Mobile Application (Design and) Development

5th class

Prof. Stephen Intille
s.intille@neu.edu

Q&A

- Anything new?

- Workspace setup – speed of emulator

Today

- Overview of UI elements and Themes
- Looking at Sudoku code (if time)

- Design paper
  - Using Mobile & Personal Sensing Technologies to Support Health Behavior Change in Everyday Life: Lessons Learned
  - Presenter: Varun Ramachandran

Tomorrow

- Saving data
- More on rapid prototyping
  - Two Readings on the wiki

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

UI fundamentals

- Views
  - Controls or widgets (not App Widgets)
  - All UI controls, including layout classes, derived from views
- View Groups
  - Extension of View that can contain multiple child groups
  - ViewGroup extended to provide layout managers that help you layout controls
- Activities

Creating UI with views

- Inflating a layout (XML option)

```
@Override
public void onCreate(Bundle icicle) {
  super.onCreate(icicle);

  setContentView(R.layout.main);
  TextView myTextView =
(TextView)findViewById(R.id.myTextView);
}
```

Creating UI with views

- Inflating a layout (manual option)

```
@Override
public void onCreate(Bundle icicle) {
  super.onCreate(icicle);

  TextView myTextView = new TextView(this);
  setContentView(myTextView);

  myTextView.setText("Hello, Android");
}
```

Widget types

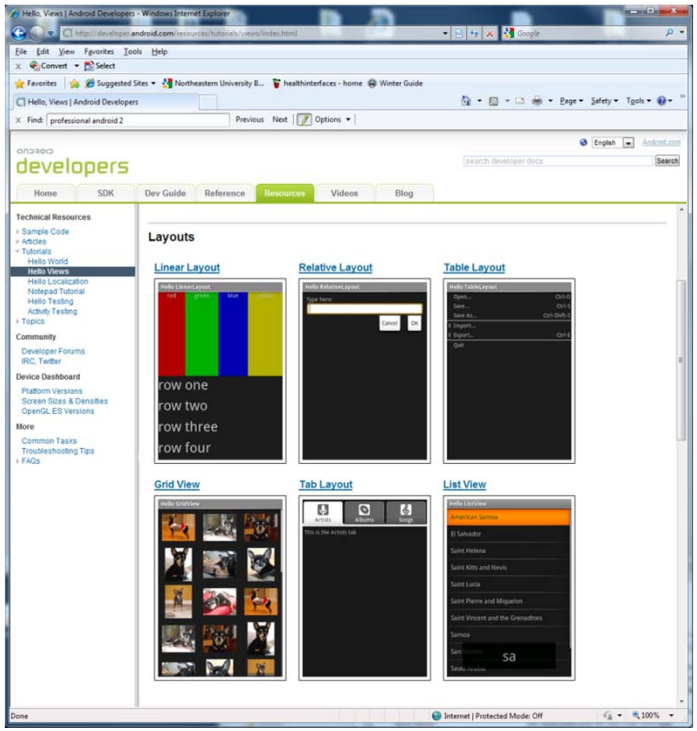
- TextView
  - Multiline, string formatting, auto word wrap
- EditText
  - Multiline entry, word wrap, hints
- ListView
  - Vertical list of views
- Spinner
  - Text view and associated List View; select an item from a list to display in the textbox

Widget types

- Button
- CheckBox
- RadioButton
- ViewFlipper
  - Collection of views as a horizontal row where only one view is visible at a time; animated transitions
- QuickContactBadge
  - Badge w/ image icon assigned to contact

Info on views

- Very helpful web page:
  - http://developer.android.com/resources/tutorials/views/index.html



Layout managers (aka layouts)

- Extensions of ViewGroup
- Can be nested for arbitrary complexity
- But, watch out! Behavior can be a little tricky with fancy nesting

- Useful website:
  - http://developer.android.com/guide/topics/ui/layout-objects.html

Common layouts

- FrameLayout
  - Pins children to top left corner
- LinearLayout
  - Vertical or horizontal line
- RelativeLayout
  - Flexible. Define positions relative to others and screen boundaries
- TableLayout
  - Grid of rows and columns
- Gallery
  - Single row of items in a horizontally scrolling list

XML for layouts

- Decouple presentation from View/Activity
- Hardware-specific variations dynamically loaded

