Πανεπιστήμιο Πατρών

Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Εισαγωγή στους Υπολογιστές (ECE_Y106)

Υπεύθυνος Καθηγητής: Βασίλης Παλιουράς

Αυτόματη πλοήγηση virtual οχήματος με Python

Ομάδα 15

Χρήστος Καρράς

Γεώργιος Κουνάβης

Ηλίας Κρίγγος

Γεώργιος-Χρήστος Κωστάκης

Περίληψη

Κληθήκαμε να επιλύσουμε το πρόβλημα της πλοήγησης ενός οχήματος από ένα αρχικό σημείο σε ένα στόχο, σε ένα επίπεδο που περιέχει εμπόδια σε τυχαία διάταξη. Οι παράμετροι που ορίζουν την διάταξη του επιπέδου καθορίζονται από τον χρήστη, οδηγώντας σε άπειρο αριθμό πιθανών επιπέδων. Για την αντιμετώπιση του προβλήματος αυτού σχεδιάσαμε έναν αλγόριθμο που βασίζεται στην τελική απόσταση από τον στόχο για να φέρει το όχημα σε αυτόν χαράσσοντας την συντομότερη δυνατή πορεία. Μετέπειτα προστέθηκε η δυνατότητα ο χρήστης να μπορεί να εισάγει τα στοιχεία που απαιτούνται για την έναρξη του προγράμματος μέσω γραφικής διεπαφής και έγινε χρήση του αλγορίθμου A* για την εύρεση της βέλτιστης διαδρομής. Συνολικά με την κάθε σταδιακή αλλαγή προέκυψαν 3 διαφορετικές εκδόσεις του προγράμματος (v1, v1.1, v2), με την έκδοση 2 να αντικαθιστά τον δικό μας αλγόριθμο επίλυσης και να χρησιμοποιεί τον αλγόριθμο A*.

Σε αυτή την έκθεση αναλύεται αρχικά ο αλγόριθμος που δημιουργήσαμε, ενώ στη συνέχεια γίνεται αναφορά στις διάφορες βελτιώσεις του προγράμματος και παρουσιάζεται το τελικό αποτέλεσμα που δημιουργήθηκε. Στο τέλος της έκθεσης παρουσιάζεται και ο τρόπος που συνέβαλε το κάθε μέλος της ομάδας για την ολοκλήρωση της εργασίας.

**Όλα τα αρχεία βρίσκονται στον σύνδεσμο github.com/EliaKr/upatras-autonavigation*

Περιεχόμενα

1 Παρουσίαση του προβλήματος

Το πρόβλημα αφορά ένα εικονικό όχημα το οποίο πρέπει να πλοηγηθεί αυτόνομα από την αρχική του θέση έως την θέση τερματισμού του. Η πλοήγηση αυτή γίνεται σε ένα επίπεδο με τυχαία τοποθετημένα εμπόδια σε όλη την επιφάνεια του. Η αρχική και τελική θέση του οχήματος πρέπει να μπορούν να είναι οποιαδήποτε σημεία του επιπέδου που δεν είναι εμπόδια. Θα πρέπει να υπάρχει πρόβλεψη σε περίπτωση που το όχημα δεν είναι δυνατόν να χαράξει πορεία προς τον στόχο (π.χ. λόγω εμποδίων που αποκλείουν την πρόσβαση στο επίπεδο πέραν ενός σημείου).

2 Τρόπος επίλυσης (Λογική)

Για την επίλυση του προβλήματος χρειάστηκε ο χωρισμός του σε άλλα, μικρότερης έκτασης έτσι ώστε να είναι πιο εύκολα διαχειρίσιμο. Πριν την υλοποίηση σε κώδικα έγινε μία ανάλυση των επιμέρους στοιχείων για τα οποία έπρεπε να βρεθεί ένας αλγόριθμος επίλυσης και δημιουργήθηκε ένας τρόπος με τον οποίο το κάθε ένα απο αυτά θα μπορούσε να αντιμετωπιστεί επιτυχώς.

2.1 Δημιουργία επίπεδου με τυχαία τοποθετημένα εμπόδια

Για την αντιμετώπιση του κυρίως προβλήματος είναι προαπαιτούμενο να είναι γνωστή η διάταξη του επιπέδου στο οποίο θα κινείται το όχημα. Αρχικά δηλαδή είναι αναγκαία η διαμόρφωση ενός επιπέδου με στοιχεία του τα εμπόδια που θα πρέπει να αποφύγει το όχημα, τυχαία τοποθετημένα σε όλη την επιφάνεια του.

Για την διαμόρφωσή του επιπέδου πρέπει να λαμβάνονται τα στοιχεία που θα το χαρακτηρίζουν ως είσοδος από τον χρήστη (διαστάσεις, πυκνότητα εμποδίων) και να γίνεται τυχαία η τοποθέτηση των εμποδίων. Μετά την εισαγωγή των στοιχείων από τον χρήστη θα πρέπει να γίνεται η δημιουργία του επιπέδου σε μορφή που να είναι αναγνώσιμη από ένα πρόγραμμα, αλλά και να μπορεί να χρησιμοποιηθεί για γραφική αναπαράσταση στον χρήστη αργότερα.

