Τ.Ε.Ι. ΠΕΙΡΑΙΑ

ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΠΕΙΡΑΙΑ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

ΤΜΗΜΑ ΑΥΤΟΜΑΤΙΣΜΟΥ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

«ΕΦΑΡΜΟΓΗ ΤΗΣ ΔΙΑΔΡΟΜΗΣ ΕΥΡΕΣΗΣ ΤΕΧΝΙΚΩΝ ΑΣΦΑΛΕΙΑΣ ΣΕ ΡΟΜΠΟΤ»

ΤΡΕΜΠΕΛΑ ΙΩΑΝΝΑ , ΦΙΛΙΠΠΟΠΟΥΛΟΥ ΣΤΕΦΑΝΙΑ

38686 **38224**

ΕΠΙΒΛΕΠΩΝ

ΤΣΕΛΕΣ ΔΗΜΗΤΡΙΟΣ

ΕΥΧΑΡΙΣΤΙΕΣ

Θα θέλαμε καταρχήν να ευχαριστήσουμε όλους όσους συνέβαλαν με οποιονδήποτε τρόπο στην επιτυχή εκπόνηση αυτής της πτυχιακής εργασίας. Θα πρέπει να ευχαριστήσουμε θερμά τον καθηγητή κ. Δημήτριο Τσελέ για την επίβλεψη αυτής της εργασίας. Ήταν πάντα διαθέσιμος να μας προσφέρει τις γνώσεις και την εμπειρία του.

Στη συνέχεια, ευχαριστούμε ιδιαίτερα τον κ. Χρήστο Δρόσο για την εξαιρετική συνεργασία που είχαμε, και ελπίζουμε πραγματικά να συνεχίσουμε να έχουμε στο μέλλον. Μέσα στον τελευταίο χρόνο ήταν πάντα διαθέσιμος να ασχοληθεί με κάθε απορία μας σχετική με ακαδημαϊκά ζητήματα, εντός και εκτός των πλαισίων της παρούσας εργασίας και με κάθε δισταγμό μας, για τα επόμενα βήματα των σπουδών μας. Τον ευχαριστούμε θέρμα για τις ιδέες που μας προσέφερε καθ' όλη τη διάρκεια εκπόνησης αυτής της εργασίας.

Θέλουμε επίσης να ευχαριστήσουμε τη σχολική μας παρέα, που ήταν, και ελπίζουμε να είναι δίπλα μας και στο μέλλον, παρά τη μεγάλη απόσταση που θα μας χωρίζει. Έπειτα, θα θέλαμε να ευχαριστήσουμε τους φίλους και τις φίλες των φοιτητικών μας χρόνων, που έκαναν τα χρόνια αυτά μία πραγματικά αξέχαστη εμπειρία. Βέβαια, το μεγαλύτερο ευχαριστώ το οφείλουμε στους γονείς μας, των οποίων η πίστη στις δυνατότητες μας αποτέλεσε αρωγός σε όλους τους στόχους και τα όνειρά μας, και οι οποίοι μας ανέθρεψαν σε ένα ειδυλλιακό περιβάλλον χωρίς καμία στέρηση.

ΠΕΡΙΛΗΨΗ

Οι τεχνικές ανεύρεσης διαδρομών είναι ένα έργο στην κινητική ρομποτική που υποβάλλεται σε εκτεταμένη έρευνα. Τα θέματα πλοήγησης έχουν μεγάλη σημασία και σε εφαρμογές εσωτερικής ασφάλειας. Μεταξύ των διαφόρων τεχνικών ανεύρεσης η χρήση των αλγόριθμων αναζήτησης είναι από της σημαντικότερες μεθόδους. Χρησιμοποιώντας το περιβάλλον προσομοίωσης ο αλγόριθμος A* υλοποιήθηκε. Μέσα από πειραματικά αποτελέσματα αποδείχθηκε ότι η απόδοση του αλγόριθμου A* βελτιώνεται δραστικά με μετά την προσθήκη ενός επιπλέον ευρετικού κανόνα.

Οι τεχνικές ανεύρεσης διαδρομών είναι η βελτίωση μιας κανονικής εύρεσης διαδρομής. Είναι αρκετά δύσκολο για ένα όχημα το οποίο έχει ήδη ορίσει μια ασφαλή διαδρομή να συμβαδίσει με το μεταβαλλόμενο περιβάλλον και να επαναϋπολογίσει ασφαλείς διαδρομές κατά τη διάρκεια της πορείας του. Τα ad-hoc αισθητήρια δικτύων χρησιμοποιούνται στην εσωτερική ασφάλεια και στις τεχνικές ανεύρεσης διαδρομών σε περιβάλλον όπως της breve. Παρακάτω θα μελετήσουμε τους σχεδιασμούς διαδρομών που έχουν εφαρμοστεί ώστε το όχημα να ελέγχει για αλλαγές στο περιβάλλον σε κάθε προσομοίωση και να επαναϋπολογίζει διαδρομές ενώ υπάρχει αλλαγή στο περιβάλλον. Η τεχνική ανεύρεσης διαδρομής με την χρήση του αλγόριθμου A* που αναπτύχθηκε στην παρούσα εργασία μπορεί να υπολογίζει ασφαλή διαδρομές για οποιοδήποτε κόμβο έναρξης και τερματισμού ή και για οποιαδήποτε τοποθέτηση αισθητήρα.

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΕΦΑΛΑΙΟ 4

ΚΕΦΑΛΑΙΟ 5

ΚΕΦΑΛΑΙΟ 1

1.1 Εισαγωγή

Ο σκοπός της εργασίας είναι να διερευνήσει και να αναπτύξει τεχνικές εύρεσης διαδρομών για αυτόνομα ρομπότ εσωτερικής ασφάλειας. Παρακάτω θα αντιμετωπιστούν τα εξής ζητήματα:

→ Η λήψη ασφαλέστερης διαδρομής για τον προορισμό από τα ρομπότ.
→ Η εφαρμογή τεχνικών ανεύρεσης διαδρομών.
→ Η διερεύνηση διαφόρων μεθόδων σχεδίασης.
→ Η σύγκριση των μεθόδων σχετικά με την απόδοση τους.
→ Η μελέτη του περιβάλλοντος προσομοίωσης της breve.
→ Ο σχεδιασμός και η δημιουργία του ρομπότ ως μοντέλο.
→ Το ρομπότ να μάθει να περιηγείται στην περιοχή εργασίας.
→ Η αντικατάσταση των αισθητήρων που χρησιμοποιούνται.

Το πρόβλημα του δυναμικών συγκρούσεων για το σχεδιασμό διαδρομής είναι για κινητά ρομπότ και ρομπότ που χρησιμοποιούνται σε εφαρμογές εσωτερικής ασφάλειας. Μια προσπάθεια έγινε για να δημιουργηθούν πιο ευέλικτα πληροφοριακά συστήματα με τη χρήση δικτυακών αισθητήρων σε κατανεμημένο περιβάλλον. Εκατοντάδες μικροί αισθητήρες είναι εξοπλισμένοι με περιορισμένη μνήμη και πολλαπλές δυνατότητες ανιχνεύσεως. Αυτοί οι αισθητήρες αυτόνομα οργανώνουν και αναδιοργανώνουν τους εαυτούς τους ως ad hoc δίκτυα για να ανταπεξέλθουν στις απαιτήσεις της εργασίας και στην ενεργοποίηση του περιβάλλοντος. Μια συλλογή από ενεργούς αισθητήρες δικτύων μπορούν να ακολουθήσουν την κίνηση μιας πηγής που πρέπει να παρακολουθείται, για παράδειγμα ένα κινούμενο όχημα. Μπορούν να καθοδηγήσουν την κίνηση ενός αντικειμένου στο έδαφος, για πχ ένα ρομπότ επιτήρησης ή μπορούν να εστιάσουν την προσοχή σε μια συγκεκριμένη περιοχή για πχ μια πυρκαγιά ώστε να εντοπίσουν την πηγή της και να παρακολουθήσουν την εξάπλωσή της .

Οι τεχνικές ανεύρεσης διαδρομής μπορεί να ομαδοποιηθούν σε τοπικές μεθόδους και ολικές. Μια πολύ γνωστή τοπική μέθοδος σχεδιασμού είναι η

μέθοδος πεδίου. Το πεδίο δημιουργείται από ελκυστικά δυναμικά, από μια αρχική θέση προς τον στόχο και από απωστικά δυναμικά που το αποτρέπουν από εμπόδια. Το δυναμικό πεδίο έχει χαμηλό υπολογιστικό κόστος. Αντίθετα οι ολικές μέθοδοι χρειάζονται πλήρεις πληροφορίες σχετικά με τον περιβάλλον και ως εκ τούτου θα απαιτούν σχετικώς μεγάλο υπολογιστικό κόστος.

1.2 Κίνητρα

Η ρομποτική στην εσωτερική ασφάλεια έχει μεγάλη σημασία στην σημερινή εποχή. Τα ρομπότ χρησιμοποιούνται ευρέως για την διάσωση επιζώντων από ακατάλληλα περιβάλλοντα. Για παράδειγμα ένα ρομπότ- φωτιά μπορεί να περάσει μέσα από την φωτιά και να διασώσει. Ένας σοβαρός περιορισμός είναι η έλλειψη των αισθητήρων που τοποθετούνται στα ρομπότ.

1.3 Περιγραφή θέσης

Στην αρχή έχει υλοποιηθεί η πιο βασική μορφή του αλγόριθμου αναζήτησης γράφων δένδρων η οποία δεν απαιτεί ευρετικούς κανόνες και βρίσκει την πρώτη δυνατή πορεία. Στην συνέχεια υλοποιείται ο αλγόριθμος A* , ένας από τους πιο γνωστούς αλγορίθμους αναζήτησης σε γράφους. Αυτός ο αλγόριθμος αξιολογεί τους κόμβους συνδυάζοντας το κόστος g(n) για την μετάβαση από τον κόμβο εκκίνησης στον κόμβο που ενδιαφέρει και h(n) το κόστος διαδρομής από τον κόμβο που ενδιαφέρει στον κόμβο τερματισμού. Ισχύει f(n) = g(n) +h(n)

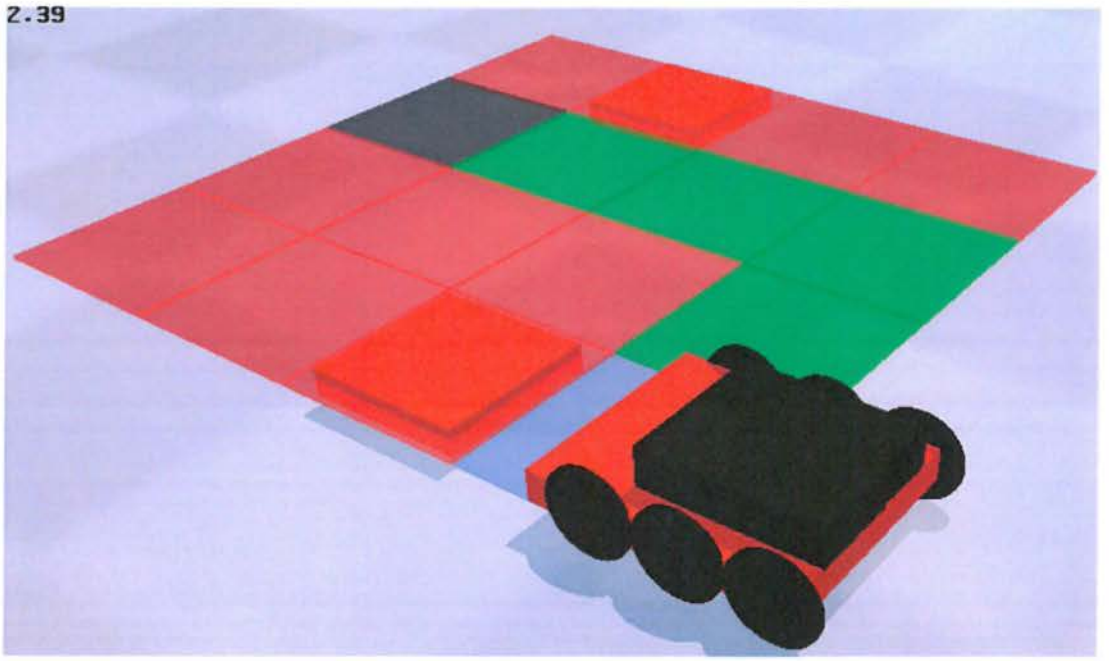
αφού g(n) δίνει το κόστος διαδρομής από την κόμβο εκκίνησης στον κόμβο n και h(n) είναι το εκτιμώμενο κόστος της φθηνότερης διαδρομής από τον στον κόμβο τερματισμού.

Έτσι f(n) είναι το εκτιμώμενο κόστος της φθηνότερης λύσης που διέρχεται από τον κόμβο n.

1.4 Μεθοδολογία

Στην breve οι αισθητήρες μπορούν να μοντελοποιηθούν με την χρήση της patch class. Η patch class είναι πιθανό να δημιουργήσει τα patches τα οποία μπορούν να ανιχνεύουν την παρουσία εμποδίων που αντιπροσωπεύουν επικίνδυνα αντικείμενα. Μπορούμε να υποθέσουμε ότι οι αισθητήρες είναι τόσο ευαίσθητοι και αποτελεσματικοί ώστε τα εμπόδια μπορούν να

δημιουργούνται και στο πρόγραμμα προσομοίωσης. Αλλά όταν τοποθετηθούν για πρακτική χρήση ίσως είναι απαραίτητο να δημιουργηθούν πολύ αποτελεσματικοί αισθητήρες ώστε να αντισταθμίσουν την έλλειψη αποτελεσματικότητας των αισθητήρων. Μόλις βρεθούν ασφαλή και μη ασφαλή patches θα πρέπει να είμαστε σε θέση να φτιάξουμε χάρτες και να βρούμε ασφαλή διαδρομές κρατώντας το όσο πιο μακριά από τα εμπόδια είναι δυνατόν.



***Εικόνα 1.1**: Στιγμιότυπο προσομοίωσης. Το ανοιχτό μπλε κελί είναι η αρχική θέση του ρομπότ. Το σκούρο μπλε κελί είναι ο στόχος. Τα patches που περιέχονται στα έντονο κόκκινο τετράγωνα αντιπροσωπεύουν τα εμπόδια. Τα άδεια και τα ανοιχτό κόκκινο κελιά αντιπροσωπεύουν τα ασφαλή. Τα κελιά με πράσινο χρώμα δείχνουν την πορεία που πρέπει να πάρει το όχημα.*

Τα αποτελέσματα που λαμβάνονται σε αυτή την προσομοίωση μπορούν να ελεγχθούν και να εφαρμοστούν πρακτικά χρησιμοποιώντας ένα Lego ρομπότ. Η απόδοση των τεχνικών ανεύρεσης διαδρομών είναι κάνοντας σύγκριση και αντίθεση. Θα πρέπει να είναι δυνατό να συνδυάζουν τα πλεονεκτήματα των διαφόρων αλγορίθμων και να δημιουργήσουν ένα νέο και αποδοτικό αλγόριθμο για τον προγραμματισμό διαδρομής.

1.5 Τεχνικοί Περιορισμοί

Μέσα από την έρευνα καταλήξαμε σε διάφορους περιορισμούς όπως η δυσκολία εφαρμογής του πειράματος μας σε περιβάλλον C και C++, η δυσκολία στην πλοήγηση του οχήματος και η έλλειψη δόμων δεδομένων.

ΚΕΦΑΛΑΙΟ 2

Διάφοροι Αλγόριθμοι Σχεδίασης Διαδρομών

Τα ρομποτικά συστήματα σχεδιάζονται για να λειτουργήσουν σε φυσικά περιβάλλοντα εργασίας. Ως περιβάλλον εργασίας ορίζεται το πραγματικό φυσικό πεδίο στο οποίο το ρομποτικό σύστημα έχει κληθεί για να εκτελέσει τις επιθυμητές εργασίες. Είναι φανερό ότι η θέση του στον περιβάλλοντα χώρο σε σχέση με ένα σημείο αναφοράς, απαιτείται να καθοριστεί με ακρίβεια. Ο χώρος εργασίας, είναι ο χώρος όλων των πιθανών θέσεων του ρομπότ μέσα στο περιβάλλον. Αυτό που έχει σημασία εδώ είναι η αντιστοίχηση του χώρου εργασίας με το χώρο των βαθμών ελευθερίας. Πολλές φορές οι δύο αυτοί χώροι έχουν διαφορετική διάσταση με αποτέλεσμα η τοποθέτηση του ρομπότ σε μια επιθυμητή θέση να απαιτεί πολύπλοκο έλεγχο.

2.1 Σχεδιασμός διαδρομής

Όπως γνωρίζουμε ένα μεγάλο μέρος της επιστήμης υπολογιστών ασχολείται με μεθόδους αναζήτησης. Οι μέθοδοι που ακολουθούνται ποικίλλουν ανάλογα με την περίπτωση και την εφαρμογή. Το πρόβλημα της αναζήτησης της διαδρομής που θα πρέπει να ακολουθήσει το ρομπότ δεν είναι ένα απλό υπολογιστικό πρόβλημα. Για να γίνει δυνατή η σχεδίαση του μονοπατιού που μπορεί να ακολουθηθεί, χρησιμοποιούνται τεχνικές που στόχο έχουν να ελαχιστοποιήσουν το χώρο εργασίας, έτσι ώστε να είναι δυνατή η εξέτασή του. Για την επίλυση του συγκεκριμένου πρόβλήματος υπάρχει ανεξάντλητη ποσότητα πληροφορίας, μιας και από τη δεκαετία του '80 έχουν δοθεί κάποιες

προσεγγίσεις επίλυσης. Στο κείμενο που ακολουθεί παρουσιάζονται κάποιες βασικές μέθοδοι σχεδιασμού διαδρομής.

2.1.1 Οδικός χάρτης

Η μέθοδος αυτή στόχο έχει να ελαχιστοποιήσει τις διαστάσεις του χώρου εργασίας του ρομπότ. Αυτό επιτυγχάνεται με τη αναπαράσταση του ελεύθερου χώρου εργασίας του ρομπότ με μονοδιάστατες καμπύλες. Η κατασκευή αυτή ονομάζεται οδικός χάρτης και το αποτέλεσμα αυτής της μεθόδου δίνει ένα ολοκληρωμένο μονοπάτι με τη μορφή γράφου που αποτελείται από τρία επιμέρους μονοπάτια:

→ Ένα αρχικό μονοπάτι διασυνδέει το αρχικό σημείο που βρίσκεται το ρομποτικό σύστημα **qstart** με το κοντινότερο σ' αυτό σημείο του οδικού χάρτη **q'**.
→ Το βασικό μονοπάτι το οποίο ξεκινάει από το **q'** του οδικού χάρτη και φθάνει στο σημείο **q''** του οδικού χάρτη.
→ Το τελικό μονοπάτι που συνδέει το σημείο τερματισμού **qend** με το κοντινότερο σ'αυτό, σημείο του οδικού χάρτη **q''**.

Διαδεδομένες τεχνικές αυτής της μεθόδου είναι τα διαγράμματα Voronoi και οι γράφοι ορατότητας . Σ' αυτές τις περιπτώσεις το πρόβλημα είναι η κατασκευή του χάρτη διαδρομής (υπολογιστικά δύσκολο) και ο τρόπος που μπορούν να τροποποιηθούν για εφαρμογή σε δυναμικά περιβάλλοντα.

