Πρόγραμμα Σπουδών: **Πληροφορικής**

Θεματική Ενότητα: ΠΛΗΠΡΟ

ID Project: 54

Ακαδημαϊκό Έτος: 2022 – 2023

Δημήτριος Ματαφιάς

Περιεχόμενα

Εισαγωγή

Σαν ομάδα κληθήκαμε να γράψουμε κώδικα στη γλώσσα προγραμματισμού python ο οποίος να δημιουργεί γραφικό περιβάλλον, στο οποίο ένας χρήστης να παίζει ηλεκτρονικά το παιχνίδι της ναυμαχίας με αντίπαλο τον υπολογιστή.

Με σκοπό τη εκπόνηση της εργασίας που μας δόθηκε να φέρουμε εις πέρας αποφασίσαμε να χωρίσουμε τη λύση της εργασίας αυτής σε 4 επιμέρους τμήματα τα οποία είναι τα παρακάτω:

- Κατασκευή μίας κλάσης η οποία θα αποθηκεύει όλες τις πληροφορίες για τον κάθε παίχτη Φίλο/Εχθρό.
- Κατασκευή του γραφικού περιβάλλοντος το οποίο θα χρησιμοποιεί ο εκάστοτε παίχτης για να εισάγει τις απαραίτητες πληροφορίες καθώς και να αλληλοεπιδρά με το παιχνίδι.
- Κατασκευή ενός αλγόριθμου που θα αναπαριστά τον ακριβή τρόπο σκέψης με τον οποίο ένας άνθρωπος επιλέγει την επόμενη κίνησή του στο συγκεκριμένο παιχνίδι.
- Και τελικά την χρησιμοποίηση κατάλληλων συναρτήσεων και μεθόδων με σκοπό τη συνένωση των παραπάνω τμημάτων έτσι ώστε να δουλεύουν αρμονικά μαζί, με τον βέλτιστο δυνατό τρόπο.

Εμένα μου ανατέθηκε το 3^{ο} κομμάτι που αφορά τον αλγόριθμο του εχθρού καθώς επίσης και ο ρόλος της σύνδεσης των 3ων πρώτων κομματιών με σκοπό την επίτευξη του επιθυμητού αποτελέσματος.

Κατασκευή Αλγορίθμου Επιλογής του Εχθρού

Όπως αναφέραμε σαν ομάδα στην παρουσίαση της εργασίας, το πρώτο πράγμα που σκέφτηκα ήταν πως θα πρέπει να φτιάξω ένα πρόγραμμα που να σκέφτεται σαν να είναι κανονικός παίχτης.

Για να το πετύχω αυτό χρειάστηκε απλά να σπάσω σε απλά διαδοχικά βήματα τον τρόπο σκέψεις που κάνει κάποιος παίχτης με σκοπό να επιλέξει κάποιο πλοίο.

Αναρωτήθηκα λοιπόν ποια είναι τα βήματα που θα έκανα εγώ;

Παρακάτω ακολουθεί ένας αλγόριθμος σε ψευδοκώδικα που θα χρησιμοποιούσα εγώ για να επιλέξω την επόμενη κίνηση μου.

```
ΔΕΔΟΜΕΝΑ
        Επιλογή, Κατεύθυνση, Άξονας
        Λίστα = [Πάνω, Δεξιά, Κάτω, Αριστερά]
ΑΡΧΗ
        ΕΠΑΝΑΛΑΒΕ
                Επιλογή = Τυχαία συντεταγμένη πάνω στο ταμπλό
                ΕΑΝ (Επιλογή = Συντεταγμένη πλοίου) Τότε
                        ΕΠΑΝΑΛΑΒΕ
                                Κατεύθυνση = Το κάθε στοιχείο της λίστας "Λίστα" με τη σειρά
                                Επιλογή = Η αμέσως κοντινή συντεταγμένη προς τη δοθείσα
                                          κατεύθυνση
                                ΕΑΝ (Επιλογή = Συντεταγμένη πλοίου) ΤΟΤΕ
                                        ΕΑΝ (Κατεύθυνση = Πάνω ή Κάτω) ΤΟΤΕ
                                                Άξονας = y΄y
                                        ΑΛΛΙΩΣ
                                                Άξονας = x΄x
                                        ΕΑΝ-ΤΕΛΟΣ
                                ΕΑΝ-ΤΕΛΟΣ
                        ΜΕΧΡΙ (Επιλογή = Συντεταγμένη πλοίου)
                        ΕΠΑΝΑΛΑΒΕ
                                Εδώ επιλέγουμε συνεχόμενα κοντινές συντεταγμένες πάνω στον
                                επιλεγμένο άξονα έως να βρούμε κενό εκατέρωθεν των δύο άκρων του
                                πλοίου
                        ΕΠΑΝΑΛΑΒΕ-ΤΕΛΟΣ
                ΕΑΝ-ΤΕΛΟΣ
        ΜΕΧΡΙ (Να τελειώσει το παιχνίδι)
ΤΕΛΟΣ
```