Συνεπώς η δημιουργία επιπέδου χωρίστηκε σε 3 επιμέρους ζητήματα:

1. Εισαγωγή παραμέτρων επιπέδου από χρήστη
2. Δημιουργία μηχανικά αναγνώσιμης μορφής επιπέδου
3. Δημιουργία μορφής επιπέδου για γραφική αναπαράσταση

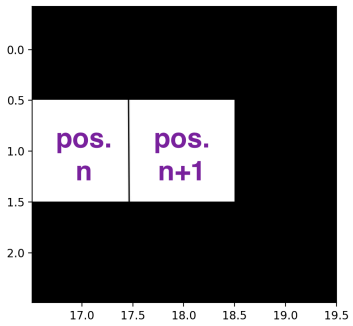
2.2 Εύρεση βέλτιστης διαδρομής και αναπαράσταση στο επίπεδο

Για την εύρεση της βέλτιστης διαδρομής το πρόγραμμα γνωρίζει την θέση του στόχου και κινεί το όχημα συνεχώς προς αυτόν αποφεύγοντας όποιο εμπόδιο συναντήσει στο δρόμο του. Για την κίνησή του το όχημα έχει 8 πιθανές θέσεις, οι οποίες είναι στα σημεία που το περιβάλλουν. Το όχημα ελέγχει για εμπόδια τα περιβάλλοντα στοιχεία και αποκλείει όλες τις κινήσεις που το οδηγούν πάνω σε εμπόδιο. Ο υπολογισμός της βέλτιστης κίνησης γίνεται με μέτρηση των αποστάσεων από τον στόχο όλων των πιθανών τελικών θέσεων μετά την κίνηση και την σύγκρισή μεταξύ τους για επιλογή αυτής που θα φέρει το όχημα πιο κοντά στον στόχο. Σε περίπτωση που δύο πιθανές κινήσεις φέρνουν το όχημα στην ίδια απόσταση από τον στόχο τότε το πρόγραμμα ελέγχει τον τρόπο που θα κινηθεί το όχημα μετέπειτα σε κάθε μία θέση και επιλέγει την βέλτιστη διαδρομή. Σε περίπτωση που υπάρχει εμπόδιο στην θέση της βέλτιστης κίνησης το όχημα επιλέγει την επόμενη βέλτιστη στην οποία δεν υπάρχει εμπόδιο. Η κάθε θέση από την οποία περνάει καταγράφεται για να χρησιμοποιηθεί εν τέλει στην γραφική αναπαράσταση της διαδρομής, αλλά και για τον υπολογισμό της απόστασης που διένυσε.

Επαναλαμβάνοντας συνεχώς τη διαδικασία αυτή μέχρι το όχημα να φτάσει στο στόχο επιτυγχάνεται η επίλυση του προβλήματος.

2.2.1 Η περίπτωση που το όχημα "παγιδεύεται"

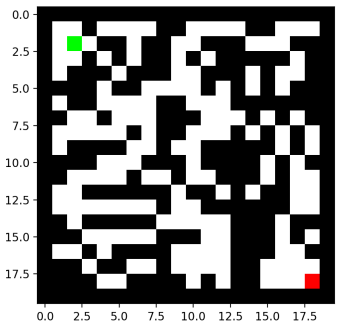
Υπάρχει περίπτωση η θέση στην οποία θα κινηθεί το όχημα να το οδηγεί σε μια κατάσταση όπου είναι παγιδευμένο και δεν μένει άλλη επιλογή από το να κινηθεί προς τα πίσω. Μία τέτοια περίπτωση οδηγεί σε μια συνεχή επανάληψη των ίδιων κινήσεων και δεν είναι δυνατή η επίλυση του επιπέδου. Συνεπώς είναι αναγκαίος ο έλεγχος πριν από κάθε κίνηση του οχήματος για να αποφευχθεί μια τέτοια περίπτωση.



Εικόνα 2.1 Παράδειγμα "παγιδευμένου" οχήματος

2.2.2 Η περίπτωση της μη επιλύσιμης διαδρομής

Μία ακόμα πιθανότητα είναι να μην υπάρχει διαδρομή μέσω της οποίας επιτυγχάνεται ο στόχος. Στην περίπτωση αυτή το πρόγραμμα εντοπίζει την αδυναμία στην επίλυση όταν μετά από επανάληψη πολλαπλών κινήσεων δεν σημειώνει πρόοδο. Ο χρήστης ενημερώνεται μέσω της γραφικής διεπαφής και το πρόγραμμα τερματίζει.

