MAD

Mobile Application Design
Mobile is different

• Smartphones and desktop computers are very different
  — One might be tempted to think of mobile devices as underpowered versions of 'real' computers

• Smartphones are actually more powerful than desktops in many ways
  — Designing for mobile is very different
    — Many more options
THINK “FAST FOOD” – SIMPLE, CHEAP AND ADDICTING.
Some key choices

- Market
- Device(s)
- Operating system
- Legacy components
- Deployment/distribution model
- Complete process
What is the target?

• Controlled set of users
  – Customers
  – Employees

• Open set of users
  – Market
Devices

• > 1000 Android devices
## Operating system

<table>
<thead>
<tr>
<th>Version</th>
<th>Code name</th>
<th>Release date</th>
<th>API level</th>
<th>DVM/ART</th>
<th>Distribution</th>
<th>First devices to run version</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.0</td>
<td>Nougat</td>
<td>August 22, 2016</td>
<td>24</td>
<td>Unknown</td>
<td>&quot;less than 0.1%&quot;</td>
<td>Nexus 5X, Nexus 6P</td>
</tr>
<tr>
<td>6.0</td>
<td>Marshmallow</td>
<td>October 5, 2015</td>
<td>23</td>
<td>Unknown</td>
<td>18.7%</td>
<td>Nexus 5X, Nexus 6P</td>
</tr>
<tr>
<td>5.1</td>
<td>Lollipop</td>
<td>March 9, 2015</td>
<td>22</td>
<td>Unknown</td>
<td>21.9%</td>
<td>Android One</td>
</tr>
<tr>
<td>5.0</td>
<td></td>
<td>November 3, 2014</td>
<td>21</td>
<td>ART 2.1.0</td>
<td>13.1%</td>
<td>Nexus 6</td>
</tr>
<tr>
<td>4.4</td>
<td>KitKat</td>
<td>October 31, 2013</td>
<td>19</td>
<td>ART 1.6.0</td>
<td>27.7%</td>
<td>Nexus 5</td>
</tr>
<tr>
<td>4.3</td>
<td>Jelly Bean</td>
<td>July 24, 2013</td>
<td>18</td>
<td>Unknown</td>
<td>2.3%</td>
<td>Nexus 7 2013</td>
</tr>
<tr>
<td>4.2</td>
<td></td>
<td>November 13, 2012</td>
<td>17</td>
<td>Unknown</td>
<td>7.7%</td>
<td>Nexus 4, Nexus 10</td>
</tr>
<tr>
<td>4.1</td>
<td></td>
<td>July 9, 2012</td>
<td>16</td>
<td>Unknown</td>
<td>5.6%</td>
<td>Nexus 7</td>
</tr>
<tr>
<td>4.0</td>
<td>Ice Cream Sandwich</td>
<td>December 16, 2011</td>
<td>15</td>
<td>Unknown</td>
<td>1.4%</td>
<td>Galaxy Nexus</td>
</tr>
<tr>
<td>2.3</td>
<td>Gingerbread</td>
<td>February 9, 2011</td>
<td>10</td>
<td>DVM 1.4.0</td>
<td>1.5%</td>
<td>Nexus S</td>
</tr>
<tr>
<td>2.2</td>
<td>Froyo</td>
<td>May 20, 2010</td>
<td>8</td>
<td>Unknown</td>
<td>0.1%</td>
<td>Droid 2</td>
</tr>
</tbody>
</table>

Different operating systems
Legacy components

- Where do we came from?
- Where do we want to go?
- What assets do we want to keep?
- Which systems are we supposed to integrate?
Deployment model

• Custom systems
• Closed world
  – Centralized distribution
  – Dedicated sites
• Market place
Main development activities

- Think/Prototype
- Design
- Develop
Some key characteristics

- Mixed teams
- Sketches and prototyping
- Designers and not just computer scientists
How to start

• Mobile mindset
  – Focused, unique, charming, user-centered

• Different classes of users
  – Clearly identify your target(s): bored, busy, lost

• First impression is key
  – Limited/No help text
  – Characteristic and intriguing look and feel
  – Just a few seconds and the app…

http://www.netmagazine.com/features/10-principles-mobile-interface-design
Be bold

• Users are captured by unique design
• Users get tired of seeing the same old thing
• Do not use Android/Apple-supplied UI elements as a always-good solution
  — They are starting to look dated
Sketching
Something a bit more complete . . .
TODAY PASTA WITH...