Το πρώτο πρόβλημα που αντιμετώπισα για την μετατροπή του παραπάνω αλγορίθμου σε κώδικα σε γλώσσα Python είναι ότι ο παραπάνω κώδικας αφορά συνεχόμενη εκτέλεση ενεργειών ενώ η εργασία απαιτεί μια "turn based" αντιμετώπιση, δηλαδή ο παραπάνω κώδικας θα πρέπει να μας επιστρέφει κάθε φορά ένα αποτέλεσμα και όχι το σύνολο τον επιλογών. Έτσι δημιουργήθηκε η ανάγκη για διατήρηση πληροφοριών έτσι ώστε κάθε φορά που τρέχουμε τον κώδικα να μην ξεκινάει από την αρχή, αλλά να επιστρέφει ένα μοναδικό αποτέλεσμα, το αμέσως επόμενο σε σειρά.

Το δεύτερο πρόβλημα που αντιμετώπισα ήταν πως η παραπάνω λογική έχει ως προϋπόθεση να μην υπάρχουν πλοία που εφάπτονται μεταξύ τους, διότι στην περίπτωση που ο εχθρός ψάχνει να βρει τον άξονα του πλοίου, αν συναντήσει άλλο κομμάτι εφαπτόμενου πλοίου, θα μπερδευτεί και θα χαθεί η

συνοχή του αλγόριθμου. Αφού λοιπόν έλεγξα πως δεν υπάρχει κάποιος περιορισμός στην εκφώνηση σχετικά με τη διάταξη των πλοίων εισηγήθηκα στην ομάδα και αποφασίσαμε πως καθολικά ούτε ο παίχτης, ούτε ο εχθρός θα μπορεί να τοποθετήσει πλοία τα οποία εφάπτονται.

Κατασκευή Αλγορίθμου

Αρχική Ιδέα

Αρχικά ο εχθρός θα πρέπει να επιλέγει τυχαία συντεταγμένες. Αποφάσισα λοιπόν πως θα υπάρχει μία λίστα με όλες τις συντεταγμένες που υπάρχουν στο ταμπλό και με τη χρήση της συνάρτησης random να γίνεται επιλογή τυχαίας θέσης μέσα στο ταμπλό και τελικά με τη μέθοδο remove να αφαιρείται αυτή από τη λίστα ώστε να μην να μη μπορεί να ξανά χρησιμοποιηθεί.