2.2 Γεωμετρική προσέγγιση

Αυτή η προσέγγιση μοντελοποιεί το περιβάλλον εργασίας, αλλά και το ίδιο το ρομπότ κάνοντας χρήση γεωμετρικών δομών (τετράγωνα, κύκλους, κλπ) . Με τη βοήθεια των δομών αυτών η μέθοδος εκτελεί τις απαραίτητες περιστροφές και μετατοπίσεις των συγκεκριμένων γεωμετρικών αντικειμένων, ώστε το ρομποτικό σύστημα να οδηγηθεί στον επιθυμητό προορισμό. Ένα πολύ γνωστό πρόβλημα αυτής της κατηγορίας είναι το πρόβλημα του «piano movers», το οποίο εξετάζει τις περιστροφές που πρέπει να γίνουν σε ένα πιάνο για να μετακινηθεί έξω από ένα δωμάτιο μέσω μιας στενής πόρτας. Οι μέθοδοι αυτοί εξαιτίας του κόστους υπολογισμού, σπάνια εφαρμόζονται σε πραγματικά ρομποτικά συστήματα, ωστόσο έχουν μελετηθεί σε βάθος.

2.3 Προσέγγιση διάσπασης κελιών

Η μέθοδος αυτή διαφοροποιεί τον χώρο εργασίας του ρομπότ σε ένα μεγάλο αριθμό από μικρότερες περιοχές που τις καλούμε κελιά. Συγκεκριμένα, κάθε δυνατή κίνηση του ρομπότ από το ένα κελί στο γειτονικό του μπορεί να πραγματοποιηθεί με βάση τα χαρακτηριστικά και τις δυνατότητες κίνησης του ρομπότ. Η συνολική διαδρομή που πρέπει να ακολουθήσει το ρομπότ για να μεταβεί μέσω των κελιών αυτών σε ένα επιθυμητό σημείο, αναπαριστάται σαν ένας μη κατευθυνόμενος γράφος. Το μέγεθος των κελιών μπορεί να μεταβάλλεται ανάλογα με τις απαιτήσεις και ο υπολογισμός του μονοπατιού μπορεί να πραγματοποιηθεί από οποιοδήποτε αλγόριθμο αναζήτησης σε γράφο. Αυτή η κατηγορία έχει πιθανώς ερευνηθεί πολύ περισσότερο από οποιαδήποτε άλλη, μιας και η πολυπλοκότητά της σχετίζεται άμεσα με τον αλγόριθμο αναζήτησης που μπορεί να χρησιμοποιηθεί. Λόγω της γενικότητας αυτών των τεχνικών, επιλέξαμε να χρησιμοποιήσουμε μια μέθοδο διαφοροποίησης και μια τεχνική αναζήτησης για το σχεδιασμό διαδρομής στο σύστημά μας.

2.4 Ανακλαστική προσέγγιση

Αυτή η μέθοδος εστιάζει σε τεχνικές που στόχο έχουν την αντιμετώπιση του υπολογιστικού κόστους, δηλαδή την κατασκευή γρήγορων λύσεων σε εφαρμογές πραγματικού χρόνου. Η ιδέα είναι να υπάρχει άμεση ανταπόκριση. Τέτοια σύνδεση μπορεί να πραγματοποιηθεί με τη μορφή συμπεριφορών που απεικονίζουν αντιλήψεις και δράσεις και να υλοποιηθεί με μια απλή δομή, π.χ. πεπερασμένα αυτόματα. Παρόλο που οι δυνατότητες της μεθόδου αυτής φαίνονται περιορισμένες, ο συνδυασμός πολλών κανόνων συμπεριφοράς

παράλληλα, θα έχει ως αποτέλεσμα τη συνολική συμπεριφορά του ρομποτικού συστήματος να είναι ένας συνδυασμός των επιμέρους συμπεριφορών. Άλλες τεχνικές ανακλαστικών ελεγκτών επιτυγχάνονται με τη χρήση ασαφούς λογικής και νευρωνικών δικτύων. Μια γρήγορη μηχανή ασαφούς λογικής σε συνδυασμό με ένα άρτια εκπαιδευμένο νευρωνικό δίκτυο μπορούν να δώσουν άμεση και ομαλή ανταπόκριση στη συμπεριφορά του ρομπότ, ιδιαίτερα σε γρήγορες απότομες μεταβολές στο περιβάλλον εργασίας. Τα τελευταία χρόνια η τεχνική του μοντέλου των συμπεριφορών γίνεται ολοένα και πιο δημοφιλής, μιας και έχει δώσει μια εντελώς διαφορετική οπτική γωνία στην κατασκευή έξυπνων ρομποτικών συστημάτων. Ωστόσο, οι δυνατότητες αυτών των τεχνικών είναι περιορισμένες, διότι οι αλληλεπιδράσεις μεταξύ συμπεριφορών ελέγχονται δύσκολα.

2.5 Εικονικά πεδία δυνάμεων

Η μέθοδος αυτή καθορίζει τον τρόπο που δρουν κάποια εικονικά πεδία δυνάμεων πάνω στο χώρο εργασίας του ρομπότ, έτσι ώστε το αποτέλεσμα της δράσης αυτής να είναι ένας συνδυασμός δυνάμεων πάνω στο ρομπότ που να το οδηγούν στο επιθυμητό σημείο. Γενικότερα, υποθέτουμε ότι το σημείο που θέλουμε να φτάσουμε εφαρμόζει στο ρομπότ μια ελκτική δύναμη, ενώ τα εμπόδια μια απωθητική. Αθροίζοντας τα επιμέρους διανύσματα όλων των δυνάμεων, θα οδηγηθεί το ρομπότ προς το συνιστάμενη. Το αποτέλεσμα είναι να μετακινήσουμε το ρομπότ προς τον στόχο και παράλληλα να αποφύγουμε τα εμπόδια. Δυστυχώς, σε κάποιες περιπτώσεις το άθροισμα των επιδράσεων αλληλοαναιρείται, κατά συνέπεια το ρομποτικό σύστημα οδηγείται σε αδιέξοδο. Αναλυτικά παραδείγματα της τεχνικής αυτής έχουν μελετηθεί από τους Oussama Khatib και Johann Borenstein .

2.6 Δυναμικά πεδία

Με τη μέθοδο αυτή η πλοήγηση μπορεί να περιγραφεί σαν μια διαδικασία που ακολουθεί το μέγιστο άνυσμα σε κάποιο ιδεατό πεδίο δυναμικού μέσα στο περιβάλλον του ρομπότ. Η φύση, τα ζώα ή ακόμα και ο άνθρωπος εφαρμόζουν αντίστοιχες τεχνικές για να προσανατολιστούν. Για παράδειγμα, όπως ένας άνθρωπος μπορεί να κατευθυνθεί από ένα ήχο και να τον ακολουθήσει καθώς ενισχύεται μέχρι να προσδιορίσει την πηγή του, έτσι αντίστοιχα κάποια ζώα κατευθύνονται με βάση τα ένστικτα και τα αισθητήρια όργανα που διαθέτουν προς κάποιο ερέθισμα (νερό, φαγητό, ήχους, κλπ). Στα

ρομποτικά συστήματα η μέθοδος αυτή υπολογίζει ένα άνυσμα με τη βοήθεια των αισθητήρων του ρομπότ και με βάση το ερέθισμα που δέχεται. Έτσι, το ρομπότ απλά ακολουθεί κάθε χρονική στιγμή *το* μέγιστο παραγόμενο άνυσμα το οποίο εντέλει το οδηγεί στο στόχο. Τα δυναμικά πεδία επιλύουν ένα μεγαλύτερο εύρος του προβλήματος της πλοήγησης σε σχέση με τις άλλες μεθόδους. Ο τρόπος αυτός λαμβάνει χώρα στο συνολικό περιβάλλον του ρομπότ και το αποτέλεσμα που προκύπτει είναι ένα διάνυσμα κατεύθυνσης σε κάθε θέση του χώρου εργασίας. Η τεχνική αυτή είναι πιθανόν από τις πιο δημοφιλείς στην πλοήγηση ρομποτικών συστημάτων.

ΚΕΦΑΛΑΙΟ 3

3.1 Τι είναι breve;

Η breve είναι ένα λογισμικό προσομοίωσης που ξεκίνησε από τον Jon Klein ως μια διατριβή στο Hampshire κολλέγιο και αναπτύχθηκε περαιτέρω σε μια διατριβή master στο πανεπιστήμιο Chalmers. Το λογισμικό αναπτύσσεται ενεργά ως μια πλατφόρμα για την οικοδόμηση σε μια διατριβή μεγάλης κλίμακας προσομοίωσης με εξελικτική δυναμική αλλά χρησιμοποιείται επίσης και για πολλές άλλες εφαρμογές . Η breve είναι ένας δωρεάν ανοιχτός κώδικας λογισμικού που καθιστά εύκολο να οικοδομήσουμε 3d προσομοιώσεις των αποκεντρωμένων συστημάτων και της τεχνητής νοημοσύνης. Οι χρήστες καθορίζουν τις συμπεριφορές μέσα σε ένα 3d κόσμο και παρατηρούμε πως αλληλεπιδρούν μεταξύ τους. Η breve περιλαμβάνει προσομοίωση σωματιδίων και ανίχνευση σύγκρουσης, για να μπορεί κανείς να προσομοιώσει ρεαλιστικά πλάσματα και μια open GL μηχανή προβολής ως στόχο να απεικονίσει αυτά σε κόσμους προσομοίωσης. Ενώ η breve εννοιολογικά είναι παρόμοια με υπάρχοντα πακέτα όπως Swarm και Star logo η εφαρμογή της breve που προσομοιώνει τόσο συνεχή χρόνο και συνεχή 3d χώρο είναι αρκετά διαφορετική ώστε το περιβάλλον να είναι κατάλληλο για πολύπλοκες προσομοιώσεις. Η breve είναι διαθέσιμη για Mac, Linux και Windows.

3.2 Γράφοντας προσομοιώσεις στην breve

Οι προσομοιώσεις breve γράφονται εύκολα με την χρήση της γλώσσας steve. Η γλώσσα steve είναι αντικειμενοστραφής και δανείζεται πολλά στοιχεία από τις γλώσσες όπως η C, PERL και objective c και ακόμη οι χρήστες θα το βρουν εύκολο να προγραμματίσουν χωρίς προηγούμενη εμπειρία στον προγραμματισμό.

3.3 Χρήση πρόσθετων-plugins στην breve

Η breve διαθέτει μια επεκτάσιμη αρχιτεκτονική pluging που μας επιτρέπει να γράψουμε τον δικό μας πρόσθετο κώδικα ώστε να αλληλεπιδρά με τον δικό μας κώδικα. Για να γράψουμε plugins είναι απλό και μας επιτρέπει να επεκτείνουμε την breve ώστε να συνεργαστεί με τον υπάρχον κώδικα. Έχουν γραφεί σε breve πρόσθετοι κώδικες για την παραγωγή μουσικής midi , για να κατεβάζουν ιστοσελίδες και να αλληλεπιδρούν με το περιβάλλον. Είναι επίσης δυνατόν να χρησιμοποιηθεί στη c και στη c++.

3.4 Χαρακτηριστικά της breve

Ακολουθούν μερικά από τα σημαντικά στοιχεία της breve:

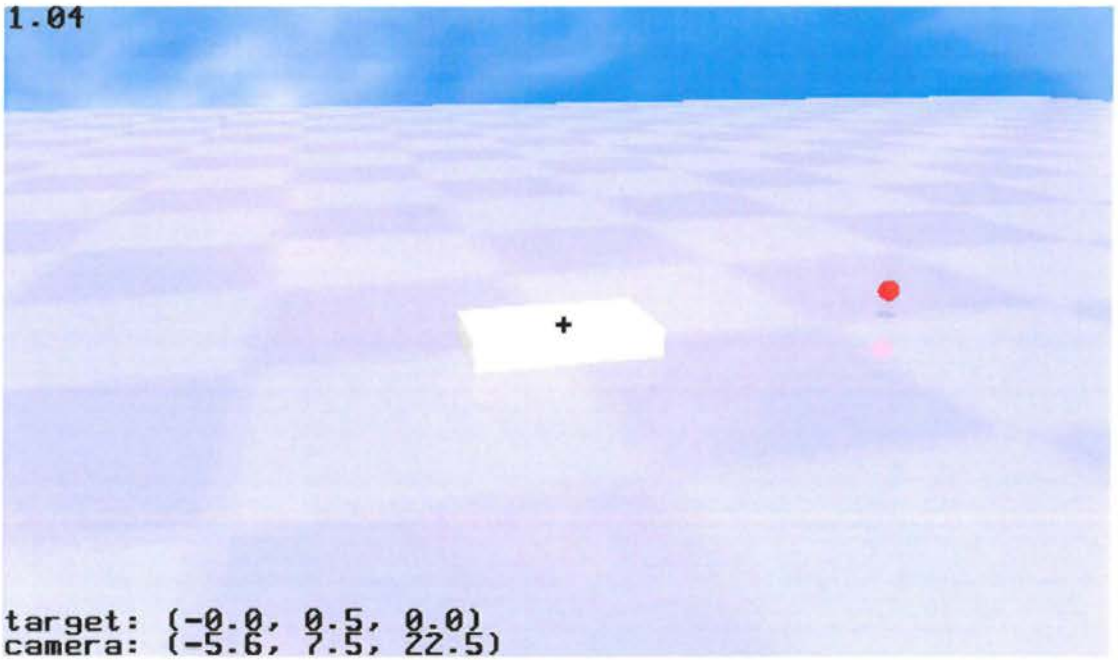
→ Αντικειμενοστραφής γλώσσα Steve
→ Μια μηχανή προβολής open GL
→ Ανίχνευση σύγκρουσης
→ Γενετική τάξη αλγορίθμου
→ Braitenberg κατηγορία
→ Plugins
→ Εγγραφή ταινιών της προσομοίωσης
→ Αποθήκευση στιγμιοτύπων της προσομοίωσης

3.5 Braitenberg Οχήματα

Ο Valentino Braitenberg περιγράφει μια σειρά πειραμάτων σκέψης σε κάποια οχήματα με απλή εσωτερική δομή που συμπεριφέρονται απρόσμενα με πολλαπλούς τρόπους. Περιγράφει απλούς μηχανισμούς ελέγχου και είναι πολύ δύσκολο να προσπαθήσει να μαντέψει την εσωτερική δομή μόνο από την παρατήρηση της συμπεριφοράς από ότι να δημιουργήσει την δομή που δίνει την συμπεριφορά.

3.5.1 Braitenberg όχημα και η εφαρμογή στη breve

Το πρότυπο του οχήματος braitenberg σε breve φαίνεται στο σχήμα 3.1

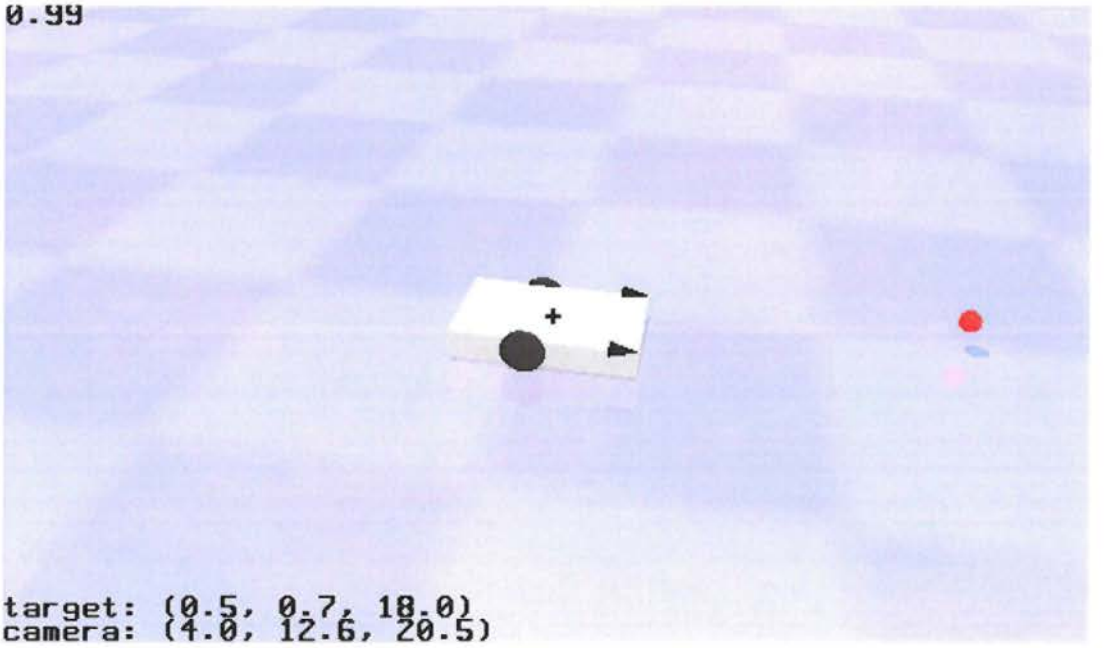


***Εικόνα 3.1**: Στιγμιότυπο που λήφθηκε από την breve προσομοίωση, δείχνει το ορθογώνιο λευκό αντικείμενο που είναι το σώμα του οχήματος και ένα κόκκινο φώς-αντικείμενο το οποίο μπορεί να μετακινηθεί.*

3.5.2 Braitenberg όχημα με πρόσθετους τροχούς και αισθητήρες

Δύο τροχοί προστέθηκαν στο όχημα ώστε να μπορεί να κινηθεί σε όλη την επιφάνεια. Δύο είχαν επίσης προστεθεί στο όχημα ώστε να μπορεί να ανιχνεύσει το φώς αντικειμένων. Το σχήμα 3.2 δείχνει αυτό το σχεδιασμό. Το όχημα μπορεί να κινηθεί με τον καθορισμό μιας ταχύτητας προς τους τροχούς. Η μέθοδος vehicle get position χρησιμοποιήθηκε για να πάρει τη θέση του οχήματος σε κάθε επανάληψη και όποτε υπήρχε η πιθανότητα το όχημα να πέσει κάτω να μετακινείται προς τα πίσω και να στρίβει από τα δεξιά και προς τα αριστερά.

