ΕΛΛΗΝΙΚΟ ΑΝΟΙΚΤΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

Ομαδικό Προγραμματιστικό Project: Παιχνίδι Μνήμης με Τράπουλα

Συμμετέχοντες: Θεολόγου Σπυρίδων, Μιχαλόπουλος Λεωνίδας

Project ID: 05

Κωδικός Τμήματος: ΠΛΗΠΡΟ-ΗΛΕ62

Αριθμός Ομάδας: Ομάδα 3

Καθηγητής Σύμβουλος: Ζαφείρης Βασίλειος

Ακαδημαϊκό Έτος: 2021-2022

Github: https://github.com/spiros-theologou/MemoryGamePython

Εισαγωγή

Το project αφορά την υλοποίηση μιας παραλλαγής του κλασικού παιχνιδιού μνήμης – ταιριάσματος καρτών, που είναι αναποδογυρισμένες και τοποθετημένες σε ένα ταμπλό. Οι κάρτες προέρχονται από την γνωστή σε όλους τράπουλα, η οποία αποτελείται από 4 «σειρές» φύλλων(κούπα ♥, σπαθί ♣, καρό ♦, μπαστούνι ♠), ενώ κάθε σειρά περιλαμβάνει τους αριθμούς από το 1(A) έως και το 10 και τις φιγούρες: Βαλές(J), Ντάμα(Q), Ρήγας(K).

Η συγκεκριμένη υλοποίηση δίνει δυνατότητα παιξίματος με έναν παίκτη, αλλά και με πολλούς(έως 4). Μόλις ξεκινήσει το παιχνίδι, οι κάρτες απλώνονται αναποδογυρισμένες στο ταμπλό, και ο Παίκτης 1 καλείται να επιλέξει 2 κάρτες, κάνοντας κλικ επάνω τους. Εάν αυτές οι κάρτες ταιριάζουν ως προς το σύμβολο(αριθμός ή φιγούρα), τότε προσάπτονται στον παίκτη πόντοι, ανάλογοι με την αξία των φύλλων που αναποδογύρισε, και οι κάρτες παραμένουν ανοικτές. Διαφορετικά αναποδογυρίζουν και πάλι. Στη συνέχεια, καλείται να παίξει ο επόμενος παίκτης. Οι αξίες των τραπουλόχαρτων είναι: i) Ο Άσσος(A) έχει αξία έναν πόντο, ii) τα φύλλα 2-10 έχουν αξία ίση με τον αριθμό τους, ενώ iii) οι φιγούρες έχουν αξία ίση με 10 πόντους. Νικητής είναι ο παίκτης με τους περισσότερους πόντους όταν ανοίξουν όλες οι κάρτες και έτσι ολοκληρώνεται το παιχνίδι.

Το παιχνίδι υποστηρίζει 3 διαφορετικές δυσκολίες οι οποίες ορίζονται ανάλογα με τον αριθμό καρτών που βρίσκονται στο ταμπλό:

I) Εύκολο: 16 κάρτες(Μόνο φιγούρες και 10) τοποθετημένες σε διάταξη 4x4.
II) Μέτριο: 40 κάρτες(Χωρίς τις φιγούρες) τοποθετημένες σε διάταξη 4x10.
III) Δύσκολο: 52 κάρτες(όλη η τράπουλα) τοποθετημένες σε διάταξη 4x16.

Ειδικές Κάρτες:

- Αν κάποιος παίκτης ανοίξει ταυτόχρονα δύο Βαλέδες (J), κερδίζει τους αντίστοιχους πόντους και παίζει ξανά.

- Αν κάποιος παίκτης ανοίξει ταυτόχρονα δύο Ρηγάδες (K), τότε κερδίζει τους αντίστοιχους πόντους και ο επόμενος παίκτης χάνει τη σειρά του.

- Αν κάποιος παίκτης ανοίξει μια Ντάμα και ένα Ρήγα, τότε θα έχει ευκαιρία να ανοίξει και μια τρίτη κάρτα. Οι κάρτες που ταιριάζουν μεταξύ των τριών μένουν ανοικτές και ο παίκτης κερδίζει τους αντίστοιχους πόντους. Οι κάρτες που δεν ταιριάζουν παραμένουν κλειστές.

Παιχνίδι με αντίπαλο τον υπολογιστή:

Σε περίπτωση που αριθμός παικτών είναι ίσος με 1, ο παίκτης θα παίζει με αντίπαλο το ίδιο το πρόγραμμα (ΠΡΟΓΡΑΜΜΑ). Συγκεκριμένα το ΠΡΟΓΡΑΜΜΑ θα θυμάται τις τελευταίες 5 κάρτες που άνοιξαν από τον παίκτη ή το ΠΡΟΓΡΑΜΜΑ (ιστορικό καρτών). Όταν έρχεται η σειρά του, το ΠΡΟΓΡΑΜΜΑ θα παίζει ακολουθώντας την παρακάτω στρατηγική:

1. Αν στο ιστορικό καρτών περιλαμβάνονται κάρτες που ταιριάζουν ως προς το σύμβολο τότε θα ανοίγει τις συγκεκριμένες κάρτες και θα τις αφαιρεί από το ιστορικό επιλογών.

2. Αν δεν υπάρχουν κάρτες που ταιριάζουν στο ιστορικό καρτών, θα ανοίγει την πρώτη κάρτα τυχαία (από τις κλειστές κάρτες). Αν το ιστορικό επιλογών περιέχει κάρτα που ταιριάζει τότε θα παίζει τη συγκεκριμένη κάρτα και θα την αφαιρεί από το ιστορικό επιλογών. Αλλιώς θα επιλέγει και τη δεύτερη κάρτα τυχαία από τις κλειστές κάρτες και αν δεν ταιριάξουν, προσθέτει και τις δυο κάρτες στο ιστορικό επιλογών.

