrete value for use in a digital system.
- Resolution could be 8-bit for example (0 - 255) that senses a continuous value in a digital system maps the continuous reading to a discrete value.
- 10-bit resolution would give a better resolution with 1023 discrete levels.
- A discrete value read from a sensor can be mapped to a real-world value (eg. Lux, degrees Celsius etc.) in software.
- Some devices may have multiple sensors of the same type.
- Multiple readings can be averaged to give better accuracy.
- But there is a greater energy requirement to maintain multiple active sensors.
- Note that accuracy and resolution are not the same.

GPS Location Coordinate System

- Location sensors use the world standard latitude and longitude system of surface coordinates.
- The earth's surface is mapped as a 2D plane.
- Latitude determines North and South and has a range of (-90, 90)
- Longitude determines East and West and has a range of (-180, 180)