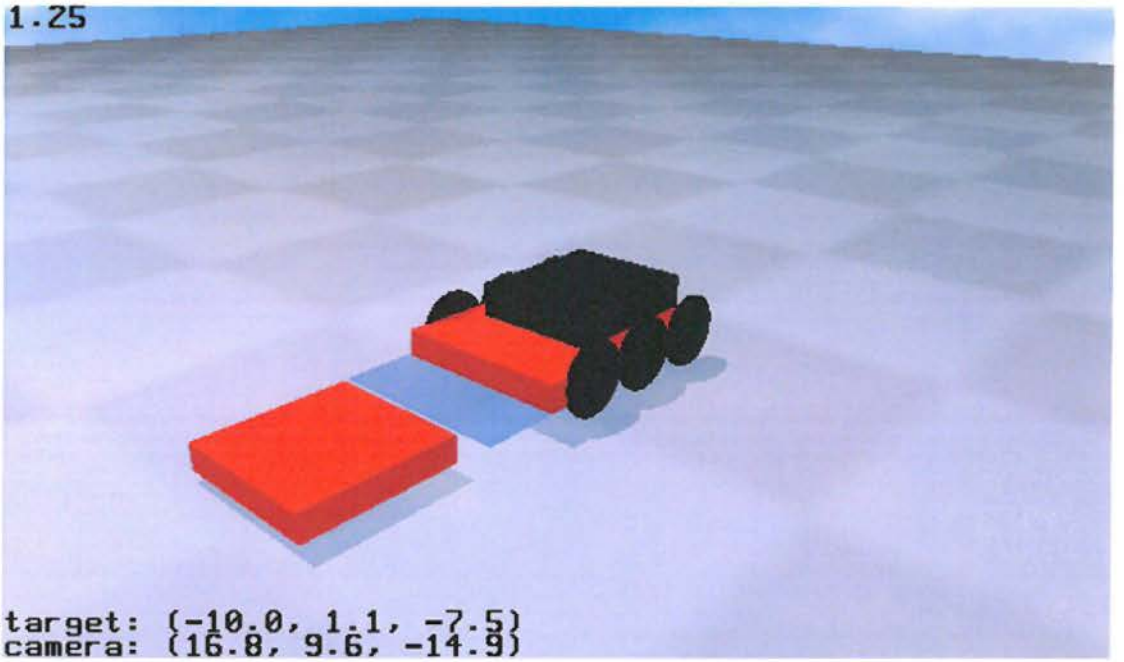
Τώρα που το όχημα ήταν σε θέση να κινηθεί σε όλη την επιφάνεια χωρίς να πέσει κάτω, προσθέτουμε τους αισθητήρες και να εισάγουμε τα εμπόδια στην πορεία του οχήματος. Χρησιμοποιώντας αυτό το σχέδιο μια ενδιαφέρουσα μελέτη των αρχών braitenberg παρατηρήθηκε τόσο οι αριστεροί όσο και οι δεξιοί αισθητήρες συζεύχθηκαν με μόνον έναν από τους τροχούς. Αυτό προκάλεσε το όχημα ικανό να κινείται σε όλη την επιφάνεια αποφεύγοντας τα εμπόδια. Αλλά παρατηρήθηκε ότι αυτή η μέθοδος δεν ήταν τέλεια. Τα εμπόδια φωτός θα πρέπει να αποφεύγονται όταν τα εμπόδια είναι είτε από τα αριστερά ή από τα δεξιά του οχήματος. Όμως το όχημα θα κινηθεί και όταν δεξιά υπάρχει εμπόδιο και όταν είναι μεταξύ των αισθητήρων . Ωστόσο καμία από τις μεθόδους δεν μπορεί να δώσει λύση ώστε το όχημα να αποφύγει το εμπόδιο με την χρήση τεχνικής.



Εικόνα 3.2: ***2 τροχοί*** *και 2 αισθητήρες προστέθηκαν στο όχημα*

3.5.3 Εισαγωγή patches

Η εισαγωγή των patches- εμπόδιων στην προσομοίωση οδήγησε σε μεταβολή των διαστάσεων στο όχημα braitenberg. Λόγω της αλλαγής των διαστάσεων του οχήματος ήταν απαραίτητο να προστεθούν περισσότεροι τροχοί. Μετά την εισαγωγή των εμποδίων η χρήση των αισθητήρων είχε καταστεί άνευ αντικειμένου και ως εκ τούτου αφαιρέθηκαν. Η τελική σχεδίαση του χρησιμοποιημένου οχήματος φαίνεται στο σχήμα 3.3.



Σχήμα 3.3: *Τελικό σχέδιο τροποποιημένο του οχήματος και του εμποδίου.*

3.6 Patch Class

Σύμφωνα με την μέθοδο διάσπασης κελιών η περιοχή αποσυντίθεται με ίσου μεγέθους πλέγματα κελιών. Τα εμπόδια αποκτούν το ίδιο μέγεθος με το όχημα , προκειμένου να διευκολυνθεί η πλοήγηση του οχήματος από το ένα κελί στο άλλο.

3.6.1 Χαρακτηριστικά της κατηγορίας Patch Class

Η patch class έχει πολλά χρήσιμα χαρακτηριστικά που διευκολύνει το έργο του προγραμματισμού:

→ patch-get θέση: επιστρέφει ένα διάνυσμα θέσης του patch object

→ get-patch-at θέση: επιστρέφει το patch στη δεδομένη θέση

→ get-patch-above: επιστρέφει το patch προς (0,1,0)

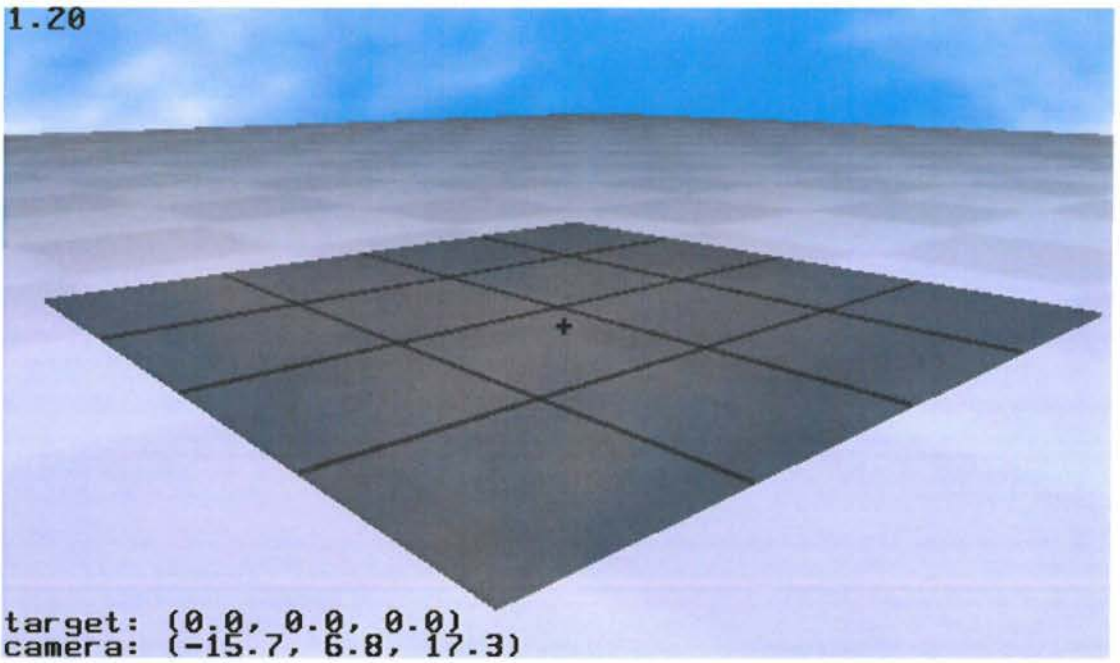
→ get-patch-below: επιστρέφει patch προς (0,-1,0)

→ get-patch προς plus-z: επιστρέφει το patch προς (0,0,1)

→ get-patch-προς minus z: επιστρέφει το patch προς (0,0,-1)

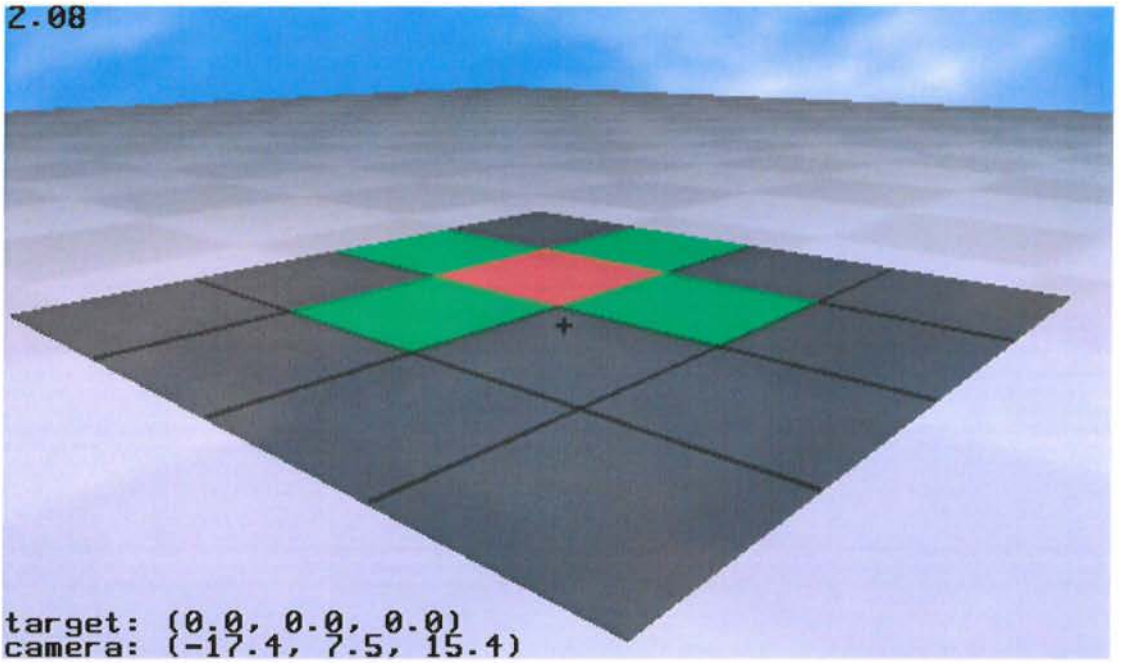
→ patch-set -color (τιμή): ορίζει το χρώμα patch στην τιμή

→ patch-set transparency to value: ρυθμίζει τη διαφάνεια στην τιμή



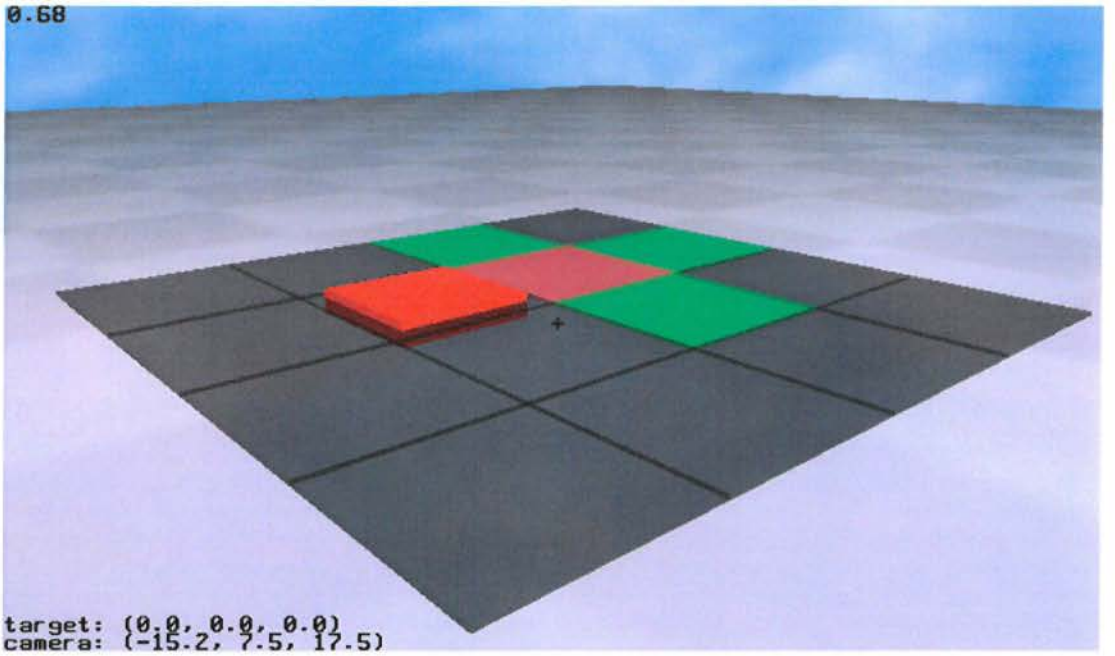
Σχήμα 3.4: *εμφανίζει ένα 4X4 πλέγμα*

Η παρακάτω συνάρτηση χρησιμοποιείται για να πάρει τα γειτονικά κελιά του κάθε εμποδίου. Το σχήμα 3.5 δείχνει τον παραπάνω κώδικα σε δράση. Λαμβάνοντας υπόψη το εμπόδιο στο κέντρο (κόκκινο) αυτό λειτουργεί επιστρέφοντας τα 4 γειτονικά κελιά (πράσινο) γύρω από αυτό. Η συνάρτηση για το σύνολο των χρωμάτων χρησιμοποιήθηκε για λόγους επίδειξης.



***Σχήμα 3.5**: για κάθε εμπόδιο είναι δυνατόν να ληφθούν τα γειτονικά κελιά*

Στη συνέχεια πρέπει να αποκλειστούν τα γειτονικά κελία τα οποία περιείχαν εμπόδιο . Το κελί στο οποίο βρίσκεται το εμπόδιο είναι κόκκινο και όχι πράσινο όπως φαίνεται στο σχήμα 3.6.



***Σχήμα 3.6**: το κελί το όποιο περιέχει εμπόδιο.*

3.7 Λίστα για τον Τύπο Δεδομένων

Ο τύπος δεδομένων χρησιμοποιείται για την αποθήκευση των κελιών στη λίστα της steve. Χρησιμοποιώντας τον κατάλογο του τύπου δεδομένων είναι δυνατόν να κρατήσει μια λίστα με άλλες μεταβλητές οποιουδήποτε τύπου, συμπεριλαμβανομένων άλλων λιστών.

ΚΕΦΑΛΑΙΟ 4

4.1 Τι είναι Εσωτερική Ασφάλεια

Εσωτερική ασφάλεια ή εσωτερική άμυνα είναι ένας νεολογισμός που αναφέρεται σε εγχώριες κυβερνητικές ενέργειες ο οποίος δικαιολογείται από πιθανές επιθέσεις ανταρτών ή τρομοκρατία. Ο όρος έγινε εμφανής στις ΗΠΑ μετά την 11 Σεπτεμβρίου του 2001.

Τέτοιες εγχώριες κυβερνητικές ενέργειες περιλαμβάνουν:

→ Κινητοποίηση έκτακτης ανάγκης, συμπεριλαμβανομένων των εθελοντών ιατρικής της αστυνομίας και πυροσβεστικής
→ Νέα εγχώρια παρακολούθηση και κατασκοπευτικές προσπάθειες ιδίως όταν αφορά την μετανάστευση τη μεταφορά τις στρατιωτικές εγκαταστάσεις και επιχειρήσεις κοινής ωφέλειας
→ Προστασία των υποδομών
→ Έλεγχος των συνόρων

4.2 Πεδίο Εφαρμογής της Εσωτερικής Ασφάλειας

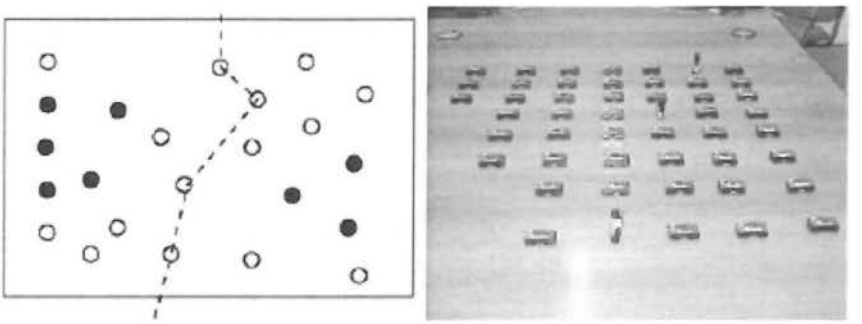
Οι έξι βασικοί τομείς της αποστολής που θεωρούνται κρίσιμοι για την εσωτερική ασφάλεια είναι:

→ Νοημοσύνη και προειδοποίηση
→ Σύνορα και ασφάλεια των μεταφορών
→ Εγχώρια αντιτρομοκρατία
→ Προστασία των υποδομών ζωτικής σημασίας
→ Υπεράσπιση της τρομοκρατίας
→ Ετοιμότητα έκτακτης ανάγκης και αντιμετώπιση

Οι τρείς πρώτες περιοχές έχει επικεντρωθεί, μεταξύ άλλων στην πρόληψη τρομοκρατικών επιθέσεων εναντίον των ΗΠΑ, τα επόμενα 2 σχετικά με τη μείωση των τρίτων σημείων εντός ΗΠΑ και η τελευταία περιοχή στην ελαχιστοποίηση των ζημιών και την ανάκτηση από τις τρομοκρατικές επιθέσεις που έχουν συμβεί στις ΗΠΑ.

4.3 Τεχνικές Πλοήγησης σε Δίκτυα Αισθητήρων

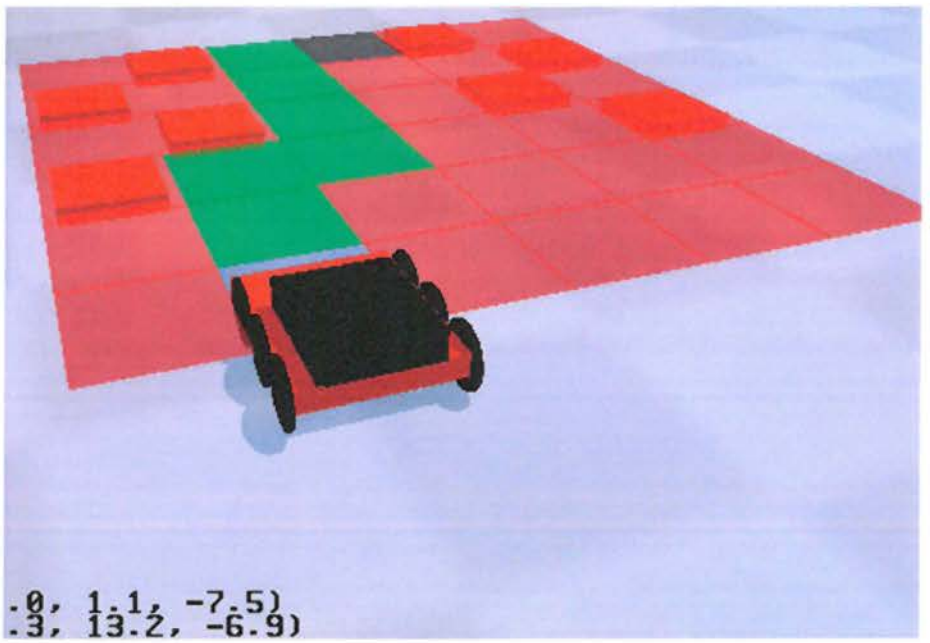
Ξεκινάμε με ένα παράδειγμα χρήσης κατανεμημένων αισθητήρων. Εκατοντάδες μικροί αισθητήρες που είναι εξοπλισμένοι με περιορισμένη μνήμη και πολλαπλές δυνατότητες ανίχνευσης και αναπροσαρμόζονται αυτόνομα ως ad hoc δίκτυα για την αντιμετώπιση απαιτήσεων της εργασίας και καθώς ενεργοποιούνται από το περιβάλλον. Κατανεμημένα προσαρμοστικά δίκτυα αισθητήριων δραστικών συστημάτων πληροφορικής κατάλληλα για εργασίες σε ακραία περιβάλλοντα ειδικά όταν το περιβαλλοντικό μοντέλο και οι προδιαγραφές του έργου είναι αβέβαιο και το σύστημα πρέπει να προσαρμοστεί σε αυτά. Μια συλλογή από ενεργά δίκτυα αισθητήριων μπορούν να ακολουθούν την κίνηση μιας πηγής που πρέπει να παρακολουθούν για παράδειγμα ένα κινούμενο όχημα. Αυτό μπορεί να κατευθύνει την κίνηση ενός αντικειμένου στο έδαφος για παράδειγμα επιτήρηση ρομπότ. Ή μπορεί να εστιάσει την προσοχή σε μια συγκεκριμένη περιοχή για παράδειγμα μια πυρκαγιά προκειμένου να εντοπίσουν την πηγή της και να παρακολουθεί την εξάπλωσή της. Ένα δίκτυο αισθητήριων αποτελείται από μια συλλογή αισθητήριων που έχουν διανεμηθεί πάνω από κάποια περιοχή που σχηματίζουν ένα δίκτυο ad hoc. Κάθε αισθητήρας είναι εξοπλισμένος με ορισμένες περιορισμένες δυνατότητες μνήμης και την επεξεργασία πολλαπλών τρόπων ανίχνευσης και δυνατότητες επικοινωνίας. Οι αισθητήρες αυτοί είναι ικανοί να ανιχνεύσουν ειδικές εκδηλώσεις που ονομάζονται «ΚΙΝΔΥΝΟΣ»(π.χ. θερμοκρασία, βιοχημικοί παράγοντες κλπ) που βρίσκονται πάνω από ένα συγκεκριμένο όριο. Οι αισθητήρες που έχουν προκαλέσει οι ειδικές εκδηλώσεις θεωρούνται εμπόδιο.



