Project

Due Friday by 12pm **Points** 100 **Available** after Apr 7 at 12pm

CS5001 Spring 2021 Final Project

Deadline: April 23rd at 12:00 noon ET

To submit your solution, compress all files together into one .zip file. Login to handins.ccs.neu.edu with your Khoury account, click on the appropriate assignment, and upload that single zip file. You may submit multiple times right up until the deadline; we will grade only the most recent submission.

NOTE: **You may NOT use late days for this assignment**. Any project turned in after the deadline will receive an automatic zero! Manage your time well and be sure to submit something well before the deadline.

End Product

Your files to turn in:

- **mastermind_game.py** (includes a main() function that lets us play your game)
- **test_mastermind_game.py** (tests for game functions, does not test turtle/view)
- **readme.txt** (short plaintext file – 1 or 2 paragraphs – explaining your design and any extra credit functions you attempted)

You may turn in other files with helper functions, but these above files are the minimum required. The code may be written as classes and objects or procedurally, although some functions will be required to be outside of classes for our testing hooks. See below for the functions required for testing hooks.

Include any and all files required for playing your game, including gifs (either ones we provided for you or custom ones that you created) and any starter code you used, whether you modified it or not. We should be able run mastermind_game.py in IDLE and play your game as you intend it to look for us.

The Game: Mastermind

Mastermind **(https://en.wikipedia.org/wiki/Mastermind_(board_game))** is a coding-breaking board game for two players. The "secret code" is a set of 4 colors chosen out of a possible 6 colors, where the player needs to guess which colors are in which positions in the 4-color secret code. After each guess, the player learns how many colors in their guess are in the right position, and how many had a color that's in the code but the color is in the wrong position. Scoring pegs of different colors were used to show them

how many correct guesses and how many correct positions they had gotten with their guess. Red pegs meant a correct color but out of position, black pegs meant a correct color in the correct position. These scoring pegs after each guess can be placed in any order – the player never knows _which_ colors are correct and/or in the correct position. Examples of the two-player version of the game being played can be seen
here (https://www.youtube.com/watch?v=dMHxyulGrEk)

(https://www.youtube.com/watch?v=dMHxyulGrEk)
and
here (https://www.youtube.com/watch?v=wsYPsrzCKiA)

(https://www.youtube.com/watch?v=wsYPsrzCKiA)
.

In an older version of Mastermind, called, Bulls and Cows, guesses with correct position were called "bulls", and correct guesses with incorrect positions were called "cows". We will use the "bull" and "cow" terminology when talking about correct guesses for Mastermind.

Our Version of Mastermind: 1 player

We'll be designing a version for one player. In your version, the program selects the 4 color secret code. Your program will also place the scoring pegs. In some versions of Mastermind, blank positions are allowed, and/or duplicate colors are allowed. You can play a one-person version of the game **here (https://www.archimedes-lab.org/mastermind.html)**; this version allows duplicate colors. Our version will not allow blank positions or duplicate colors (but see "Extra Credit", below). Our permitted colors are: `colors = ["red", "blue", "green", "yellow", "purple", "black"].`

In the one-player game, the score is the number of guesses it takes the player to guess the code. The player loses if they don't guess the code in 10 tries. Lower scores/fewer guesses are better. Our scoring pegs will be red for cows and black for bulls.

Prof Keith posted walkthroughs and examples of the game play and expected functions for the game, which can be found on Piazza here (https://piazza.com/class/kjyf0wctcts1no?cid=313).