Αποθήκευση του παιχνιδιού:

Στο τέλος κάθε γύρου παίκτη, η τρέχουσα κατάσταση του παιχνιδιού αποθηκεύεται σε δυαδικό αρχείο, μέσω της βιβλιοθήκης pickle και δίνεται στον χρήστη η δυνατότητα να συνεχίσει το παιχνίδι από το σημείο που σταμάτησε την προηγούμενη φορά.

Οδηγίες Εγκατάστασης:

Το πρόγραμμα αποτελείται από 8 .py αρχεία(Memory_game.py, Root.py, class_tile.py, Menu.py, class_ContinueGame.py, class_GameState.py, class_player.py, class_Computer.py) και έναν φάκελο(gui) με .png εικόνες που χρησιμοποιούνται για την αναπαράσταση των καρτών. Όλα έρχονται σε ένα .zip αρχείο.

Όλα τα modules που χρησιμοποιούνται υπάρχουν στην Python Standard Library, επομένως δεν χρειάζεται εγκατάσταση περαιτέρω modules, παρά μόνο της Python.

Εφόσον πληρούνται οι παραπάνω προϋποθέσεις, ο χρήστης αρκεί να εξάγει όλα τα αρχεία από το συμπιεσμένο αρχείο(.zip) σε κοινό directory και μέσω της κονσόλας να εκτελέσει το αρχείο Root.py(κατευθύνοντας στο directory και πληκτρολογώντας την εντολή `python Root.py`).

Το πρόγραμμα δημιουργήθηκε σε Python 3.9 και δοκιμάστηκε σε περιβάλλον Windows 10 και ενώ μπορεί να λειτουργήσει σε άλλα λειτουργικά συστήματα ή άλλες εκδόσεις της Python, η συμπεριφορά του πιθανώς να είναι απρόβλεπτη.

Πηγές/Παραπομπές:

Γενικές πληροφορίες/tutorials για το tkinter:

- https://www.geeksforgeeks.org/flipping-tiles-memory-game-using-python3/
- https://stackoverflow.com/questions/tagged/tkinter
- https://www.geeksforgeeks.org/python-gui-tkinter/
- https://www.activestate.com/resources/quick-reads/how-to-position-widgets-in-tkinter/
- https://docs.python.org/3/library/tkinter.html
- https://www.tutorialspoint.com/how-to-use-an-image-as-a-button-in-tkinter
- Tile Matching Game - Python Tkinter GUI Tutorial:
    - https://www.youtube.com/watch?v=tlMPVGSEEDw
- Let's Code Python: Memory Tile Game with GUI:
    - https://www.youtube.com/watch?v=-jlfRMbjCMc
- Create Graphical User Interfaces With Python And TKinter:
    - https://www.youtube.com/watch?v=yQSEXcf6s2I
- Python Object Oriented Programming (OOP) - For Beginners:
    - https://www.youtube.com/watch?v=JeznW_7DlB0
- Serialize Python Objects With Pickle:
    - https://www.youtube.com/watch?v=qt15PnF8x-M
- Πηγή των .png εικόνων των καρτών της τράπουλας:
    - https://opengameart.org/content/playing-cards-vector-png

Υλοποίηση της Εφαρμογής και μια Περιήγηση

Η υλοποίηση της εφαρμογής έγινε σε Python 3.9, και οι βιβλιοθήκες που χρησιμοποιήθηκαν είναι οι:

i) tkinter για την υλοποίηση του GUI, ii) time για την χρήση της μεθόδου sleep() που «παγώνει» το πρόγραμμα για κάποιο δεδομένο χρόνο και χρησιμοποιήθηκε για την παύση του προγράμματος κατά την αποκάλυψη των καρτών, iii) pickle για την αποθήκευση ως binary file, των δεδομένων της κατάστασης του παιχνιδιού(στιγμιότυπο κλάσης Gamestate()) και iv) random για τη μέθοδο shuffle που ανακατανέμει τυχαία τα στοιχεία μιας λίστας και την choice που επιλέγει τυχαία ένα στοιχείο λίστας.

Το πρόγραμμα σχεδιάστηκε με 7 κλάσεις(Menu, NewGame, ContinueGame, Player, Tile, Gamestate, Computer) και ένα αρχείο κώδικα(Root.py) που υλοποιεί το κύριο παράθυρο και καλεί την κλάση Menu που θα προχωρήσει στην υλοποίηση του Αρχικού Μενού. Στη συνέχεια ανάλογα με τις επιλογές του χρήστη(Νέο Παιχνίδι, Συνέχεια.. → Επιλογή Δυσκολίας →Επιλογή αριθμού παικτών), καλούνται οι Κλάσεις MemoryGame ή ContinueGame.

Αναλυτική περιγραφή των κλάσεων:

I) **Η κλάση Menu:** παρουσιάζει και διαχειρίζεται τις επιλογές του αρχικού μενού και μεταβαίνει στις επόμενες κλάσεις υλοποίησης του παιχνιδιού. Δέχεται ως όρισμα το master window, που έχει δημιουργηθεί στο Root.py, δημιουργεί ένα frame στο οποίο τοποθετούνται τα κουμπιά και οι λεζάντες που προωθούν τον χρήστη στο να κάνει τις επιλογές του. Αρχικά ο χρήστης καλείται να επιλέξει αν θέλει να ξεκινήσει Νέο Παιχνίδι ή να συνεχίσει το τελευταίο παιχνίδι που ξεκίνησε(Συνέχεια..).

