ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ
ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΣΗΜΑΤΩΝ, ΕΛΕΓΧΟΥ ΚΑΙ ΡΟΜΠΟΤΙΚΗΣ

ΣΧΕΔΙΑΣΗ ΚΑΙ ΑΝΑΠΤΥΞΗ ΓΡΑΦΙΚΗΣ ΔΙΕΠΑΦΗΣ ΑΝΘΡΩΠΟΥ-ΥΠΟΛΟΓΙΣΤΗ ΓΙΑ ΤΟΝ ΤΗΛΕΠΡΟΓΡΑΜΜΑΤΙΣΜΟ ΕΝΟΣ ΑΥΤΟΚΙΝΟΥΜΕΝΟΥ ΡΟΜΠΟΤ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Αθανασίου Γ. Δαρκαδάκη

Επιβλέπων: Κωνσταντίνος Τζαφέστας
Επίκουρος Καθηγητής Ε.Μ.Π.

Αθήνα, Οκτώβριος 2009

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΣΗΜΑΤΩΝ, ΕΛΕΓΧΟΥ ΚΑΙ ΡΟΜΠΟΤΙΚΗΣ

ΣΧΕΔΙΑΣΗ ΚΑΙ ΑΝΑΠΤΥΞΗ ΓΡΑΦΙΚΗΣ ΔΙΕΠΑΦΗΣ ΑΝΘΡΩΠΟΥ-ΥΠΟΛΟΓΙΣΤΗ ΓΙΑ ΤΟΝ ΤΗΛΕΠΡΟΓΡΑΜΜΑΤΙΣΜΟ ΕΝΟΣ ΑΥΤΟΚΙΝΟΥΜΕΝΟΥ ΡΟΜΠΟΤ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Αθανασίου Γ. Δαρκαδάκη

Επιβλέπων: Κωνσταντίνος Τζαφέστας
Επίκουρος Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή στις 29 Οκτωβρίου 2009

.....	.....	.....
Κωνσταντίνος Τζαφέστας	**Νικόλαος Μαράτος**	**Τρύφων Κουσιουρής**
Επίκουρος Καθηγητής Ε.Μ.Π	Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.

Αθήνα, Οκτώβριος 2009

.....

Αθανάσιος Γ. Δαρκαδάκης

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

ΠΕΡΙΛΗΨΗ

Ο κύριος στόχος αυτής της διπλωματικής εργασίας είναι η δημιουργία μιας γραφικής διεπαφής ανθρώπου-υπολογιστή (Graphical User Interface) με σκοπό τον τηλεπρογραμματισμό ενός ρομπότ το οποίο θα βρίσκεται σε ένα χώρο μακριά από το χρήστη. Η γραφική αυτή διεπαφή εκτός από τη δυνατότητα εντολών κίνησης του ρομπότ θα περιλαμβάνει ένα δισδιάστατο χάρτη του ρομπότ (κάτοψη), έναν τοπικό χάρτη γύρω από το ρομπότ, ο οποίος θα δημιουργείται την ίδια στιγμή βάσει των ενδείξεων των sonars του ρομπότ και μία τρισδιάστατη απεικόνιση της περιοχής μπροστά από το ρομπότ. Επίσης θα δίνεται στο χρήστη η δυνατότητα περιήγησης στον τρισδιάστατο χώρο και θα εμφανίζονται ενδείξεις των sonars του ρομπότ καθώς και της κατάστασης της μπαταρίας. Τέλος θα υπάρχει και η προβολή της κάμερας η οποία θα βρίσκεται επάνω στο ρομπότ. Σχετικά με τη μετακίνηση του ρομπότ από μία τοποθεσία σε μια άλλη έχει ενσωματωθεί ένα σύστημα οπτικής αναγνώρισης χειρονομιών, με το οποίο θα δίνονται οι εντολές κίνησης.

ΛΕΞΕΙΣ - ΚΛΕΙΔΙΑ

γραφική διεπαφή ανθρώπου-υπολογιστή, τηλεπρογραμματισμός, τηλερομποτική, δημιουργία χάρτη, *C#*, *OpenGl*, αισθητήρες, επικοινωνία ανθρώπου-υπολογιστή

ABSTRACT

The main scope of this thesis is the design and development of a human-computer graphical user interface for teleprogramming of a mobile robot.This graphical interface, apart from the capability of robot movement commands, will include a two-dimensional map of the robot (ground plan), a local map nearby the robot,which will be created the same time based on the signals from the robot sonars and a three-dimensional view of the area in front of the robot. In addition to this, the user will have the capability of moving in the three-dimensional area and there will be signs showing the robot sonars' state and the condition of the battery. Moreover, we will be able to see the view from the camera which is placed on the robot. As far as robot movement from a place to another is concerned, there is integrated an image-based hand gestures recognition system for human-robot interaction in order to give the movement commands.