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent">
  <TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Enter Text Below"
  />
  <EditText
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Text Goes Here!"
  />
</LinearLayout>
```

In code an option

- Avoid where you can
- But sometimes you'll need to "inflate"

```
LinearLayout ll = new LinearLayout(this);
ll.setOrientation(LinearLayout.VERTICAL);

TextView myTextView = new TextView(this);
EditText myEditText = new EditText(this);

myTextView.setText("Enter Text Below");
myEditText.setText("Text Goes Here!");

int lHeight = LinearLayout.LayoutParams.FILL_PARENT;
int lWidth = LinearLayout.LayoutParams.WRAP_CONTENT;

ll.addView(myTextView, new LinearLayout.LayoutParams(lHeight, lWidth));
ll.addView(myEditText, new LinearLayout.LayoutParams(lHeight, lWidth));
setContentView(ll);
```

Keep in mind

- Inflating expensive
- Complex layouts confusing
  - For you
  - For user
- Avoid unnecessary nesting
- Void too many Views
- Avoid deep nesting
- To help: layoutopt command line tool

Keep in mind 2

- Tempting to want to create custom views
  - Adds complexity
  - User is familiar with standards (perhaps apply a nice theme, but don't force them to learn something new)
- Best to extend appearance of existing views or combine views

Modifying views

```
public class MyTextView extends TextView {

  public MyTextView (Context context, AttributeSet ats, int defStyle) {
    super(context, ats, defStyle);
  }

  public MyTextView (Context context) {
    super(context);
  }

  public MyTextView (Context context, AttributeSet attrs) {
    super(context, attrs);
  }

  @Override
  public void onDraw(Canvas canvas) {
    [ ... Draw things on the canvas under the text ... ]

    // Render the text as usual using the TextView base class.
    super.onDraw(canvas);

    [ ... Draw things on the canvas over the text ... ]
  }

  @Override
  public boolean onKeyDown(int keyCode, KeyEvent keyEvent) {
    [ ... Perform some special processing ... ]
    [ ... based on a particular key press ... ]

    // Use the existing functionality implemented by
    // the base class to respond to a key press event.
    return super.onKeyDown(keyCode, keyEvent);
  }
}
```

Creating compound controls

```
public class MyCompoundView extends LinearLayout {
  public MyCompoundView(Context context) {
    super(context);
  }

  public MyCompoundView(Context context, AttributeSet attrs) {
    super(context, attrs);
  }
}
```

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent">
  <EditText
    android:id="@+id/editText"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
  />
  <Button
    android:id="@+id/clearButton"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Clear"
  />
</LinearLayout>
```

Creating compound controls

```
public class ClearableEditText extends LinearLayout {

  EditText editText;
  Button clearButton;

  public ClearableEditText(Context context) {
    super(context);

    // Inflate the view from the layout resource.
    String infService = Context.LAYOUT_INFLATER_SERVICE;
    LayoutInflater li;
    li = (LayoutInflater)getContext().getSystemService(infService);
    li.inflate(R.layout.clearable_edit_text, this, true);

    // Get references to the child controls.
    editText = (EditText)findViewById(R.id.editText);
    clearButton = (Button)findViewById(R.id.clearButton);

    // Hook up the functionality
    hookupButton();
  }
}
```

Creating compound controls

```
public class ClearableEditText extends LinearLayout {

  EditText editText;
  Button clearButton;

  public ClearableEditText(Context context) {
    super(context);

    // Inflate the view from the layout resource.
    String infService = Context.LAYOUT_INFLATER_SERVICE;
    LayoutInflater li;
    li = (LayoutInflater)getContext().getSystemService(infService);
    li.inflate(R.layout.clearable_edit_text, this, true);

    // Get references to the child controls.
    editText = (EditText)findViewById(R.id.editText);
    clearButton = (Button)findViewById(R.id.clearButton);

    // Hook up the functionality
    hookupButton();
  }

private void hookupButton() {
  clearButton.setOnClickListener(new Button.OnClickListener() {
    public void onClick(View v) {
      editText.setText("");
    }
  });
}
```

Completely new views

- Complete control over look and feel
- Extend View or SurfaceView classes
- View
  - Provides Canvas w/ draw methods
  - Paint classes
  - Use with bitmaps and rasters
- SurfaceView
  - Surface object that supports openGL and drawing from background thread

Menus

- Device's menu button: icon menu
- Context menus
  (long press on View in focus)

