Janet Weber ***Fall 2015***

iOS/Swift Project Recipe/Proposal

Project: Simon like game (pattern recognition game of the late '70s and early '80's using 4 colors each associated with a sound).

General Description:
For this variation of the original Simon:

1. Begin with 4 characters (no sound, no color) and call it Level 1.
2. Game displays 1 of the characters for .5 seconds, hides, it, then waits for user to repeat that 1 character.
3. Upon successful repeat by user, the game then displays first character again for .5 seconds, hides it, then displays a second character, hides it and waits for user to repeat the sequence.
4. And so on, adding additional characters each iteration until the user is unsuccessful at repeating the sequence.
5. After a certain number (10?) of successful repetitions of 12 (number can be changed) characters or more with a set of 4 characters, increase level -> add an additional character to the set. So if there are 4 characters in the set for level 1, there are 5 characters in the set for level 2.

Variables & Constants

Constants:

charset4 = constant array of 4 characters to be used at first level
(i.e. {a, b, c, d} or {1, 2, 3, 4} or {# $ & *})
charset5 = charset4 + one additional character
charset6-14 = additional character sets each with one more character than last.

Variables:

genSeq: String - characters generated randomly one at a time and appended.
repSeq: String - characters entered by user trying to repeat the generated sequence.
rndm : integer - random number generated.
success: bool – result of comparison of generated sequence to repeated sequence
successCnt: Int – keeps count of number of successes at current level.

Directions:

Initialize success to true to start game.

```
success = true
successCnt = 0;

while (success)
    rdnm = (generate a random number for a range of 0-3 or 1-4).
    Use rndm as index into charset4 array – Append to the generated sequence array.
    genSeq+=[charset4[rndm]]

    Display the generated sequence one char at time, hiding each character after .5 seconds.
    for (i= 0; i<genSeq.count; i++) {
        print (genSeq[i])
        wait .5 seconds
        hide (genSeq[i])
        }
```

Now wait for user to repeat sequence, inserting it into repSeq array.
repSeq = read from console

Compare repSeq to the genSeq
If (repSeq = genSeq) then
success = true
successCnt +=1
Else success = false
Back to top of loop
} *end of while loop*

Enhancements to this basic implementation:

- Need to implement movement to different levels based upon successCnt.
- Possibly implement use of different character sets.
- Suggestions?

Concerns:

- User input – haven't covered that yet.
  - User required to hit "enter"?
  - Or just grab one char at time from input?
- How to display each sequence, both the generated sequence and the repeated sequence, one char at a time (timing and hiding – backspace?).
- Color? Maybe beyond the scope of this project.
- Sound? Maybe beyond the scope of this project.
- Does this idea use enough of the concepts learned in this course?
- Not sure this is robust enough for 5-7 weeks? Thoughts? I do have another idea.

Thoughts:

Maybe adding more characters makes it easier, not harder. Maybe it's harder to remember a sequence of only 3 or 2 characters in the set.

iOS App Development I : MySimon
Final Project (a continuation from last semester)
Janet Weber

SCRUM 3/14/16 : Get organized

What features do the apps already out there have?

- Party mode Multi player (one device – players must be together – up to 4 players)
- Multi player (remote devices)
- Sound selection (classic plus ?? moos, farts, …)
- "Button" Selection (classic plus ?? gems, cows, droids…)
- Options for other than 4 colors (more or less).
- Different background options.

- no repeat mode – only show added visual (instead of repeating entire pattern first)
- blind mode – no colors highlighted, only sounds.
- Crazy mode – colored buttons can move, increase or decrease in number, swap positions and more!
- Speed mode – you need to go faster and faster the longer the pattern goes.

What Views (screens) will I need?

- Basic SIMPLE instructions
- Settings
- Score – maybe not a separate view
- Game play view

What are my view controller options in Xcode?

- Simple buttons one view (just hide/display appropriately – can get messy in storyboard)
- Segmented Control (would work like simple buttons)
- Simple buttons multiple views (like chapter 6 View Switcher)
- Tab controller ?? Probably not the most elegant solution, but maybe.
- Navigation controller (allow for most flexibility in adding features).

Unlikely choices

- Picker
- Table view

Next week – Classic Simon

- Decide on controller and Implement the skeleton to include:
  - Instructions
  - settings
  - Main game screen display (working buttons, highlights, correct match count tally) – no real functionality – just implement alerts to show proper wiring.
  - Detailed score
- Launch screen

- Possibly import code from last semester to play (generate the random number sequence, etc). Not all of it is usable, but much is.
- Explore Sound
- Always be on lookout for app icon.

Toolbar button item images/backgrounds
View switch transitions

Basic framework done (will change transitions and make fancy AFTER functionality is there)
Begin porting code from last semester.
Data, View, Controller ??

Sprint/Scrum (Monday, April 11)

✳Discovered that my view switcher had several logic errors. The book assignment implementation used an "If" (both for creating a view controller if necessary and also switching the view), but with more than 2 views to switch between, use a case statement (sender.tag) and "ifs". I tried doing it all with just "ifs" and wasn't working.

✳Implemented a recursive function to highlight a sequence of buttons with timing.
- ➡ Might be a better way to handle timing - suggestions?

✳Struggled with view controller class and game class - going back and forth. Going from view controller class to game class wasn't the problem, but when I'd try to call a function in my view controller class from the game class, my instance variables weren't in scope AND my storyboard buttons became nil. Spinning wheels. For some reason, I was stuck on using this big chunk of code from last semester. After spinning my wheels, decided all that was needed was to break that big chunk up into smaller chunks. Basically decided the view controller should handle only anything in the view and my game class should handle the control (not much data involved so that's included with control for now).

