Android: n-Puzzle Walkthrough

Tommy MacWilliam

Harvard University

February 22, 2012
Announcements

- Lecture videos: https://www.cs76.net/Lectures
- Section videos: https://www.cs76.net/Sections
- Walkthrough videos: https://www.cs76.net/Projects
Today

- dynamic GUIs
- bitmaps
- gameplay
- saving state
- menus
UI can also be created from pure Java

- necessary when you don’t know what your GUI will be until runtime

- make sure to import android.widget.*; etc.

- wildcard puts a small hit on the compiler
Dynamic GUIs

- **XML:**
  ```xml
  <LinearLayout>
    <Button />
  </LinearLayout>
  ```

- **Java:**
  ```java
  LinearLayout layout = new LinearLayout(this);
  Button button = new Button(this);
  layout.addView(button);
  ```
IDS

- `R.id.<id>` no longer applies
  - we need to maintain IDs now
- `view.setId(int id);`
  - set the numerical ID of a view
  - `id`: unique numerical integer
Event Handling

- `setOnClickListener(View view);`
  - attaches an event handler that will fire on a tap
  - `view`: the object that was tapped
  - `view.getId()` will return the equivalent of `R.id.<id>`
view.getParent();

- get the parent element of a view
- so we can get the layout containing this element

layout.getChildCount();

- get the number of children of a layout

layout.getChildAt(int index)

- index: numerical index of element
Subviews

- `layout.addView(View view)`
  - add a subview to a view
  - `view`: view to add

- `layout.removeView(View view)`
  - remove a subview from a view
  - `view`: view to remove

- `layout.removeViewAt(int index)`
  - remove a subview from a view
  - `index`: index of child to remove
Dynamic GUIs

- AndroidWalkthroughApp1
ListView

➤ displays a scrollable list of data
➤ template for each item fined in res/layouts
➤ every ListActivity has a ListView
ListView displays data, ArrayAdapter holds data

- a = new ArrayAdapter(Context c, int resource, List objects)
- getListView().setListAdapter(a)
ListViews

- AndroidWalkthroughApp6
Breaking up an image

➤ Google is so helpful
➤ http://lmgtfy.com?q=android+break+up+image
must be able to create a background from:
  - image smaller than n-Puzzle board
  - image larger than n-Puzzle board
  - image with portrait orientation
  - image with landscape orientation
  - image with square orientation
**Bitmaps**

- **Bitmap** class works with images in `res/drawable`
- `BitmapFactory.decodeResource(Resources r, int id)`
  - `r`: `getResources()`;
  - `id`: ID of drawable (i.e. `R.drawable.<image>`)

---

**Android:**
- n-Puzzle Walkthrough
- Dynamic GUIs
- ListViews
- Bitmaps
- Gameplay
- Saving State
- Menus

**Tommy MacWilliam**
Resizing Bitmaps

- images we load into our app might not fit on the screen
- `Bitmap.createScaledBitmap(Bitmap bitmap, int width, int height, boolean filter);
- `bitmap`: image to scale
- `width`: desired width of image
- `height`: desired height of image
Maintaining Aspect Ratio

- simply modifying the width and height to fit the device will distort the image
- need to modify width/height at the same rate
  - equivalent of resizing a photo by dragging a corner, not an edge
Cropping Bitmaps

- image also might not be square
- `Bitmap.createBitmap(Bitmap bitmap, int x, int y, int width, int height);
  - `bitmap`: image to crop
  - `x, y`: origin for cropped image
  - `width, height`: dimensions for cropped image, in pixels
Getting Screen Size

- `getResources().getDisplayMetrics()` returns an object containing properties corresponding to screen size
  - `getResources().getDisplayMetrics() .pixelHeight`
  - `getResources().getDisplayMetrics() .pixelWidth`
Error-Checking

- **Bitmap objects can be pretty memory intensive**
- **surround code in** `try {} catch {}` **blocks to avoid OutOfMemoryError**
- `bitmap.recycle()` **when an image is no longer needed**
  - garbage collects the `Bitmap`, freeing up memory
Bitmaps