Εάν ο χρήστης επιλέξει Νέο Παιχνίδι, καλείται η μέθοδος new_game() η οποία αρχικοποιεί τον τύπο του παιχνιδιού ως “New Game”, αφαιρεί τα υπάρχοντα κουμπιά και παρουσιάζει το μενού επιλογής δυσκολίας:

Εκεί, ανάλογα με την επιλογή του χρήστη αρχικοποιείται η δυσκολία του παιχνιδιού(μέσω της μεθόδου set_difficulty(dif). Επίσης, αφαιρούνται οι επιλογές από την οθόνη(μέθοδος clear_screen()) και καλείται η μέθοδος num_of_players().

Στη συνέχεια ο χρήστης καλείται να επιλέξει τον αριθμό παικτών από το αντίστοιχο μενού που υλοποιείται από τη num_of_players().

Ο αριθμός παικτών αρχικοποιείται, μέσω της μεθόδου set_number_of_players(players) και τώρα υπάρχουν όλα τα απαραίτητα δεδομένα για την έναρξη του νέου παιχνιδιού. Τέλος, καλείται η μέθοδος start_game() η οποία δημιουργεί ένα αντικείμενο της κλάσης NewGame(master, difficulty, players) που αναλύεται παρακάτω.

Στην περίπτωση που στο αρχικό μενού ο χρήστης επιλέξει Συνέχεια.. Τότε καλείται η μέθοδος start_game() η οποία ελέγχει αν υπάρχει αποθηκευμένο αρχείο παιχνιδιού. Εάν υπάρχει, δημιουργεί ένα αντικείμενο κλάσης ContinueGame("saved_game_data.pickle", master) το οποίο αναλύεται παρακάτω. Διαφορετικά εμφανίζει ένα popup window με το κείμενο «Δεν βρέθηκε αποθηκευμένο παιχνίδι».

II) **Η κλάση NewGame:** δημιουργεί το ταμπλό και τις ανάλογες λεζάντες πληροφορίας(εμφανίζει ποιος παίζει, το τρέχων σκορ..) και υλοποιεί το παιχνίδι καθώς και το κομμάτι του AI. Δέχεται ως ορίσματα το master window, την επιλεγμένη δυσκολία και τον αριθμό των παικτών.

Αρχικά, δημιουργεί ένα frame για το ταμπλό στο οποίο θα τοποθετηθούν οι κάρτες. Καλεί τη μέθοδο init_tiles() που δημιουργεί τις κατάλληλες κάρτες(στιγμιότυπα της κλάσης Tile που αναλύεται στη συνέχεια, το ποιες κάρτες θα δημιουργηθούν εξαρτάται από τη δυσκολία που επιλέχθηκε) και τις εισάγει σε μια λίστα tiles[]. Όταν δημιουργηθούν όλες οι επιθυμητές κάρτες, η λίστα «ανακατεύεται» με τη μέθοδο shuffle() της random βιβλιοθήκης για να υπάρχει τυχαία κατανομή στο ταμπλό. Επίσης, καλείται η μέθοδος add_tile_functionality η οποία προσάπτει λειτουργικότητα στις κάρτες.

Στη συνέχεια, καλείται η μέθοδος create_board που τοποθετεί τις κάρτες στο ταμπλό, περνώντας την εικόνα της πίσω όψης της κάρτας σε κάθε μια από αυτές. Η γραμμή και η στήλη που θα τοποθετηθεί κάθε κάρτα ορίζονται μέσα σε μια for, με δεδομένα τον αριθμό γραμμής που αρχικοποιείται ως 0 και αυξάνεται κάθε φορά που φτάσουμε τον επιθυμητό αριθμό στηλών(που καθορίζεται από τη δυσκολία) και τον αριθμό της στήλης, ο οποίος ορίζεται από τον index της κάρτας μέσα στη λίστα tiles[] και τον αριθμό των στηλών(column = index % columns).

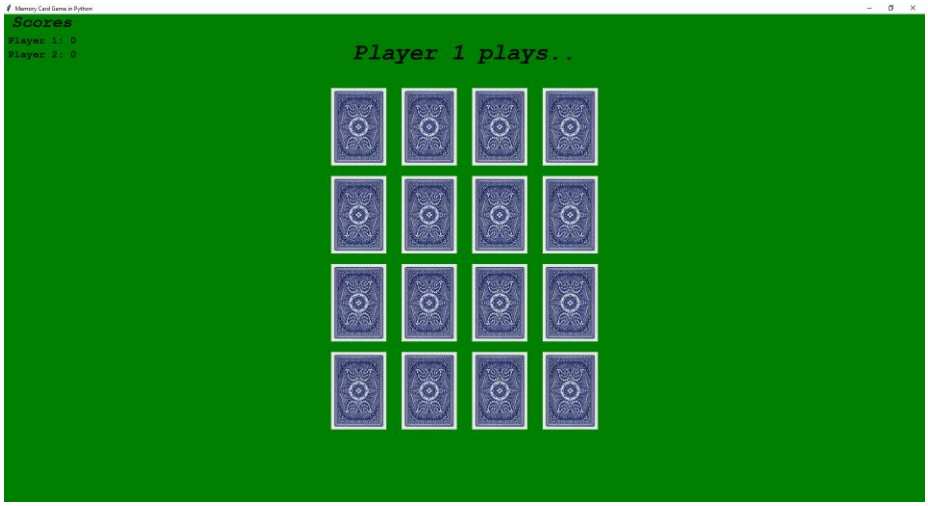
Έπειτα, καλείται η μέθοδος init_players() η οποία δημιουργεί στιγμιότυπα της κλάσης Player(αναλύεται μετέπειτα), δεδομένου του αριθμού παικτών που έχει επιλεγεί, και στη συνέχεια τα εισάγει σε μια λίστα players[].