Έπειτα ξεκίνησα να γράφω μία συνάρτηση την οποία ονόμασα enemy_choice και θα λειτουργούσε ως μεσάζοντας μεταξύ του προγράμματος και των συναρτήσεων επιλογής. Αυτή θα τροφοδοτεί τις συναρτήσεις με δεδομένα και σύμφωνα με τα αποτελέσματα τους θα αλλάζει και το γραφικό περιβάλλον. Η λειτουργία της συνάρτησης αυτής έχει ως εξής:
Χρησιμοποιεί μία μεταβλητή με το όνομα hit έτσι ώστε να γίνεται έλεγχος αν οι συντεταγμένες ανήκουν σε τμήμα πλοίου οπότε και παίρνει την τιμή True. Αν η μεταβλητή αυτή έχει την τιμή False τότε απλά επιλέγεται νέα τιμή στην επόμενη επανάληψη.
Στην περίπτωση που είναι True αρχικά γίνεται κλήση μίας συνάρτησης την οποία ονόμασα check_hit_validity. Αυτή η συνάρτηση επιστρέφει δύο μεταβλητές (κατεύθυνση και περάσματα). Ουσιαστικά η συνάρτηση αυτή ελέγχει αν γίνεται να χρησιμοποιηθεί η δοθείσα κατεύθυνση. Αν δεν γίνεται, δηλαδή στην περίπτωση που αν ακολουθηθεί η κατεύθυνση, θα βγούμε εκτός ορίων, ή εκτός πλοίου, τότε θα δοκιμάσει όλες τις δυνατές κατευθύνσεις με τη σειρά, μέχρι να βρει μία η οποία θα είναι έγκυρη την οποία και θα επιστρέψει. Η μεταβλητή για τα περάσματα χρησιμοποιείται για το τελευταίο κομμάτι του αλγορίθμου. Στην περίπτωση που η κατεύθυνση έχει μετακινηθεί στους άξονες πρέπει να υπάρχει ένας έλεγχος έτσι ώστε να γίνεται αναζήτηση προς τη μία κατεύθυνση του πλοίου αρχικά και έπειτα άλλη μία αναζήτηση προς την άλλη κατεύθυνση. Δηλαδή δεν πρέπει τελικά να πραγματοποιηθούν πάνω από δύο αναζητήσεις καθώς μετά τη δεύτερη αναζήτηση θεωρητικά έχει βρεθεί και το τελευταίο κομμάτι του πλοίου και ο αλγόριθμος θα πρέπει να αναζητήσει κάποια επόμενη τυχαία συντεταγμένη με σκοπό αναζήτησης νέου πλοίου. Με κάθε αλλαγή κατεύθυνσης λοιπόν στους άξονες αυξάνεται και η μεταβλητή του περάσματος κατά 1 το οποίο σηματοδοτεί και το πέρας του αλγορίθμου καθώς μετά την εκτέλεση αυτής της συνάρτησης γίνεται έλεγχος αυτής της μεταβλητής. Αν αυτή είναι πάνω από 1, δηλαδή έχουν πραγματοποιηθεί δύο διαπεράσεις, τότε αρχικοποιούνται όλες οι τιμές και ξανά ξεκινάει η τυχαία αναζήτηση από όλες τις εναπομένουσες συντεταγμένες.

Στη συνέχεια, σε περίπτωση που χτυπηθεί πλοίο, η μεταβλητή hit παίρνει την τιμή True, αρχικοποιείται η τιμή της κατεύθυνσης σε “up” και αυξάνονται κατά 1 δύο άλλες βοηθητικές μεταβλητές, η score και η counter.

Η μεταβλητή score χρησιμοποιείται για να καταγράψει τον αριθμό επιτυχών βολών. Η μεταβλητή counter χρησιμοποιείται για να καταγράφει τον αριθμό τον χτυπημάτων με σκοπό στο τέλος να γνωρίζουμε ποιο είναι το πλοίο που καταστρέψαμε, έτσι ώστε να μπορέσουμε να αφαιρέσουμε της περιμετρικές συντεταγμένες του από τη λίστα τυχαίας επιλογής, καθώς αποφασίσαμε πως τα πλοία δεν εφάπτονται.

Τέλος καλείται η συνάρτηση check_if_tile_is_ship η οποία δέχεται δεδομένα και επιστρέφει πληροφορίες σχετικά με την επόμενη θέση που θα εξεταστεί, την επόμενη κατεύθυνση, αλλαγές που έγιναν στον πίνακα και τελικά πληροφορίες σχετικά με το χρώμα καθώς και το αντικείμενο του γραφικού περιβάλλοντος που θα αλλάξει.

Στη συνέχεια θα αναλύσω τον τρόπο λειτουργείας και τη σκέψη πίσω από τις δύο βασικές συναρτήσεις επιλογής του εχθρού check_hit_validity και check_if_tile_is_ship.