Game Play Functions

1. We should be able to input the player name to your game through a pop-up window rather than in the terminal.
2. Your program should draw a board as seen in the demo videos. This board for the Mastermind game should include:
   - a Leaderboard on the right that lists the previous best scores in the game (lowest scores are best).
   - A “Quit” button that exits the game. We provide a gif for this button in the starter code.
   - A set of clickable, colored guess buttons that allows a player to choose one color at a time for a guess, and does not allow them to choose duplicate colors (unless you have implemented the complete option in “Extra Credit”). We provide starter code for these buttons in Marble.py
   - A green check mark button that confirms that a guess should be checked/entered. We provide a gif for this button in the starter code.
   - A red “X” button that removes a guess that has not yet been checked/entered and resets the clickable, colored guess buttons. We provide a gif for this button in the starter code.
   - A representation of the board that displays the current guess and previous guesses in this game. We provide starter code for these buttons in Marble.py .
   - A part of the display that shows the scoring pegs. These scoring pegs are filled in after a guess is checked, with black pegs indicating colors guessed in the right position, and red pegs indicating colors guessed correctly, but not in the right position. Blank circles can represent neither of those conditions.
3. Your program should save the best scores in a leaderboard file, and when the program is re-launched, those scores should be visible with the player name in the leaderboard. A minimum of 2 best scores should be saved. The list should update with new best scores when someone achieves a new best score. Do not submit your leaderboard.txt file with your submission; a new file should be created if one does not exist yet.
4. The program should display a visual error message popup if the leaderboard file cannot be found. We provide a gif for this message in the starter code.
5. The program should display a visual message popup if the player presses the Quit buttons. We provide a gif for this message in the starter code.
6. The program should display a visual message popup if the player wins or loses. We provide gifs for these messages in the starter code.

All user interaction should be via your turtle-based user interface; users must use the mouse to play your game (the only keyboard actions are when you capture the player name at the start of the game)

Test Suite

Your test file should test all the functions in your game that don’t involve turtle (and remember, for good design and separation of concerns, turtle drawing functions should be separate from game functions).

Test edge cases and exceptions.

Starter Code

The starter code described below can be found **here** (https://northeastern.instructure.com/courses/63359/files/8964749/download?download_frd=1) .

Using our starter code is optional. You can customize this game as you like, modify the starter code, or not use it at all. We will be providing you with these files:

- **`Point.py`**
  - This file contains a class Point with two attributes, x and y. It represents a geometric point with x and y coordinates. It has two methods:
    - delta_x, which takes as input another Point, and returns the absolute distance of the different between this point and the other point's x coordinates,
    - delta_y, which takes an input another Point, and returns the absolute value of the difference between this point and the other point's y coordinates.
- **`Marble.py`**
    - This file contains a class Marble with the attributes pen, color, position, visible, is_empty, and size. This class can draw an empty Marble and set its color, get its color, erase itself, and determine if it has been clicked. This class requires the use of the class Point.
- A whole bunch of .gif files:
    - checkbutton.gif, a green check mark for the "check" button
    - file_error.gif, a file error popup message
    - leaderboard_error.gif, a leaderboard access error popup message
    - lose.gif, a popup message for when the player loses
    - quit.gif, a red square for the "quit" button
    - quitmsg.gif, popup message for when the player quits
    - winner.gif, a popup message for when the player wins
    - xbutton.gif, a red "X" for the "cancel" button

Note: none of the popup messages actually pop up, they're just gifs that you have to write the code to make them open in a new window, etc.

Required Testing Hooks

Your mastermind_game.py file must include the following function, provided outside of any classes if you are using classes:

- `count_bulls_and_cows(secret_code, guess)`
  This function takes as input two lists: the first parameter is the secret code and the second parameter is the user guess.
  This function returns a 2-tuple (a tuple with two elements) containing the number of bulls and cows

by comparing with the secret code. The secret code and guesses are in the form of a list of strings from the options:

```
colors = ["red", "blue", "green", "yellow", "purple", "black"]
```

   - this function doesn't need to run anything in your game view – it just needs to work as stated.

Program Evaluation

We will test your code using `count_bulls_and_cows`() with automated testing, and we will also play your game.

- We'll be looking for the functionality described in this document and shown in the demo videos for program correctness.
- Document your code extensively, following all the guidelines you've learned through the semester and what's in the Style Guide for the course.
- Make your code readable, following the style guide with good variable names, good function writing, well-formed and clear classes if you're using classes, good use of whitespace, and all the other code readability guidelines you've learned throughout the semester. Readability and documentation get even more important the longer and more complicated your programs get!
- Keep your code as efficient as possible – don't do anything twice that you only need to do once, etc.

Instead of AMAZING points, we'll use those 2 points for "UI Aesthetics", which we will use to assess the overall UI aesthetics and user experience.

Extra Credit

Bonus points above 100 can be earned for adding the following functionalities. Do _not_ try to work on these functionalities until you're sure you have a finished, working project to hand in that has all of the required functionalities!

If you include any of these extra credit options, document them extensively in your code as well as in your readme.txt.