KEYWORDS

graphical user interface, teleprogramming, telerobotics, mapping, C#, OpenGl, sensors, human-computer interaction

Περιεχόμενα

1 Εισαγωγή

1.1 Οργάνωση της διπλωματικής

Στο δεύτερο κεφάλαιο γίνεται μια εισαγωγή αναφέροντας κάποιες θεωρητικές βάσεις που πρέπει να υπάρχουν. Οι βάσεις αυτές είναι σχετικές με την τεχνητή νοημοσύνη, την εικονική πραγματικότητα και την τηλερομποτική. Επίσης γίνεται αναφορά στις εργασίες τις οποίες εκτελούν τα ρομπότ στα περιβάλλοντα στα οποία βρίσκονται. Τέλος αναφερόμαστε στις γραφικές διεπαφές ανθρώπου - υπολογιστή καθώς και στην επικοινωνία μέσω χειρονομιών.

Στο τρίτο κεφάλαιο γίνεται η περιγραφή της εφαρμογής που έχει υλοποιηθεί και δίνονται κάποιες οδηγίες για τη χρήση της. Επίσης βλέπουμε μερικά παραδείγματα εκτέλεσης της εφαρμογής.

Στο τέταρτο κεφάλαιο αναφερόμαστε στην πρώτη εργασία αυτής της διπλωματικής, δηλαδή τη δημιουργία ενός χάρτη μέσω των ενδείξεων sonars του ρομποτ και στον τρόπο με τον οποίο αυτή υλοποιήθηκε. Επιπλέον γίνεται αναφορά γενικότερα στις αισθητήριες διατάξεις.

Το πέμπτο κεφάλαιο περιγράφει το δεύτερο κομμάτι της διπλωματικής εργασίας, δηλαδή την κατασκευή ενός τρισδιάστατου χάρτη, ο οποίος θα απεικονίζει αυτό που θα έδειχνε μια κάμερα που βρίσκεται πάνω στο ρομπότ. Περιγράφεται η μέθοδος που ακολουθήθηκε και δίνονται και επιπλέον πληροφορίες σχετικά με τον φωτισμό, την πρόβλεψη κίνησης, το πολυγωνικό μοντέλο και τις προβολές.

Στο έκτο κεφάλαιο διατυπώνονται κάποια συμπεράσματα, που προέκυψαν από την εκπόνηση αυτής της διπλωματικής εργασίας, και κάποιες σκέψεις για περαιτέρω έρευνα.

Τέλος στα δύο τελευταία κεφάλαια αναφέρεται η βιβλιογραφία που χρησιμοποιήθηκε και δίνεται ο κώδικας C της εφαρμογής.

2 Θεωρητικό υπόβαθρο

2.1 Εικονική και επαυξημένη πραγματικότητα

Η εικονική πραγματικότητα (virtual reality) είναι ένας όρος που χρησιμοποιείται, για να περιγράψει συστήματα στα οποία υπάρχει αλληλεπίδραση σε πραγματικό χρόνο μεταξύ ανθρώπου και υπολογιστή. Ιδανικά, μία τέτοια αλληλεπίδραση περιλαμβάνει όλες τις αισθήσεις και όχι μόνο την όραση. Θα μπορούσε για παράδειγμα να χρησιμοποιείται ήχος ή ακόμη και η αίσθηση της αφής μέσω ειδικών συσκευών. Όλα αυτά έχουν ως σκοπό να δημιουργηθεί στο χρήστη η ψευδαίσθηση της πραγματικότητας.

Η επαυξημένη πραγματικότητα (augmented reality) χρησιμοποιείται, για να περιγράψει ένα περιβάλλον το οποίο περιλαμβάνει τόσο στοιχεία εικονικής πραγματικότητας, όσο και στοιχεία του πραγματικού κόσμου. Ακόμη δύο προϋποθέσεις που χρειάζονται, για να χαρακτηριστεί ένα σύστημα ως επαυξημένη πραγματικότητα, είναι να υπάρχει διάδραση (interaction) με το χρήστη σε πραγματικό χρόνο και να βρίσκεται στις τρεις διαστάσεις. Σκοπός της δεν είναι να υποκαταστήσει την πραγματικότητα αλλά να την αλλάξει κατά κάποιο τρόπο προσθέτοντας στο σύστημα επιπλέον πληροφορίες.