***Σχήμα 4.1**: Η αριστερή εικόνα δείχνει μια τυπική εγκατάσταση για την πλοήγηση σε κατευθυντήριο έργο. Οι στερεοί μαύροι κύκλοι αντιστοιχούν σε αισθητήρες των οποίων ανιχνεύεται ο κίνδυνος. Οι λευκοί κύκλοι αντιστοιχούν σε αισθητήρες που δεν αισθάνονται τον κίνδυνο. Η διακεκομμένη γραμμή δείχνει την κατευθυντήρια πορεία σε όλη την περιοχή που καλύπτεται από το δίκτυο αισθητήρων και σημειώνεται από τις περιοχές κινδύνου ενώ προχωράει στην περιοχή εξόδου. Η δεξιά εικόνα δείχνει κάποιους αισθητήρες που χρησιμοποιούνται για τα πειράματά μας. Οι τρείς αισθητήρες τοποθετούνται σε όρθια θέση που υποδηλώνουν δύο εμπόδια (δηλαδή σε ακατάλληλες ζώνες) και έναν στόχο.*

4.4 Το Δίκτυο Αισθητήρων που διαμορφώθηκε στη breve

Το δίκτυο αισθητήρων που αναφέρονται στην προηγούμενη ενότητα διαμορφώθηκε στην breve χρησιμοποιώντας τα patch στην κατηγορία path class. Στην παρούσα εργασία τα patch ήταν ισομερώς κατανεμημένα πάνω από την περιοχή. Αλλά η απόδοση θα μπορούσε να βελτιωθεί με την τοποθέτηση του patch σε ένα συγκεκριμένο μοτίβο ώστε να ελαχιστοποιηθεί ο αριθμός των patches και να μεγιστοποιηθεί η ασφάλεια ενός οχήματος πλοήγησης μέσα από την περιοχή με «κίνδυνο». Στο δίκτυο αισθητήρων οι αισθητήρες θα αισθανθούν τις εξαιρετικές καταστάσεις ηλεκτρονικά. Προσομοιώνοντας στην breve τοποθετούμε φωτεινά εμπόδια κατά την έκταση των κελιών και μπορεί να καθορίσουν τα patch που έχουν εμπόδια ή τα patch που δεν είναι ασφαλή.



Σχήμα 4.2 *Το δίκτυο αισθητήρων που φαίνεται στο σχήμα 3.1 έχει διαμορφωθεί με patch στην breve. Τα κελιά που περιέχουν τα κόκκινα φωτεινά αντικείμενα αντιπροσωπεύουν επικίνδυνες ζώνες. Το κελί που είναι το όχημα είναι το κελί-αφετηρία. Το σκούρο μπλε κελί είναι το κελί-στόχος. Οι πράσινες κηλίδες δείχνουν το δρόμο τον οποίο το όχημα πρέπει να λάβει.*

4.5 Επικοινωνίες Αισθητήρων

Τα μελλοντικά οπλικά συστήματα του στρατού θα εξαρτηθούν σε μεγάλο βαθμό από την χρήση μη επιβλεπόμενων δικτύων αισθητήρων για την ανίχνευση , τον εντοπισμό και την αναγνώριση των στόχων του εχθρού προκειμένου να επιβιώσουν με λιγότερη προστασία στα μελλοντικά πεδία των μαχών. Η τελευταία εσωτερική ασφάλεια όσον αφόρα τα αντιτρομοκρατικά μέτρα θα βασιστεί επίσης σε μεγάλο βαθμό από μη επιβλεπόμενα δίκτυα αισθητήρων για την ανίχνευση, τον εντοπισμό και την αναγνώριση τρομοκρατικών επιθέσεων στις κρίσιμες πολιτικές υποδομών. Η επιτυχής εφαρμογή αυτών των κρίσιμων δικτύων επικοινωνίας θα απαιτήσουν την συλλογή των δεδομένων του αισθητήρα, την επεξεργασία και την αντιπαραβολή με διαθέσιμη την νοημοσύνη, την μεταφορά. Στόχος όλων των παραπάνω η μορφή να είναι τέτοια ώστε να τα ευνοεί να παίρνουν γρήγορες και ακριβείς αποφάσεις. Οι επικοινωνίες δικτύων πρέπει να υποστηρίζουν την ασφαλή μετακίνηση των δεδομένων από αισθητήρες , σε συνθήκες συμφόρησης του δικτυού.

4.6 Τεχνικές Ανάπτυξης Αισθητήρων

Οι αισθητήρες τοποθετήθηκαν σε ομοιόμορφη απόσταση και συνδέθηκαν μεταξύ τους. Το πρόβλημα εγκατάστασης του αισθητήρα στο πλαίσιο της αβεβαιότητας έλεγχου της θέσης του αισθητήρα λόγω διαλείψεων. Η εγκατάσταση αισθητήρων σε τέτοιες συνθήκες είναι εγγενώς μη-ντετερμινιστική και υπάρχει ένας ορισμένος βαθμός τυχαιότητας που σχετίζεται με την θέση ενός αισθητήρα στο πεδίο ενός άλλου αισθητήρα. Τα ασύρματα δίκτυα των αισθητήρων επιφορτισμένα με την παρατήρηση του περιβάλλοντος, την επεξεργασία δεδομένων και την λήψη αποφάσεων με βάση αυτές τις παρατηρήσεις έχουν προσελκύσει την ιδιαίτερη προσοχή. Τέτοια δίκτυα μπορούν να χρησιμοποιηθούν για την παρακολούθηση του περιβάλλοντος να ανιχνεύουν, να ταξινομούν, να εντοπίζουν συγκεκριμένα γεγονότα και να παρακολουθούν τους στόχους σε συγκεκριμένη περιοχή. Η τοπολογία του πεδίου του αισθητήρα, δηλαδή οι θέσεις των αισθητήρων καθορίζουν σε μεγάλο βαθμό την ποιότητα της κάλυψης που παρέχεται από το δίκτυο αισθητήρων. Ωστόσο ακόμα και όταν οι θέσεις του αισθητήρα έχουν προϋπολογιστεί για βέλτιστη κάλυψη και την αξιοποίηση των πόρων υπάρχουν εγγενείς αβεβαιότητες της θέσης αισθητήρων όταν διασκορπίζονται οι αισθητήρες ή υπάρχουν διαλείψεις. Έτσι μια βασική πρόκληση στην ανάπτυξη του αισθητήρα είναι να καθοριστεί μια αβεβαιότητα που θα γνωρίζει την αρχιτεκτονική του αισθητήρα ώστε να μειώνει το κόστος και να παρέχει υψηλή κάλυψη έστω και αν η ακριβής θέση των αισθητήρων μπορεί να είναι

ελεγχόμενη. Σε εφαρμογές όπως η επιτήρηση του πεδίου μάχης και η παρακολούθηση του περιβάλλοντος οι αισθητήρες μπορούν να πέσουν από τα αεροπλάνα. Οι εν λόγω αισθητήρες δεν μπορούν να αναμένονται να πέσουν ακριβώς στις προκαθορισμένες θέσεις. Σε υποβρύχια εγκατάσταση οι αισθητήρες μπορούν να κινούνται λόγω της ολίσθησης του νερού ή των ρευμάτων που δημιουργούνται από τα νερά. Έτσι η θέση των αισθητήρων μπορεί να μην είναι ακριβώς γνωστή και για κάθε σημείο στο πεδίο του αισθητήρα να υπάρχει μόνο μια πιθανότητα, ο αισθητήρας να βρίσκεται σε εκείνο το σημείο. Δύο αλγόριθμοι για την ανάπτυξη αισθητήρων παρουσιάστηκαν χωρίς την προϋπόθεση οι θέσεις του αισθητήρα να μην είναι ακριβώς προκαθορισμένες. Είχε επίσης σημειωθεί ότι οι θέσεις των αισθητήρων υπολογίστηκαν πριν από την αποστολή και έγινε μια προσπάθεια κατά την ρίψη να τοποθετήσουν αισθητήρες στα προαποφασισμένα σημεία.. Ωστόσο ο αισθητήρας υπολογισμού τοποθέτησης και βελτιστοποίησης, κάλυψης βασίστηκαν σε ένα μοντέλο Gaussian η οποία θεωρούσε ότι ένας αισθητήρας προορίζεται για ένα συγκεκριμένο σημείο p στο πεδίο ενός άλλου αισθητήρα. Το πεδίο του αισθητήρα αναπαριστάται ως πλέγμα μονάδων δύο ή τριών διαστάσεων .Ένας στόχος στον τομέα του αισθητήρα είναι συνεπώς ένα λογικό αντικείμενο το οποίο αναπαρίσταται από ένα σύνολο αισθητήρων που μπορούν να το δουν.. Το πλαίσιο βελτιστοποίησης ωστόσο είναι ενδογενώς στοχαστικό λόγω της αβεβαιότητας που συνδέεται με αισθητήρα ανίχνευσης.

Δύο αλγόριθμοι προτάθηκαν για την τοποθέτηση του αισθητήρα που απευθύνεται στην βελτιστοποίηση της κάλυψης υπό τους περιορισμούς της ασάφειας ανίχνευσης και τις ιδιότητες του εδάφους. Οι αλγόριθμοι τοποθέτησης έδωσαν τις θέσεις του αισθητήρα πριν από την πραγματική τοποθέτηση και υποθέτουμε ότι οι αισθητήρες έχουν αναπτυχθεί σε ένα μόνο βήμα.

4.7 Αισθητήρες Κίνησης

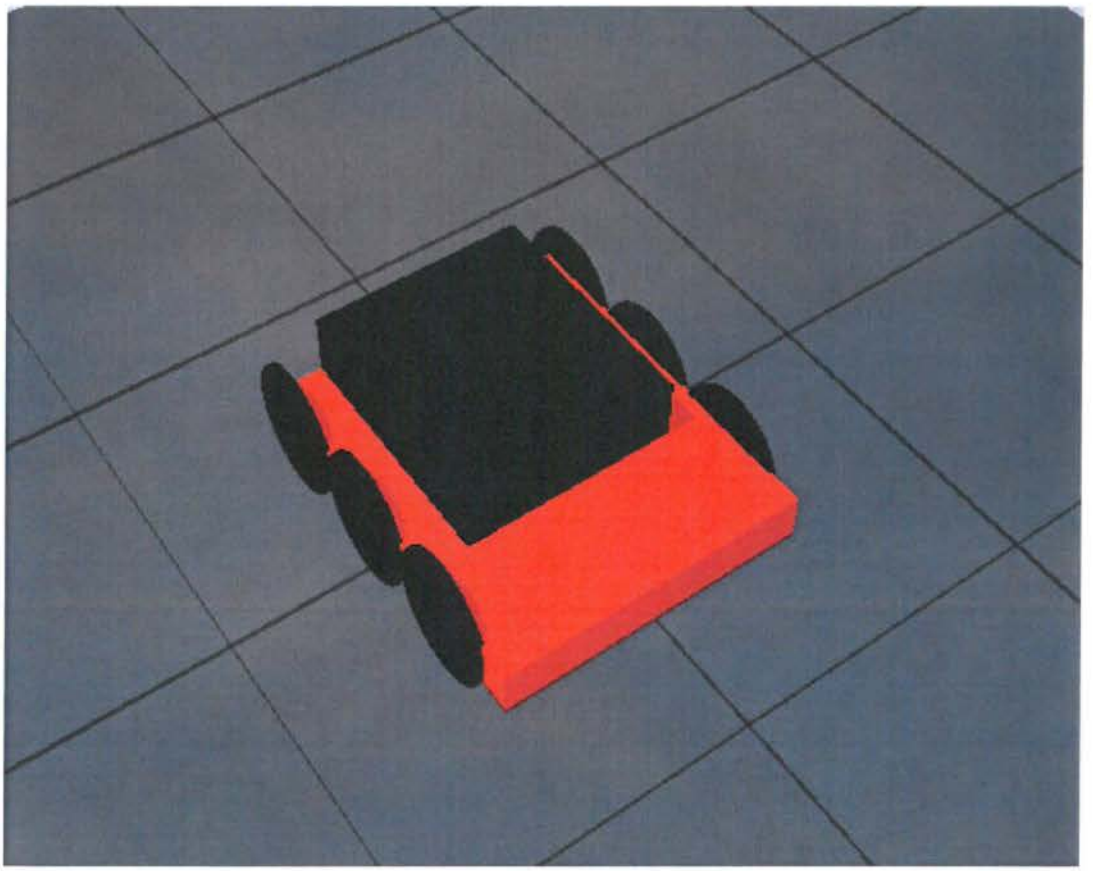
Σε ένα δυναμικό περιβάλλον όπου οι τιμές του αισθητήρα μπορούν να αλλάξουν από κίνδυνο για ασφάλεια ή και αντίστροφα πρέπει να υπολογίσει εκ νέου διαδρομή για να πλοηγηθεί το όχημα και αντί του υπολογισμού ολόκληρης διαδρομής μπορεί να χρησιμοποιηθεί ο αλγόριθμος A*.

ΚΕΦΑΛΑΙΟ 5

5.1 Εισαγωγή

Σε αυτό το κεφάλαιο γίνεται αναφορά στην κωδικοποίηση που αναπτύχτηκε για τον σχεδιασμού του οχήματος που χρησιμοποιήθηκε. Απεικονίζονται τα κελιά και τα Φώτο-αντικείμενα που έχουν μοντελοποιηθεί ως εμπόδια καθώς και ορισμένες από τις λειτουργίες τους. Η εφαρμογή του γενικού αλγορίθμου αναζήτησης και ο αλγόριθμος A* μελετήθηκαν μαζί με το ψευδοκώδικα και τα πειραματικά αποτελέσματα .Συζητήθηκε επίσης ένας πρόσθετος ευριτικός κανόνας που προστέθηκε στον αλγόριθμο A*.

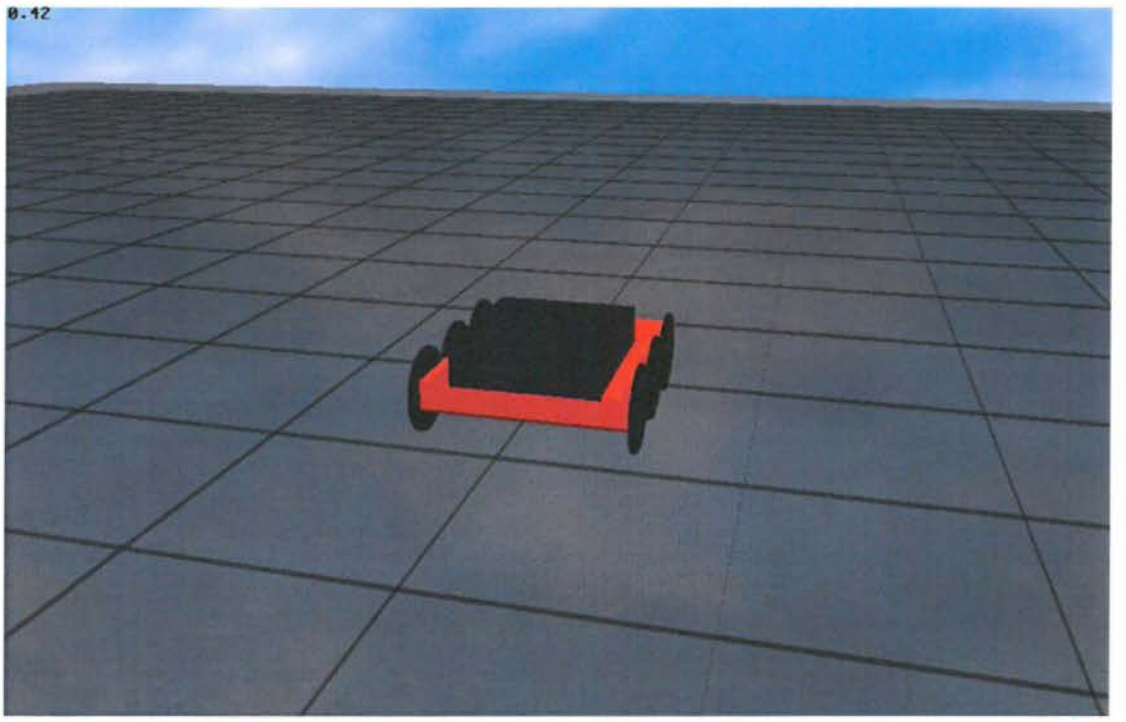
5.2 Οι αλλαγές στη braitenberg class



Εικόνα 5.1 *Σχέδιο οχήματος που χρησιμοποιείται στην προσομοίωση*

5.3 Patches

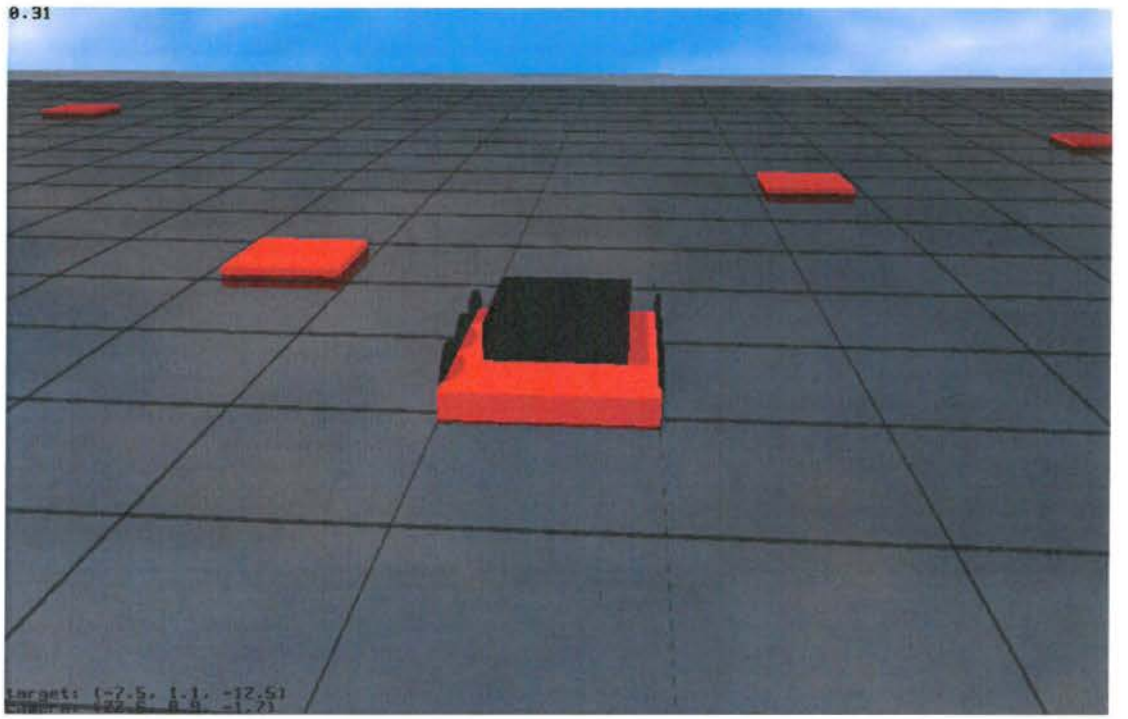
Σύμφωνα με την αρχή της διάσπασης κελιών ο χώρος καλύφθηκε από ίσου μεγέθους κελιά. Τα κελιά έχουν μια αίσθηση θέσης. Το σχήμα 5.2 δείχνει τα κελιά που καλύπτουν ολόκληρο το χώρο εργασίας. Δεν είναι δυνατόν να συλλάβει το σύνολο του χώρου εργασίας και ως εκ τούτου και ένα μέρος έχει αποδειχτεί. Με την αλλαγή των τιμών x και z δεν είναι δυνατόν να δημιουργηθούν κελιά από διαφορετική διάσταση όπως 4*4 ή 6*6 κλπ. Ένα 32*32 πλέγμα θα καλύψει το σύνολο του χώρου εργασίας που δημιουργείται από την προσομοίωση.