Η συνάρτηση check_hit_validity

Η συνάρτηση αυτή δέχεται τα εξής ορίσματα:

- position = x, y μέσα σε [] --> [x, y]
- direction = ένα από όλα τα παρακάτω:
  Σε περίπτωση που δεν έχουμε βρει την κατεύθυνση του πλοίου:
  ("up", "right", "down", "left")
  Σε περίπτωση που το πλοίο είναι τοποθετημένο κατακόρυφα:
  ("y_up", "y_down")
  Σε περίπτωση που το πλοίο είναι τοποθετημένο οριζόντια:
  ("x_right", "x_left")
- random_choice_list = Μία λίστα η οποία περιέχει όλες τις πιθανές εναπομένουσες επιλογές του εχθρού ως προς την επιλογή της επόμενης θέσεις που θέλει να χτυπήσει.
- passes = Μία μεταβλητή η οποία δείχνει πόσες διελεύσεις έχουν γίνει κατά μήκος ενός πλοίου. (Αν το passes γίνει > 1, δηλαδή έχουμε ψάξει για κομμάτια και στις δύο κατευθύνσεις, τότε το πλοίο έχει καταστραφεί.)
- rows = Ο αριθμός των γραμμών του πίνακα.
- columns = Ο αριθμός των στηλών του πίνακα.
- enemy_set_up_table = Το ταμπλό του αντιπάλου από αυτόν που κάνει την επιλογή το οποίο περιέχει τις θέσεις όλων των πλοίων συμβολισμένες με "*".

Αρχικά στη συνάρτηση αυτή δημιουργούνται οι εξής λίστες:

directions = [["up", [-1, 0]], ["right", [0, 1]], ["down", [1, 0]], ["left", [0, -1]]]
y_directions = [["y_up", [-1, 0]], ["y_down", [1, 0]]]
x_directions = [["x_right", [0, 1]], ["x_left", [0, -1]]]

Η κάθε λίστα περιέχει στοιχεία που αντιπροσωπεύουν και αυτά λίστες όπου σε αυτές το πρώτο στοιχείο αποτελείται από μία κατεύθυνση και το δεύτερο στοιχείο από μία λίστα με τους αριθμούς που πρέπει να προστεθούν στο x και στο y με σκοπό να λάβουμε τις συντεταγμένες που υποδεικνύει η κατεύθυνση.

Έπειτα γίνεται έλεγχος ως προς την δοθείσα κατεύθυνση. Στην πρώτη περίπτωση εξετάζεται απλά συνεχώς η επόμενη θέση αν υπάρχει μέσα στη λίστα με τις τυχαίες επιλογές (αν δεν έχει γίνει δηλαδή ήδη επεξεργασία της). Αν δεν υπάρχει προχωράμε στην επόμενη κατεύθυνση και ξανά πραγματοποιείται ο έλεγχος.

Σε όλες τις υπόλοιπες περιπτώσεις εξετάζεται αρχικά η πρώτη κατεύθυνση του άξονα. Αν η θέση δεν υπάρχει στη λίστα πιθανών επιλογών η μεταβλητή περάσματος αυξάνεται κατά 1 που υποδηλώνει πως πραγματοποιήθηκε έλεγχος του πλοίου προς τη μία κατεύθυνση του άξονα και εξετάζονται όλες οι θέσεις

της αντίθετης κατεύθυνσης μέχρι να βγούμε εκτός των ορίων του πίνακα ή να συναντήσουμε μία περιμετρική θέση πλοίου η οποία δεν υπάρχει στη λίστα τυχαίων επιλογών. Τότε η μεταβλητή περάσματος αυξάνεται κατά 1 που υποδηλώνει ότι πραγματοποιήθηκε πέρασμα και προς την άλλη κατεύθυνση, δηλαδή το πλοίο έχει καταστραφεί.

Τέλος η συνάρτηση αυτή επιστρέφει τη διαθέσιμη κατεύθυνση καθώς και τον αριθμό περασμάτων.

Η συνάρτηση check_if_tile_is_ship

Η συνάρτηση αυτή επεξεργάζεται μία θέση (η οποία ξέρουμε πως είναι ορθή, δηλαδή δεν αποτελεί σφάλμα τύπου εκτός ορίων κλπ.) και σύμφωνα το αν υπάρχει πλοίο ή όχι, επιστρέφει πληροφορίες για την αλλαγή του γραφικού περιβάλλοντος καθώς και πληροφορίες που θα χρειαστούν για τον αμέσως επόμενο έλεγχο.

