COSC 3V97

Mobile Computing

1.1

Prerequisites
- COSC 2P13, 2P32

Staff
- Instructor: Dave Hughes (J312)
- Teaching assistant: Ron Bond

Lectures (EA102), lab (J310)

Web
- COSC: http://www.cosc.brocku.ca/
- COSC 3V97: http://www.cosc.brocku.ca/offerings/cosc3v97/

Email
- Dave Hughes: dhughes@brocku.ca

1.2

Course Procedures

Outline
Text
Software
- Android Studio 0.2.6, SDK Tools 22.0.5, Android SDK 2.2 & 4.3, Java 1.7

Hardware
Assignments
Project
- pairs
- proposal
- platform

Presentation
Lecture Schedule
Mobile Computing

- Software development for mobile equipment
  - smartphones, tablets
- Constraints/Considerations
  - differ from desktop systems
  - typically
    - power consumption
    - screen real estate
    - security model (sandbox)
    - unexpected interruption of execution
    - extended "running" time

Marketplace

- Platforms
  - Android, iOS, Windows 8, Blackberry
  - cross platform (Corona)
- Apps
  - smaller, cheaper, wider distribution
  - marketplaces run by drivers of platform
  - differing levels of control/validation
  - private individuals to large corporations

Android

- Driven by Google
  - Open Handset Alliance (OHA)
  - multiple H/W vendors
  - many different configurations
- Built around open-source and international standards
  - Java, XML, LINUX, Open GL ES, SQLite, …
  - no licensing fees for development
  - multi-platform development (Windows, OSX and LINUX)
- Now a mature platform
Android Platform

- Runs on Linux OS
  - each application runs as a separate user
  - kernel provides core functions
    - e.g. permissions & security, memory management, threading, network stack, H/W access
- Dalvik VM
  - each application runs on its own instance of VM
- Access to services/data/personal info
  - e.g. browser, contacts, phone, ...
  - permissions (granted by user on install)
  - via managers
    - e.g. LocationManager, AudioManager
  - apps can share data with other apps

Development Environment

- Two options:
  - Eclipse - using ADK plug-in
  - Android Studio - new Google environment
- For instructions see: http://developer.android.com/sdk/index.html
  - Eclipse - ADT Bundle for Windows
  - Android Studio - Android Studio for Windows
- Components
  - Java JDK – at least 1.5, preferably 1.7 (set 1.6 compatibility in Eclipse)
  - SDK manager
    - download/update SDK components
  - platforms
    - at least API 8 (2.2) and a recent one (e.g. API 18 (4.3))
Testing Environment

- Emulator
  - emulators for various generic Android devices (API levels)
  - AVD (Android Virtual Device) Manager
    ° set up emulators for various API levels
- On-device debugging
  - need USB driver for device
  - configure device
    ° select Unknown Sources and USB Debugging

Development Demo

- Run Android Studio
- Create a new Android project
  - fill in details
  - automatically creates necessary files
- Create an emulator
  - ADV Manager
- Run on emulator

Project Files

- Generated directories
  - src/main
    ° project code
  - src/main/java
    ° java code
  - res
    ° resources
    ° /layout
      - GUI
    - GUI editor
    - AndroidManifest
    ° project description
Logging

- Logging messages for debugging
- Set a debug tag for filtering
- Import android.util (for Log class)
- Levels
  - errors, warnings, information, debug, verbose

Debugging

- Debug
  - available as perspective in Eclipse
  - breakpoints
  - stepping through code
  - examining variables
- DDMS (Monitor)
  - perspective
  - devices
  - threads
  - profiling

Distribution

- Generate an Application Package (.apk)
  - generates .dex code (Dalvik Executable)
  - includes manifest
  - has an unique ID (unchanged through releases)
  - signed with a certificate
- Distributed
  - Android Marketplace
    - controlled by Google
    - register as Android developer
  - directly from developer
Android Application Basics