- Allow duplicate colors to be used in the secret code and in guesses for your game. This makes the game substantially harder to win in 10 guesses. +5 points
- Allow blanks in the secret code and in guesses in the game. In `count_bulls_and_cows()`, blanks should be represented by the empty string. No points are given for only implementing this in `count_bulls_and_cows()`: it must be a part of the visual representation of your game. +3 points
- Allow players to play multiple rounds of the game without quitting out of the application. Make sure the high scores update between games, but don't require the player to put their name in again. +5 points

Notes and Guidance to Help You

Important: **Your project MUST be runnable and do something non-trivial to get credit** for this assignment. However, it does not need to be functionally complete to get partial credit.

Resources

This project will have you using a bit of almost everything we learned this semester (except recursion), so be sure to allocate appropriate time to complete it. If you wait until the last minute to work on this, you are almost guaranteed NOT to finish on time. You cannot use late days on this project, because we have a deadline for turning in grades. Plan to start early!

At the same time, don't panic. The project is bigger than anything you've worked on this term, but the concepts are all within your reach. You'll need to do a bit of reading on Turtle to brush up on the elements you'll likely need for the graphics. In particular, review the turtle objects themselves and think about how you can use multiples of them to do things on the gameboard for you.

A link to Python Turtle documentation is here: **https://docs.python.org/3/library/turtle.html#turtle.update (Links to an external site.) (https://docs.python.org/3/library/turtle.html#turtle.update)**

Feel free to search the web for other Turtle resources to help you learn whatever you think you might need to know for the UI portion of your application.

Pro-Tip

You likely will NOT be able to complete this project in one day. Work a little bit EVERY day (even if it's refactoring your code) to make progress. Have a plan and work towards smaller milestones to keep your development on pace for completion. If you haven't planned a project before, a sample work plan is below. Feel free to use it as a rough guide if you're not sure how to get started. (You'll need to act as your own project manager and writing down some timeboxed goals will likely be helpful).

Work Plan

Here are some of the things I did when creating the sample solution (NB: I'm a Turtle novice like many of you, so I had a learning curve here too). I'm including some sample timeboxes – you can adjust these to suit your working style and progress.

Milestone 1: *Mastermind / Bulls & Cows model*

Goal: Implement the fundamentals for basic game without worrying about the graphics yet. Develop the functionality to generate a randomized "secret code" and the ability to test user input via count_bulls_and_cows(). Don't spend a lot of time with an elegant textual interface; you just want to get the core of the "game model" working so you can use it later.

[1 day]

Milestone 2a: *Turtle Pre-work Exploration* (In agile terms, this would be called a “spike” where we explore any technology we’re a bit shaky on, so we can learn & gain confidence before trying to use it).

Goal: Review how to draw shapes, capture mouse clicks, use turtle objects to render .gif images in the proper locations, etc.. Turtle also has some rudimentary "dialog box" functionality like what you see me use in the video to get the player name. It's pretty straightforward, but you'll need to read the documentation to see how to use it. Some of this code will be “throw-away” but it’s a learning exercise before digging in.

[1 day]

Milestone 2b: *Develop Gameboard* Write the code to create the entire game board: The play area, status area & leader board. Validate the code works with “dummy” data.

[1 day]

Milestone 3: *Marble placement & behavior:* Write the code to manage proper Marble placement for the game AND the Marble “Guess Panel” (at the bottom of the screen where the user selects the colors they want). If you’re using our included Marble code, you’ll likely want to spend a little time examining that code and seeing how to use it. Check your layout algorithm to make sure it works for placing each User Guess in the correct spot. Implement code to handle mouse clicks & selecting Marbles appropriately.

[3 days]

Milestone 3: *Game behavior:* Write the code to implement the game rules. Plug your count_bulls_and_cows()code into your application code so that the guess pegs can be updated properly given a user guess. Write the code to update the “row marker” every time the user enters a new guess.

[3 days]

Milestone 4: *Leaderboard & clean up:* Write the code to implement saving/retrieving and showing real contents of the leader board. Clean up code, continue to test.

[2 days]

Milestone 5: *Write readme.txt:* Write readme.txt. Do any extra work for bonus points as desired. Any other outstanding clean up.

[1 day]

Release: Before Noon, April 23rd: *Ship It!*