PROGRAMMING FOR MOBILE DEVICES

Programming for Mobile Devices

A McMonnies
School of Computing

Module Overview

1. The Characteristics of Mobile Devices & Applications
   - Mobile vs. Desktop
   - Design for Mobile
   - Design for Interaction & Usability
2. Mobile Web-Apps
   - HTML & HTML5
   - Javascript / Dynamic HTML
   - Event handlers
   - Web-app structure – HTML, CSS, Javascript
   - Organizing Web-Apps
   - A simple mobile web-app structure (based on jQuery Mobile)
3. Mobile App Frameworks
   - There are many of these – new ones arriving frequently
   - The mainstream ones are:
     - jQuery Mobile – based on jQuery: well known, easy to learn
     - Wakanda – a more complex (and more complete) framework, including IDE
     - Chocolate-Chip UI – gaining popularity
     - Kendo UI – many languages and platforms supported, corporate support (free on a trial basis)

Module Overview II

4. jQuery Mobile (jQM) – a web-app framework for mobile devices
   - ❑ Raw web-apps from scratch – problems and issues
   - ❑ jQM App Structure
   - ❑ Organizing a jQM Project
   - ❑ Page-links, transitions, dialogs
   - ❑ Listviews, dialogs, collapsible (concertina) content
   - ❑ Device hardware access – touch interface, orientation & accelerometers, sensors etc.
   - ❑ General mobile development principles
5. Javascript programming (revision – *you should know this!*)
   - ❑ Core principles – execute everything!
   - ❑ Variables, structures etc.
   - ❑ String Objects and Array Objects
   - ❑ Constructors and the Class Prototype
   - ❑ Classes, Inheritance & Polymorphism
   - ❑ OOP in Web Apps
   - ❑ "Business" classes

Module Overview III

6. CLASS TEST – A 1-hour multiple-choice paper
7. HTML5 features in Web Apps
   - ❑ Detecting browser support
   - ❑ Drawing & Graphics
   - ❑ Video and Audio
   - ❑ Geo-location, sensors, touch interfaces
   - ❑ Local Storage
   - ❑ Offline operation
8. WWW Data Feeds and the Single-Origin Policy
   - ❑ Online services – threading a needle in boxing gloves
   - ❑ Accessing online data
   - ❑ Online/offline functionality
   - ❑ Incorporating Online Data Using JSONP
   - ❑ <script> tags and Javascript
   - ❑ Finding JSONP Services
   - ❑ Creating a data-feed Proxy

Module Overview IV

9. RSS Feeds and AJAX
   - ❑ AJAX – event-driven access to HTTP request/ response
     - ❑ jQuery Mobile supports this with well-established functions
   - ❑ RSS Feeds and your Project
     - ❑ jGFeed – the Google feed component
10. HTML5 installable Apps
    - ❑ The HTML5 Manifest
    - ❑ Debugging Manifest problems
    - ❑ Data-Feeds and Offline Access
11. Coursework Workshop
12. Module Overview

Module Assessment & Info.

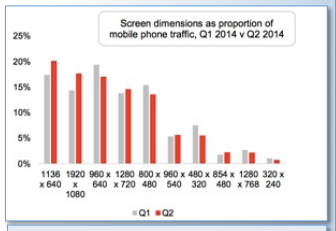
- Assessment for this module will be by:
  - A Class test – 20%
    - Done during lecture time – week 6
  - A Practical Development Project
    - Part 1: A Project Specification – 30%, due for submission week 8
      - You will create a document that fully describes what you intend to build for your final project
      - You will choose from a number of project briefs, or can specify your own (provided I get to vet it first)
    - Part 2: The project development – 50%, due for submission week 12
      - Create a working application that meets specification submitted in week 8
      - The application uses specific features of a mobile platform
      - This will be group-work for groups of TWO students
        - » One is possible (but not advised)
        - » Groups of 3 or more, ***not allowed***
- Moodle:
  - All materials (including assessment info, slides, labs, example programs, additional notes etc.) will be posted on Moodle
  - You should already be enrolled on this via your Banner registration for the module