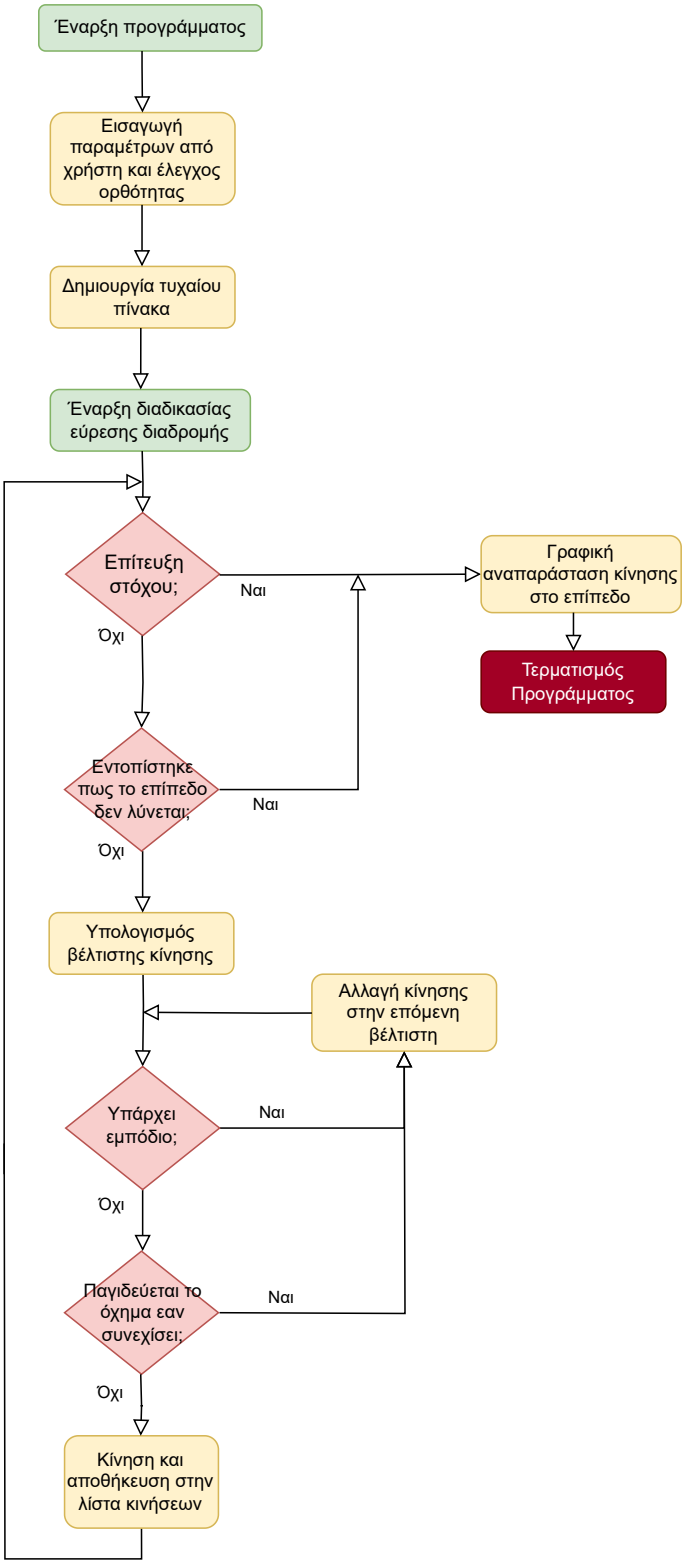


Εικόνα 2.2 Παράδειγμα αδύνατου στη λύση επιπέδου

2.3 Τερματισμός επίλυσης και γραφική αναπαράσταση διαδρομής

Η διαδικασία επίλυσης σταματά όταν επιτευχθεί ο στόχος. Τότε εμφανίζεται η διαδρομή που έχει διανύσει το όχημα και χρήσιμες πληροφορίες όπως απόσταση που διανύθηκε και χρόνος εκτέλεσης.

2.4 Αναπαράσταση σε διάγραμμα ροής



3 Υλοποίηση τρόπου επίλυσης σε Python (v1)

Μετά την ολοκλήρωση του λογικού τρόπου επίλυσης άρχισε η συγγραφή του κώδικα σε Python για δημιουργία του προγράμματος που χρησιμοποιεί ο τελικός χρήστης. Ο κώδικας οργανώθηκε σε επιμέρους συναρτήσεις με βάση την λογική που αναλύθηκε στην προηγούμενη ενότητα.

3.1 Δημιουργία επίπεδου με τυχαία τοποθετημένα εμπόδια

3.1.1 Δημιουργία πίνακα x*y με τα στοιχεία του επιπέδου

Αρχικά ο χρήστης καλείται να ορίσει τις παραμέτρους που θα χαρακτηρίσουν το επίπεδο. Οι παράμετροι αυτές είναι οι διαστάσεις του πίνακα, τα σημεία εκκίνησης και τερματισμού και η πυκνότητα των εμποδίων. Η διαδικασία λήψης των παραμέτρων από τον χρήστη γίνεται με τη χρήση της συνάρτησης initialise() η οποία επιστρέφει τις τιμές που εισήγαγε ο χρήστης.

