Mobile Application (Design and) Development

15th class

Prof. Stephen Intille
s.intille@neu.edu

Today

- Schedule revisions and grading
- Q&A
- Design mashup checklisk
- Track Stephen programming assignment
- Review of services and broadcast receivers
- Start SQL
- Two design paper presentations

Schedule

- http://www.ccs.neu.edu/home/intille/teaching/MobileApplicationDevelopment2011Syllabus.htm

Grading reminder

- Categories
  - A: Superior, striking, or unexpected pieces of work with excellent effort demonstrating a mastery of the subject matter and a skillful use of concepts and/or materials discussed in class; work robustly and fully implemented; work that shows exceptional imagination, elegance of presentation, originality, creativity, and effort.

Grading reminder

- Categories
  - B: Good work demonstrating a capacity to use the subject matter and the ability to handle problems encountered in the course.
  - C: Work that is adequate but that would benefit from increased effort or preparation.

Grading reminder

- Final grade weighting from syllabus
  - Quick quizzes on reading (5%)
  - Class paper presentation and/or code reviews (10%)
  - Design assignments (25% total)
  - Project programming lead-up assignments (30% total)
  - Final project interaction design and robustness (30%)

Q&A

Your design mashup checklist

- Simple gameplay
- “Just one more” reward mentality (incremental reveals) “so close!”
- Simple, curious storyline (that is easy to remember when take a break and come back)
- Some mystery
- Slowly adding complexity

Your design mashup checklist

- Bright, colorful yet simple graphics
- Easy to pause/quit and return later
- Layered complexity for experts to keep them engaged (e.g., at first pass/fail then scoring on levels)
- Can play in bits and spurts and make progress

Your design mashup checklist

- Rules can be either easily explained (pictorially) or figured out by trial and error
- Boards create many different combinations that keep game interesting
- Some games auto-restart to “hook” people in after an unsuccessful level
- Some games less fun with sound muted (create tension but in non-irritating way)

***Your*** design mashup checklist

- Simple controls
- Intellectually stimulating (some)
- Positive rewards (e.g. victory sounds)
- User sets pacing (some games)
- Fast and responsive controls
- If menus at all, short and appealing

Your design mashup checklist

- Cute animations (and speed can vary pace of game)
- Cute sounds (increase competitive feel)
- Simple gameplay
- Many levels
- Can start playing without knowing the rules
- Rules can be either easily explained (pictorially) or figured out by trial and error

Your design mashup checklist

- Boards create many different combinations that keep game interesting
- Some games auto-restart to “hook” people in after an unsuccessful level
- Some games less fun with sound muted (create tension but in non-irritating way)
- Simple controls
- Intellectually stimulating (some)

Your design mashup checklist

- Positive rewards (e.g. victory sounds)
- User sets pacing (some games)
- Fast and responsive controls
- If menus at all, short and appealing
- Finger approved. Manipulations of objects right size for fingers.
- One handed play helps.

Your **design mashup checklist**

- Explicit use of short term memory (7 +/- 2) to create challenge
- No unintentional demand on short-term memory
- Scarcity of stuff (and innate tendency to want to acquire it; badges)
- Social interaction, but in a manageable, local way (e.g., FarmVille)

**Your** **design mashup checklist**

- Reward simple behaviors that show engagement (e.g. logging in)
- Variation in pacing (some games)
- Very limited simple options for what user needs to do (press buttons, swipe across screen, etc)
- Leaderboards
- Scoring mechanism easy to understand

Your design mashup checklist

- Micro-goals (e.g., high score under score)
- Remembers the history of previous games (creating individual challenge when you return)
- Age group appropriate
- Use of other human ingenuity ... sharing of solutions
- Variable length of time of gameplay possible

Your design mashup checklist

- Use of randomness/uncertainty in a natural way in the gameplay
- (Visual) cues that let someone fix past mistakes and iterate
- “Invited into world” by visuals that also set expectation on moral behavior
- Minimal text
- Small set of rules create large set of interesting behaviors and gameplay

Your design mashup checklist

- Think about real-world time vs. game-world time and mapping
- Hints can create challenge
- Clever use of muscle memory (unique physical feel during play)

- Miss any?