✳Ended up using a couple of global variables (declared/initialized above my game class in same file). The user response array of button tags and a boolean flag for detecting if the round is finished. Not sure if there is a better way. Maybe make a property of game class?

✳Expanded game class definition to include a couple more properties.
- ➡Button taps (counted so game knows when to compare user response to sequence). This means compare doesn't occur until the user has tapped as many buttons as are in in the match sequence thus far. A wrong button press won't be flagged until the correct number of taps have occurred. This needs to be changed so that an incorrect button tap flags the end of a round.
- ➡indexMatched - keeps track of where the user is (that is what's been matched correctly)

Spent most of time getting buttons to highlight and go thru the pattern - adding one more each time. Then this morning was adding remaining logic so its kind of messy.

Needs to be cleaned up - but it is working - a very rudimentary implementation.

Sprint/Scrum (Wednesday, April 20)

✳Game is working - functional, but no pretty notifications (only print statements for now).

- ➡Implemented timer (3 Sec) to end round if user doesn't press a button for specified amount of time. This timer is started with each round (for first element), then stopped with a button press and after processing that button press, the timer is started again (setting limit for next button press). If the timer ever fires, the round is over (3 seconds of inactivity).
- ➡Also, the round will end immediately after an incorrect button is pressed (instead of waiting for user to complete sequence so far before indicating incorrect button press - last week).
- ➡Must play 3 rounds before possible to move to next level. Level 1 starts out at 4 matches (for debugging purposes - will be 10 for real). If total number of matches for previous 3 rounds meets the threshold, then the level is bumped up one and the level threshold is increased by 2. Then continue.

✳Moved code, most of which was in GamePlayViewController, to SimonGame.swift. Only left code dealing specifically with the view (i.e. highlighting buttons or handling button presses) in the GamePlayViewController.

✳Game has 3 view controller classes (GamePlayVC, InstructionsVC and SettingsVC) and 1 game class. Was able to "link" GamePlayVC to game class by making view controller a property of the game class, then after instantiating a gameClass object in the GamePlayVC, set the object's view controller property to self.

Cleaned up and started commenting/leaving notes to myself.

Next week:
Implement GUI for all of the print statements in my code.

Sprint/Scrum (Wednesday, April 27)

- ✳ Created the game icon
- ✳ Implemented "Again" Button - searched for image to match theme. After a round (playing one sequence), user has to hit this button to continue this game (trying to increase levels, etc)
- ✳ Discovered a few bugs:
  - When tookToLong timer fired, game wasn't reporting matches that round or finishing up the round properly.
  - Modified roundFinished() method to have an integer argument indicating the reason the round is finished. 1 -> incorrect button press, 2 -> took too long, 3 -> matched entire sequence.
- ✳ Added Sound :
  - required searching for sounds (takes a while - I can see this time shrinking as you happen upon sites that become your favorites, etc). Had luck with a site called FreeSound. They had packages of sounds - the one I implemented is from a glocknspiel. There were several others (piano chords, acoustic guitar) that were interesting.
  - playTone() function and calls to it
- ✳ Searched for GUI elements to be used instead of print statements. Also took quite a while. Show these.

Next Week:

- ✳ GUI / labels instead of print statements in code (game statistics, etc).
- ✳ Implement Settings (options for sounds and game modes - get some in Storyboard and hooked up. But they may all point to a "default" for now. But it should be easy to add in later.

- Implement Instructions for default game mode

Future:
data permanence: Simple game so maybe not necessary. But it might be nice to have game pick up where you left off.
Gestures - ?? not sure I need more than taps. Could think about allowing some swipe gesture (like swipe texting - so user doesn't have to lift finger).

Final Sprint/Scrum (Wednesday, May 4)

- implemented reset() method for mySimon class for new game (basically the same as the init() method of the class (but can't run init() because that creates a new instance - this is the same instance).

- Created series of 8 labels in bottom view (4 static titles and 4 dynamic/updated with game stats for level, matches this round, matches this level, and the level threshold).
- Implemented updateLabel() method that is used in place of my print statements to display the game stats. It is also used to update the 2 message labels in the top view. The first message label is used to inform user of the reason the round is over (3 reasons: no match, took too long, or matched entire sequence). The second message label is used every 3 rounds to inform user whether they are repeating the level or moving on to the next level and each round to announce either New Game or Next Round. Then decided I didn't need the method - just as easy to change the label text as needed within the code.

- implemented sound for "mistake" including appropriate attribution.
- implemented sound for "winning" including appropriate attribution.
- Added these sounds to my playTone method and they played twice (although the other game tones only played once). Researched and ended up changing the .numerOfLoops property of the audioPlayer to 0 (because 1 means it will loop 1 time or play twice - https://www.safaribooksonline.com/library/view/ios-swift-game/9781491920794/ch04.html)

- Made sure the end of game was detected (after achieving level 10) and no more rounds will fire until "New game" button pressed. Plays winning sound for this as well as if entire sequence is matched.

- If continue button is hit first - nothing happens (until new game is initiated)

- Implemented smiley images at various appropriate times (correct sequence match, incorrect sequence match, took too long. entire sequence match, level 10 passed)

- Modified the animated transitions when the view switches between play mode, instructions and settings.

- Added launch screen

- Instructions: tried labels and a pdf, but ended up using an html file with a web view. Only have instructions for the classic game.

- Settings - looks nice but doesn't do anything (classic mode, blind mode, speed mode)

Future:

data permanence: Simple game so maybe not necessary. But it might be nice to have game pick up where you left off. Remember Settings
implement other game modes, sounds (put sounds into an array - this would make it easier)
move some code from button press into game class
