MACHINE PROBLEM 2

SIX SUITS: A GAME OF MEMORY

Ramon Valeriano, Adriel Padernal, Coleen Carolino

December 2018

PROGRAMMER'S GUIDE AND DOCUMENTATION

1. Images Used and Other Files

By default, the game uses the #2 cards from a proposed six suit deck found on Google, which links to http://www.jamesrobertwatson.com/deck6.html. The filenames of each card are saved in **CardImages.txt**. The single back image of all cards was also found on google and links to https://www.ellusionist.com/playing-cards. The back face was saved simply as **CardBack.jpg**. A high score file (which is written to) is also included, titled **HighScores.txt**.

2. The Engine Module

The Engine module contains important functions needed for the game to run properly. A "Card Class" is also defined here. Certain functions defined here need functions from the **pyglet**, **random**, and **time** modules.

Card Objects

A **Card** object represents a card to be used in the game. Much like a real, physical card, **Card** objects have front and back faces.

class engine.**Card**(backLoad,frontLoad,listOfPairs,batch)

All arguments are required. Note that backLoad and frontLoad are images loaded via **pyglet.image.load**. These images may be PNGs, JPEGs or any loadable image format. For the sake of this game, it also takes a list of pairs of x-y coordinates. It shuffles the list and takes the last pair of coordinates. This last pair is also removed so another card won't take the same coordinates.

Other constructors, all class methods:

classmethod Card.**flip**()

> If the card is still marked as 'unmatched', the card's **displayFaceStatus** attribute is changed (if it is 'front', it becomes 'back' and vice versa). The sprite of the card is now a 0.5 scale version of the opposite face (if the image was **frontLoad** it is now **backLoad**, and vice versa). This does nothing to cards that are already marked as 'matched'.

Class attributes:

Card.**batch**

> The batch which the card's sprite will be a part of

Card.**status**

> A string (either 'unmatched' or 'matched')

Card.**backLoad**

> The image of the back face loaded via **pyglet.image.load**

Card.**frontLoad**

> The image of the front face loaded via **pyglet.image.load**

Card.**displayFaceStatus**

> A string (either 'front' or 'back')

Card.**x**

> The x coordinate of where the card's sprite will be drawn.
> Note: this x coordinate comes from the randomly chosen x-y pair when the card object was initialized.

Card.**y**

> The y coordinate of where the card's sprite will be drawn.
> Note: this y coordinate comes from the randomly chosen x-y pair when the card object was initialized.

Card.**displayFace**

> A sprite of either Card.**backLoad** or Card.**frontLoad** at 0.5 scale. The sprite's x attribute is Card.**x** and the sprite's y attribute is Card.**y**. The **subpixel** attribute is set to True and the sprite's batch is Card.**batch**. Upon initialization, Card.**displayFace** is a sprite of Card.**frontLoad** as the board starts with all cards "face up" before being flipped down.

Other Functions:

engine.**cardReader(***cardFile***)**

Opens *cardFile*, a file with filenames of the back face and front faces of the cards, reads it, and returns a list of filenames of all faces.

engine.**highScoreUpdated(***scoreFile,finishTime,dateTime***)**

Opens *scoreFile*, a file with completion times and timestamps, reads it, and makes a list of these scores. The time recorded upon finishing (*finishTime*) and the date and time at which it was recorded (*dateTime*) are combined into a single string. This string is appended to the list of scores, sorted and then written to *scoreFile*.

engine.**flipChecker(***flippedCards***)**

First, this function assumes that *flippedCards* is a list that contains exactly 2 **Card** objects whose **displayFaceStatus** attributes are both 'front'. If the **frontLoad** attributes of the two cards are the same (e.g. both spades), then the **status** attributes of both cards are updated to 'matched', the cards remain permanently flipped up, and are no longer affected by **flip**(). If the two cards have different **frontLoad** attributes (e.g. one is a spade and one is a club), then the program sleeps for 0.5 of a second, both cards are run through **flip**() so that they face down again. In either of the two cases, *flippedCards* is always reset at the end as an empty list.

engine.**gameChecker(***cardList***)**

Checks if the game is "finished" by examining the **status** attribute of each **Card** object in *cardList*. If at any time, it detects a **status** attribute of 'unmatched', the function returns the string 'unfinished' because there are still some cards which have not been matched yet. Otherwise, if the function iterates through each card and never detects a **status** attribute of 'unmatched', then it returns the string 'finished' because all cards have been matched.

engine.**TimeElapsed(***startTime***)**

Returns the difference between **time.time()** and *startTime.*

engine.**secondsConverter(***seconds***)**

Converts seconds into the format: MINUTES:SECONDS.MILISECONDS.