Your design mashup checklist

- Checklist to compare your projects against

Track Stephen assignment

- Much learning about what information is at your disposal ... Great!

- Examples...

Information gathered

- Network information (phone number, service provider, country)
- Application info (Packages, apps installed (with icons), application changes)
- Hardware info (Phone brand, device type and model, device manufacturer, CPU)
- OS info/version

Information gathered

- Location (instantaneous)
- Location (type of data being used)
- Location (successfully tracked)
- Plot on graph

Information gathered

- Incoming calls (during test, for entire history of use of the phone, including missed or answered)
- Outgoing calls (during test, for entire history of use of the phone)
- Time spent on phone screen
- Phone-related behavior (screen, headset, power connections)

Information gathered

- Memory usage on phone an SD card
- Available sensors
- Display size and refresh rate
- Text message log
- Phone contacts
- Email
- Accelerometer data (moved phone, raw data)
- Battery info

Information gathered

- Battery info
- Phone usage time
- User interacting with phone time
- Record audio snippets
- Take pictures
- Wallpaper changed
- Airplane mode activated
- Recording audio of phone calls

Issues with logging

- Accelerometer issues in OS prior to v2.1
- App service should start on reboot
  - Can’t assume phone won’t get restarted
- Many apps don’t seem to be saving state and actually logging
- Services crashing regularly
  - Most errors will be because of null objects
  - Some errors due to UI thread locking/slow

Service/wakelock/logging

- Step through an example

SQLite

- Alternative to SharedPreferences for saving data
- Best for data that you plan to run small DB queries on from your app
  - Open source
  - Standards-compliant
  - Lightweight
  - Robust (hopefully)

SQLite (http://www.sqlite.org)

- Pros
  - Open source
  - Standards-compliant
  - Lightweight
  - Robust (hopefully)
- Cons
  - Loosely type columns

Syncing

- "Automatic" sync with web server database?
  - Typical scenario

- Recommended strategy
  - Keep data management/user simple on the phone
  - Keep complexity on the server
  - You don't want problems out of your control...

Basics

- ContentValues used to insert new rows into tables
- Queries are returned as Cursor objects
  - Pointers to result set within underlying data
  - Managed way of controlling position (row) in result set of a DB query
- startManagingCursor (stopManagingCursor)
  - Integrates Cursor lifetime into calling Activity's

Cursor class

- moveToFirst, moveToNext, moveToPrevious
- getCount
- getColumnIndexorThrow (from name)
- getColumnName (from index)
- getColumnNames (in current cursor)
- moveToPosition (to row)
- getPostion (cursor position)

Helper classes

- Think through what you need and make the helper classes that will make use of DB go smoothly
  - Typing
  - Error checking
  - Syncing
  - Handle queries
  - Expose methods for creating, opening, closing
  - Publish DB constants

ContentProvider

- Generic, well-defined interface for using and sharing data
- Convention for URI:
  - Content://com.<CompanyName>.provider.<ApplicationName>/<DataPath>
    - Content://com.company.provider.myapp/elements (request for all values of type elements)
    - Content://com.company.provider.myapp/elements/5 (request for single, 5th element)

ContentProvider

- Generic, well-defined interface for using and sharing data
- Convention for URI:
  - Content://com.<CompanyName>.provider.<ApplicationName>/<DataPath>
    - Content://com.company.provider.myapp/elements (request for all values of type elements)
    - Content://com.company.provider.myapp/elements/5 (request for single, 5th element)

ContentProvider

- Typically exposing access to a SQLite DB
- But, can also expose access to any source of data (files, application instance variables)
- Use ContentResolver object to modify and query ContentProviders
- Query results returned as Cursors

ContentProvider

- Using query in ContentResolver, pass in:
  - URI of the ContentProvider you want to query
  - Projection that lists the columns you want in result set
  - A where clause that defines the rows to be returned (can use wildcards: ?)
  - An array of selection argument strings that replace wildcards (?)
  - A string that describes the order of the returned rows

Step through example