Εικόνα 5.2 *Κελιά που χρησιμοποιούνται στην προσομοίωση*

5.4 Τα Φώτο-αντικείμενα διαμορφώνονται ως εμπόδια

Τα Φώτο-αντικείμενα τα οποία αποτελούν μέρος της κατηγορίας braitenberg χρησιμοποιήθηκαν ως εμπόδια. Τα ελαφριά αντικείμενα είναι κινητά αντικείμενα και μπορούν να μετακινούνται κατά την διάρκεια της προσομοίωσης η οποία μας δίνει δυναμικά τα εμπόδια που είναι τα εμπόδια που θα αλλάξει τη θέση με το χρόνο και σε αυτή την εργασία ήταν σε θέση να αναλάβει την δυναμική φύση των εμποδίων στην προσοχή και να πάρει μια ασφαλή διαδρομή αποφεύγοντας τα εμπόδια σε κάθε επανάληψη του κύκλου. Το μέγεθος των patches του οχήματος και τα εμπόδια είναι όλα ίδια. Το μέγεθος του οχήματος και του patch προκειμένου να διευκολυνθεί η πλοήγηση του οχήματος όταν κινείται από το ένα κελί στο άλλο.



Σχήμα 5.3 *: Το patch του οχήματος και το μέγεθος του εμποδίου είναι ίδια.*

5.5 Εφαρμογή του γενικού αλγόριθμου αναζήτησης

Ο πρώτος αλγόριθμος που υλοποιήθηκε στην εργασία αυτή ήταν ο γενικός αλγόριθμος αναζήτησης. Αυτός ο τυχαίος αλγόριθμος αναζήτησης υλοποιήθηκε προκειμένου να βεβαιωθεί ότι μπορεί να εφαρμοστεί ένας αλγόριθμος αναζήτησης χρησιμοποιώντας την γλώσσα Steve.

Ο αλγόριθμος κολλάει κατά την υπέρβαση ενός ορίου στον αριθμό των κελιών ή των κόμβων που το λογισμικό μπορεί να υπολογίσει σε μια δεδομένη στιγμή λαμβάνοντας υπόψη την διαθέσιμη μνήμη. Ως εκ τούτου απαιτείται μελέτη της χωρικής πολυπλοκότητας . Οι μέγιστες διαδρομές και οι μέγιστοι κόμβοι ορίζονται ως εξής:

→ Μέγιστες διαδρομές: Το μέγιστο πλήθος διαδρομών που ο αλγόριθμος υπολογίζει κατά την διάρκεια μιας επανάληψης όπως πηγαίνει για να βρει το τελικό ασφαλές μονοπάτι.

→ Μέγιστοι κόμβοι: Ο αριθμός των κόμβων στο μονοπάτι που περιέχει τον μέγιστο αριθμό των κόμβων πριν από ένα ασφαλές κελί που βρέθηκε.

Έτσι διαπιστώνεται ότι ο αλγόριθμος αυτός σταματά να λειτουργεί όταν έχει να υπολογίσει ένα τεράστιο αριθμό των μέγιστων διαδρομών σε μια επανάληψη.

5.6 Εφαρμογή του αλγόριθμου A*

Ο A* είναι ένας αλγόριθμος που χρησιμοποιείται για την εύρεση διαδρομών. Πρόκειται για μια επέκταση του αλγορίθμου djkstra. Ο A* επιτυγχάνει την καλύτερη απόδοση χρόνου με την χρήση ευρετικών μηχανισμών.

Όπως όλοι οι αλγόριθμοι αναζήτησης πρώτα αναζητούν τις διαδρομές που είναι πιο πιθανό να οδηγηθούν προς το στόχο. Ο A* «θυμάται» τη διαδρομή που ήδη έχει ταξιδέψει στο παρελθόν. Η g(x) είναι το κόστος από το σημείο εκκίνησης, δεν είναι το κόστος που έχει επεκταθεί σε έναν κόμβο. Ο A* χρησιμοποιεί μια εικονική εκτίμηση του κόστους μιας διαδρομής από κάθε κόμβο που θεωρεί ότι το πραγματικό κόστος μιας διαδρομής μέσω αυτού του κόμβου και του στόχου θα πρέπει να είναι τουλάχιστον τόσο μεγάλη όσο η εκτίμηση.

Ξεκινώντας από τον αρχικό κόμβο έχει δημιουργηθεί μια «ουρά» με προτεραιότητα κόμβων που πρέπει να διέλθει . Προτεραιότητα έχει ο κόμβος x με την μικρότερη τιμή της f(x). Σε κάθε βήμα ο κόμβος με την χαμηλότερη τιμή της f(x) αφαιρείται από την «ουρά» και οι τιμές των f και g των «γειτονικών» κόμβων ενημερώνονται και προστίθενται οι κόμβοι στη «ούρα». Ο αλγόριθμος συνεχίζει έως ότου ο κόμβος-στόχος έχει μικρότερη τιμή από οποιονδήποτε κόμβο στην «ουρά» ή έως ότου η «ουρά» να είναι κενή. Από τον κόμβο-στόχος μπορεί να περάσει πολλές φορές εάν εξακολουθούν να υπάρχουν άλλοι κόμβοι με χαμηλότερες τιμές της f για να οδηγηθεί στον κόμβο-στόχο με την συντομότερη διαδρομή.

Εάν ο ευρετικός κανόνας είναι αποδεκτός σημαίνει ότι ποτέ δεν εκτιμά το πραγματικό κόστος για την επίτευξη του στόχου

Αυτό σημαίνει ότι για κάθε ζεύγος γειτονικών κόμβων x και y όπου d(x ,y) συμβολίζεται το μήκος της ακμής μεταξύ τους ισχύει:

$h(x) \leq d(x, y)+h(y)$

Αυτό εξασφαλίζει ότι για κάθε x διαδρομή από τον αρχικό κόμβο στο x:

$L(x)+h(x) \leq L(x)+d(x,y)+h(y)=L(y)+h(y)$ όπου

- L η συνάρτηση που υποδηλώνει το μήκος μιας διαδρομής και το
- y είναι το x μονοπάτι που επεκτάθηκε για να συμπεριλάβει το y.

Μονοτονία σημαίνει δεκτό όταν σε οποιονδήποτε κόμβο ο ίδιος ο στόχος είναι μηδέν με P=(f,v1,v2,v3..,vn,g) είναι η συντομότερη διαδρομή από οποιονδήποτε κόμβο f στο πλησιέστερο κόμβο g ισχύει:

$h(f) \leq d(f,v1)+h(v1) \leq d(f,v1)+d(v1,v2)+h(v2) \leq \ldots \leq L(P)+h(g)=L(P)$

5.6.1 Πολυπλοκότητα

Η χρονική πολυπλοκότητα του αλγόριθμου A* εξαρτάται από τον ευρετικό μηχανισμό. Στη χειρότερη περίπτωση ο αριθμός των κόμβων που επεκτείνονται είναι εκθετικός προς το μήκος της συντομότερης διαδρομής αλλά είναι πολυώνυμο όταν ο χώρος αναζήτησης είναι δέντρο(=tree) υπάρχει μία ενιαία κατάσταση στόχου και η ευρετική συνάρτηση h πληρεί τον ακόλουθο όρο:

$|h(x)-h^*(x)| = O(\log h^*(x))$

όπου h* είναι ο βέλτιστος ευρετικός κανόνας το ακριβές κόστος για να πάρει από το x στο στόχο δηλαδή το σφάλμα h δεν θα αυξηθεί γρηγορότερα από τον λογάριθμο του ευρετικού κανόνα h* που επιστρέφει την πραγματική απόσταση από το x στο στόχο.

5.6.2 Λήψη της Βέλτιστης Διαδρομής

Όταν υπάρχουν δύο ή περισσότερες διαδρομές όπου τελειώνουν στον ίδιο κόμβο είναι λογικό να ακολουθήσει την βέλτιστη δυνατή διαδρομή και να αγνοεί τις υπόλοιπες. Αυτό θα μας βοηθήσει να μειώσουμε την πολυπλοκότητα χώρου που υπάρχει στον βασικό αλγόριθμο αναζήτησης . Έτσι κάθε φορά που διαπιστώνουμε ότι υπάρχουν περισσότερες από δύο διαδρομές με τον ίδιο κόμβο τερματισμού υπολογίζουμε το άθροισμα της διανυόμενης απόστασης και της απόστασης που απομένει μέχρι τον κόμβο στόχο και επιλεγούμε την διαδρομή με το χαμηλότερο κόστος χωρίς τα υπόλοιπα να λαμβάνονται καν υπόψη. Ο αλγόριθμος A* χρησιμοποιεί αυτόν τον ευριτικό κανόνα και ταξινομεί τις διαδρομές με το χαμηλότερο κόστος διαδρομής πρώτα έτσι ώστε η διαδρομή χαμηλότερου κόστους να επεκτείνεται πρώτη.

5.7. Τα πλεονεκτήματα-μειονεκτήματα του αλγόριθμου A*

Ο αλγόριθμος A* αποδίδει πολύ καλύτερα από ότι ο γενικός αλγόριθμος αναζήτησης. Δεδομένου ότι ένας αλγόριθμος A* επεκτείνει το χαμηλότερο κόστος μονοπατιών πρώτα, αναμένεται να δώσει τη βέλτιστη διαδρομή.

Με δεδομένη μια λίστα που περιέχει διαδρομές που λήγουν στο ίδιο κόμβο ένας αλγόριθμος A* επιλέγει αυτή με το χαμηλότερο κόστος κ αγνοεί τα υπόλοιπα. Αυτό δεν είναι πολύ αποτελεσματικοί για 2 λόγους:

- Η διαδρομή που έχει περισσότερο κόστος προς το παρόν ίσως μακροπρόθεσμα να είναι φθηνότερη.
- Η βιαστική διαγραφή διαδρομών μπορεί να διαγράψει και τον ο μόνο δρόμος που μπορεί να διατίθεται σε ορισμένες περιπτώσεις

.

5.8 Πρόσθεση έναν επιπλέον ευριτικού κανόνα στον αλγόριθμο A*

Μετά την προσθήκη του ευρυτικού κανόνα μπορεί να εξαχθεί το συμπέρασμα ότι η προσθήκη αυτή επιπλέον ευριστικού κανόνα έχει βελτιώσει σίγουρα την απόδοση του αλγόριθμου A* σε όσον αφορά το χρονικό διάστημα που απαιτείται. Όσον αφορά τα κελιά και τα πλέγματα έχει μειώσει τον αριθμό των μονοπατιών που επεκτάθηκε από τον αλγόριθμο A* ακόμη και αν δεν υπάρχει βελτίωση στο μέγιστο αριθμό κόμβων.

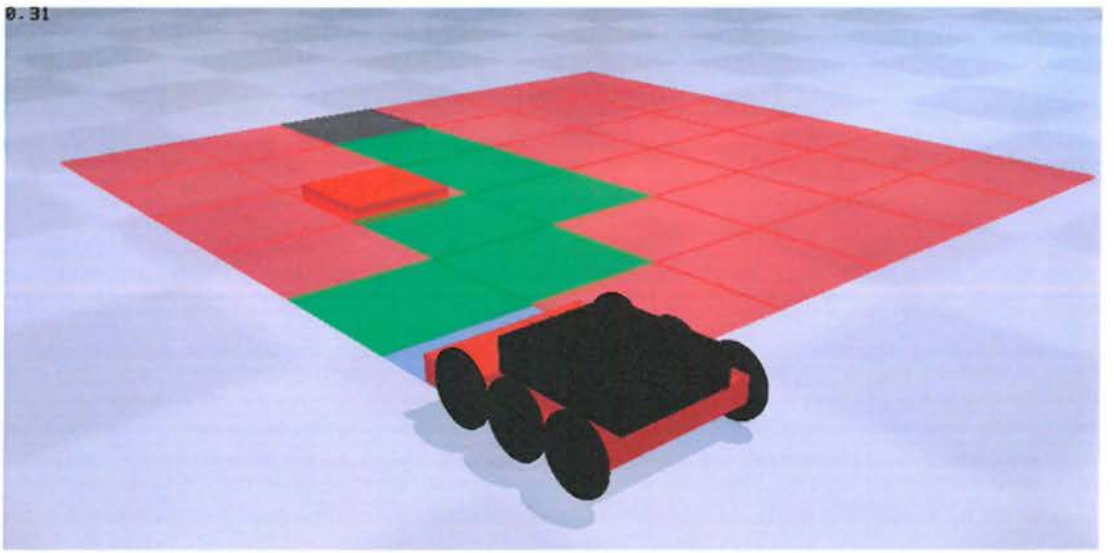
5.9 πλεονεκτήματα- μειονεκτήματα του τροποποιημένου Α* αλγόριθμου

Η προσθήκη ενός απλού κοινού ευριτικού κανόνα έχει βελτιώσει την απόδοση του αλγόριθμου Α* με την διαγραφή των αναποτελεσματικών μονοπατιών.

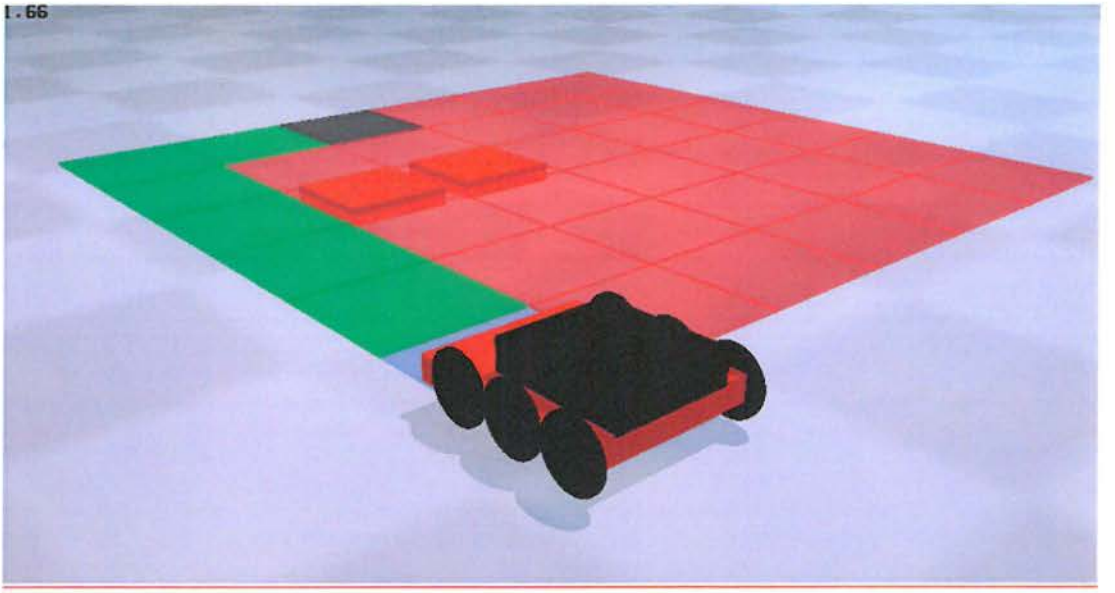
Μαζί με τα μειονεκτήματα που αναφέρθηκαν για τον αλγόριθμο Α* η προσθήκη του νέου ευριτικού κανόνα εισάγει ένα άλλο θέμα όταν είναι εμπόδια παρόντα. Η παρουσία εμποδίων θα δώσει διαδρομές οι οποίες είναι μεγαλύτερες από τη μέγιστη έγκυρη διαδρομή. Έτσι όταν τα εμπόδια είναι παρόντα αυτός ο νέος ευριτικός κανόνας δεν θα μπορούσε να είναι σε θέση να μας δώσει μια λύση δεδομένου ότι διαγράφει όλα τα μονοπάτια μεγαλύτερα από την απόσταση μεταξύ του κόμβου έναρξης και του κόμβου στόχου. Αλλά το ζήτημα αυτό μπορεί να επιλυθεί κάνοντας την απόσταση μεγαλύτερη από τον αριθμό των παρόντων εμποδίων.

5.10 Δυναμική Αναζήτηση διαδρομής

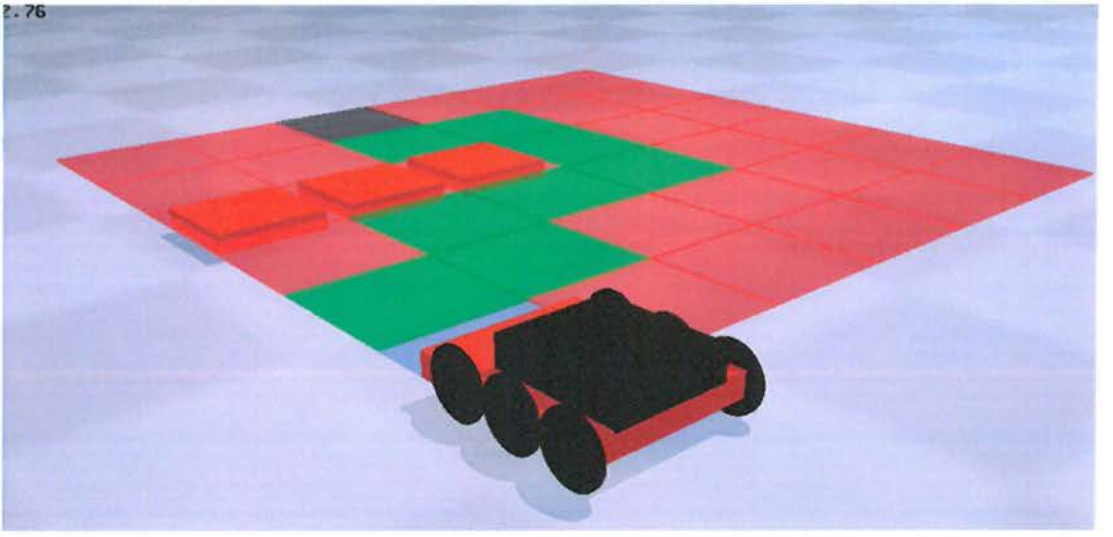
Δυναμική Αναζήτηση διαδρομής έχουμε όταν τα εμπόδια που κινούνται και αλλάζουν θέση με το χρόνο. Αυτό αποτελεί μια πρόκληση για το κινούμενο όχημα, δεδομένου ότι πρέπει να αλλάξει την πορεία της καθώς τα εμπόδια αλλάζουν την θέση τους. Τα στιγμιότυπα που λαμβάνονται από ένα μόνο «τρέξιμο» της προσομοίωσης σε διαφορετικά χρονικά βήματα εμφανίζονται δίπλα.



