iOS: Evil Hangman Walkthrough

Tommy MacWilliam

tmacwilliam@cs.harvard.edu

April 4, 2012

Today

- Evil Hangman
- Setup
- Getting Input
- Property Lists
- Equivalence Classes
- Settings

Evil Hangman

- it's evil.

Evil Hangman

- goal: dodge user's guess as best as possible
- strategy: be able to switch among the most words
  - maximize ability to cheat

Evil Hangman

- `_ _ _`
- words: `OWN, PUN, LOL, NVM, NOT, WON, NEW`

Evil Hangman

- – – –
- words: `OWN, PUN, LOL, NVM, NOT, WON, NEW`
- guess: `N`
    - – – –: `LOL`
    - N – –: `NVM, NOT, NEW`
    - – – N: `OWN, PUN, WON`

Evil Hangman

- `_ _ N`
- words: `OWN, WON, PUN`
- guess: `P`
  - `_ _ N`: `OWN, WON`
  - `P _ N`: `PUN`

Evil Hangman

- `_ _ N`
- words: `OWN, WON, PUN`
- guess: `O`
  - `_ _ N: PUN`
  - `O _ N: OWN`
  - `_ O N: WON`

Evil Hangman

- `W – N`
- words: `WON`
- guess: `O`
  - you win!

Evil Hangman

- optimize at each step to find best solution
- at each guess, leave yourself maximum number of words remaining

Setup

- Utility Application
  - contains two controllers: `MainViewController` and `FlipsideViewController`
  - flipside often used for settings, etc.
  - e.g., Weather app

Delegate

- define object to handle actions rather than handling them yourself
- `delegate` object implements a protocol
  - guarantees definitions for methods exist

Protocol

```
@protocol SomeProtocol
- (void)something;
- (int)calculateSomething:(int);
@end
```

Setup

- `AnnotatedUtilityApp`

Getting Input

- must come from a `UITextField`
- `UITextFieldDelegate` provides greater access
- probably don't want to show the text field

UITextFieldDelegate

- `textFieldShouldReturn`: “done” button pressed
- `textFieldShouldBeginEditing`: user about to edit text
- `textFieldShouldEndEditing`: text field about to lose focus

Getting Input

- `HiddenUITextFieldExample`

plists

- key/value pairs, frequently used to store settings
- stored as XML, editable in raw form or using XCode's plist editor

Syntax

```
array
    <string>David</string>
    <string>Rob</string>
</array>
```

Loading plist Files

- `NSDictionary` can `initWithContentsOfFile`
  - can also `writeToFile`
- `NSBundle` accesses filesystem via `pathForResource:ofType:`

Using plists

- `PropertyListExample`

Equivalence Classes

- Wikipedia says: $[a] = \{x \in X | x \sim a\}$
- clear? okay we’re done, have a nice night

Equivalence Classes

- no, there's no `NSEquivalenceClass` either

Equivalence Classes

- define a set of words sharing a given letter at a location
- order matters!
  - `– – N, N – –` represent different equivalence classes
  - don't forget the set of words not containing the guessed letter

Equivalence Classes

- group words into equivalence classes based on user input
- how do we define the collection of words in an equivalence class?
  - `NSMutableDictionary`
  - `NSMutableArray`
  - `NSMutableSet`

Equivalence Classes

```
for each word in set:
    determine equivalence class for word;
    add word to equivalence class;
determine largest equivalence class;
remove all words in complement of
largest equivalence class;
update UI;
```

Equivalence Classes

- each equivalence class contains a set of words
- also need to keep track of all equivalence classes
  - make sure to pick the largest class
  - ties must be broken psuedorandomly

Time Consumption

- iterating through long lists is SLOW
- indexing into arrays/dictionaries is FAST

Space Consumption

- `words.plist` is pretty big!
- keep data structures as small as possible
- don’t bother keeping things in memory that don’t need to be

Design

- “We should forget about small efficiencies, say about 97% of the time: premature optimization is the root of all evil. Yet we should not pass up our opportunities in that critical 3%.” – Donald Knuth

NSUserDefaults

- `NSUserDefaults` capable of storing persistent key/value pairs
    - without the hassle of implementing a database

NSUserDefaults

- `[NSUserDefaults standardUserDefaults]` gets defaults associated with the app
- `setObject:forKey:` saves a key/value pair into defaults
- `objectForKey:` retrieves a value
- `removeObjectForKey:` removes an item
- `synchronize` commits changes

NSUserDefaults

- can read/write anything saveable in a plist
  - `NSData`, `NSString`, `NSNumber`, `NSDate`, `NSArray`, or `NSDictionary`
- convenience methods
  - `arrayForKey`, `dictionaryForKey`, `integerForKey`, etc.
  - `setBool`, `setInteger`, etc.

NSUserDefaults

- `NSUserDefaultsExample`