Ακολουθεί η δημιουργία ενός frame που θα τοποθετηθεί το scoreboard και ενός frame που θα τοποθετηθεί η λεζάντα των μηνυμάτων του παιχνιδιού. Αρχικοποιούνται το index του τρέχοντος παίκτη, ο τρέχων παίκτης, ο αριθμός των ανοικτών καρτών ανά γύρο, η λίστα στην οποία τοποθετούνται προσωρινά οι κάρτες που ανοίγονται σε κάθε γύρο, καθώς και ο συνολικός αριθμός των ανοικτών και των κλειστών καρτών.

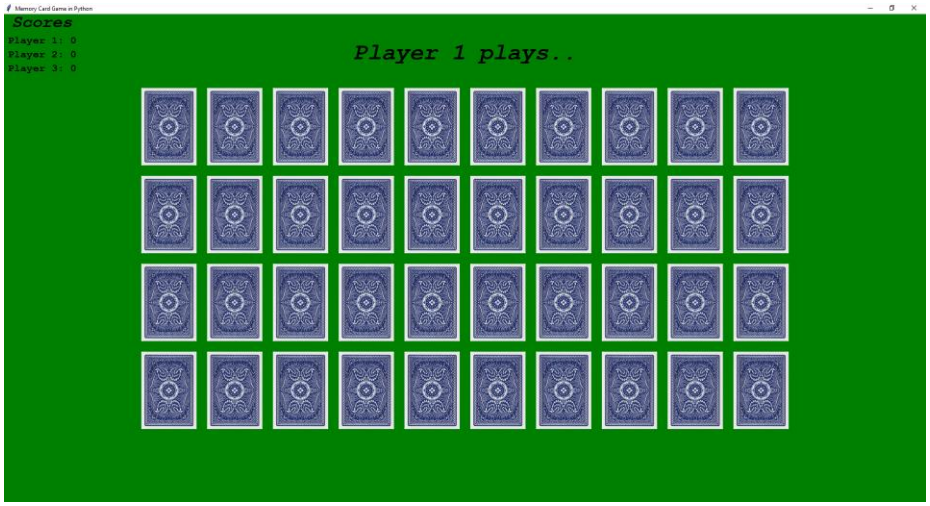
Στη συνέχεια, καλείται η μέθοδος create_scoreboard() που δημιουργεί τις λεζάντες του scoreboard έχοντας ένα dictionary με κλειδιά τους παίκτες και τιμές το σκορ του κάθε παίκτη{Player: Player.score}, και η μέθοδος create_message_board() που δημιουργεί και τοποθετεί την λεζάντα των μηνυμάτων(αρχικά εμφανίζει τον τρέχων παίκτη). Επίσης δημιουργείται ένα στιγμιότυπο της κλάσης Gamestate(αναλύεται παρακάτω) που χρησιμοποιείται για την αποθήκευση της κατάστασης του παιχνιδιού.

Παρατίθενται 3 screenshots με επιλογές «Εύκολο», «2 Παίκτες» και «Μέτριο», «3 Παίκτες» και «Δύσκολο», «4 Παίκτες» αντιστοίχως:

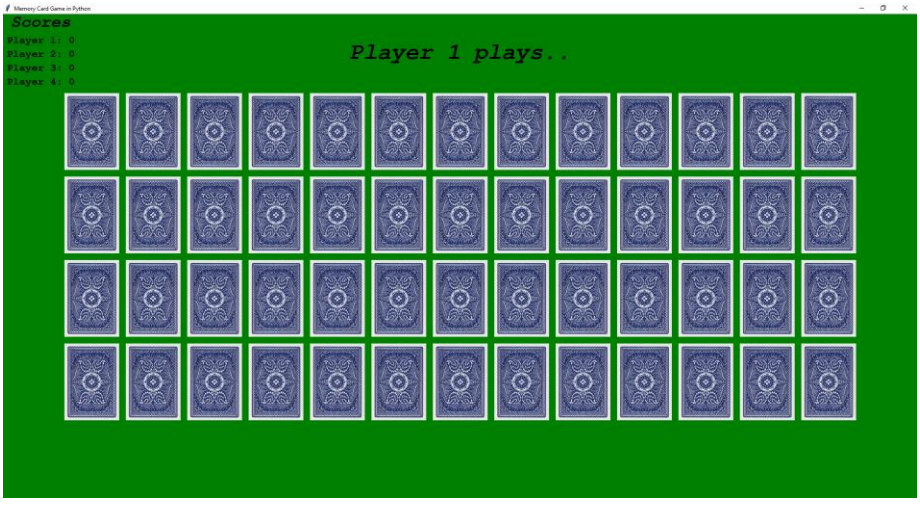
Εύκολο/2 παίκτες:



Μέτριο/3 παίκτες:



Δύσκολο/4 παίκτες:



Μόλις ο παίκτης επιλέξει μια κάρτα, καλείται η μέθοδος button_click(tile). Εάν αυτή είναι γυρισμένη ανάποδα(φαίνεται η πίσω όψη της), τότε αυξάνεται το click_count κατά 1 καλείται η μέθοδος flip(tile) η οποία αλλάζει την εικόνα της κάρτας, σε αυτή που είναι στην μπροστινή όψη της και απενεργοποιεί τη λειτουργία της(ώστε να μην μπορεί να κλικαριστεί όταν είναι face up).