Bucatini with Calabrian Style Pesto, with Fresh Ricotta and Basil-Infused Oil

DISCOVER THE RECIPE
Find your journey:
From: Milano Centrale
To: Roma Termini
Round-trip: Outward, Round-trip

When do you want to leave?
Date: 10/16/14, 14:16

Passengers (Max 5):
Adults: 1
Children: 0

Search
How many features

• Users do not spend time discovering features
• Users do not complain about “advanced” features
  – More features imply more apps
• Users complain about features that do not work
Android design principles

• Enchant me
  – Delight me in surprising ways
  – Real objects are more fun than buttons and menus
  – Let me make it mine

• Simply my life
  – Keep it brief
  – Pictures are faster than words
  – Decide for me but let me have the final say
  – I should always know where I am
Further iOS design suggestions

• Use Layout to Communicate
• Avoid asking people to supply setup information
  – Focus on the needs of 80% of your users
• Launch in the device’s current orientation
• When your app restarts, restore its state so users can continue where they left off
• An iOS app never displays a Close or Quit option
  – Never quit an iOS app programmatically
Final suggestions

• Single and appropriate navigation model
• Minimal user inputs (through the proper means)
  — Auto-correct can be so frustrating
• Gestures are not really standardized
  — They are nice to have, but not mandatory
• Support orientations
  — Be consistent and exploit orientation locks
• Communications
  — Provide polite feedback, modal alerts, confirmations
• Postpone sign up

http://www.netmagazine.com/features/10-principles-mobile-interface-design
Flat design

- If your app looks outdated, users will note that
Flat design

• Not boring
• Ornamental elements are viewed as unnecessary clutter
• Bright, contrasting colors make illustrations and buttons pop from backgrounds
• Minimalistic nature

http://www.creativebloq.com/graphic-design/what-flat-design-3132112
Consistent layout

- Can be very “expensive”
- Extremely important
- Design libraries exist to help decide which layout is the best for a particular problem
Anti-patterns

- Metaphor mismatch
  - Control, icon, or mental model mismatch
- Idiot boxes
- Too many chart elements
- Oceans of buttons
Avoid PCisms

Images courtesy of Mobile Design Pattern Gallery by Theresa Neil
Development options
Model-View-Controller

**Model**
- Encapsulates application state
- Responds to state queries
- Exposes application functionality
- Notifies views of changes

**View**
- Renders the models
- Requests updates from models
- Sends user gestures to controller
- Allows controller to select view

**Controller**
- Defines application behavior
- Maps user actions to model updates
- Selects view for response
- Uses one for each functionality
Web-based solution

**Pros**
- It is not installed on the device
- Being server-based, it can easily be updated
- The same user experience can be reused on different platforms

**Cons**
- Being internet-based, performance can be an issue
- The interactions with local software and hardware components is limited
- It is not distributed through a marketplace
Hybrid solution

**Pros**
- The user experience can be based on native elements and be reused
- It can (partially) interact with the hardware components of the device
- It can be distributed through a marketplace

**Cons**
- Performance can be an issue given the need for an interpreter
- JavaScript might be interpreted differently on different devices
- The user experience is only close to the native one
Interpreted solution

**Pros**
- The user experience corresponds to the (basic) native one
- The business logic can be reused
- It can be distributed through a marketplace

**Cons**
- Performance can be an issue
- The reuse of the user experience depends on the abstraction level of the framework
- The actual development depends on the specific framework
Cross-compiled solution

**Pros**
- It can offer all the characteristics of a native solution
- Hardware and software components can be exploited
- Performance is usually good

**Cons**
- The user experience usually cannot be reused
- There could be some limitations in the way hardware components can be used
- The result is usually not too sophisticated
Native solution

Pros
- It can be efficient and special-purpose
- It can fully exploit any single characteristic
- It can (easily) provide a completely native user experience

Cons
- Development costs tend to become high
- One development for each platform
- Almost no reuse