Foundational Classes

- **Context**
  - android.content.Context
  - application-wide configuration and data
  - instantiated as an Application object

- **Activity**
  - android.app.Activity
  - single task in an application
  - usually associated with a screen (UI)
  - subclass of Context

- **Fragment**
  - android.app.Fragment
  - new in Android 3.0 (11)
  - single function (UI) within a task
  - can be shared between activities

- **Intent**
  - android.app.Intent
  - encapsulates an asynchronous message to/from another activity
  - for inter-task communication/coordination

- **Service**
  - android.app.Service
  - task without user interaction
  - time-consuming or periodic tasks
  - background
  - subclass of Context
**Context**

- **Accessing (in Context)**
  - `Context context = getApplicationContext();`
- **Access to**
  - `resources`
    - `getResources`
    - `res directory`
    - `R.java`
  - `preferences`
    - `SharedPreferences`
    - `getSharedPreferences`
  - also: private files, assets, services, database, permissions

**Activity**

- Typically one per task that has UI
- Handles interaction with one screen
- Must be defined in the manifest
- e.g. simple game
  - splash screen
  - main menu
  - high scores, play, help

---

![Activity Diagram]

Figure 5.1: A simple game with five activities.
Processes

- Multi-processing
  - concurrent applications
  - application can multi-process
  - interrupts (e.g., phone call)
  - foreground/background
- Activity is unit for multi-processing
- Activity stack
- Application must manage resources and pause and resume seamlessly

Activity Lifecycle

- States
- Callbacks
  - Activity methods to be overridden
    - onCreate(Bundle savedInstanceState)
    - first started or restarted after killed
    - bundle may have state information if killed
    - initial setup including setContentView
    - onResume()
    - before becoming active
    - obtain resources needed to run
    - start resource hungry activities (e.g., audio play)
- onPause()
  - about to move to background
  - save resources loaded in resume
  - stop resource hungry activities
  - cleanup so less likely to be killed
- onDestroy()
  - not necessarily called
  - example

**Fragment**

- New in API 11
- Subdividing UI processing
  - compartmentalizing UI processing
  - common UI between activities
- e.g. multi-function (drill down) in music player
Intent

- Transitioning between activities
  - permanent
    - e.g. splash screen to main menu
    - new activity takes over, previous discarded
      - use startActivity and finish
  - temporary
    - e.g. main menu to functions
    - expect to return (back)
    - new activity pushed on top of previous
      - use startActivityForResult and onActivityResult

- Intent object
  - used to determine which activity to start
  - used to pass data to activity

- Starting activity
  - can start activity by name (class)
  - can start activity by intent filter
    - activities register intents they handle

- Passing data
  - intent has associated bundle (called Extras)
    - name/value pairs
    - putExtra and getExtra
Manifest

- XML file that defines the application
- Part of the package (.apk) that is generated from your project for distribution
- Describes properties for installation
  - name & icon
  - system requirements
  - permissions
  - registering components (activities, services etc.)

Identity

- Package name
- Version
  - code vs name
- App name
- Icon
System Requirements

- SDK version
  - min: backwards compatibility
  - target: most up-to-date
  - max: only in special cases
- Versions
  - most recent 4.3 (18) Jelly Bean

Platform Requirements

- Input
  - user-configuration
    - keyboards, navigation devices, touch screen settings
- Features
  - user-feature
    - optional device features/hardware
- Screen sizes and density
  - supports-screens
    - small/normal/large
Activities

- Activities must be defined in the manifest to be used
- Must be within the package declared in the manifest
  - dot notation or full pathname
- Primary entry point defined with an intent filter
  - action: MAIN
  - category: LAUNCHER
- Other intent filters can be specified to indicate that an activity, service or receiver can respond to implicit intents

Permissions

- Access to shared resources
  - e.g. contacts database, hardware such as camera
- Must be requested
  - uses-permission
  - upon install user must grant permission
- May not be enforced by all devices but request anyway
- Activities can also define and enforce their own permissions
  - when used by other applications