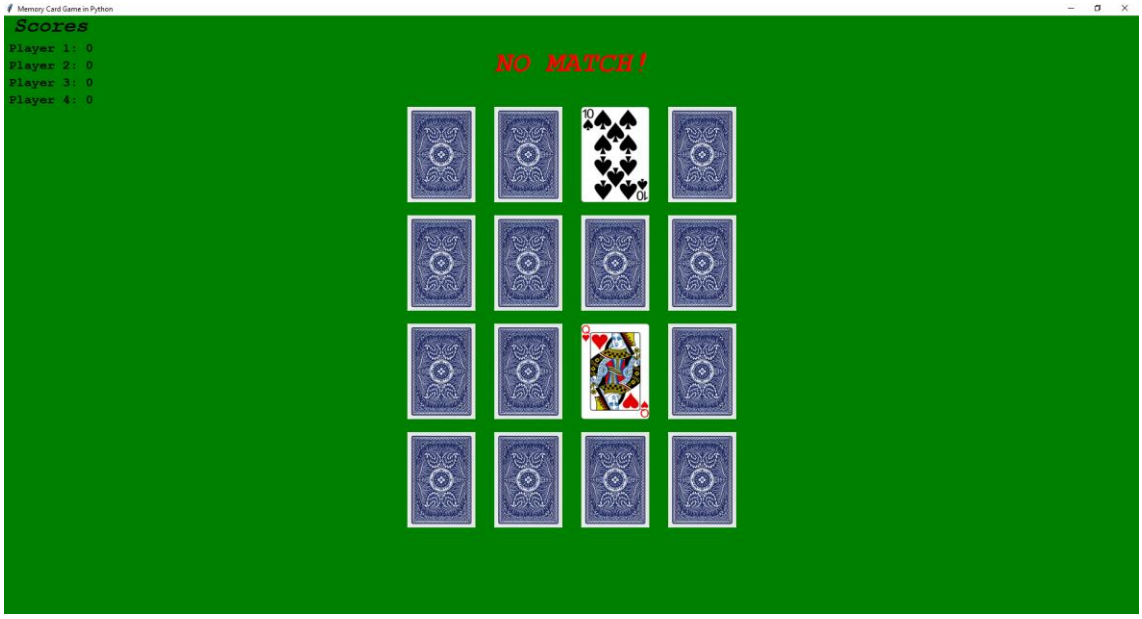
Στη συνέχεια, όταν ο παίκτης επιλέξει και δεύτερη κάρτα, πραγματοποιείται ο έλεγχος για το αν οι κάρτες είναι Ρήγας και Ντάμα. Στην περίπτωση αυτή μειώνουμε το click_count κατά 1 και δίνουμε τη δυνατότητα στον παίκτη να ανοίξει και μία τρίτη κάρτα. Στη συνέχεια καλείται η μέθοδος compare_tiles(tile_list) που δέχεται ως όρισμα την λίστα με τις κλικαρισμένες κάρτες του τρέχοντος γύρου. Καταμετρούμε τους Ρηγάδες και τις Ντάμες στις κάρτες που έχουν ανοιχτεί, και εάν ο πλήθος ενός εξ αυτών είναι ίσο με 2 τότε έχουμε επιτυχία, εμφανίζουμε στον χρήστη το μήνυμα MATCH και μένουν ανοιχτές οι κάρτες που ταιριάζουν, ενώ χρησιμοποιώντας τη μέθοδο flip() η κάρτα που δεν ταιριάζει, αναποδογυρίζει και πάλι και τις επαναπροσθέτεται λειτουργικότητα. Εάν δεν ταιριάζει κανένα ζεύγος καρτών τότε επιστρέφουν όλες οι κάρτες στην face down κατάσταση.