Η συνάρτηση αυτή δέχεται τα εξής ορίσματα:

- position = x, y μέσα σε [] --> [x, y]
- direction = ένα από όλα τα παρακάτω:

  Σε περίπτωση που δεν έχουμε βρει την κατεύθυνση του πλοίου:

  ("up", "right", "down", "left")

  Σε περίπτωση που το πλοίο είναι τοποθετημένο κατακόρυφα:

  ("y_up", "y_down")

  Σε περίπτωση που το πλοίο είναι τοποθετημένο οριζόντια:

  ("x_right", "x_left")
- random_choice_list = Μία λίστα η οποία περιέχει όλες τις πιθανές εναπομένουσες επιλογές του εχθρού ως προς την επιλογή της επόμενης θέσεις που θέλει να χτυπήσει.
- enemy_set_up_table = Το ταμπλό του αντιπάλου από αυτόν που κάνει την επιλογή το οποίο περιέχει τις θέσεις όλων των πλοίων συμβολισμένες με "*"

Αρχικά κάνει ένα αντίγραφο τις θέσης. Ελέγχει τη θέση που υποδεικνύει η δοθείσα κατεύθυνση.
Αν στη θέση αυτή βρίσκεται τμήμα πλοίου τότε επιστρέφει τη νέα θέση καθώς και το χρώμα κόκκινο.
Επίσης αν η κατεύθυνση είναι μία από τις “up”, “right”, “down”, “left” τότε περνάει τον κυκλικό έλεγχο σε έλεγχο αξόνων.
Αν στη θέση αυτή δεν βρίσκεται τμήμα πλοίου τότε επιστρέφει το χρώμα άσπρο.
Τέλος είτε υπάρχει ή δεν υπάρχει πλοίο, θα γίνει αφαίρεση της θέσης η οποία υποβλήθηκε σε επεξεργασία από τη λίστα τυχαίων επιλογών έτσι ώστε να μην υπάρχει περίπτωση να επιλεχθεί εκ νέου από τον εχθρό.

Κατασκευή του Τελικού Προγράμματος

Για την κατασκευή του τελικού προγράμματος θεώρησα ορθό την κατασκευή ενός αρχείου το οποίο θα περιέχει πληροφορίες και θα ενώνει τα 3 αρχεία που κατασκεύασε το κάθε μέλος της ομάδας.

Αρχικά το αρχείο αυτό καλεί όλες τις βιβλιοθήκες που χρειάζονται για την εκτέλεση του προγράμματος καθώς και τα αρχεία των άλλων μελών.

Έπειτα καλεί τη συνάρτηση random.seed(time.time( )) η οποία αρχικοποιεί κάθε φορά διαφορετικό seed έτσι ώστε οι τυχαίες επιλογές με κάθε έναρξη του παιχνιδιού να είναι διαφορετικές.

Η Συνάρτηση center_window

Αρχικά, από τη στιγμή που έχουμε να κάνουμε με διάφορα παράθυρα τα οποία εμφανίζονται και εξαφανίζονται σκεφτικά ότι χρειαζόμαστε μία συνάρτηση η οποία θα δέχεται ένα παράθυρο και πριν το εμφανίσει θα εκτελούνται όποιες αλλαγές έχουν γίνει στα αντικείμενά του, και έπειτα θα κεντράρεται στην οθόνη. Έτσι κάθε φορά που θα καλούμε ένα παράθυρο, πριν την κλήση του θα καλούμε τη συνάρτηση center_window.

Η Συνάρτηση create_ship_button

Είναι μία συνάρτηση η οποία ανοίγει ένα καινούριο παράθυρο και ελέγχει όλα τα πλοία, αν έχουν τοποθετηθεί ή όχι. Σε περίπτωση που κάποιο από αυτά έχει ήδη τοποθετηθεί, τότε το κουμπί για την εισαγωγή του θα είναι απενεργοποιημένο, έτσι ώστε ο χρήστης να μη μπορεί να επιλέξει ένα πλοίο το οποίο έχει ήδη επιλεγεί.