Εικόνα 5.4 *: αρχική εύρεση διαδρομής*



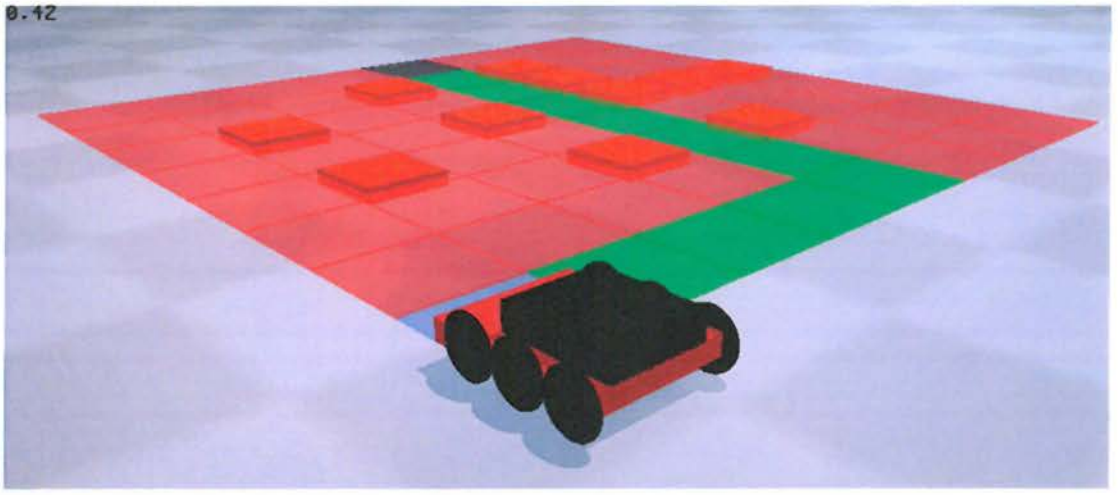
Σχήμα 5.5 *: Δείχνει την εισαγωγή ενός νέου εμποδίου το οποίο ήταν στο δρόμο του στην αρχική διαδρομή που βρέθηκε. Ο αλγόριθμος έχει βρει ένα νέο ασφαλές μονοπάτι αποφεύγοντας το νέο εισαχθέν εμπόδιο.*



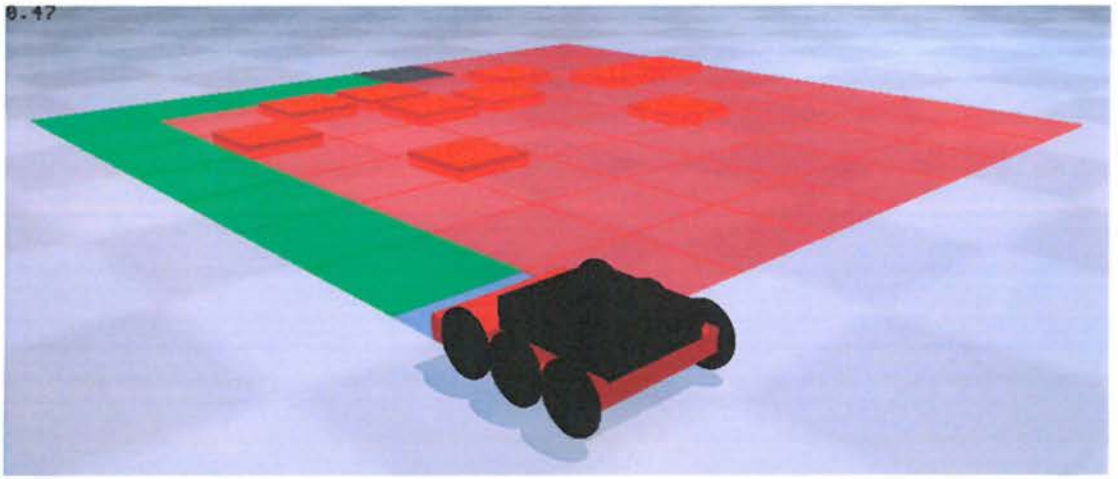
Σχήμα 5.6*: Δείχνει την εισαγωγή ενός άλλου νέου εμποδίου που ήταν και πάλι στη διαδρομή του οχήματος και μια νέα διαδρομή βρέθηκε.*

5.11 Αναζήτηση Διαδρομής στην εσωτερική ασφάλεια των ρομπότ

Τέλος μπορούμε να εξετάσουμε τώρα μια πορεία βρίσκοντας τεχνικές για τα ρομπότ που χρησιμοποιούνται στην εσωτερική ασφάλεια. Οι ακόλουθες φωτογραφίες δείχνουν το όχημα, λαμβάνοντας ασφαλή διαδρομή αποφεύγοντας τα «σημεία κινδύνου».

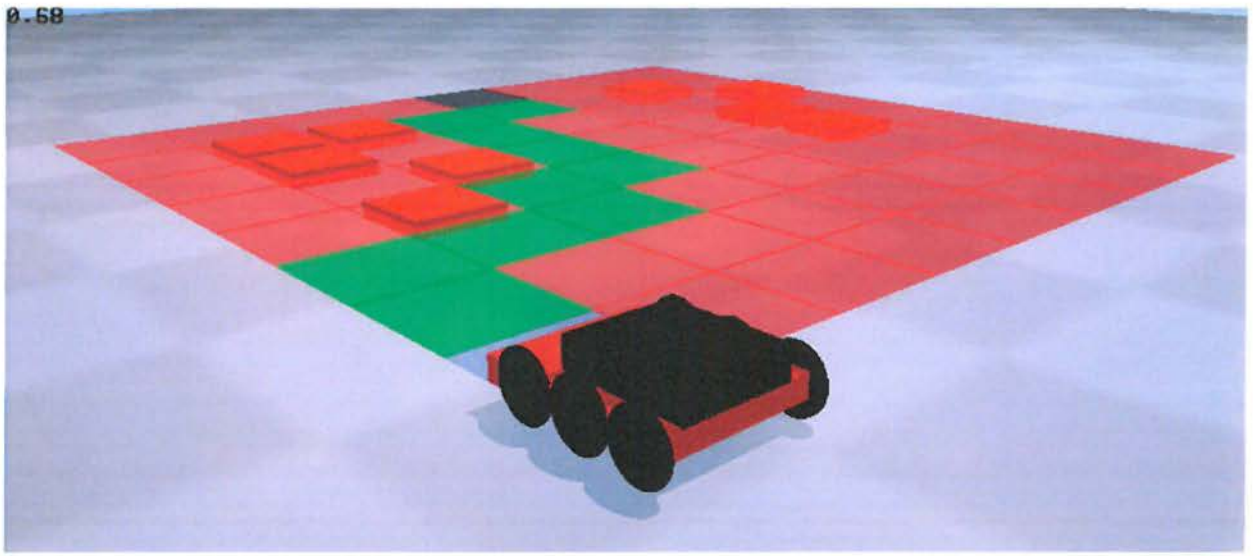


Σχήμα 5.7 *: μια ασφαλή διαδρομή βρίσκεται για το ρομπότ για να περιηγηθεί.*



Σχήμα 5.8*: Ένα άλλο στιγμιότυπο που δείχνει μια ασφαλή που βρέθηκε.*

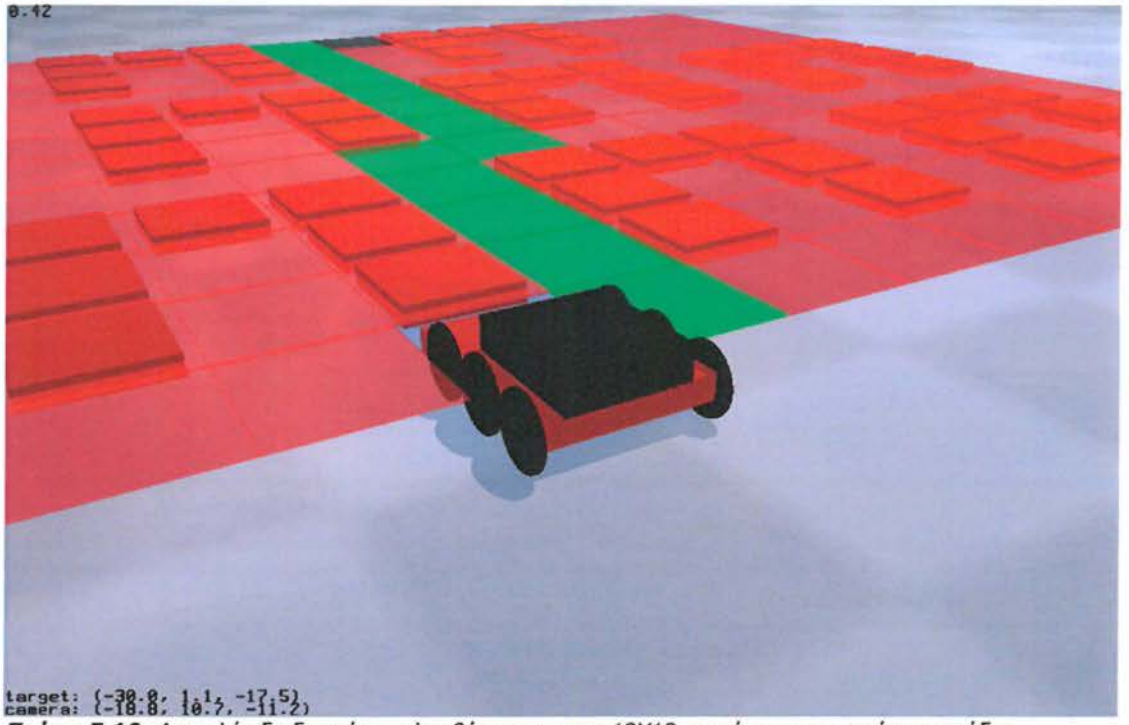
Τα στιγμιότυπα φαίνονται στο σχήμα 5.7 και 5.8 δείχνουν τα ασφαλή μονοπάτια που βρέθηκαν. Το στιγμιότυπο που φαίνεται στο σχήμα 5.9 είναι πολύ ενδιαφέρον διότι αν και το όχημα έχει βρει ένα ασφαλές μονοπάτι δεν έχει βρει την ασφαλέστερη διαδρομή. Μια προσεκτική εξέταση δείχνει ότι βρήκε μια διαδρομή που δεν φρόντισε να κρατήσει το όχημα σε μια μέγιστη απόσταση από τον εχθρό. Αυτό θα μπορούσε να είναι ένα σοβαρό πρόβλημα. Σκεφτείτε για παράδειγμα μια μονάδα του στρατού που διακινείται μέσω ενός στρατοπέδου σε πεδίο μάχης. Για την αποφυγή του εχθρού θα είναι πολύ επιθυμητό να περιηγηθεί η μονάδα του στρατού η του οχήματος όσο πιο μακριά από τα «επικίνδυνα σημεία».



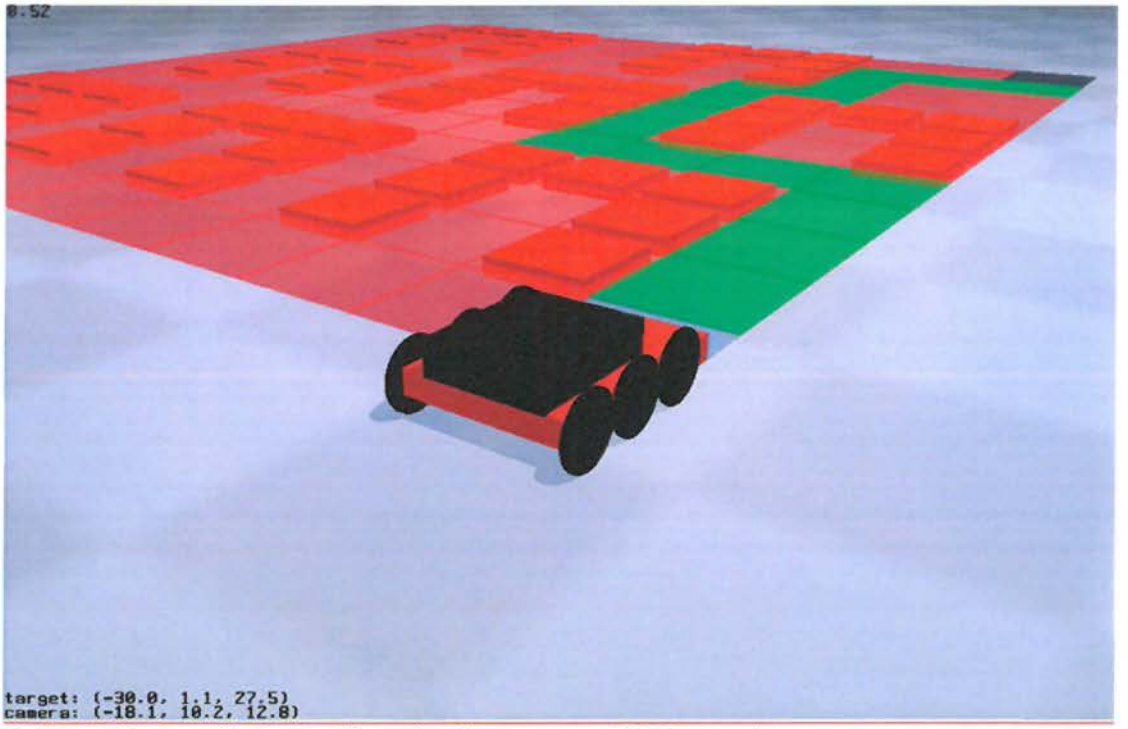
Σχήμα 5.9. *μια ασφαλή διαδρομή αλλά όχι η ασφαλέστερη που βρέθηκε.*

5.12 Τοποθέτηση Εμποδίων

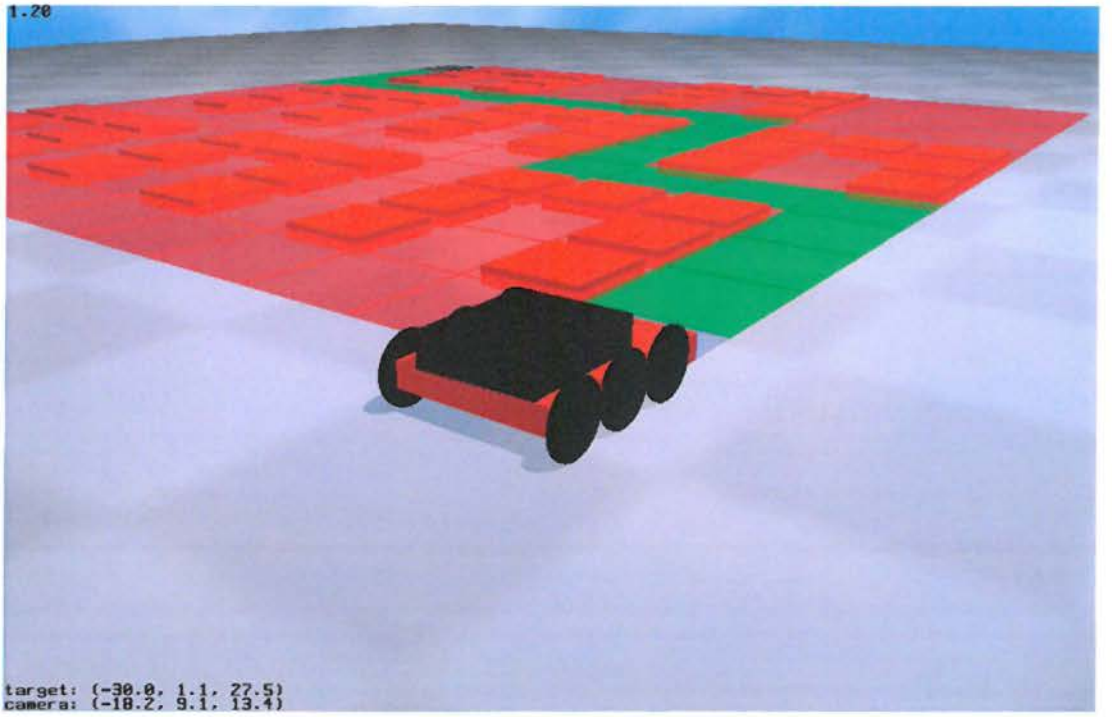
Οι ακόλουθες φωτογραφίες δείχνουν τις διαδρομές που βρέθηκαν για πολύπλοκες ρυθμίσεις των εμποδίων.