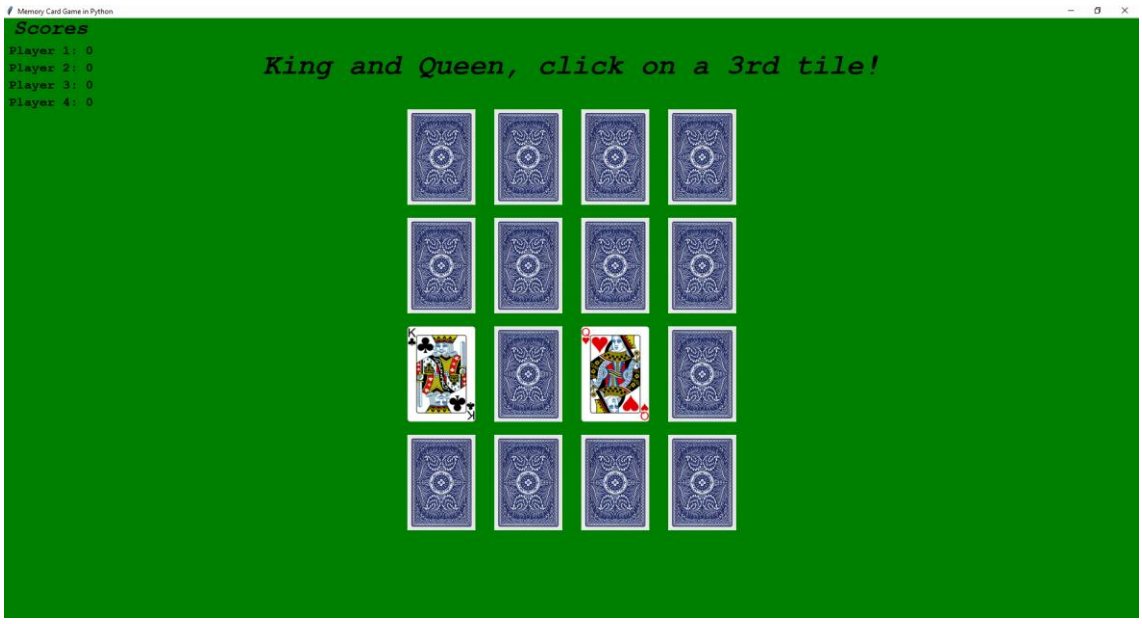
Στην περίπτωση που επιλεγούν 2 κάρτες, αλλά δεν είμαστε στην ειδική περίπτωση του ζεύγους Ρήγας-Ντάμα, καλείται και πάλι η μέθοδος compare_tiles όπου συγκρίνει τις 2 κάρτες μεταξύ τους ως προς το σύμβολο(A, 2, 3, … 10, J, Q, K) και στη συνέχεια εάν ταιριάζουν ανανεώνει τη λεζάντα μηνύματος σε “MATCH!”. Μετά από μια μικρή αναμονή, ανανεώνει το σκορ του τρέχοντος πάικτη, καλώντας τη μέθοδο add_score και ανανεώνει το scoreboard καλώντας τη μέθοδο update_scoreboard(). Τέλος, αυξάνει κατά 2 τον αριθμό των ανοικτών tiles. Στην περίπτωση που δεν ταιριάζουν οι κάρτες, ανανεώνουμε τη λεζάντα μηνυμάτων σε “NO MATCH!”, και μετά από μια μικρή χρονική καθυστέρηση οι κάρτες επανέρχονται στην αρχική τους κατάσταση(face down).

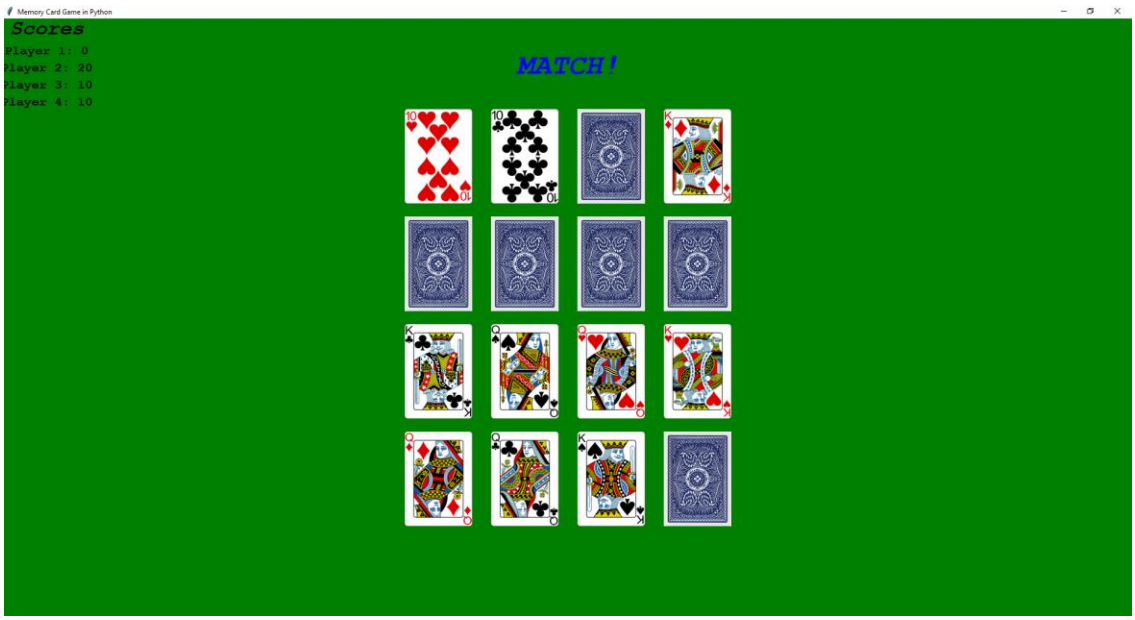
Έπειτα, επανερχόμαστε στη μέθοδο button_click(), όπου καλείται η μέθοδος change_player(). Εάν οι κάρτες που ανοίξαμε δεν ανήκουν σε κάποια ειδική περίπτωση, αυξάνουμε το index του παίκτη κατά 1, και το αλλάζουμε στο υπόλοιπο της διαίρεσής του με τον συνολικό αριθμό των παικτών για να βρούμε το index του επόμενου πάικτη χωρίς να ξεπεράσουμε τα όρια της λίστας. Οι ειδικές περιπτώσεις είναι όταν ανοίξουμε 2 Βαλέδες, όπου ξαναπαίζει ο ίδιος παίκτης, οπότε δεν χρειάζεται κάποια ενέργεια, και όταν ανοίξουμε 2 Ρηγάδες όπου αυξάνουμε το player_index κατά 2 για να χάσει ο επόμενος παίκτης τη σειρά του.