Ο έλεγχος αυτός γίνεται από τα λεξικά των πλοίων στο αντικείμενο friend, αν η λίστα με τη λέξη κλειδί “position” δεν είναι κενή.

Η Συνάρτηση choose_ship_length

Η συνάρτηση αυτή έχει δοθεί σε κάθε ένα από τα κουμπιά του προηγούμενου παραθύρου, με διαφορετικό όρισμα length για κάθε κουμπί.

Αρχικά με την επιλογή ενός κουμπιού, αποθηκεύει το όρισμα length σε μία μεταβλητή αντικειμένου και έπειτα αλλάζει τη συνάρτηση που έχει τεθεί σε όλα τα κουμπιά που απαρτίζουν το ταμπλό του παιχνιδιού.

Η συγκεκριμένη δουλειά αποτέλεσε εμπόδιο αρχικά καθώς, όταν προσπαθούσα να ορίσω μία συνάρτηση που είχα γράψει με απλή επανάληψη για διαφορετικά x και y για το κάθε κουμπί, η συνάρτηση αυτή δεν κρατούσε ποτέ διαφορετικά x, y αλλά εκτελούνταν μόνο οι πληροφορίες της τελευταίας επανάληψης, δηλαδή (9, 10).

Η πρώτη λύση που σκέφτηκα ήταν να δώσω σε κάθε ένα από τα κουμπιά ξεχωριστά από μία συνάρτηση με x και y τη θέση που έχει το κάθε κουμπί στο ταμπλό. Αυτό όμως δεν θα ήταν προγραμματιστικά κομψό, ούτε χρονικά αποδοτικό.

Για την αντιμετώπιση σκέφτηκα ως εξής. Κάπου στην ύλη έγραφε πως το όνομα μίας συνάρτησης ουσιαστικά είναι ένας δείκτης που δείχνει πού είναι αποθηκευμένη αυτή μέσα στη μνήμη. Ακόμη, μάθαμε πως η εμβέλεια μίας συνάρτηση που βρίσκεται μέσα σε μία άλλη είναι μεγαλύτερη, δηλαδή μία συνάρτηση η οποία ορίζεται ή εκτελείται μέσα σε μία άλλη μπορεί να δει και να χρησιμοποιήσει

εξωτερικές μεταβλητές χωρίς να της δοθούν ως ορίσματα. Επομένως θα μπορούσα να φτιάξω μία συνάρτηση, η οποία θα δέχεται κάποια ορίσματα και μέσα της θα ορίζεται μία άλλη συνάρτηση η οποία δεν θα δέχεται ορίσματα. Αυτή θα εκτελεί τις ενέργειες που θέλουμε χρησιμοποιώντας τα ορίσματα της εξωτερικής συνάρτησης. Τέλος η εξωτερική συνάρτηση θα επιστρέφει την εσωτερική, η οποία όταν κληθεί πλέον θα εκτελεί την ενέργειες που της έχουν οριστεί, έχοντας αποθηκευμένες τις διάφορες τιμές που δόθηκαν στη συνάρτηση που την επέστρεψε.

Έτσι κάθε φορά που θα τρέχουμε την επανάληψη για κάθε διαφορετικό κουμπί του πίνακα, θα καλούμε και την εξωτερική συνάρτηση με διαφορετικά x και y, και θα δίνουμε σε κάθε κουμπί την επιστρεφόμενη συνάρτηση χωρίς ορίσματα, καθώς θα έχει κάθε φορά αποθηκευμένα τα διαφορετικά x και y για το κάθε κουμπί.

Τέλος η εσωτερική συνάρτηση αυτή την οποία ονομάσαμε και choose_ship_head θα ελέγχει όλες τις κατευθύνσεις από το x και το y της για το αποθηκευμένο μήκος current_length και αν το πλοίο συναντήσει κάποιο εμπόδιο, θα απενεργοποιεί το κουμπί προς τη συγκεκριμένη κατεύθυνση πριν εμφανιστή το παράθυρο.