▶ AndroidWalkthroughApp3
Tags

- each `View` can have an associated object
  - use object to store any data you want in view
  ```java
  view.setTag(Object o)
  view.getTag()
  ```
n-Puzzle

▶ also called Game of Fifteen or Fifteen Puzzle
Representing the Board

- board is an $n \times n$ grid
  - 1 blank tile
  - $n^2 - 1$ image tiles
- access tiles using $(x, y)$?
- access tiles using (row, column)?
Representing a Tile

- think about good software design!
- board tile has $x$-position, $y$-position, etc.
  - tiles are actually images, but Game of Fifteen uses numbers
- encapsulate that information!
Representing the Blank

► blank tile is a special case
  ▶ what’s the best way to represent it?
► blank tile is most important
  ▶ is repeatedly searching for the blank tile efficient?
Layout

- layout describes how elements are displayed
  - `LinearLayout`: children aligned in single direction
  - `TableLayout`: children placed into defined rows and columns
  - `RelativeLayout`: children specify relation to parent
TableLayout

- similar to HTML tables
- `<TableRow>` defines rows
- specify same number of elements per `<TableRows>`
TableLayout

Android: n-Puzzle Walkthrough
Tommy MacWilliam

Dynamic GUIs
ListViews
Bitmaps
Gameplay
Saving State
Menus

★ AndroidWalkthroughApp5
GridView

- ViewGroup, just like a ListView
- has an associated adapter to display data
  - just like a ListView
- events triggered with setOnItemClickListener
  - omg ListView
Swapping Tiles

- tile should be swapped iff next to empty tile
- be sure to check:
  - two non-blank tiles cannot be swapped
  - tile is never swapped off board
  - swapping occurs visually and internally
Swapping Tiles

- find empty tile on board
- find tapped tile on board
- swap if two tiles are adjacent
  - checking up, down, left, and right separately doesn’t seem DRY...
Swapping Tiles

- **ImageView** displays an image
  - `setImageBitmap()`, `setImageDrawable`
- image displayed within **ImageView** can change
  - tiles can also move
#Winning

- game is won when image is reconstructed
  - in Game of Fifteen, equivalent to increasing order
- if any tile is not in order, then game is not won
#Winning

- show congratulatory message on win
- remember Intents are used to swap between activities
Saving State

▶ need to save:
  ▶ the state of the board
  ▶ number of moves made
  ▶ difficulty
Saving State

- `onPause()` called whenever `Activity` is removed from foreground state
  - seems like a good place to save state
- `onCreate()` called when `Activity` is first created
  - seems like a good place for `View` setup, can retrieve saved state here
- `onResume()` called immediately before `Activity` is put into foreground state
  - state retrieval also possible here
- be sure to call `super.onPause()`, `super.onResume()`, and `super.onCreate()`!
Retrieving State

- **SharedPreferences** allows us to retrieve saved state
  - preferences for Activity retrieved via
    `getPreferences(int mode);`
  - get saved key/value pairs via
    `sharedPreferences.get<type>(String name, int value)`
    - where `<type>` can be `Boolean`, `String`, `Int`, etc.
    - `name`: key corresponding to desired value
    - `value`: value to return if name is not found
Saving State

- `SharedPreferences.Editor` used to save state
  - created via `getPreferences(int mode).edit()`
- save key/value pairs via
  ```java
  sharedPreferencesEditor.put<type>(String name, int value)
  ```
  - where `<type>` can be `Boolean`, `String`, `Int`, etc.
  - `name`: key corresponding to desired value
  - `value`: value to associate with `name`
- `sharedPreferencesEditor.commit()` writes changes
Saving/Retrieving State

▶ AndroidWalkthroughApp4
Menus

- menu should appear when we press the menu button
- menus go in res/menu
- <menu> has multiple <item> children
Menus

- `onCreateOptionsMenu(Menu menu)` fired when user requests to open menu
  - `MenuInflater` used to display menu
- `onOptionsItemSelected(MenuItem item)` fired when user selects a menu item
  - `item.getItemId()` returns an ID that can be used with `R.id.<id>`
Menus

Android Walkthrough App 2