Twenty-One

Due 11:59 PM, Thursday, April 25, 2018

1. Introduction

After the movie Boss Baby comes out, the puppies of the world are finally at wit's end with the attention the frickin' babies are getting! They decide to challenge those tiny humans to a match of the classic card game, twenty-one, using hugs as betting tokens to decide once and for all who is the cutest. They enlist your help in setting up the game. But little do the babies know, those sneaky puppies have a secret weapon up their sleeve, which they also ask you to help complete.

The Twenty-one assignment has been scaffolded to let you focus on applying the MATLAB skills you've learned in lecture and in section to construct an entire game of twenty-one. Don't be worried if you don't understand some of the stencil code, since we've explained everything you need in this handout. We've also provided you a test suite to test your code as you go and a simulator to see if the puppies' secret weapon has worked (both are explained below).

2. Assignment

In this assignment, you will code a game of twenty-one in Matlab. Each round of a game of twenty-one consists of a player either drawing a card or ending their turn with the objective of having the sum of their hand be as close to 21 as possible without exceeding 21. The will be twofold. In part one, you will be implementing several classes necessary to set up a game of 21. In part two, you will implement two AI players and compare their performance in the game.

3. Installation

For each project, there may be support files that you will need to complete for the assignment. These can be copied to your home directory by using the `cs4_install` command in a CIT Terminal window. For this project, type the command:

```
cs4_install twentyone
```

There should now be a `twentyone` folder within your `projects` directory. Using Terminal, you can move into the folders with the cd command:

```
cd ~/course/cs0040/projects/twentyone
```

If you are working on a department machine, you can run `cs4_matlab` from this folder to open all files in that folder in MATLAB R2018a

4. Instructions

The twentyone stencil code consists of the following files. You need only **edit** the bolded files

Class Files	Functions	Testing Scripts

`Player.m`
`AIOptimistPlayer.m`
`AIPessimistPlayer.m`
`AIPlayerXray.m`
`Card.m`
`Game.m`
`Hand.m`
`Shoe.m`
`Shuffler.m`
`TwentyOneHand``.m`
	`compare_results.m`
	`run_player.m`
	`make_hand.m`
	`play_round.m`
	`sample_mean.m`
		`twentyone_tests.m`
		`simulate.m`

As you may recall, MATLAB doesn't allow multiple public functions in one file; this same restriction also applies to Object Class definitions. Since this is a lot of files to wade around in, you can use the MATLAB **"Find Files"** button on the editor tab to allow you to easily navigate to each TODO item that your need to complete in order to get your twenty-one system up and running.