Αφού οριστούν οι παράμετροι αυτές δημιουργείται ένας πίνακας με τα χαρακτηριστικά που έδωσε ο χρήστης. Ο πίνακας μπορεί να διαβαστεί από το πρόγραμμα έτσι ώστε να εντοπιστούν τα σημεία που περιέχουν εμπόδια, η θέση έναρξης και η θέση τερματισμού. Αυτό επιτυγχάνεται με την αναπαράσταση των ελεύθερων σημείων με το στοιχείο 1, των εμποδίων με 0, της θέσης εκκίνησης με το string 'start' και την θέσης τερματισμού με το 'end'. Ο πίνακας περιβάλλεται από τοίχο εμποδίων για την διατήρηση του οχήματος εντός του επιπέδου. Σε περίπτωση που ο πίνακας έχει κάποιο χαρακτηριστικό που κάνει αδύνατη την κίνηση του οχήματος αρχικά ή την αρχική τοποθέτηση του η δημιουργία του πίνακα επαναλαμβάνεται συνεχώς με νέα τυχαία διάταξη έως ότου πληρούνται όλες οι προϋποθέσεις για την έναρξη της επίλυσης. Η διαδικασία αυτή της δημιουργίας του πίνακα γίνεται με την συνάρτηση gen_matrix() που παίρνοντας ως εισόδους τις παραμέτρους που ορίστηκαν από το χρήστη δημιουργεί και μορφοποιεί τον πίνακα.

3.1.2 Δημιουργία πίνακα κατάλληλου για χρήση σε γραφική αναπαράσταση

Για την γραφική αναπαράσταση με το πέρας της διαδικασίας εύρεσης της διαδρομής καλείται η συνάρτηση gen_layout() η οποία αντικαθιστά με τιμές RGB τις θέσεις του επιπέδου ανάλογα με το περιεχόμενό τους. Τα ελεύθερα σημεία του επιπέδου περιέχουν τις τιμές που αντιστοιχούν στο λευκό χρώμα, τα εμπόδια τις τιμές που αντιστοιχούν στο μαύρο, ενώ οι θέσεις εκκίνησης και τερματισμού περιέχουν το πράσινο και κόκκινο αντίστοιχα. Η έξοδος της συνάρτησης αυτής θα χρησιμοποιηθεί στο τέλος της επίλυσης για σχεδίαση της διαδρομής με την Matplotlib[1].

3.2 Εύρεση βέλτιστης διαδρομής και αναπαράσταση στο επίπεδο

3.2.1 Έλεγχος πιθανών θέσεων για εμπόδια

Ο έλεγχος για εμπόδια ή κενές θέσεις περιμετρικά του οχήματος επιτυγχάνεται με την συνάρτηση detect_poss() η οποία σε κάθε θέση που βρίσκεται το όχημα επιστρέφει μια λίστα με τις συντεταγμένες των διαθέσιμων σημείων στα οποία μπορεί να μεταβεί. Για τον έλεγχο των εμποδίων χρησιμοποιήθηκε και μια άλλη συνάρτηση η check_blockage() η οποία ελέγχει εάν η θέση στην οποία θα οδηγηθεί το όχημα το παγιδεύει (βλ. 2.2.1). Σε αυτή την περίπτωση η θέση αυτή παραλείπεται και το όχημα συνεχίζει στην επόμενη βέλτιστη.

3.2.2 Εύρεση επιθυμητής κατεύθυνσης κίνησης και κίνηση προς τον στόχο

Με την συνάρτηση find_path() υπολογίζονται και ταξινομούνται κατά αύξουσα σειρά οι αποστάσεις από τα πιθανά σημεία που επιστρέφει η detect_poss() σε σχέση με τον στόχο σε ένα λεξικό της μορφής (x,y) : απόσταση. Στην συνέχεια αν προκύψει ότι το λεξικό αυτό περιέχει δυο ή περισσότερα σημεία τα οποία ισαπέχουν από τον στόχο μέσω της same_distances() δημιουργείται ένα λεξικό της μορφής κοινή απόσταση:(x1,y1),(x2,y2).

Το λεξικό αυτό εισάγεται στην bsdr() (best_same_distance_route) και επιστρέφεται το σημείο που δεν πρέπει να μεταβεί το όχημα. Η κύρια συνάρτηση move() αλλάζει τη θέση του οχήματος επιλέγοντας το σημείο με την μικρότερη απόσταση από το τελικό λεξικό θέσεων που διαμορφώθηκε. Η όλη διαδικασία επαναλαμβάνεται και η κάθε κίνηση του οχήματος καταγράφεται σε μια λίστα (done_moves) έως ότου επιτευχθεί ο στόχος. Εάν παρατηρηθεί ότι στην λίστα done_moves επαναλαμβάνονται περιοδικά οι ίδιες κινήσεις συμπεραίνεται ότι η το επίπεδο δεν έχει επιλύσιμη διαδρομή και το πρόγραμμα τερματίζει.

3.2.3 Έλεγχος επίτευξης του στόχου

Ο έλεγχος επίτευξης στόχου γίνεται με την συνάρτηση check_position() η οποία περιλαμβάνεται στην move(). Κάθε φορά που το κινητό αλλάζει θέση η συνάρτηση καλείται συγκρίνοντας την θέση με το σημείο τερματισμού. Εάν οι δυο συντεταγμένες ταυτίζονται επιστρέφει False με σκοπό να διακόψει την διαδικασία κίνησης αλλιώς παραμένει True έτσι ώστε να συνεχιστεί η προσπάθεια για να φτάσει στο στόχο.

3.2.4 Γραφική αναπαράσταση της διαδρομής και στοιχείων εκτέλεσης του προγράμματος με χρήση της Matplotlib

Κατά την επίλυση της διαδρομής οι θέσεις που περνάει το όχημα αποθηκεύονται σε μια λίστα της μορφής [(x1,y1),(x2,y2),...]. Επιπρόσθετα, γίνεται υπολογισμός του χρόνου που παρήλθε έως την επίλυση του επιπέδου με χρήση της time.process_time() και της απόστασης που διένυσε το όχημα με χρήση της συνάρτησης calcdist().