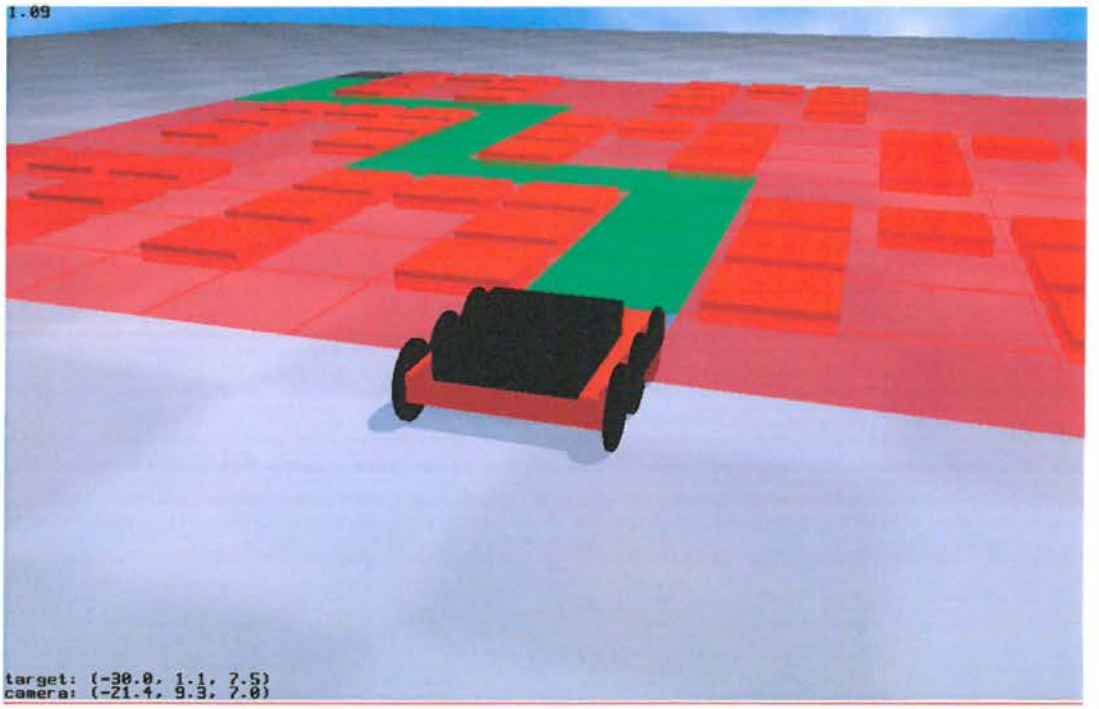
Σχήμα 5.10: *Ασφαλής διαδρομή που λαμβάνεται σε μια 12X12 τετράγωνη περιοχή με εμπόδια.*



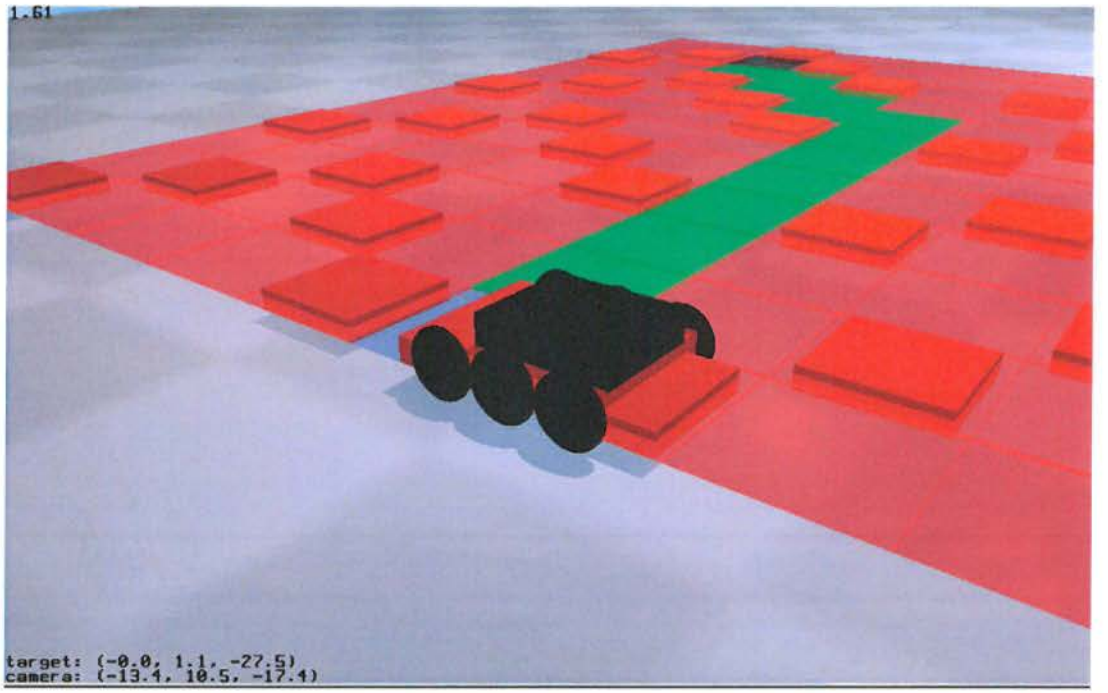
Σχήμα 5.11*: Ασφαλής διαδρομή που λαμβάνεται σε μια 12X12 περιοχή*



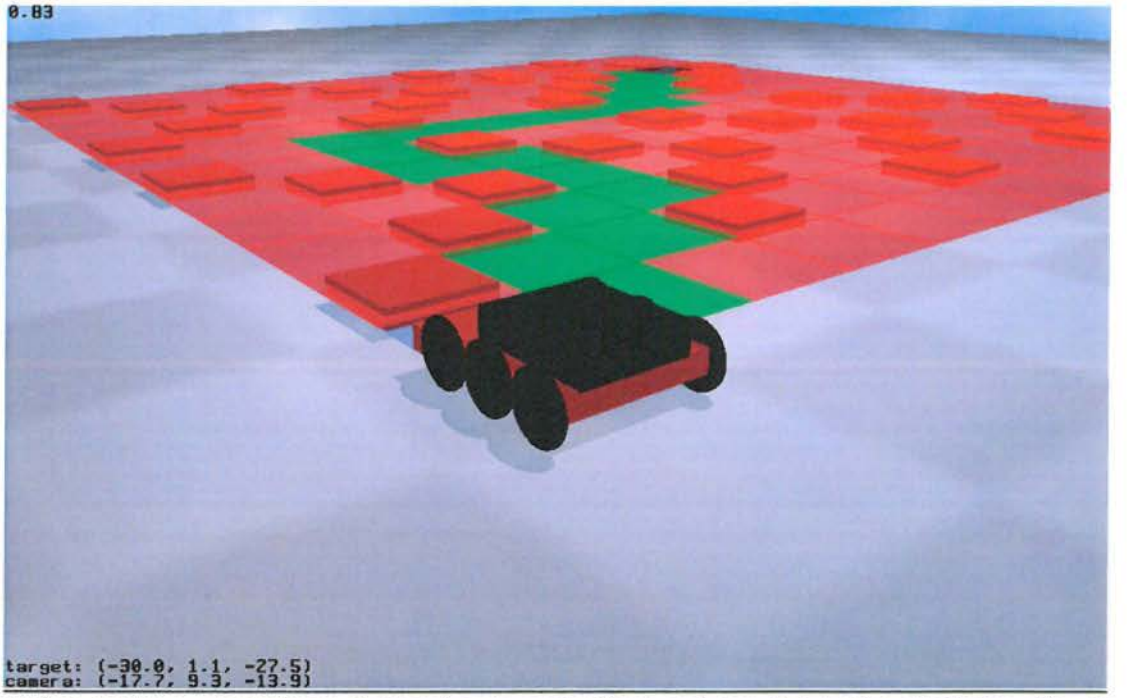
Σχήμα 5.12: *Ασφαλής διαδρομή που λαμβάνεται σε μια 12χ12 περιοχή*



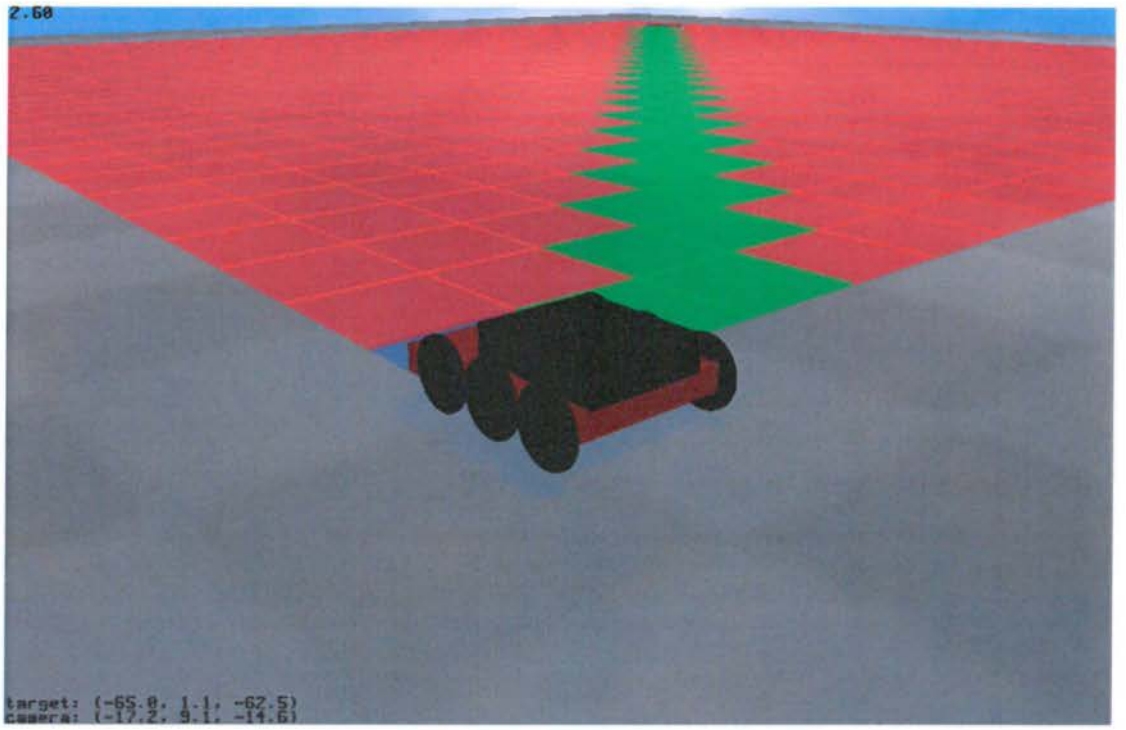
Σχήμα 5.13 *Ασφαλής διαδρομή με τοποθετημένα εμπόδια σε τετραγωνική περιοχή 12χ12*



Σχήμα 5.14: *η μεγαλύτερη διαδρομή που υπολογίζεται από τον βελτιωμένο αλγόριθμο A**



Σχήμα 5.15: *η μεγαλύτερη διαδρομή που υπολογίζεται από τον βελτιωμένο αλγόριθμο A* σε τετραγωνική περιοχή 12χ12*



Σχήμα 5.16 *:η μεγαλύτερη διαδρομή που υπολογίζεται από τον βελτιωμένο από τον αλγόριθμο A*.*

ΚΕΦΑΛΑΙΟ 6

ΣΥΜΠΕΡΑΣΜΑΤΑ – ΑΠΟΤΕΛΕΣΜΑΤΑ

Η εργασία αυτή ανέπτυξε και ανέλυσε τη χρήση διάφορων τεχνικών εύρεσης διαδρομών βρίσκοντας τεχνικές στον ίδιο χώρο εργασίας που έχει σημασία για τα ρομπότ που χρησιμοποιούνται σε εφαρμογές εσωτερικής ασφάλειας.

Ο στόχος του έργου ήταν να αναλύσει την απόδοση των διάφορων τεχνικών εύρεσης διαδρομών και να καταλήξει σε μια τεχνική συνδυάζοντας τις δυνάμεις του και εξουδετερώνοντας τα μειονεκτήματά τους.

Διαφορετικοί αλγόριθμοι αναζήτησης χρησιμοποιήθηκαν στον ίδιο χώρο εργασίας και ο χρόνος που απαιτείται για τον υπολογισμό διαδρομών χρησιμοποιώντας τους διάφορους αλγόριθμους που καταγράφηκαν πιο αποτελεσματικοί στη breve.

Η αναζήτηση του αλγόριθμου A* χρησιμοποιεί ένας πολύ αποτελεσματικός ευρετικός κανόνας και ταξινομεί επίσης τις διαδρομές έτσι ώστε η διαδρομή εύρεσης είναι πιο αποτελεσματική. Τέλος ένα πρόσθετος ευρετικός κανόνας προστέθηκε στον αλγόριθμο A* και αποδείχθηκε μέσω πειραματικών στοιχείων ότι βελτιώνει την απόδοση του αλγόριθμου.

Αυτός ο βελτιωμένος αλγόριθμος A* χρησιμοποιήθηκε στην συνέχεια για τα ρομπότ που χρησιμοποιούνται σε εφαρμογές εσωτερικής ασφάλειας . Διαπιστώθηκε ότι ο αλγόριθμος μας ήταν πολύ αποτελεσματικός όσον αφορά την εύρεση ενός ασφαλούς μονοπατιού αποφεύγοντας όλα τα «σημεία κινδύνου». Διαπιστώθηκε επίσης ότι ο αλγόριθμος αυτός βρίσκει μερικές φορές την συντομότερη διαδρομή αντί για την ασφαλέστερη διαδρομή που θα είναι απαραίτητη σε εφαρμογές εσωτερικής. Ως εκ τούτου μπορεί να εξαχθεί το συμπέρασμα ότι αν ο αλγόριθμος που αναπτύχθηκε στην παρούσα εργασία μπορούν να χρησιμοποιηθούν σε_εφαρμογές εσωτερικής ασφάλειας, δεν μπορεί να χρησιμοποιηθεί σε ευαίσθητες αποστολές κρίσιμων εφαρμογών εσωτερικής ασφάλειας όπου είναι ζήτημα ζωής και θανάτου. Μια άλλη ενδιαφέρουσα τεχνική εύρεσης διαδρομής χρησιμοποιήθηκε σε δυναμικά πεδία όπου συζητείται στην ενότητα 2.6. Ωστόσο στο δυναμικό πεδίο μέθοδοι τείνουν να υποφέρουν από τυπικά ελάχιστα δηλαδή η πλοήγηση του οχήματος μπορεί να κολλήσει στις περιφέρειες τοπικών ελαχίστων πρέπει να ληφθεί μέριμνα για να αποφευχθεί αυτό.

Τα πλεονεκτήματα και τα μειονεκτήματα όλων των χρησιμοποιούμενων αλγορίθμων ήταν σαφείς. Αυτό θα μας επιτρέψει να διερευνήσουμε περισσότερο την πορεία βρίσκοντας τεχνικές και να καταλήξουμε σε μια πιο αποτελεσματική τεχνική συνδυάζοντας τα πλεονεκτήματα των διάφορων τεχνικών και διόρθωση των μειονεκτημάτων σε κάθε αλγόριθμο.

Οι αλγόριθμοι υποθέτουν ότι ο αισθητήρας θέσης είναι γνωστός. Η BREVE έχει μια αποτελεσματική μέθοδο χειρισμού σύγκρουσης χρησιμοποιώντας αυτό θα πρέπει να είναι δυνατή η εφαρμογή των τεχνικών κτίριο στο χάρτη. Το ρομπότ μπορεί να εξερευνήσει το περιβάλλον και να σηματοδοτήσει τα ασφαλή σημεία και τα επικίνδυνα σημεία. Ο χάρτης κτιρίου είναι ακόμα υπό αναπτυξιακό στάδιο στη κινητή ρομποτική. Ο ταυτόχρονος εντοπισμός είναι μια από τις πιο χρησιμοποιημένες τεχνικές για την κατασκευή του χάρτη.

Τα μειονεκτήματα των αλγόριθμων που εφαρμόζονται σε αυτή την εργασία μπορούν να ξεπεράσουν την χρήση του αλγόριθμου S*M*A το οποίο είναι μια τροποποίηση του αλγόριθμου A*. Οι αλγόριθμοι που εμφανίζονται στην παρούσα εργασία εξετάζουν μόνο την καλύτερη διαδρομή και δεν δίνουν την ευκαιρία για μια διαδρομή η οποία είναι προς το παρόν το κακό στο να εξελιχθεί σε ένα αποτελεσματικό δρόμο στο μέλλον. Ο αλγόριθμος A* από την άλλη πλευρά θυμάται την καλύτερη διαδρομή από τα «ξεχασμένα» μονοπάτια και επανέρχεται στο «ξεχασμένο» μονοπάτι αναδρομικά όταν διαπιστώνει ότι η νέα πορεία δεν κάνει όλα αυτά πολύ καλά.

Η εύρεση τεχνικής διαδρομής που αναπτύχθηκε σε αυτή την εργασία υλοποιήθηκε σε προσομοιωμένο περιβάλλον εσωτερικής ασφαλείας. Η τεχνική αυτή αποδείχθηκε να λάβει το μακρότερο χρόνο από ότι ο αλγόριθμος A*. Δεδομένης κάθε κόμβο αρχή, ο κόμβος στόχος και η τοποθέτηση εμποδίων αλγόριθμος επέστρεψε πάντα μια ασφαλή διαδρομή αποφεύγοντας όλα τα επικίνδυνα σημεία. Το μόνο μειονέκτημα βρέθηκε σε αυτό το αλγόριθμο ήταν ότι υπολογίζει το βέλτιστο μονοπάτι αντί το ασφαλέστερο μονοπάτι το οποίο μπορεί να είναι κρίσιμο σε ορισμένες εφαρμογές εσωτερικής ασφαλείας. Ως εκ τούτου μπορεί να καταλήξουμε στο συμπέρασμα ότι ο αλγόριθμος αυτός μπορεί να χρησιμοποιείται σε μη κρίσιμες εφαρμογές εσωτερικής ασφάλειας και ο αλγόριθμος πρέπει να τροποποιηθεί για χρήση σε κρίσιμες εφαρμογές εσωτερικής ασφάλειας.

ΒΙΒΛΙΟΓΡΑΦΙΑ

1. Artifical Intelligence Center PREVIEW
   http://www.ai.sri.com/shakey/
2. Αλγόριθμοι και Πολυπλοκότητα
   http://www.softlab.ntua.gr/~fotakis/algorithms.html
3. A* Pathfinding για Αρχάριους
   http://www.policyalmanac.org/games/aStarTutorial_greek.htm
4. Αλγόριθμοι Ευριστικής Αναζήτησης
   http://aibook.csd.auth.gr/include/slides/Chap04.pdf
5. http://en.wikipedia.org/wiki/A*_search_algorithm
6. Τεχνολογίες Υλοποίησης Αλγορίθμων
   https://www.ceid.upatras.gr/webpages/faculty/zaro/teaching/alg-eng/
7. http://www.faa.gov/documentLibrary/media/Advisory_Circular/TCAS%20II%20V7.1%20Intro%20booklet.pdf
8. Dalong Wang, Dikai Liu, and Gamini Dissanayake. A variable speed force eld method for multi-robot collaboration. In IROS, pages 2697{2702, 2006.
9. Διαδικτυακός Τόπος της ομάδας Arduino http://arduino.cc/en/
10. Σύγκριση των αλγορίθμων IDA* και A* στο 15-puzzle http://www.icte.uowm.gr/uploads/thesis/manoliadhs_am55_diplomatikh.pdf
11. B. Goertzel, Advances in Artificial General Intelligence: Concepts, Architectures and Algorithms (Frontiers in Artificial Intelligence and Applications)
12. http://intelligence.worldofcomputing.net/ai-search/iterative-deepening-a-star.html#.UtW3LScyM24
13. Βλαχάβας, Κεφαλάς, Βασιλειάδης, Κόκκορας, Σακελλαρίου, Τεχνητή Νοημοσύνη - Γ' Έκδοση, Εκδόσεις Γκιούρδας
14. http://www.daniweb.com/software-development/cpp/threads/351851/thread-safe-timer-for-c-callback
15. Stuart Russel και Peter Norvig, Τεχνητή Νοημοσύνη, μια σύγχρονη προσέγγιση, εκδόσεις Κλειδάριθμος
16. http://www.informit.com/articles/article.aspx?p=1881386&seqNum=2
17. http://www.apl.jhu.edu/~hall/AI-Programming/IDA-Star.html

ΚΩΔΙΚΑΣ

- Main

```
package astar22;
import java.lang.*;
/*
 * To change this template, choose Tools | Templates and open the template in
 * the editor.
 */
/**
 *
 * @author giannis
 */
public class Main {
   public static void main(String args[]){
        Startup s=new Startup();
        s.Initialise();
     }
  }
```

- Map

```
package astar22;
import java.awt.*;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.Serializable;
public class Map extends java.awt.Panel implements Serializable
{
  int w;
  int h;
  transient Image buffer;
  GridCell gridCell[][];
        public Map(int dim, int dim1)
        {
                super();
         w=dim;
         h=dim1;
         gridCell=new GridCell[w][h];
                setLayout(new GridLayout(w,h));
                setSize(insets().left + insets().right + 430,insets().top + insets().bottom
+ 270);
                //}}
    for(int i=0;i<w;i++){
       for(int j=0;j<h;j++){
         gridCell[i][j] = new GridCell();
         gridCell[i][j]:setPosition(new Point(i,j));
         add(gridCell[i][j]);
```

```
        }
      }
  }
  @Override
  public void paint(Graphics g){
     if(buffer == null){buffer = createImage(getBounds().width,getBounds().height);}
     Graphics bg = buffer.getGraphics();
     super.paint(bg);
     bg.setColor(Color.black);
     g.drawImage(buffer,0,0,null);
  }
  @Override
  public void update(Graphics g){
     paint(g);
  }
  public Point getStartPosition(){
     GridCell start = GridCell.getStartCell();
     return start.getPosition();
  }
  public GridCell[] getAdjacent(GridCell g){
     GridCell next[] = new GridCell[4];
     Point p = g.getPosition();
     if(p.y!=0){next[0]=gridCell[p.x][p.y-1];}
     if(p.x!=w-1){next[1]=gridCell[p.x+1][p.y];}
     if(p.y!=h-1){next[2]=gridCell[p.x][p.y+1];}
     if(p.x!=0){next[3]=gridCell[p.x-1][p.y];}
     return next;
  }
```

```
public GridCell getLowestAdjacent(GridCell g){
    GridCell next[] = getAdjacent(g);
    GridCell small = next[0];
    double dist = Double.MAX_VALUE;
    for(int i=0;i<4;i++){
      if(next[i]!=null){
        double nextDist = next[i].getDistFromStart();
        if(nextDist<dist && nextDist>=0){
          small = next[i];
          dist = next[i].getDistFromStart();
        }
      }
    }
    return small;
  }
  private void readObject(ObjectInputStream ois) throws IOException,
ClassNotFoundException{
    GridCell.tidy=false;
    ois.defaultReadObject();
    GridCell.setShowPath(false);
  } }
```