Icon menu

- Typically six items
- Expanded menu
  - Opens when “more” pressed
  - Scrollable list of items not visible before

Submenus

- Floating window
- No submenus in submenu (Keep menus simple)

Adding a menu item

```
static final private int MENU_ITEM = Menu.FIRST;

@Override
public boolean onCreateOptionsMenu(Menu menu) {
  super.onCreateOptionsMenu(menu);

  // Group ID
  int groupId = 0;
  // Unique menu item identifier. Used for event handling.
  int menuItemId = MENU_ITEM;
  // The order position of the item
  int menuItemOrder = Menu.NONE;
  // Text to be displayed for this menu item.
  int menuItemText = R.string.menu_item;

  // Create the menu item and keep a reference to it.
  MenuItem menuItem = menu.add(groupId, menuItemId,
                               menuItemOrder, menuItemText);

  return true;
}
```

- Pass ID to menu findItem to get MenuItem

Menu item options

- Checkboxes
- Radio buttons
- Shortcut keys
- Condensed titles
- Icons
- Better to use onOptionsItemSelected than
  - Menu item click listener
  - Intents

Dynamic updates of items

- Modify menu based on current state just before menu displayed

```
Override
public boolean onPrepareOptionsMenu(Menu menu) {
  super.onPrepareOptionsMenu(menu);

  MenuItem menuItem = menu.findItem(MENU_ITEM);

  [ ... modify menu items ... ]

  return true;
}
```

Handling menu selections

- Single event handler for menus: onOptionsItemSElected method

```
public boolean onOptionsItemSelected(MenuItem item) {
  super.onOptionsItemSelected(item);

  // Find which menu item has been selected
  switch (item.getItemId()) {

    // Check for each known menu item
    case (MENU_ITEM):
      [ ... Perform menu handler actions ... ]
      return true;
  }

  // Return false if you have not handled the menu item.
  return false;
}
```

Assigning context menu to view

```
@Override
public void onCreate(Bundle icicle) {
  super.onCreate(icicle);

  EditText view = new EditText(this);
  setContentView(view);

  registerForContextMenu(view);
}
```

- onCreateContextMenu handler triggered when context menu needed for View

Assigning context menu to view

```
@Override
public void onCreateContextMenu(ContextMenu menu, View v,
                                ContextMenu.ContextMenuInfo menuInfo) {
  super.onCreateContextMenu(menu, v, menuInfo);

  menu.setHeaderTitle("Context Menu");
  menu.add(0, menu.FIRST, Menu.NONE,
           "Item 1").setIcon(R.drawable.menu_item);
  menu.add(0, menu.FIRST+1, Menu.NONE, "Item 2").setCheckable(true);
  menu.add(0, menu.FIRST+2, Menu.NONE, "Item 3").setShortcut('3', '3');
  SubMenu sub = menu.addSubMenu("Submenu");
  sub.add("Submenu Item");
}

@Override
public boolean onContextItemSelected(MenuItem item) {
  super.onContextItemSelected(item);

  [ ... Handle menu item selection ... ]

  return false;
}
```

Usually more than one way

- Menus via XML as well

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
  android:name="Context Menu">
  <item
    android:id="@+id/item01"
    android:icon="@drawable/menu_item"
    android:title="Item 1">
  </item>
  <item
    android:id="@+id/item02"
    android:checkable="true"
    android:title="Item 2">
  </item>
  <item
    android:id="@+id/item03"
    android:numericShortcut="3"
    android:alphabeticShortcut="3"
    android:title="Item 3">
  </item>
  <item
    android:id="@+id/item04"
    android:title="Submenu">
    <menu>
      <item
        android:id="@+id/item05"
        android:title="Submenu Item">
      </item>
    </menu>
  </item>
</menu>
```

Usually more than one way

- Menus via XML as well

```
public void onCreateContextMenu(ContextMenu menu, View v,
                                ContextMenu.ContextMenuInfo menuInfo)
{
  super.onCreateContextMenu(menu, v, menuInfo);
  MenuInflater inflater = getMenuInflater();
  inflater.inflate(R.menu.my_menu, menu);
  menu.setHeaderTitle("Context Menu");
}
```

Drawable resources

- XML definable (res/drawable folder)
- Types
  - ColorDrawable
  - ShapeDrawable
  - GradientDrawable (require gradient radius defined in pixels)
- Specify attributes with density independent pixels
- Run time scaled dyanamically

ColorDrawable

- Solid color