Η λίστα των θέσεων και οι μεταβλητές που περιέχουν τον χρόνο εκτέλεσης σε ms και την απόσταση δίνονται ως ορίσματα στην συνάρτηση plot(), η οποία με τη χρήση της βιβλιοθήκης Matplotlib[2] προβάλλει το επίπεδο με χρώματα κατανοητά προς τον χρήση (βλ. 3.1.2), ενώ πάνω στο επίπεδο σχεδιάζει την διαδρομή που ακολούθησε το όχημα ενώνοντας τα σημεία που περιέχονται στη λίστα θέσεων.

4 Επιπλέον ανάπτυξη του προγράμματος

Μετά την αρχική ανάπτυξη του προγράμματος έγιναν ορισμένες επιπλέον προσθήκες έτσι ώστε να βελτιωθεί η εμπειρία χρήσης, καθώς επίσης και να προστεθεί επιπλέον λειτουργικότητα στο ήδη υπάρχον πρόγραμμα. Για αυτό προστέθηκε προστέθηκε η δυνατότητα εισαγωγής των παραμέτρων του επιπέδου και χρήσης του προγράμματος μέσω γραφικού περιβάλλοντος.

4.1 Γραφική έναρξη προγράμματος με χρήση της Flet (v1.1)

Ως επόμενη βελτίωση στο πρόγραμμα θεωρήθηκε σκόπιμο να προστεθεί μία γραφική διεπαφή όπου ο χρήστης μπορεί να εισάγει τις παραμέτρους που θα καθορίσουν την δημιουργία του επιπέδου με τρόπο πιο εύκολο και γρήγορο από την εισαγωγή στην γραμμή εντολών. Η βιβλιοθήκη Flet[3] επιλέχθηκε λόγω των λειτουργιών που προσφέρει, της ευκολίας στη χρήση της και του καλύτερου αποτελέσματος που προσφέρει καθώς χρησιμοποιεί την γλώσσα σχεδιασμού Material [4].

Η διάταξη του παραθύρου αρχικά δημιουργήθηκε με χαρτί και μολύβι (Wireframes) και ο κώδικας γράφτηκε για να έχει το παράθυρο την διάταξη που περιγράφηκε στο προσχέδιο. Το παράθυρο οργανώθηκε με χρήση στηλών και γραμμών για να υπάρξει ομοιόμορφη στοίχιση των διάφορων στοιχείων. Επιτρέπει την είσοδο όλων των στοιχείων που είναι αναγκαία για την δημιουργία του επιπέδου και την έναρξη της επίλυσης.

Σε περίπτωση λανθασμένης εισόδου προβάλλεται στον χρήστη μήνυμα που τον προτρέπει να ελέγξει τα στοιχεία που έχει εισάγει.

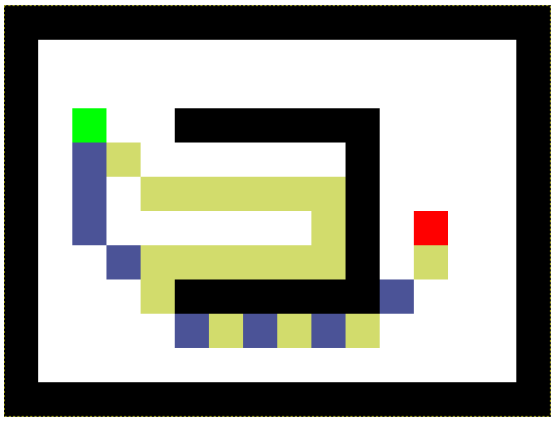
4.2 Υλοποίηση του τρόπου επίλυσης με χρήση του αλγορίθμου A* (v2)

4.2.1 Σύντομη περιγραφή τρόπου λειτουργίας του αλγορίθμου A*

Ο αλγόριθμος A*[5] είναι ένας αλγόριθμος που επιτρέπει την εύρεση της συντομότερης διαδρομής μεταξύ δύο σημείων. Αυτό το επιτυγχάνει ελέγχοντας σε όλα τα πιθανά σημεία που μπορεί να βρεθεί από την θέση του κάθε στιγμή ποιο θα είναι το αποτέλεσμα της κίνησης του και επιλέγοντας αυτό που θα το φέρει στον στόχο με το μικρότερο κόστος κίνησης. Ο υπολογισμός αυτός γίνεται με την χρήση της τιμής f που προκύπτει από το άθροισμα δύο άλλων τιμών των g, h, όπου g είναι το κόστος μετακίνησης από την αρχή που ορίσαμε έως το σημείο εκείνο και h η εκτίμηση για το κόστος μετακίνησης από εκείνο το σημείο έως τον στόχο που έχουμε ορίσει με βάση τον υπολογισμό της απόστασης από το στόχο. Ο υπολογισμός της απόστασης γίνεται με διαφορετικό τρόπο ανάλογα με τις κινήσεις που επιτρέπεται να κάνει το όχημα. Το κάθε σημείο στο οποίο θα κινηθεί το όχημα επιλέγεται με βάση αν έχει το μικρότερο f από τα σημεία που έχουν ελεγχθεί, με τους υπολογισμούς να γίνονται για κάθε σημείο που έχει νόημα να ελεγχθεί στο επίπεδο έως ότου το όχημα φτάσει στο στόχο του.