3. The Main Module

The Main module makes use of both **engine** and **pyglet** to deliver a working memory game. It also needs functions from the **random**, **time**, and **datetime** modules.

The program starts by assigning the filenames of the list of card faces and the list of completion times (CardImages.txt and HighScores.txt by default, respectively) to variables.

It then runs the "Card Images" variable through **cardReader**. A list with 7 filenames is returned. The very first filename is designated as the "Back Face" for all cards and is loaded via **pyglet.image.load**. The remaining 6 filenames are put into a list of their own and each filename is again loaded via **pyglet.image.load**.

Then, a graphics batch is created for the sprites of each card, along with a "Card List" to be used to store the card objects. 12 pairs of x-y coordinates are also defined in a list such that there will be 12 cards in a 4 X 3 grid.

12 Card objects are created (2 for each front face). As mentioned in **engine**, the cards take the x-y coordinates list as one of the arguments when being initialized. These card objects are then added to the "Card List".

A list of "Flipped Cards" to be updated as the game progresses is also initialized but is empty for now.

The game window is then initialized along with the timer label. Initially, all 12 cards are face up, but the scheduled **allFlip** function "sleeps" for 3 seconds then runs all cards through the **flip**() method so that they become face down. After which, the game immediately takes note of the start time, unschedules **allFlip**, schedules **timerUpdate**, schedules **gameUpdate**, and schedules **flipChecker**.

The **timerUpdate** function constantly calls on the **timeElapsed** function (from **engine**), taking the recorded starting time as an argument. This result is then run through **secondsConverter** (from **engine** again). The converted form is then designated as the text of the timer label created earlier.

The **gameUpdate** function constantly calls on **gameChecker** (from **engine**), taking the "Card List" as an argument. If **gameChecker** returns 'unfinished', nothing will happen. What happens when **gameChecker** returns 'finished' will be discussed later.

Finally, the **flipCheck** function constantly checks the number of cards in the 'Flipped Cards' list. If the list has 2 cards, it runs the **flipChecker** (from **engine**) function, taking the 'Flipped Cards' list as an argument. Otherwise, it does nothing.

At this point the game begins. All cards are face down, and a timer is ticking. When a user clicks with the left mouse, the program checks if the x and y coordinates of the click are in the boundaries of any card. If they are, the corresponding card is "flipped' via the **flip**() method, and is added to the list of "Flipped Cards". Note: at most only 2 cards can be flipped at once. The user can however choose to click the same card again to face the card down (which removes the card from the "Flipped Cards" list). As mentioned earlier, if the player has 2 flipped up cards, then the **flipChecker** function is executed.

The moment the program detects the game is "finished" (i.e. **gameChecker** returns 'finished' because all cards have a **status** attribute of 'matched'), the total elapsed time at that exact moment is recorded, along with the date and time at which it was recorded (via **datetime.now**()). The "now" variable is then converted into a string made from combining its attributes (year, month, day, hour, minute, second, microsecond). The string of total elapsed time, and the "now" variable are passed as arguments to **highScoreUpdater** along with the "High Score File" variable. All functions are unscheduled, and the window is closed. The player may choose to play again by executing **main** again.

4. References

Install a Python package into a different directory using pip? (n.d.). Retrieved December 4, 2018, from https://stackoverflow.com/questions/2915471/install-a-python-package-into-a-different-directory-using-pip

Playing Cards. (n.d.). Retrieved December 4, 2018, from https://www.ellusionist.com/playing-cards

Pyglet.org. (n.d.). Pyglet.clock. Retrieved December 4, 2018, from https://pyglet.readthedocs.io/en/pyglet-1.2-maintenance/api/pyglet/pyglet.clock.html

Python Software Foundation. (n.d.). Datetime - Basic date and time types. Retrieved December 4, 2018, from https://docs.python.org/3/library/datetime.html#module-datetime

Python Software Foundation. (n.d.). 7. Input and Output. Retrieved December 4, 2018, from https://docs.python.org/3/tutorial/inputoutput.html

Python Software Foundation. (n.d.). Venv - Creation of virtual environments. Retrieved December 4, 2018, from https://docs.python.org/3/library/venv.html

Watson, J. (n.d.). DECK6. Retrieved December 4, 2018, from http://www.jamesrobertwatson.com/deck6.html