Lecture 1

THE CHARACTERISTICS OF MOBILE DEVICES AND APPLICATIONS

The Distinctive Features of Mobile Platforms

- Restrictions
  - Limited Display Size: Typical desktop: 1200x1024pixels, mobile: bigger range (~1000x700), but need to support limited screen devices
  - Limited Power: Typical desktop-3GHz, mobile ~1GHz (2GHz is now reaching some devices – debatable benefits)
  - Limited Memory: Typical desktop- 4GBytes, mobile-128-512MBytes
  - Battery life: Smartphone typical – 2Days standby, 1 day with normal use
  - Intermittent Connectivity (e.g. GSM/GPRS/3G/4G) – possible cold spots
  - Input Methods/Devices – touch screen, mini keyboard
- Advantages
  - Portable
  - Convergence of Devices (Phone, PDA, Camera, Internet)
  - Mobility (and use of Location Services)
  - Built-in: Accelerometers, GPS, SMS, Wireless Networking

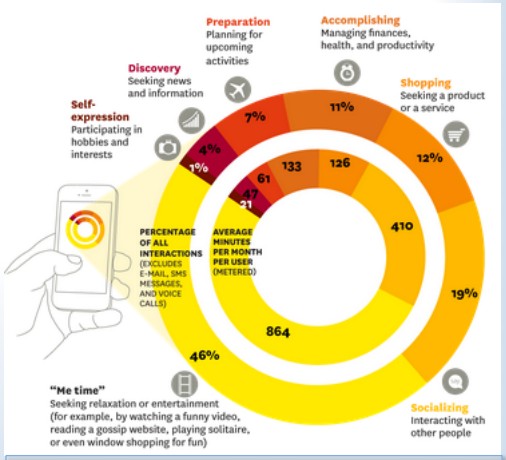


http://www.marketingprofs.com/charts/2014/25740/mobile-trends-most-popular-phones-screen-sizes-and-resolutions

Typical Tasks for Mobile Devices

- Phone & Messaging
- Internet & eMail
- Access Cloud Services
- GeoLocation & Mapping
- Games platform
- Diary & Alarms
- Contacts
- Photography
- Music Player
- Video Player

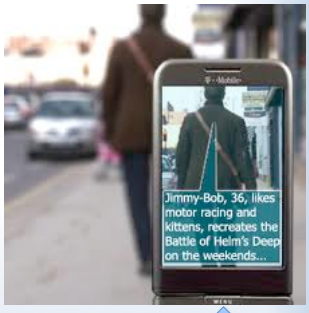
Note – most mobile APIs provide access to these features



Source: Harvard Business Review https://hbr.org/2013/01/how-people-really-use-mobile

Developer Opportunities

- Mobile Platforms provide a range of opportunities for developers
  - ***Combining core applications and services***
    - Finding the route to a contact's workplace
    - Geo-Tagging photographs
    - Looking up prices via bar-codes (or photographs)
    - Identifying landmarks
  - ***Taking data-sets out of the office***
    - Customer info
    - Catalogs
    - PNC lookups for License plates/fingerprints/resident ID
  - ***Emerging platforms***
    - Near-Field Communications for payments e.g. Oyster, iOS Payment, Google Checkout
    - Context Aware Apps – e.g. in office, at football, shopping
    - Augmented reality – i.e. overlay virtual data on a live camera image
      - Potential for games, better GPS directions, emergency medical treatment, ?



Specification of a Mobile Device

- Typically
  - Smaller memory (e.g. iPhone: 1GByte) (Android, up to 2GB)
  - Lower speed (e.g. iPhone 6: 1GHz – 1.3GHz) (HTC One – 2GHz)
  - Small offline storage (Typical: 16-64GBytes – i.e. Flash card size)
  - Slower Internet Connection (Most *claim* up to 7.2 Mbit/Sec – up to 2MBit/Sec is more common); 4G access is growing, but still nowhere near Broadband speeds
  - Screen size (e.g. iPhone 6 Retina Display is 750px X 1334px in a 4.7inch diagonal screen – more typical is 640x960 or 640x1136 HD))
  - Camera Resolution (e.g. iPhone 5: 8MPix, iPhone 6: 8MPix, HTC Desire: 5MPix, HTC One: 4MPix, Samsung Galaxy S5: 16MPix)
- These specs are quite high, even compared to quite recent desktop systems and mobile devices typically perform several high-end functions (photo/video/audio) in addition to phone and messaging, but
  - Developers still need to support the MAJORITY of lower spec devices out there
- Biggest limitations are in input devices, speed, storage and connectivity, so…
  - Apps need to be written with efficiency of these factors in mind