Η Συνάρτηση cancel_direction

Η συνάρτηση αυτή απλά εξαφανίζει το παράθυρο και επιστρέφει τον έλεγχο στο ταμπλό σε περίπτωση που ο χρήστης δεν επιθυμεί τελικά να τοποθετήσει το πλοίο αυτό. Έχει δοθεί στο κουμπί «Back» του προηγούμενου παραθύρου.

Η Συνάρτηση choose_ship_direction

Η συνάρτηση αυτή έχει δοθεί σε κάθε κουμπί του προηγούμενου παραθύρου για τις διαφορετικές κατευθύνσεις “up”, “right”, “down”, “left” κυκλικά αντίστοιχα. Αρχικά χρησιμοποιεί τις μεθόδους της κλάσεις για να ενημερώσει τον πίνακα is_ocupied σχετικά με τη θέση του πλοίου που τοποθετήθηκε και τελικά χρησιμοποιεί τα λεξικά των πλοίων έτσι ώστε να απενεργοποιήσει τα κουμπιά που καταλαμβάνει το τοποθετημένο πλοίο στο ταμπλό, όπως επίσης και το χώρο περιμετρικά από αυτό καθώς ένα πλοίο δεν πρέπει να ξεκινάει κοντά σε άλλο πλοίο.

Η Συναρτήσεις remove_ship και remove

Η συνάρτηση remove_ship κάνει την ίδια δουλειά που κάνει και η πιο πάνω συνάρτηση create_ship_button απλά με την αντίθετη δουλειά. Πιο συγκεκριμένα, απενεργοποιεί τα κουμπιά του παραθύρου , αν η λίστα με τη λέξη κλειδί “position” είναι κενή.

Η συνάρτηση remove έχει δοθεί σε κάθε κουμπί του παραθύρου που αφαιρεί πλοία για συγκεκριμένο πλοίο και αρχικά αδειάζει το λεξικό του εκάστοτε επιλεγμένου πλοίου και τελικά επαναφέρει το ταμπλό κάνοντας τις αλλαγές που χρειάζεται και να είναι οπτικά ορθό μετά την αφαίρεση του πλοίου.

Η Συνάρτηση check_buttons

Κατά τις δοκιμές λειτουργίας του προγράμματος παρουσιάστηκε το εξής θέμα. Αν είχαν τοποθετηθεί πλοία, το κουμπί εισαγωγής πλοίου ήταν άχρηστο και όταν γινόταν επιλογή του όλα τα πλοία ήταν απενεργοποιημένα. Ομοίως με το κουμπί της αφαίρεσης. Επιπλέον το κουμπί της έναρξης εμφάνιζε σφάλμα αν δεν είχαν οριστεί όλα τα πλοία. Έτσι έκρινα αναγκαίο τον ορισμό και χρήση μίας συνάρτηση

κάθε φορά που θα επιστρέφουμε από τα παράθυρα στο αρχικό ταμπλό, η οποία θα ελέγχει και θα απενεργοποιεί τα κουμπιά αυτά, με τις κατάλληλες προϋποθέσεις.

- Για το κουμπί εισαγωγής, αν έχουν τοποθετηθεί όλα τα πλοία.
- Για το κουμπί της αφαίρεσης, αν δεν έχει τοποθετηθεί κανένα πλοίο.
- Για το κουμπί της έναρξης αν δεν έχουν τοποθετηθεί όλα τα πλοία.

Η Συνάρτηση new_game

Η συνάρτηση αυτή χρησιμοποιεί τη μέθοδο reset για τα επαναφέρει τα αντικείμενα στην αρχική τους κατάσταση. Επιπλέον επαναφέρει το ταμπλό και καλεί το παράθυρο ονόματος παίχτη για να ξεκινήσει καινούριο παιχνίδι.

Η Συνάρτηση start_game

Η συνάρτηση αυτή αρχικά δέχεται ένα όρισμα print_ships για λόγους επαλήθευσης και ελέγχου. Το όρισμα αυτό δέχεται τιμές True ή False και ανάλογα με την τιμή, με την κλήση της συνάρτησης εκτυπώνει τη φίλια και την εχθρική διάταξη, καθώς και τις συντεταγμένες των πλοίων τους και της περιοχής γύρω από αυτά.