4.2.2 Πλεονεκτήματα χρήσης του αλγορίθμου A*

Ο αλγόριθμος A* βρίσκει την πιο σύντομη διαδρομή ανεξάρτητα των χαρακτηριστικών του επιπέδου. Αντίθετα ο αλγόριθμος που αναπτύξαμε εμείς σε συγκεκριμένες διατάξεις αντιμετωπίζει προβλήματα, καθώς δεν λαμβάνει υπ'όψιν όλα τα στοιχεία και ελέγχει κάθε φορά μόνο τις γειτονικές του θέσεις.



Εικόνα 4.1 Παράδειγμα περίπτωσης όπου ο αλγόριθμος A* υπερτερεί.
(Μπλε: Διαδρομή A*, Κίτρινο: Διαδρομή αλγορίθμου που αναπτύξαμε)

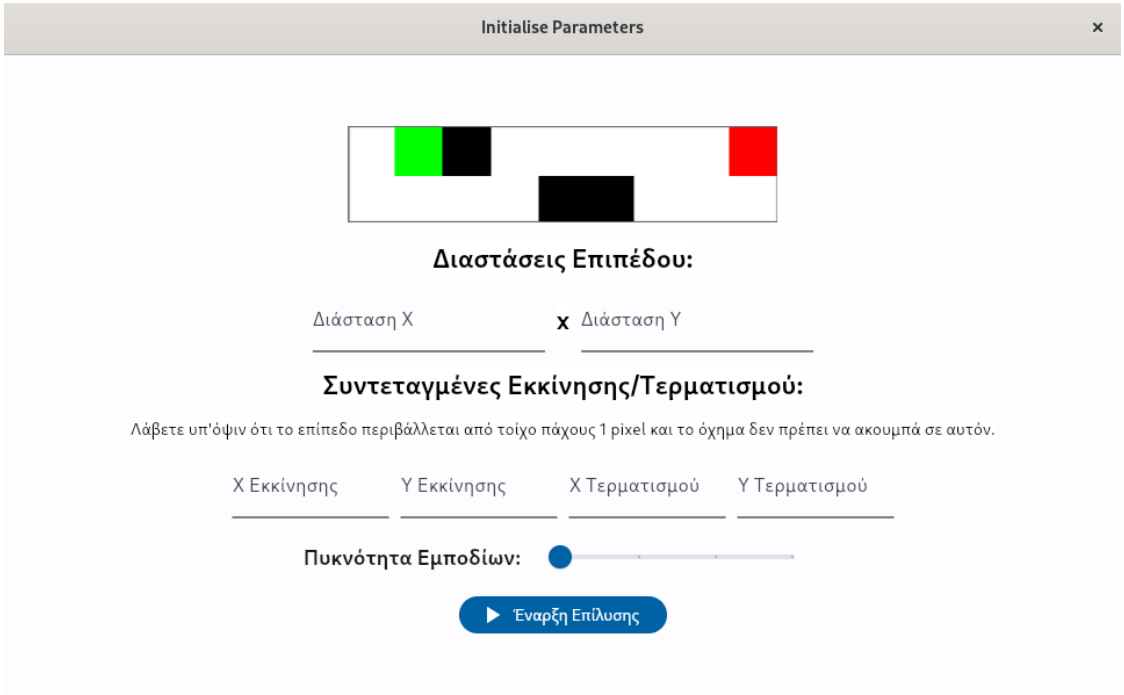
4.2.3 Υλοποίηση σε Python με χρήση της βιβλιοθήκης pathfinding

Για την επίλυση ενός επιπέδου με τον αλγόριθμο A* χρησιμοποιήθηκε η βιβλιοθήκη pathfinding[6]. Για την χρήση της βιβλιοθήκης αυτής ήταν αναγκαίες ορισμένες αλλαγές στον κώδικα και τον τρόπο που δημιουργείται ο πίνακας. Αρχικά, τα εμπόδια έπρεπε να αναπαρασταθούν με 0, ενώ τα ελεύθερα σημεία με 1, κάτι που έγινε αλλάζοντας τις τιμές που ορίζει η συνάρτηση gen_matrix(). Αντίστοιχα άλλαξε και η δημιουργία του πίνακα για χρήση με την Matplotlib.