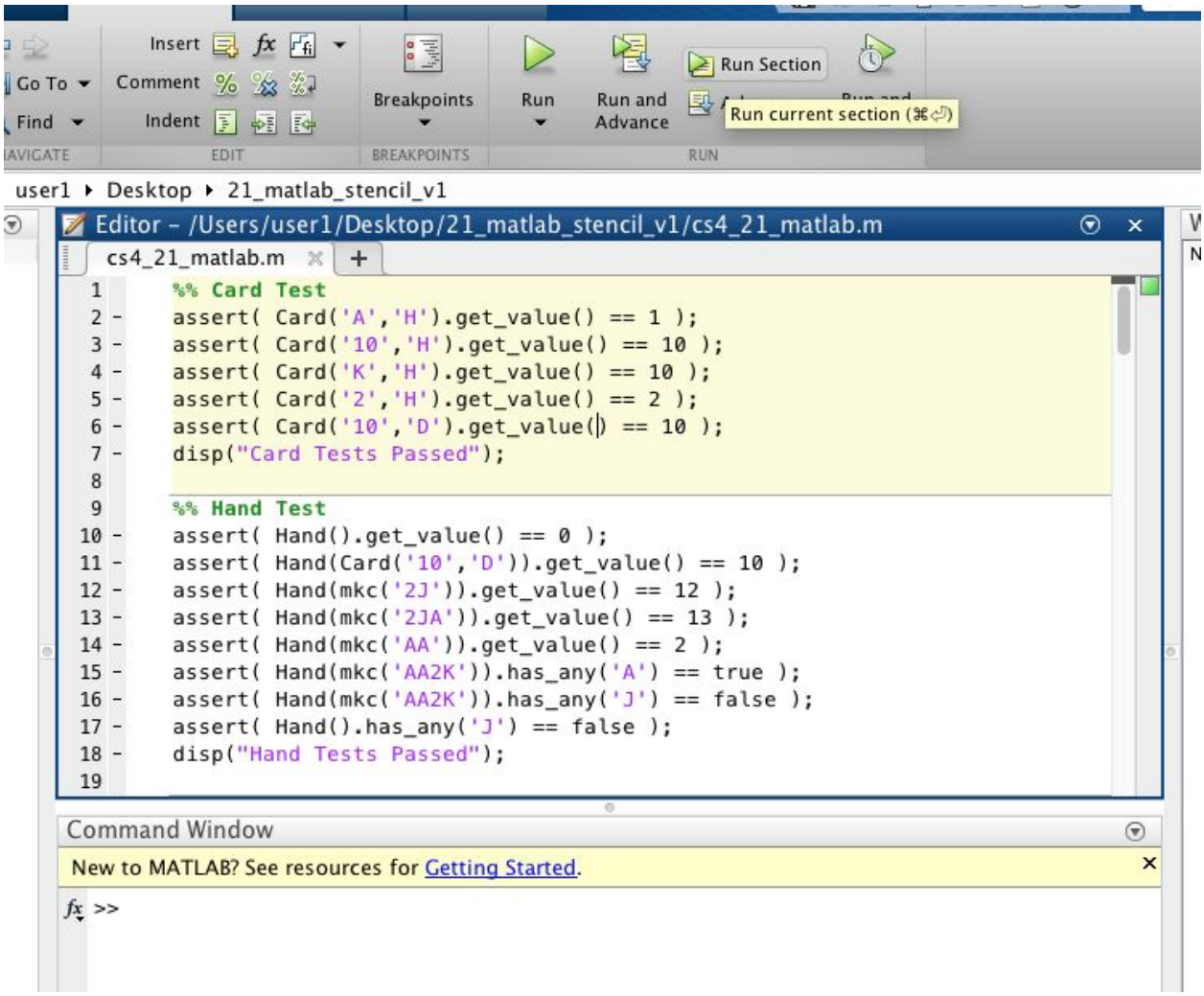
5. Part One: Game Setup

In this part of the project, you will be implementing several classes that are necessary for the 21 game to work. These classes can be found in `Card.m`, `Hand.m`, `TwentyOneHand.m`, and `Shoe.m`. We've provided the stencil code for each of these classes, along with TODO comments to detail what still needs to be done.

5.1 Testing

The **`twentyone_tests.m`** script contains a full set of tests for the MATLAB 21 system. As you implement your TODO items, you can open the **`twentyone_tests.m`** script and use the **"Run Section"** button in the editor to run just the tests associated with each section of code you are working on (which should be clearly labelled in the test suite). If you get an error when running any given section of the test suite, that means you haven't reached a working implementation of the corresponding class.

Run Section Example



5.2 Files

Card.m

For `Card.m`, you must compute the value of each card in the function `get_value`. The passed in parameter, `obj`, will be an array of card objects, and you should be returning an array of the same size where each value corresponds to the card at the same index in the card array. Remember that an Ace is worth 1, face cards are worth 10, and number cards are worth the same number value.

*Note: The suit of a card and the name of face cards are both represented by the first letter of its name. For example, a king of hearts should have the name “K” and the suit “H”.

Hand.m

For `Hand.m`, you must first initialize the cards in the hand (which should be a nx1 sized array). Then you must add new cards to the hand and calculate the total value of all cards in the hand. Finally, you must check if a given card is in the hand in the function `has_any`.

TwentyOneHand.m

`TwentyOneHand.m` represents a hand of cards in a 21 game. In the `get_value` function, you must first calculate the value of the hand of cards – the difficulty here is that an Ace can be determined to have a value of 1 or 11 depending on the situation, so you cannot just use the value returned from the Hand object's `get_value` function. For example, you may want an Ace to hold a value of 1 if you have the Ace and two face cards, and you may want an Ace to hold a value of 11 if you have the Ace and just one face card. A hand is soft if it uses an Ace as 11, and you must determine if the hand is soft in the function `soft_value`.

Shoe.m

`Shoe.m` represents a card shoe containing deck(s) of cards that can contain either a finite number of cards or infinite cards. You must first initialize the deck(s) of cards contained in the shoe and then implement the shuffling mechanism in shuffle. Then, you must perform the distribution of cards (refer to the `distribute` function) for both the case where there are finite cards (just return the next n cards in obj.cards) and infinite cards (sample with replacement from obj.cards). The TODO comments here should be extremely helpful. Remember the return value should be stored in the variable to which the function is assigned.
*Note: `next_card` denotes the index of the next card, not the actual card itself.

6. Part Two: AI

After you've finished implementing the game object classes, you'll be filling out `AIOptimist.m` and `AIPlayerXray.m` to create new players with different strategies for playing the game. We've given you `AIPessimistPlayer.m` as a point of comparison. The pessimistic player always assumed that the next hit will put them over 21, so they always stand.