- GridCell

```
package astar22;
import java.awt.*;
import java.awt.event.MouseEvent;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.Serializable;
```

```
import java.util.Vector;
public class GridCell extends java.awt.Component implements Serializable
{
   public static final int SET_BLOCKS=0,SET_START=1,SET_FINISH=2;
   public static final double NORMAL = 1,BLOCK=Double.MAX_VALUE;
   private static double newBlockStrength = BLOCK;
   private static int editMode = SET_BLOCKS;
   private static GridCell startCell;
   private static GridCell finishCell;
   private boolean isStart = false;
   private boolean isFinish = false;
private static Vector cells = new Vector();
public static boolean tidy = false;
   private static boolean showPath = true;
   private static boolean showDist = true;
   private double cost = 1.0;
   private transient boolean used = false;
   private transient double distFromStart = -1;
   private transient double distFromFinish = -1;
private boolean partOfPath = false;
   private Point position;
  public GridCell(){
     cells.addElement(this);
     tidy=true;
     enableEvents(AWTEvent.MOUSE_EVENT_MASK);
  }
public GridCell(boolean block){
     this();
```

```
        setTotalBlock(block);
    }
public void setPosition(Point p){
        position = p;
    }
public Point getPosition(){
        return position;
    }
public static void setEditMode(int mode){
        editMode = mode;
        System.out.println("mode set");
    }
@Override
    public void processMouseEvent(MouseEvent e){
        super.processMouseEvent(e);
        if(e.getID()==e.MOUSE_CLICKED){
            setShowPath(false);
            switch(editMode){
                case(SET_BLOCKS):
                    if(cost!=newBlockStrength){cost=newBlockStrength;}
                    else{cost=NORMAL;}
                    repaint();
                    break;
                case(SET_START):
                    setStart(true);
                    break;
                case(SET_FINISH):
                    setFinish(true);
```

```
                break;
                }
        }
    }
@Override
    public Dimension getPreferredSize(){
        return new Dimension(10,10);
    }
public void addToPathFromStart(double distSoFar){
        used = true;

        if(distFromStart == -1){
            distFromStart = distSoFar+cost;
            return;
        }
        if(distSoFar+cost<distFromStart){
            distFromStart = distSoFar+cost;
        }
    }
public void addToPathFromFinish(double distSoFar){
        used = true;
        if(distFromFinish == -1){
            distFromFinish = distSoFar+cost;
            return;
        }
        if(distSoFar+cost<distFromFinish){
            distFromFinish = distSoFar+cost;
        }
```

```
    }
 public double getCost(){
        return cost;
    }
 public void setCost(double c){
        cost=c;
    }

    public static GridCell getStartCell(){
        return startCell;
    }
 public boolean isStart(){
        return startCell == this;
    }
public void setStart(boolean flag){
          if(flag){
             GridCell temp = this;
             if(startCell !=null){temp = startCell;temp.setStart(false);}
             startCell=this;
             isStart=true;
             repaint();
             temp.repaint();
          }
          else{
             isStart=false;
          }
  }
public static GridCell getFinishCell(){
```

```
        return finishCell;
    }
  public boolean isFinish(){
        return finishCell == this;
    }

    public void setFinish(boolean flag){
      if(flag){
         GridCell temp = this;
         if(finishCell!=null){temp=finishCell;temp.setFinish(false);}
         finishCell=this;
         isFinish=true;
         repaint();
         temp.repaint();
      }
      else{
           isFinish=false;
      }
   }
 public boolean isTotalBlock(){
      return cost==BLOCK;
   }
 public void setTotalBlock(boolean flag){
      if(flag){cost = BLOCK;}
      else{cost = NORMAL;}
   }
 public boolean isUsed(){
      return used;
```

```
    }

    private void resetCell(){
        used = false;
        setPartOfPath(false);
        distFromStart = distFromFinish = -1;
    }
  public static void reset(){
        for(int i=0;i<cells.size();i++){
            ((GridCell)cells.elementAt(i)).resetCell();
        }
    }
  private void clearCell(){
        setCost(NORMAL);
    }
  public static void clearAll(){
        for(int i=0;i<cells.size();i++){
            ((GridCell)cells.elementAt(i)).clearCell();
        }
    }
  public static void setNewBlockStrength(double s){
        if(s<0){newBlockStrength = BLOCK;}
        else{newBlockStrength = s;}
    }
  public static void setShowPath(boolean flag){
            showPath = flag;
    }
```

```
    public static boolean isShowPath(){
        return showPath;
    }
public boolean isPartOfPath(){
        return partOfPath;
    }
public void setPartOfPath(boolean flag){
        partOfPath = flag;
    }
public double getDistFromStart(){
        if(GridCell.startCell == this){return 0;}
        if(isTotalBlock()){return -1;}
        return distFromStart;
    }
@Override
    public void paint(Graphics g){
        Dimension size = getSize();
        g.setColor(Color.white);
        if(cost!=NORMAL){
                if(cost==BLOCK){g.setColor(Color.black);}
        }
        if(showPath && partOfPath){
            g.setColor(Color.yellow);
        }
        if(startCell == this){
            g.setColor(Color.green);
        }
        if(finishCell == this){
```

```
            g.setColor(Color.red);
        }
        g.fillRect(0,0,size.width,size.height);
    g.setColor(Color.black);
        if(showDist &&
    distFromStart>0){g.drawString(""+distFromStart,1,(int)(size.height*0.75));}
        g.drawRect(0,0,size.width-1,size.height-1);
    }
    private void readObject(ObjectInputStream ois) throws IOException,
    ClassNotFoundException{
        if(!tidy){
                cells = new Vector();
            tidy=true;
        }
    ois.defaultReadObject();
        cells.addElement(this);
        if(isStart){setStart(true);}
        if(isFinish){setFinish(true);}
    }
}
```

- HeuristicStar

```
package astar22;
/*
 * To change this template, choose Tools | Templates and open the template in
 * the editor.
 */
/**
 *
 * @author giannis
 */
import java.awt.Point;
import java.util.Vector;
public class HuristicAStar extends OneTailAStar implements PathFinder
{
   double minCost;
   public HuristicAStar(){
      super();
      stepSpeed=20;
   }
   /**
    * calculates the waighted manhattan distance from a to b
    */
   public double cbDist(Point a,Point b,double low){
      return low * (Math.abs(a.x-b.x)+Math.abs(a.y-b.y)-1);
   }
   public GridCell[] findPath(Map map)
   {
```

```
        minCost = Double.MAX_VALUE;
        for(int i=0;i<map.w;i++){
          for(int j=0;j<map.h;j++){
            minCost = Math.min(map.gridCell[i][j].getCost(),minCost);
          }
        }
        //minCost=0.9;
        System.out.println("Cheepest Tile = "+minCost);
        return super.findPath(map);
    }
public int step(){
        int tests = 0;
        boolean found = false;
        boolean growth = true;
        GridCell finish = GridCell.getFinishCell();
        Point end = finish.getPosition();
        Vector temp = (Vector) edge.clone();
        //find the most promesing edge cell
        double min = Double.MAX_VALUE;
        double score;
        //int best = -1;
        GridCell best = (GridCell)temp.elementAt(temp.size()-1); ;
        GridCell now;
        for(int i=0;i<temp.size();i++){
          now = (GridCell)temp.elementAt(i);
          if(!done.contains(now)){
            //score =now.getDistFromStart();
            score =now.getDistFromStart();
```

```
            score += cbDist(now.getPosition(),end,minCost);
            if(score<min){
              min = score;
                best = now;
            }
          }
        }
        now = best;
        edge.removeElement(now);
        done.addElement(now);
        GridCell next[] = map.getAdjacent(now);
        for(int i=0;i<4;i++){
          if(next[i]!=null){
            if(next[i]==finish){found=true;}
            if(!next[i].isTotalBlock()){
              next[i].addToPathFromStart(now.getDistFromStart());
              tests++;
              if(!edge.contains(next[i]) &&
!done.contains(next[i])){edge.addElement(next[i]);growth=true;}
            }
          }
          if(found){return FOUND;}
        }
        map.repaint();
        if(edge.size()==0){return NO_PATH;}
      //now process best.
        return NOT_FOUND;
    }
}
```

- PathFinder

```
package astar22;
 public interface PathFinder
{
public GridCell[] findPath(Map map);
}
```

- StartUp

```
package astar22;
import java.awt.Dimension;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.*;
/*
 * To change this template, choose Tools | Templates and open the template in
 * the editor.
 */
/**
```

```
 *
 * @author giannis
 */
public class Startup implements ActionListener
{
int dim, dim1;
private JTextField tex=new JTextField(3);
   private JTextField tex1=new JTextField(3);
  void Initialise()
  {
     System.out.println("ΑΡΧΙΚΟΠΟΙΗΣΗ");

     JFrame basic=new JFrame("ΕΡΓΑΣΙΑ Α*");
     JPanel p1=new JPanel();
    p1.setLayout(new GridLayout(3,2));
     basic.setContentPane(p1);
    JLabel lab=new JLabel("ΕΙΣΑΓΕΤΑΙ ΑΡΙΘΜΟ ΓΡΑΜΜΩΝ");
     JLabel lab1=new JLabel("ΕΙΣΑΓΕΤΑΙ ΑΡΙΘΜΟ ΣΤΗΛΩΝ");
     JButton but1=new JButton("OK");
    but1.addActionListener(this);
     but1.setActionCommand("OK");
    p1.add(lab);
     p1.add(tex);
     p1.add(lab1);
     p1.add(tex1);
     p1.add(but1);
     p1.setBorder(BorderFactory.createEmptyBorder(20, 40, 20, 40));
    basic.pack();
```

```
    basic.setMinimumSize(new Dimension(200, 20));
    basic.setLocationRelativeTo(null);
    basic.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    basic.setVisible(true);
  }
   void spawnWindow(int dim,int dim1)
  {
    System.out.println("ΔΗΜΙΟΥΡΓΙΑ ΔΙΑΣΤΑΣΕΩΝ");

    AStar as=new AStar(dim,dim1);
   as.pack();
    as.setLocationRelativeTo(null);
    as.setVisible(true);
  }
   @Override
  public void actionPerformed(ActionEvent e)
  {
   if("OK".equals(e.getActionCommand()))
    {
   if(tex.getText().isEmpty())
        JOptionPane.showMessageDialog(null,"ΠΑΡΑΚΑΛΩ ΕΙΣΑΓΕΤΑΙ ΑΡΙΘΜΟ
ΓΡΑΜΜΩΝ",
                "Error",JOptionPane.ERROR_MESSAGE);
      else
      {
        try
        {
   int dim2=Integer.parseInt(tex.getText());
   if(dim2<2||dim2>10)
```

```
            {
              JOptionPane.showMessageDialog(null,"ΟΙ ΓΡΑΜΜΕΣ ΠΡΕΠΕΙ ΝΑ
ΕΙΝΑΙ " +
"ΜΕΤΑΞΥ 2 ΚΑΙ 10","Error",JOptionPane.ERROR_MESSAGE);
              return;
            }
            dim=dim2;
          }
          catch(NumberFormatException nfe)
          {
            JOptionPane.showMessageDialog(null,"ΜΟΝΟ ΑΚΕΡΑΙΕΣ ΤΙΜΕΣ
ΕΠΙΤΡΕΠΟΝΤΑΙ",
              "Error",JOptionPane.ERROR_MESSAGE);
            return;
          }
        }
    if(tex1.getText().isEmpty())
        JOptionPane.showMessageDialog(null,"ΠΑΡΑΚΑΛΩ ΕΙΣΑΓΕΤΑΙ ΑΡΙΘΜΟ
ΣΤΗΛΩΝ",
            "Error",JOptionPane.ERROR_MESSAGE);
      else
      {
        try
        {
int dim3=Integer.parseInt(tex1.getText());
if(dim3<2||dim3>10)
          {
            JOptionPane.showMessageDialog(null,"ΟΙ ΣΤΗΛΕΣ ΠΡΕΠΕΙ ΝΑ
ΕΙΝΑΙ" +
              "ΜΕΤΑΞΥ 2 ΚΑΙ 10","Error",JOptionPane.ERROR_MESSAGE);
```

```
                return;
              }

              dim1=dim3;
            }
            catch(NumberFormatException nfe)
            {
              JOptionPane.showMessageDialog(null,"MONO AKEPAIEΣ TIMEΣ
ΕΠΙΤΡΕΠΟΝΤΑΙ",
                  "Error",JOptionPane.ERROR_MESSAGE);
              return;
            }
        }
    }
    spawnWindow(dim,dim1);
}
}
```

- OneTailA*

```
package astar22;

import java.util.Vector;

public class OneTailAStar extends java.lang.Object implements
PathFinder,Runnable

{

public final int NO_PATH=-1,NOT_FOUND=0,FOUND=1;

  protected Vector edge;

  protected Vector done;

  protected Map map;

  int stepSpeed = 100;

  private int maxSteps = 1000;

  Thread loop;

  double distFromStart = 0;

  private boolean findFirst = false;

  @Override

  public GridCell[] findPath(Map map)

  {

    this.map = map;

    GridCell.reset();

    edge = new Vector();

    done = new Vector();

    System.out.println("calculating route");

    if(GridCell.getStartCell() == null){

      System.out.println("No start point set");

      return null;

    }

    if(GridCell.getFinishCell() == null){
```

```
        System.out.println("No finish point set");
        return null;
    }
    System.out.println("Starting from "+map.getStartPosition());
    loop = new Thread(this);
    loop.start();
    return null;
}
  @Override
public void run(){
    edge.addElement(GridCell.getStartCell());
    int pass =0;
    boolean found = false;
    double start,diff;
    int state=NOT_FOUND;
    while(state==NOT_FOUND && pass<maxSteps){
        pass++;
        start = System.currentTimeMillis();
        state = step();
        diff = System.currentTimeMillis()-start;
        try{
            loop.sleep(Math.max((long)(stepSpeed-diff),0));
        }
        catch(InterruptedException e){}
       // System.out.println(diff);
    }
    if(state==FOUND){
        setPath(map);
```

```
        }
        else{System.out.println("No Path Found");}
    }
  public int step(){
        int tests = 0;
        boolean found=false;
        boolean growth=false;
        GridCell finish = GridCell.getFinishCell();
        Vector temp = (Vector) edge.clone();
        for(int i=0;i<temp.size();i++){
          GridCell now = (GridCell)temp.elementAt(i);
          GridCell next[] = map.getAdjacent(now);
          for(int j=0;j<4;j++){
            if(next[j]!=null ){
              if(next[j]==finish){found=true;}
              next[j].addToPathFromStart(now.getDistFromStart());
              tests++;
              if(!next[j].isTotalBlock() &&
!edge.contains(next[j])){edge.addElement(next[j]);growth=true;}
            }
          }
          if(found){return FOUND;}
          done.addElement(now);
    }
     map.repaint();
     if(!growth){return NO_PATH;}
    return NOT_FOUND;
  }
  public void setPath(Map map){
```

```
        System.out.println("Path Found");
        GridCell.setShowPath(true);
        boolean finished = false;
        GridCell next;
        GridCell now = GridCell.getFinishCell();
        GridCell stop = GridCell.getStartCell();
        while(!finished){
          next=map.getLowestAdjacent(now);
          now=next;
          now.setPartOfPath(true);
          now.repaint();
          if(now == stop){finished = true;}
          try{
            loop.sleep(stepSpeed);
          }
          catch(InterruptedException e){}
      }
      System.out.println("Done");
}
}
```

- A*

```
package astar22;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.ItemEvent;
import java.awt.event.ItemListener;
```

```
import javax.swing.JFrame;
public class AStar extends JFrame implements ItemListener,ActionListener
{
   boolean isDesign = false;
   CheckboxGroup group;
   Checkbox blocks,start,finish;
   Choice level = new Choice();
Map map;
 Choice preset = new Choice();
   Choice user = new Choice();
   Button go,clear;


  PathFinder finder = new HuristicAStar();
    public AStar (int dim,int dim1)
    {
     map = new Map(dim,dim1);
    Container cp=getContentPane();
    setLayout(new BorderLayout());
    setSize(612,482);
     cp.add(map,"Center");
     cp.setBackground(Color.BLACK);
     Panel m = new Panel();
     m.setLayout(new GridLayout(1,0));
     m.add(new Label(" Classic A* "));
    m.setBackground(Color.LIGHT_GRAY);
    cp.add(m,"North");
     Panel p = new Panel();
```

```
p.setLayout(new GridLayout(4,1));
p.setBackground(Color.LIGHT_GRAY);
Panel b = new Panel();
b.setLayout(new GridLayout(2,1));
b.setBackground(Color.LIGHT_GRAY);
level.add("ΕΜΠΟΔΙΟ");
level.addItemListener(this);
preset.addItemListener(this);


group = new CheckboxGroup();
blocks = new Checkbox("ΕΜΠΟΔΙΟ",group,true);
start = new Checkbox("ΑΡΧΗ",group,false);
finish = new Checkbox("ΤΕΛΟΣ",group,false);
blocks.addItemListener(this);
start.addItemListener(this);
finish.addItemListener(this);
b.add(blocks);
b.add(level);
p.add(b);
p.add(start);
p.add(finish);
Panel g = new Panel();
g.setBackground(Color.LIGHT_GRAY);
if(!isDesign){
   g.setLayout(new GridLayout(3,1));
}
else{
```

```
        g.setLayout(new GridLayout(2,2));
      }
      go = new Button("ΕΝΑΡΞΗ");
      clear = new Button("ΚΑΘΑΡΙΣΜΟΣ");
      g.add(go);
      g.add(clear);


      p.add(g);
      go.addActionListener(this);
      clear.addActionListener(this);
     cp.add(p,"East");
      }
     @Override
          public void itemStateChanged(ItemEvent e){
            if(e.getSource()==level){
              blocks.setState(true);
              GridCell.setEditMode(GridCell.SET_BLOCKS);
              switch(level.getSelectedIndex()){
                case 0:
                  GridCell.setNewBlockStrength(GridCell.BLOCK);
                  return;
default:
                  GridCell.setNewBlockStrength(GridCell.NORMAL);
                  return;
              }
            }
```

```
                Checkbox box = group.getSelectedCheckbox();
                if(box ==
blocks){GridCell.setEditMode(GridCell.SET_BLOCKS);return;}
                if(box == start){GridCell.setEditMode(GridCell.SET_START);return;}
                if(box == finish){GridCell.setEditMode(GridCell.SET_FINISH);return;}
}
@Override
        public void actionPerformed(ActionEvent e){
          if(e.getSource() == go){
            finder.findPath(map);
          }
          if(e.getSource() == clear){
            GridCell.clearAll();
            map.repaint();
          }
        }

@Override
        public Dimension getPreferredSize(){
          return new Dimension(520,420);
        }
}
```