Assessment: due Thursday, February 19

Hangman game

Create a **Hangman** game that lets the user take up to **7 wrong guesses** on a word (each one representing head, each arm, body, each leg).

Game requirements:
1. You must have at least two scenes/view controllers.

2. One of the scenes represents your game dashboard, where the user is informed of how many guesses they have left, a graphical representation of wrong guesses, and letters they have guessed correctly.

3. The user must guess letters in a new scene, and pass the guess back to the main scene through delegation (every time the user takes a guess, they select the “Guess” button from the first/main scene, that takes them to that second scene). The second scene should have at least a text field and button that lets the user take the guess and select the button once they have made their letter guess.
*Recommendation: pass wrong guesses to a **wrongGuesses** array.*

Hint: optional route to check if a string contains another string:

```
// note: using lowercaseString so that all user inputs are not case sensitive
if myWord.lowercaseString.rangeOfString("e") != nil {
    // Enter code that does something if the word contains the letter
}
```

4. Display the incomplete word with _ replacing the not yet guessed letters.
Hint: add the provided for loop to a function that returns the current guessed word (as string):

```
var correctGuesses = ["e", "a"]
var incompleteWord = ""

for letter in myWord {
    if contains(correctGuesses, String(letter).lowercaseString) {
        incompleteWord += String(letter)
        incompleteWord += " "
    }else {
```

```
            incompleteWord += "_ "
        }
    }
```

5. Programatically draw 7 grey squares. For every wrong guess, color a square red.
Optional: have one square with a different gradient of red. The closer to bright red the color, the closer the user is to running out of lives. If going this route you must provide a legend.
Preferred route (bonus): programatically draw the hangman body as the user adds wrong guesses.

6. Use springs and struts to make sure that the programatically drawn shapes stick to one of the corners.
Bonus: lay out all the elements in your scenes with springs and struts.

7. When the user is out of guesses, present them with a game over message. Listen for a game to be over through a notification. This can either be in the form of two notifications (for **gameOverPlayerWin** or gameOverOutOfGuesses) or one notification (**gameOver**) where you calculate the result after.

It is recommended that you build the game rules in a hangman/game class.