6.1 AIOptimistPlayer.m

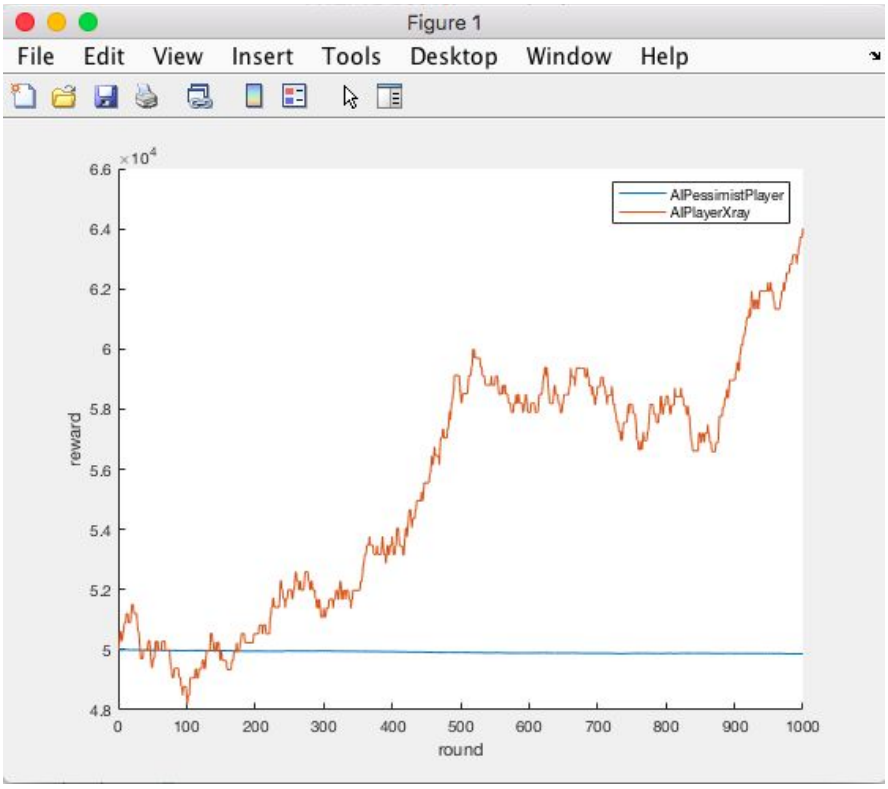
The optimistic player assumes that the next hit won't put them over 21 so they always hit unless their own hand value is 10 above the shuffler's (or dealer's) visible hand value. If 10 + the shuffler's visible hand value is over 21, then they would also stand. Remember, `classdef AIOptimistPlayer < Player` means that the `AIOptimistPlayer` inherits from the

`Player` class, so it might be useful to look over the `Player` class (in `Player.m`) to see what's going on. Like in part one of the project, you'll find tests in `twentyone_tests.m` to test your implementation.

6.2 AIPlayerXray.m

The puppies' secret weapon is that they've convinced Superman to share with them the secret of x-vision. However, they're not entirely sure how to use it except to see the shuffler's hole (hidden) card and/or to peak ahead in the shoe's cards, so they've asked you to help implement a strategy to beat the babies' pessimistic strategy which you will place in the function `bet` to determine how many hugs to bet. We've already added code to find the `hole_card` value. It's your job to determine how much to bet from this, which you should store in the return value `b`. This is open-ended and there is no right answer, so feel free to get creative! However, make sure you document your approach in the README, which will be factored into your grade. You don't need to go overboard but your explanation should be detailed enough that a TA could replicate your function.

You can run `simulate.m` to see how your strategy stacks up to that of `AIPessimistPlayer.m`. After waiting for the program to finish running, you should see a graph that looks something like this (obviously, it doesn't have to be exact!):



The reward accumulated by your x-ray bequeathed strategy in red should be higher than that of the pessimistic strategy in blue by the end of all 1000 rounds.

7. Handin

When you are ready to hand-in your project run the command

```
cs4_handin twentyone
```

from a CIT Terminal window in your `~/course/cs0040/projects/twentyone` directory, and the entire contents of the directory will be handed in. **NOTE: Be sure to turn in all files needed to run all aspects of your project. Including your AIPlayer class definition files.**

You can re-submit this assignment using the `cscs4_handin` command at any time, but only your most recent submission with be graded.

Please include a README (a simple text file) that details any parts of the project that aren't working. If a section does not pass, if you explain in detail why it does not pass, you may receive some partial credit for that verification section. You should also explain your approach to the AIPlayerXray for Part II in your README.

Be sure to turn in all files needed to run all aspects of your project.

Good luck and happy coding!

Please let us know if you find any mistakes, inconsistencies, or confusing language in this document or have any concerns about this and any other CS4 document by posting on Piazza or filling out our anonymous feedback form.