Η επαυξημένη πραγματικότητα βρίσκει όλο και περισσότερες εφαρμογές. Ένας τομέας στον οποίο χρησιμοποιείται είναι η τηλερομποτική (telerobotics), δηλαδή ο τομέας της ρομποτικής που περιέχει το χειρισμό συσκευών από κάποια απόσταση (teleoperation), ο οποίος συνήθως επιτυγχάνεται μέσω μιας διεπαφής (interface) κι ενός δικτύου. Συνήθως χρησιμοποιείται σε περιβάλλοντα όπου οι συνθήκες, για να βρεθεί ένας άνθρωπος, είναι επικίνδυνες ή σε εφαρμογές στις οποίες η χρησιμοποίηση ανθρώπων είναι δαπανηρή (είτε σε χρόνο είτε σε χρήμα).

2.2 Τηλερομποτική

2.2.1 Ορισμοί

Πρώτα απ΄ όλα πρέπει να ξεκινήσουμε ορίζοντας τι είναι το ρομπότ. Το 1979 το Ινστιτούτο Ρομποτικής της Αμερικής όρισε σαν ρομπότ μία επαναπρογραμματιζόμενη πολυλειτουργική συσκευή σχεδιασμένη, για να κινεί υλικά, μέρη, εργαλεία ή εξειδικευμένες συσκευές μέσω διάφορων προγραμματισμένων κινήσεων με σκοπό την εκτέλεση διάφορων εργασιών.

Θεωρούμε ως τηλερομπότ ένα ρομπότ το οποίο ελέγχεται από απόσταση από έναν ανθρώπινο χειριστή ανεξάρτητα από το βαθμό αυτονομίας του ρομπότ. Πιο εξειδικευμένα θα μπορούσαμε να έχουμε διακρίσεις σε ρομπότ των οποίων η κίνηση ελέγχεται συνεχώς από τον χειριστή και σε ρομπότ τα οποία έχουν μερική αυτονομία.

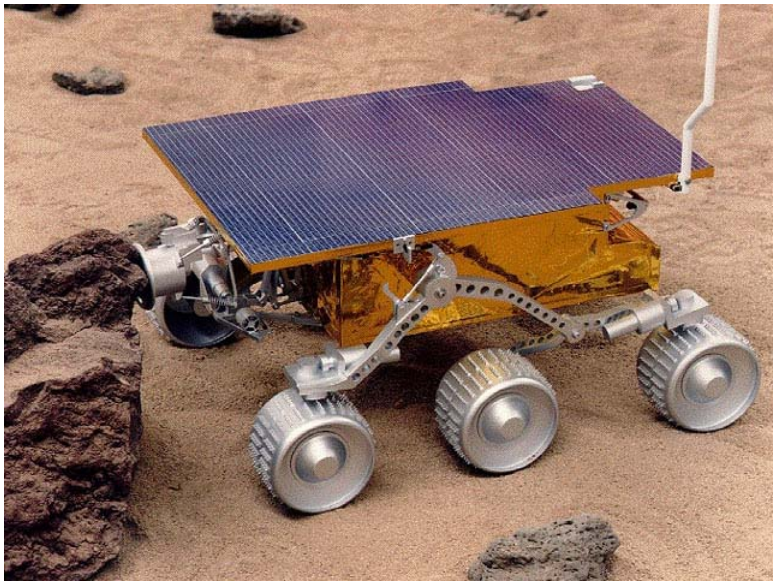
2.2.2 Πεδία εφαρμογής

Τα τηλερομπότ χρησιμοποιούνται:

- Σε συνθήκες και περιβάλλοντα που είναι επικίνδυνα για την ανθρώπινη υγεία ή επιβίωση (βαθιά νερά, διάστημα, τοξικά περιβάλλοντα, ναρκοπέδια, φωτιές, αστυνομικές ή στρατιωτικές επιχειρήσεις). Το GNOM είναι ένα υποβρύχιο τηλερομπότ το οποίο ενδείκνυται για επαγγελματική αλλά και για ερασιτεχνική χρήση.

- Σε περιβάλλοντα που δεν είναι άμεσα προσβάσιμα στον άνθρωπο λόγω μικρών περασμάτων (εξέταση και επιδιόρθωση σωλήνων και αγωγών, εγχειρήσεις)

- Όταν η εργασία που πρέπει να γίνει είναι πέραν των ανθρωπίνων δυνατοτήτων (τρύπημα οστών για τοποθέτηση τεχνητών μελών



Όταν το διαστημόπλοιο Mars Pathfinder προσεδαφίστηκε στον Άρη, ελευθέρωσε ένα τηλερομπότ, το Sojourner, στην πρώτη προσπάθεια κίνησης οχήματος στον Άρη. Η επικοινωνία δεν γινόταν σε πραγματικό χρόνο αλλά με καθυστέρηση 11 περίπου λεπτών, εξαιτίας του χρόνου μεταφοράς του σήματος σε τόσο μακρινή απόσταση.