```
<color
xmlns:android="http://schemas.android.com/apk/res/android"
   android:color="#FF0000"
/>
```

ShapeDrawable

- Shape by defining dimensions, background, stroke/outline

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
  android:shape="rectangle">
    <solid
      android:color="#f0600000"/>
    <stroke
      android:width="10dp"
      android:color="#00FF00"/>
    <corners
      android:radius="15dp" />
    <padding
      android:left="10dp"
      android:top="10dp"
      android:right="10dp"
      android:bottom="10dp"
    />
</shape>
```

Oval, rectangle, or ring

GradientDrawable

- Smooth transitions 2-3 colors in linear, radial, or sweep pattern
  (sweep along outer edge of parent shape)

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
  android:shape="rectangle"
  android:useLevel="false">
  <gradient
    android:startColor="#ffffff"
    android:endColor="#ffffff"
    android:centerColor="#000000"
    android:useLevel="false"
    android:type="linear"
    android:angle="45"
  />
</shape>
```

GradientDrawable

```
<!-- Oval with Radial Gradient -->
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
  android:shape="oval"
  android:useLevel="false">
  <gradient
    android:type="radial"
    android:startColor="#ffffff"
    android:endColor="#ffffff"
    android:centerColor="#000000"
    android:useLevel="false"
    android:gradientRadius="300"
  />
</shape>

<!-- Ring with Sweep Gradient -->
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
  android:shape="ring"
  android:useLevel="false"
  android:innerRadiusRatio="3"
  android:thicknessRatio="8">
  <gradient
    android:startColor="#ffffff"
    android:endColor="#ffffff"
    android:centerColor="#000000"
    android:useLevel="false"
    android:type="sweep"
  />
</shape>
```

Gradient examples

- http://escomic.net/217

Composite Drawables

- Combine bitmaps, shapes, and colors, and other composite drawables

- Four types
  - Transformative
  - Layer
  - State List
  - Level List

Transformative Drawables

- ScaleDrawable
- RotateDrawable

```
<!-- Rotation Drawable Resource -->
<?xml version="1.0" encoding="utf-8"?>
<rotate xmlns:android="http://schemas.android.com/apk/res/android"
  android:drawable="@drawable/icon"
  android:fromDegrees="0"
  android:toDegrees="90"
  android:pivotX="50%"
  android:pivotY="50%"
/>

<!-- Scale Drawable Resource -->
<?xml version="1.0" encoding="utf-8"?>
<rotate xmlns:android="http://schemas.android.com/apk/res/android"
  android:drawable="@drawable/icon"
  android:scaleHeight="100%"
  android:scaleWidth="100%"
/>
```

Transformative Drawables

- In code...

```
ImageView rotatingImage =
(ImageView)findViewById(R.id.RotatingImageView);
ImageView scalingImage =
(ImageView)findViewById(R.id.ScalingImageView);

// Rotate the image 50% of the way to it's final orientation.
rotatingImage.setImageLevel(5000);

// Scale the image to 50% of it’s final size.
scalingImage.setImageLevel(5000);
```

Layer Drawables

- Stack transparent images

```
<?xml version="1.0" encoding="utf-8"?>
<layer-list
xmlns:android="http://schemas.android.com/apk/res/android">
  <item android:drawable="@drawable/bottomimage"/>
  <item android:drawable="@drawable/image2"/>
  <item android:drawable="@drawable/image3"/>
  <item android:drawable="@drawable/topimage"/>
</layer-list>
```

StateList Drawables

- Composite resource; specify a different drawable to display based on state of the View to which it is assigned
- **Most native Android Views use** (e.g., buttons, backgrounds listviews)

```
<selector xmlns:android="http://schemas.android.com/apk/res/android">
  <item android:state_window_focused="false"
        android:drawable="@drawable/widget_bg_normal"/>
  <item android:state_pressed="true"
        android:drawable="@drawable/widget_bg_pressed"/>
  <item android:state_focused="true"
        android:drawable="@drawable/widget_bg_selected"/>
  <item android:drawable="@drawable/widget_bg_normal"/>
</selector>
```

LevelList Drawables

- Overlay several specifying an integer index (useful for App Widgets)
- In code: imageView.setImageLevel(3)