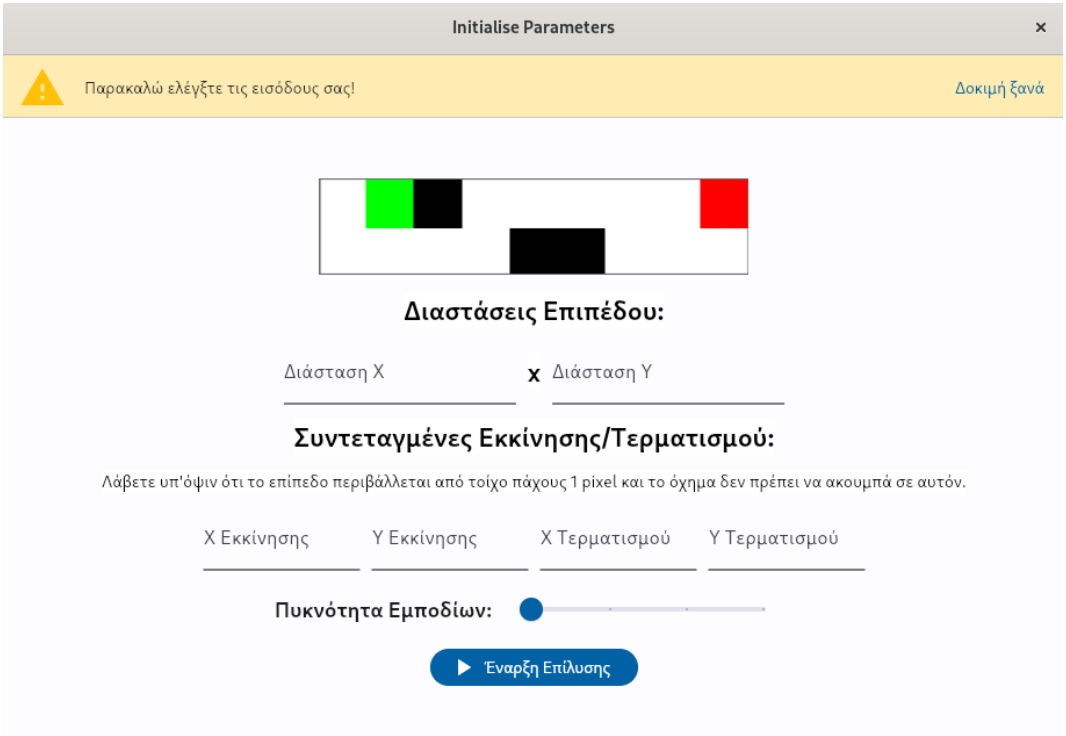
Με την σωστά διαμορφωμένη διάταξη ο πίνακας που αναπαριστά το επίπεδο δίνεται ως όρισμα στις κατάλληλες συναρτήσεις της βιβλιοθήκης και μας επιστρέφεται λίστα κινήσεων η οποία μετά από κατάλληλη επεξεργασία με την συνάρτηση poslist() δίνεται ως όρισμα στην plot() για να εμφανιστεί γραφικά η διαδρομή.

5 Τελικό αποτέλεσμα χρήσης του προγράμματος

Ανεξάρτητα του αλγορίθμου που χρησιμοποιείται (v1.1, v2) τα γραφικά του προγράμματος είναι ίδια. Το μόνο που αλλάζει είναι ο αλγόριθμος επίλυσης. Με την εκκίνηση της εφαρμογής αναδύεται ένα νέο παράθυρο στο οποίο υπάρχουν πεδία τα οποία ο χρήστης θα πρέπει να συμπληρώσει για να καθοριστούν οι ιδιότητες του επιπέδου (διαστάσεις επιπέδου πίστας, συντεταγμένες εκκίνησης, συντεταγμένες τερματισμού και πυκνότητα εμποδίων). Όταν συμπληρώσει τα πεδία αυτά θα πρέπει να επιλέξει το κουμπί 'Έναρξη Επίλυσης' για να συνεχίσει το πρόγραμμα στην δημιουργία ενός τυχαίου επιπέδου και την επίλυσή του. Σε περίπτωση που τα δεδομένα που έδωσε ο χρήστης δεν πληρούν ορισμένες προϋποθέσεις για να είναι δυνατή η δημιουργία του επιπέδου ζητείται από τον χρήστη να ελέγξει τις εισόδους του.