«Αδειάζει» η λίστα με τις ανοιγμένες κάρτες, μηδενίζεται ο αριθμός των κλικαρισμένων καρτών και αλλάζει η λεζάντα μηνυμάτων, υποδεικνύοντας τον επόμενο παίκτη. Τέλος καλούνται οι μέθοδοι save_game() που αποθηκεύει την τρέχουσα κατάσταση του παιχνιδιού(διαγράφοντας την προηγούμενη) και η μέθοδος check_game_end() που ελέγχει αν έχει ολοκληρωθεί το παιχνίδι συγκρίνοντας τον αριθμό των γυρισμένων καρτών με τον συνολικό αριθμό των καρτών. Αν, μετά τον έλεγχο, το παιχνίδι έχει ολοκληρωθεί, αφαιρούνται οι κάρτες από το ταμπλό και παρουσιάζεται ανάλογο μήνυμα που αναφέρει τον/τους νικητή/ες και το τελικό τους σκορ.

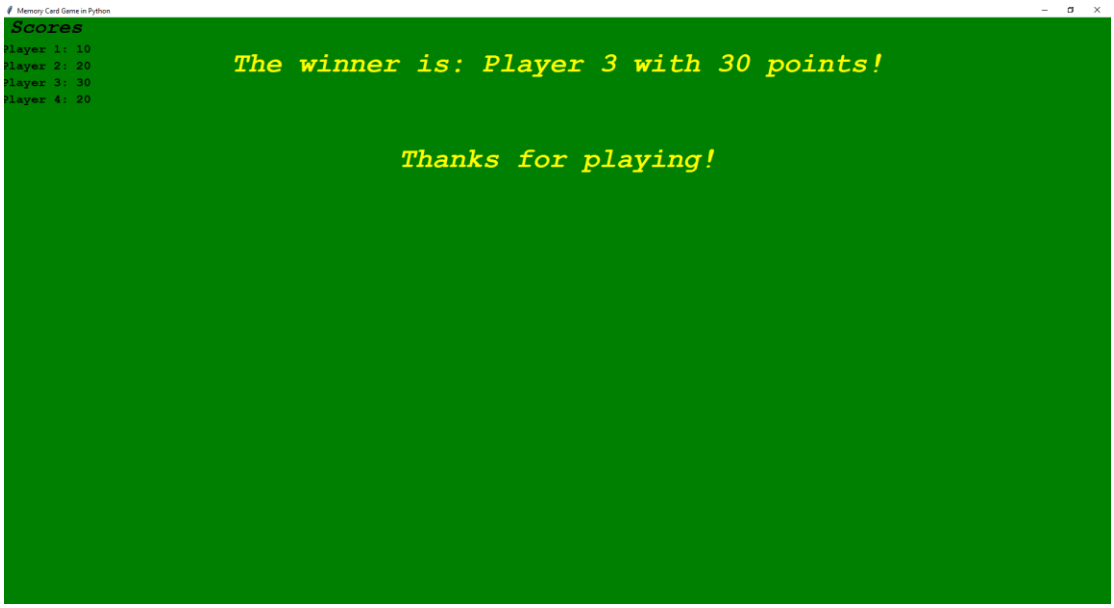
Παρατίθενται μερικά screenshots από την εκτέλεση του παιχνιδιού(Εύκολο, 4 Παίκτες):





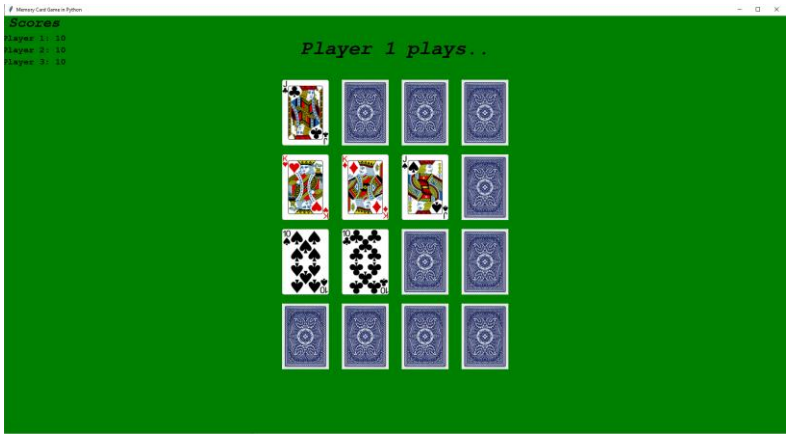

Memory Card Game in Python
Scores
Player 1: 0
Player 2: 20
Player 3: 10
Player 4: 10
MATCH!


Memory Card Game in Python
Scores
Player 1: 10
Player 2: 20
Player 3: 30
Player 4: 20
GAME OVER!



III) **Κλάση ContinueGame(NewGame):** Η κλάση ContinueGame κληρονομεί μεθόδους από την κλάση NewGame. Δέχεται ως ορίσματα το master window, και το path του αρχείου που βρίσκεται αποθηκευμένη η τελευταία κατάσταση του παιχνιδιού. Καλεί τη μέθοδο extract_data() και λαμβάνει τα δεδομένα για τον Τρέχωντα Παίκτη και το Index του, τη λίστα με τις κάρτες, τον αριθμό των ανοικτών καρτών και τον συνολικό αριθμό των καρτών, καθώς και τη δυσκολία του παιχνιδιού. Με αυτά τα δεδομένα καλεί τη μέθοδο tile_reconstruction(), η οποία ανακατασκευάζει τις κάρτες με τις κατάλληλες εικόνες και την κατάλληλη λειτουργικότητα(αναλόγως αν είναι face up ή face down) και τη μέθοδο board_reconstruction(), η οποία ανακατασκευάζει το ταμπλό, τοποθετόντας τις κάρτες στις κατάλληλες θέσεις. Εάν το τελευταίο παιχνίδι έχει ολοκληρωθεί, επανεμφανίζει τον τελευταίο νικητή και τους πόντους του.