Development for a Mobile Device

- There is a great deal to learn:
  - Windows Mobile 7
    - Based on .NET (.NET CE) / Program in C#, Visual Basic or (for specialist apps) C++
  - iPhone/iPad
    - iOS based on OS X (BSD Unix with a GUI layer and services) / Program in ObjectiveC and Swift (a new language)
  - Android
    - Based on a custom-built Java Virtual Machine (Dalvik or ART) hosted on a small Linux platform
    - Program in a variant of Java / App-Inventor is a visual development system – limited feature set, Google gave it away
  - Blackberry
    - Based on a variant Java Virtual Machine with a custom host / Program in Java
  - Nokia Phones
    - A Symbian platform (developed by Psion) / Program in C++, but also Python and Java ME
  - Sony/Ericsson
    - A custom platform (developed by Ericsson) / Program in Java
- Although it seems that Java is a common denominator, application design and implementation is different for each device – can't run Android apps on Blackberry, for example
- It seems developers would have to specialize or a particular device family, but…

Mobile Development II

- Many of these devices have a browser based on the WebKit browser core
  - iPhones, Android, Symbian and Sony Ericsson (via the Qt Toolkit)
- WebKit supports much of HTML5
  - HTML5 is HTML enhanced to support a number of native processing tasks – graphics, geo-location, forms, Video and Audio etc.
  - This gives us a common platform for developing apps
  - HTML5 coupled with a web-kit based mobile library (e.g. jQuery Mobile, Kendo UI etc.) provides a single development path that targets ***most*** devices
- WebKit supports a 'native app' mode, where the address bar is hidden to make the app run full-screen, and the app remains available even when the device is offline
- Downside
  - The main programming language for HTML5 is Javascript
  - This is a horror of a language with few redeeming features (as you all know by now)
- That said, Javascript is now the most significant development language available because of its cross-platform capabilities, and it is improving
- There are good frameworks available for WebApp development
  - Notably jQuery Mobile, Ionic, Kendo UI, Ratchet etc.

Design for Mobile Devices

- Desktop Applications
  - Will be used in one place
  - Typical use will be for a significant period of time
  - User will be
    - comfortable
    - seated
    - easily able to manipulate mouse and keyboard
    - able to take breaks
  - Also, desktop applications are often based on familiar paradigms
    - Documents
    - Multiple information sets allow comparisons
- Mobile Applications
  - Used anywhere
  - Typical use will be in short bursts
    - Look up info
    - Take short notes
    - Read messages
    - Send messages
  - User may be constrained
    - carrying stuff
    - driving (not recommended)
    - In a rush for a train
  - Information should be presented in small chunks
  - Applications should be tolerant of user inaccuracy

Design Goals

- Avoid making a miniature version of the desktop application
  - Squeezing down screen elements is exactly what not to do
  - Aim to provide two types of view
    - Overview of an information set
    - Detailed view of one item
- Mobile kit knows…
  - Time and location
  - Incorporate this into apps
  - e.g. Location can be used to filter the available information
    - Don't show EVERY store in your chain – show the LOCAL ones
    - Provide access to services that are CURRENTLY available
- Mobile usually means lower precision
  - User will make mistakes (e.g. entering data on "keyboard")
  - Allow for this
    - Provide 'undo' operations
    - Ask for confirmation before doing anything irreversible
    - Make buttons BIG
    - Make lists BIG
    - Use graphics where possible
  - Typical UI development kit is designed for this already
- Aim for elegance in a solution
  - Less is more
    - Fewer items in a list
    - Fewer text fields
  - Stash entered information for re-use

Usability Issues

- Oops – I made a mistook
  - Always provide an opportunity to confirm entered data
  - Where possible, provide Undo-Redo
  - Go for clarity in irreversible operations
    - What will pressing this button DO!
- Feedback
  - Tell the user if something will take time
  - Allow user to back-off
    - Ideally, do the thing later with the currently entered data
- Understandable
  - Avoid jargon in user messages
  - Keep messages short (and BIG)
  - Provide labels or placeholders for all inputs
- Respond quickly
  - Always confirm a user-action immediately
    - Even when the actual operation will take time
- Use animations
  - to indicate the passage of time
- Use sound
  - A Quiet Click when button pressed
- Minimize the number of times the user has to interact to perform an action
  - E.g. in Android, you can
    - Display an alert (requires a click from the user)
    - Pop up a timed message (giving time for user to read it)
  - Choose the latter – no need for the user to hit a confirm button while running for a train