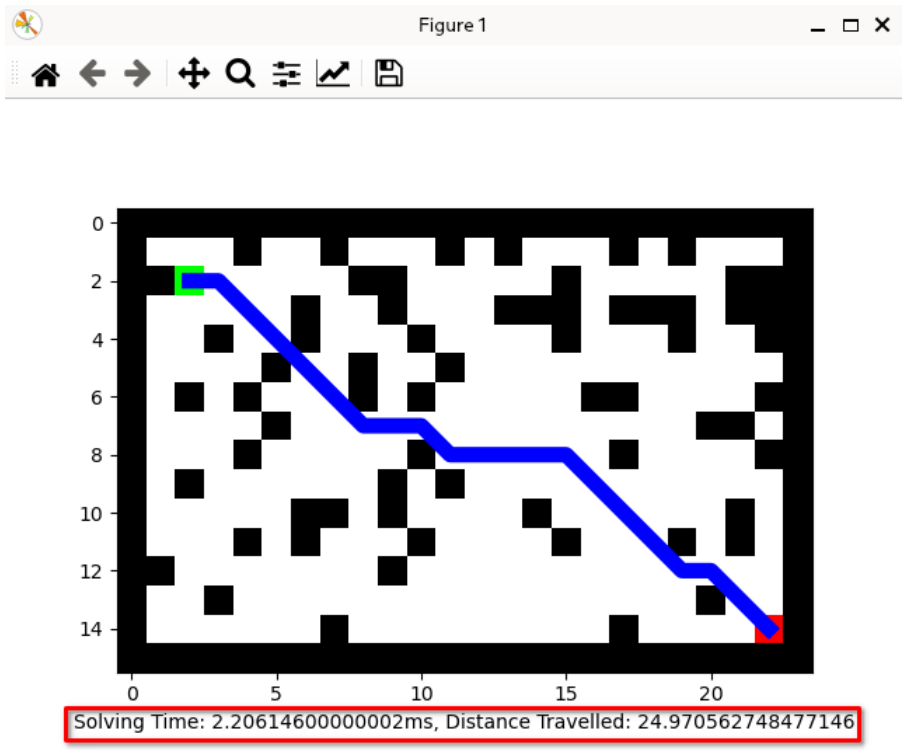


Εικόνα 5.1 Το παράθυρο που δημιουργείται με την εκτέλεση του προγράμματος.



Εικόνα 5.2 Το banner που εμφανίζεται όταν οι είσοδοι του χρήστη δεν είναι έγκυρες.

Όταν πατάμε το κουμπί 'Έναρξη επίλυσης' το αρχικό παράθυρο κλείνει και ανοίγει η γραφική διεπαφή της Matplotlib[1]. Στο παράθυρο αναπαριστάται γραφικά η βέλτιστη διαδρομή που θα εκτελέσει το όχημα με μπλε χρώμα ενώ η αφετηρία, ο τερματισμός και τα εμπόδια έχουν χρώματα πράσινο, κόκκινο και μαύρο αντίστοιχα. Υπάρχουν επίσης βαθμονομημένοι άξονες και ειδικό πεδίο, έτσι ώστε να είναι εύκολη η εύρεση των συντεταγμένων κάποιου σημείου. Ταυτόχρονα, στο κάτω μέρος δίνεται ο χρόνος επίλυσης της διαδρομής σε χιλιοστά του δευτερολέπτου και η απόσταση που διήνυσε το όχημα. Μέσω των εργαλείων που περιλαμβάνει η Matplotlib[1] ο χρήστης έχει την δυνατότητα να μεγεθύνει την αναπαράσταση και να μετακινηθεί σε αυτή, καθώς επίσης και να αποθηκεύσει σε αρχείο εικόνας το επίπεδο και την διαδρομή.



Εικόνα 5.3 Το τελικό παράθυρο εμφάνισης της διαδρομής (Στο κόκκινο πλαίσιο τα στοιχεία της επίλυσης).

6 Τρόπος εργασίας

6.1 Καταμερισμός Εργασιών

- Χρήστος Καρράς: Ο Χρήστος Καρράς συνέβαλε στη δημιουργία του τρόπου επίλυσης, του κώδικα για την εύρεση της βέλτιστης διαδρομής, τον εντοπισμό αν μια διαδρομή είναι επιλύσιμη, καθώς επίσης και στη συγγραφή μερών της παρούσας αναφοράς και της παρουσίασης.

- Γεώργιος Κουνάβης: Ο Γεώργιος Κουνάβης συνέβαλε στη δημιουργία του τρόπου επίλυσης, του κώδικα για την εύρεση της βέλτιστης διαδρομής, τον εντοπισμό αν μια διαδρομή είναι επιλύσιμη, καθώς επίσης και στη συγγραφή μερών της παρούσας αναφοράς και της παρουσίασης.

- Ηλίας Κρίγγος: Ο Ηλίας Κρίγγος συνέβαλε στη δημιουργία του τρόπου επίλυσης για την εύρεση της βέλτιστης διαδρομής και τη δημιουργία του τυχαίου επιπέδου, στη συγγραφή μερών της παρούσας αναφοράς, των σχετιζόμενων αρχείων και της παρουσίασης, καθώς επίσης και υλοποίησε τα γραφικά μέρη της εργασίας με Flet και Matplotlib και την επίλυση με χρήση του αλγορίθμου A*. Επιπρόσθετα είχε συντονιστικό ρόλο στην διαδικασία ολοκλήρωσης της εργασίας.

- Γεώργιος-Χρήστος Κωστάκης: Ο Γεώργιος-Χρήστος Κωστάκης συνέβαλε στη δημιουργία του τρόπου επίλυσης, του κώδικα για την εύρεση της βέλτιστης διαδρομής, τον εντοπισμό αν μια διαδρομή είναι επιλύσιμη, καθώς επίσης και στη συγγραφή μερών της παρούσας αναφοράς και της παρουσίασης.

Το κάθε μέλος της ομάδας έγραψε επίσης την δική του προσωπική έκθεση για το που συνέβαλε, τον τρόπο και τις ώρες που εργάστηκε, καθώς επίσης και για τις διάφορες πηγές που μελέτησε.

Αναφορές

[1] Matplotlib: Visualization with python. https://matplotlib.org/.

[2] Pyplot tutorial. https://matplotlib.org/stable/tutorials/pyplot.html, (accessed: 21.11.2023).

[3] Flet documentation. https://flet.dev/docs/.

[4] Material design. https://m3.material.io/.

[5] Amit Patel. Introduction to a*. https://theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html.

[6] Pathfinding library repository. https://github.com/brean/python-pathfinding.