Παρουσιάζεται screenshot από το τελευταίο παιχνίδι που είχε μείνει ανολοκλήρωτο, αφού κλικαριστεί η επιλογή «Συνέχεια..» στο αρχικό μενού:



IV) **Κλάση Player:** Πρόκειται για τον παίκτη. Ένα αντικείμενο της κλάσης παίκτης ορίζεται από τον μοναδικό του αριθμό (player_number), το όνομά του (name, string της μορφής f"Player {player_number}") το score του(που αρχικοποιείται σε 0 κατά τη δημιουργία του στιγμιοτύπου) και τις μεθόδους __str__() που επιστρέφει ένα string με το όνομα του παίκτη και την add_score(value) η οποία προσθέτει μια ακέραια τιμή στο σκορ του παίκτη.

V) **Κλάση Tile(Button):** Είναι το αντικείμενο της κάρτας. Κληρονομεί χαρακτηριστικά και μεθόδους του Button της βιβλιοθήκης tkinter. Δέχεται ως ορίσματα το rank("ace", 2, 3, 4, …, "queen", "king") και το suit("spades", "clubs", "diamonds", "hearts"). Διαθέτει τα γνωρίσματα rank, suit, back_image(η εικόνα της πίσω μεριάς της κάρτας που υπάρχει στο path "gui/Images/card_back.png"), front_image(η εικόνα της μπροστινής μεριάς της κάρτας που υπάρχει στο path f"gui/Images/{self.rank}_of_{self.suit}.png"), το width και το height που είναι 142 και 200 αντίστοιχα για όλες τις κάρτες, και το is_flipped(False όταν είναι face down, True όταν είναι face up). Διαθέτει τις μεθόδους __repr__ και __str__ που επιστρέφουν ένα string με τα στοιχεία της κάρτας και χρησιμοποιήθηκαν κυρίως για debugging, τη μέθοδο is_figure() που επιστρέφει True αν η κάρτα είναι φιγούρα, τη μέθοδο flip() η οποία αλλάζει την is_flipped κατάσταση της κάρτας και τη value που επιστρέφει την αριθμητική αξία της κάρτας ως ακέραιο αριθμό.

VI) **Κλάση GameState:** Αντικείμενο που περιέχει τις απαραίτητες πληροφορίες που χρειαζόμαστε για την ανακατασκευή του παιχνιδιού. Περιέχει τα γνωρίσματα current_player, player_index, tiles_info, players_list, open_tiles, total_tiles, difficulty. Ένα τέτοιο αντικείμενο αποθηκεύεται με τη μέθοδο του pickling ως δυαδικό αρχείο, όταν αποθηκεύουμε την κατάσταση του παιχνιδιού.

VII) **Κλάση Computer(Player):** Κληρονομεί από τον παίκτη(Player). Διαθέτει τις ίδιες μεθόδους και γνωρίσματα με τον Player, ωστόσο διαθέτει σαν επιπλέον γνώρισμα το ιστορικό καρτών(history). Πρόκειται για μια λίστα στην οποία αποθηκεύονται οι τελευταίες 5 κάρτες που αποκαλύφθηκαν και δεν υπήρξε ταίριασμα. Διαθέτει επίσης της μεθόδους memorize_tile() που αποθηκεύει μια κάρτα στη λίστα history και remove_tile που αφαιρεί μια κάρτα από το history.

Συμπεράσματα – Μελλοντικές Επεκτάσεις

Πρόκειται για μια εφαρμογή με πολλές δυνητικές επεκτάσεις, κάποιες από αυτές είναι:

- Δυνατότητα επιλογής background και μορφής καρτών από τον παίκτη, ή ακόμα και ένα implementation που ο παίκτης μπορεί να χρησιμοποιήσει δικές του εικόνες.
- Διαφορετικά επίπεδα δυσκολίας για το Single Player κομμάτι του παιχνιδιού, ως προς την ευφυία του Υπολογιστή-Αντιπάλου.
- Δημιουργία online έκδοσης, όπου το παιχνίδι παίζεται μέσω του browser, πιθανώς χρησιμοποιώντας κάποιο framework της Python.
- Δημιουργία ενός installer που μπορεί να κάνει extract τα απαραίτητα αρχεία αυτόματα στο directory της αρεσκείας του χρήστη, και δημιουργεί ένα .exe αρχείο, κλικάροντας το οποίο ο χρήστης μπορεί να τρέξει το πρόγραμμα.
- Δυνατότητα πληκτρολόγησης των ονομάτων του κάθε παίκτη και επιλογής διαφορετικού χρώματος λεζάντας, ή ακόμα και επιλογή εικόνας για την απεικόνιση του κατά τη διάρκεια του παιχνιδιού.
- Επιδιόρθωση ενός bug όπου κατά τη διάρκεια του παιχνιδιού, ένα μέρος των καρτών της πρώτης σειράς επικαλύπτεται από τα μηνύματα του παιχνιδιού, σε συστήματα που χρησιμοποιούν scaling από 150% και πάνω. Προς το παρόν, ο τρόπος να αποφευχθεί αυτό είναι να μειώσει ο χρήστης το scaling σε 125% ή λιγότερο, κάνοντας δεξί κλικ στην Επιφάνεια Εργασίας→Display Settings, στην περιοχή Scale and Layout.