Threelow

Learning Objectives

Today we're going to build a simple Dice game using objects, and some basic OOP structure.

Introduction

The game we're going to make is called 'Threelow'. It's similar to Yahtzee, but with much simpler scoring. To play, a player rolls five dice. The goal is to get the lowest score possible. Each roll they must choose to hold at least one of the dice, but they can choose to hold as many as five dice. If they have un-held dice, they re-roll until all five dice are held.

Threes are worth 0 points (hence the name of the game), every other roll is worth it's face value. Whoever has the lowest score after everyone has had a complete turn wins the game (note: gambling is illegal).

Setup & Tools

- Create a new OS X command line project named `Threelow`

- You should use the following unicode symbols, or roman numerals, to represent the dice values: ⚀ ⚁ ⚂ ⚃ ⚄ ⚅

Source Control: Remember to create and push your project to github. You should be committing at *least* once per user story.

MVP User Stories

Below are some user stories that outline the functionality of this app. We've also outlined a list of some of the tasks you'll need to do for each story.

As a user, I want to play Threelow but I don't have physical dice.

- Create a `Dice` class.
- It should have a property to store its current value, and a method to randomize that value.
- Make five instances of the `Dice` class, randomize their values and print them.

As a user, I want to be able to roll the dice.

- When a user inputs the word `roll`, randomize the values and print them.

- Place this code inside a `while` loop, so users can re-roll as often as needed.

As a user, I want to be able to "hold" one or more of the dice and re-roll the rest.

Step 1. Refactor

- If you are not already doing so, now is a good time to refactor your code to store your `Dice` instances in an array that is created before your `while` loop.
- Add a collection object outside your `while` loop to store `Dice` that have been "held". You should consider which of the Foundation collection types (`NSMutableArray`, `NSMutableDictionary`, or `NSMutableSet`) is best for this task.
- Now that you have two collections that seem logically linked (your array of dice, and your collection of held dice), it's time to build a controller object that encapsulates them. Call this your `GameController`, and make the two collection objects properties of it.
- You now have a **data model** and a **controller**.

Step 2. Add the "hold" feature

- Add a method to your controller called `holdDie:`, that "holds" the given dice (by number).
- After each roll, allow the user to input dice indexes to hold them.
- You will need to visually indicate which dice have been "held". Consider bracketing [] the held values when you print them out.

As a user, I want to be able to un-"hold" a die if I hold one by mistake.

- use your `holdDie:` method to toggle dice between the "held" and "un-held" state in your data model.

As a user, I want to reset all "held" dice quickly.

- Add a `resetDice:` method that changes all the buttons to the un-held state in the data model.
- When the user inputs `reset`, call the `resetDice` method.

As a user, I want to see my score as I play.

- Add method to your controller that calculates and returns the current score by adding up all the held dice values.
- Call that method whenever you print the values of all the dice.
- This is a good time to refactor so you are calling a single method that prints all the dice values, held state, and the current score.

Final product

You should have a playable command-line game of Threelow. It should be well encapsulated, and have a `GameController` class, that contains your data model (two collections of `Dice` objects), and a number of methods that effect that model. You should have a main game loop, in `main.m`, that takes user input, runs methods on the game controller, and displays the state of the game for the user.

Stretch Goals

- Display the number of rolls taken since last reset.
- Don't let the player roll until they've selected at least one die each turn.
- Don't let the player roll more than 5 times without resetting.
- Display a "score to beat", so you can track what the lowest score so far has been. Add a `new game` command to allow users to reset this when every player has had a turn.
- Make a secret trigger that allows you to always win. Perhaps `rolll` or `roIl`?