Με την κλήση της η συνάρτηση αυτή αρχικά απενεργοποιεί όλα τα κουμπιά εισαγωγής και αφαίρεσης.

Έπειτα εκτελεί έναν αλγόριθμο με τον οποίο ο εχθρός τοποθετεί τυχαία πλοία στο ταμπλό του.

Ο αλγόριθμος αυτός δουλεύει ως εξής:

Αρχικά δημιουργείται μία λίστα που περιέχει κάθε πλοίο. Έπειτα εκτελείται μία επανάληψη όσο η προηγούμενη λίστα έχει κάτι μέσα. Μέσα στην επανάληψη, δημιουργείται μία λίστα με τις 4 πιθανές κατευθύνσεις και γίνεται τυχαία επιλογή ενός πλοίου καθώς και μίας τυχαίας θέσης μέσα στο ταμπλό. Στη συνέχεια, εξετάζεται κάθε κατεύθυνση από τη λίστα και σε περίπτωση που το πλοίο τοποθετηθεί, αφαιρείται αυτό από τη λίστα, αλλιώς αν τελειώσουν οι κατευθύνσεις επιλέγεται πλοίο από την αρχή. Η διαδικασία αυτή εκτελείται μέχρι να τοποθετηθούν όλα τα πλοία.

Τέλος η συνάρτηση, όμοια με τη συνάρτηση choose_ship_length, δίνει σε κάθε κουμπί μία επιστρεφόμενη συνάρτηση χωρίς ορίσματα, για διαφορετικά x και y για το κάθε κουμπί. Τη συνάρτηση αυτή την έχουμε ονομάσει player_choice και αρχικά ελέγχει αν οι συντεταγμένες του κουμπιού που επέλεξε ο χρήστης αντιστοιχούν σε κάποιο από τα πλοία του εχθρού. Στη συνέχεια εκτελεί τη συνάρτηση enemy_choice που έχουμε αναφέρει στον αλγόριθμο του εχθρού.

Τελικά ελέγχει τη συνθήκη λήξης του παιχνιδιού η οποία είναι αν κάποιος από τους δύο παίχτες έχει 14 επιτυχημένες βολές, ο οποίος είναι τελικά και ο νικητής.

Η Συνάρτηση end_game

Η συνάρτηση αυτή απλά εμφανίζει το παράθυρο λήξης του παιχνιδιού.

Βιβλιογραφία

Ο κώδικας της εργασίας δεν έχει αντιγραφεί από πουθενά και είναι αποκλειστικά δική μας δημιουργία. Οι γνώσει που χρειαστήκαμε για την κατασκευή του προγράμματος προήλθαν αποκλειστικά από το διδακτικό υλικό που παρέχει το ΕΑΠ το οποίο περιλαμβάνει τα εξής βιβλία:

1. Μανής, Γ., 2015. Εισαγωγή στον Προγραμματισμό με αρωγό τη γλώσσα Python. Αθήνα: Σύνδεσμος Ελληνικών Ακαδημαϊκών Βιβλιοθηκών.
   Διαθέσιμο στο σύνδεσμο: http://hdl.handle.net/11419/2745

2. Μαγκούτης, Κ., Νικολάου, Χ., 2015. Εισαγωγή στον αντικειμενοστραφή προγραμματισμό με Python. Αθήνα: Σύνδεσμος Ελληνικών Ακαδημαϊκών Βιβλιοθηκών.
   Διαθέσιμο στο σύνδεσμο: http://hdl.handle.net/11419/1708

3. Αγγελιδάκης, Ν.Α., 2015. Εισαγωγή στον προγραμματισμό με την Python, Διαθέσιμο στο σύνδεσμο: http://aggelid.mysch.gr/pythonbook/

4. Swaroop, C. H. 2013. A Byte of Python, Επιμ. μετάφρασης Ubuntu community,
   Διαθέσιμο στο συνδέσμο: http://dide.flo.sch.gr/Plinet/Meetings/Meeting23/A_Byte_of_Python-el.pdf

   Πρωτότυπο στο σύνδεσμο: https://open.umn.edu/opentextbooks/textbooks/581