```
<level-list xmlns:android="http://schemas.android.com/apk/res/android">
  <item android:maxLevel="0"  android:drawable="@drawable/im_0"/>
  <item android:maxLevel="1"  android:drawable="@drawable/im_1"/>
  <item android:maxLevel="2"  android:drawable="@drawable/im_2"/>
  <item android:maxLevel="4"  android:drawable="@drawable/im_4"/>
  <item android:maxLevel="6"  android:drawable="@drawable/im_6"/>
  <item android:maxLevel="8"  android:drawable="@drawable/im_8"/>
  <item android:maxLevel="10" android:drawable="@drawable/im_10"/>
</level-list>
```

Resolution and density

- UIs running on a variety of screen sizes and densities
- Use resource qualifiers to deal with this "floodgate" of devices
- Potential hazard!

(They also have small changes in hardware and other quirks to keep in mind!)

Resolution and density

- Screen size
  - Small (smaller than 3.2" screen)
  - Medium ("Typical" size)
  - Large
- Pixel density
  - ldpi (100-140 dpi)
  - mdpi (140-180 dpi)
  - hdpi (190-250 dpi)
  - nodpi (no scaling regardless screen density)

Resolution and density

- Aspect ratio
  - long (significantly wider in landscape)
  - notlong (typical ratio)

```
res/layout-small-long/ // Layouts for small, long screens.
res/layout-large/      // Layouts for large screens.
res/drawable-hdpi/     // Drawabless for high density screens.
```

To force screen size

- Use manifest

```
<supports-screens
  android:smallScreens="false"
  android:normalScreens="true"
  android:largeScreens="true"
  android:anyDensity="true"
/>
```

- anyDensity = false forces Android to scale

Tips

- Don't make assumptions about screens
- Avoid hard-coded pixel values
  - Use wrap_content, fill_parent, dp, and sp
- Avoid AbsoluteLayout class
- RelativeLayout for complex UIs is probably what you need

Tips 2

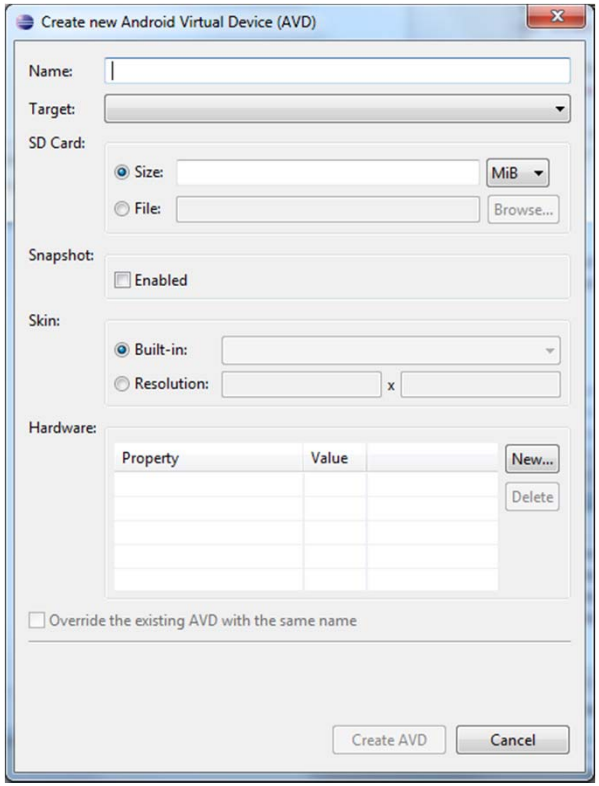
- Use Drawable resources rather than bitmaps
  - NinPatches
  - Shape Drawables
  - Gradient Drawables
  - Composite and transformable drawables
    - Rotate and Scale Drawables
    - LevelListDrawables
    - StateListDrawables

Asset that can't be scaled

- Image assess for each category to avoid aliasing/artifacts
  - res/drawable-ldpi
  - res/drawable-mdpi
  - res/drawable-hdpi
- Layouts for different phones
  - res/layout-small
  - res/layout-normal
  - res/layout-large

Test, test, test...

- Using AVD you can define arbitrary screen resolutions and pixel densities



Questions on reading

1. What sensor was used in Houston?
2. What sensor was used in UbiFit?
3. What does the butterfly represent?

Question on reading