(χρειάζεται μεγάλη ακρίβεια), ανασκαφές σε κατασκευαστικές περιοχές (χρειάζεται εφαρμογή μεγάλων δυνάμεων))

- Όταν η μεταφορά ανθρώπων στο χώρο εργασίας είναι ακριβή, είτε σε χρόνο είτε σε χρήμα (εξερεύνηση άλλων πλανητών, συντήρηση υποθαλάσσιων καλωδίων, τηλεδιάγνωση ή τηλεεγχείρηση που απαιτεί εξειδικευμένους ιατρούς των οποίων ο χρόνος δεν πρέπει να σπαταλάται σε ταξίδια).

- Σε μέρη όπου η παρουσία ανθρώπων μπορεί να βλάψει το περιβάλλον (εξερεύνηση ευαίσθητων περιβαλλόντων ή αρχαιολογικών χώρων).

2.2.3 Προβλήματα

2.2.3.1 Χρονοκαθυστέρηση Πολλές φορές, ανάλογα με τη θέση που βρίσκεται το τηλερομπότ, υπάρχει καθυστέρηση χρόνου στην

Ο καθηγητής Jacques Marescaux και η ομάδα του από το IRCAD τον Σεπτέμβρη του 2001 πραγματοποίησαν την πρώτη υποβοηθούμενη από ρομπότ εξ' αποστάσεως εγχείρηση, μια χολοκυστεκτομή, σε έναν ασθενή σε απόσταση 7000χλμ. χρησιμοποιώντας το Zeus, ένα ειδικά σχεδιασμένο τηλερομπότ.

επικοινωνία του χειριστή με το ρομπότ. Για παράδειγμα η καθυστέρηση χρόνου στα ρομπότ που βρίσκονται στο διάστημα εξαρτάται από την απόσταση και την ταχύτητα του φωτός. Αντίστοιχα ένα τηλερομπότ που βρίσκεται σε μεγάλο βάθος σε ωκεανούς θα μπορούσε να επηρεάζεται από την ταχύτητα του ήχου μέσα στο νερό (1700 m/s). Ένας τρόπος, για να αποφύγουμε τις ασταθείς καταστάσεις που δημιουργούνται, θα ήταν να περιμένουμε να πάρουμε την απόκριση του ρομπότ στην προηγούμενη εντολή μας και στη συνέχεια να δώσουμε την επόμενη (move and wait strategy). Αυτό όμως θα έκανε το σύστημά μας πολύ αργό.

2.2.3.2 Αισθητήρες και εμφάνιση Η θέση του ρομπότ προσδιορίζεται από αισθητήρες, οι οποίοι όμως δεν "αισθάνονται" όλα

τα δεδομένα που θα αντιλαμβανόταν ένας άνθρωπος αν βρισκόταν σ΄ εκείνο το μέρος. Έτσι υπάρχει απώλεια πληροφορίας και τα δεδομένα δεν μπορούν να αναπαρασταθούν με υψηλό βαθμό αξιοπιστίας.

2.2.3.3 Ασφάλεια Όπως είπε από το 1941 ο Ισαάκ Ασίμωφ στον πρώτο του νόμο περί ρομποτικής: "Ενα ρομπότ δε θα κάνει κακό σε άνθρωπο, ούτε θα επιτρέψει με την αδράνειά του να βλαφτεί ανθρώπινο ον". Γενικεύοντας περισσότερο θα πρέπει να προσέχουμε μήπως μέσω του τηλεχειρισμού το ρομπότ συγκρουστεί με άλλα έμβια ή αντικείμενα της φύσης με αποτέλεσμα την καταστροφή οποιουδήποτε από τα δύο.

Υπάρχουν τέλος και κάποια άλλα λιγότερο σημαντικά -όχι όμως ασήμαντα- ζητήματα σχετικά με την τηλερομποτική, όπως για παράδειγμα η αυτονομία του ρομπότ, η οποία έχει να κάνει τόσο με την ενεργειακή αυτονομία του ρομπότ όσο και με τη δυνατότητα αυτοκυβέρνησης, μάθησης και προσαρμοστικότητας.

2.2.4 Πραγματικά και εικονικά περιβάλλοντα

Τα εικονικά περιβάλλοντα παίζουν σημαντικό ρόλο στην επίβλεψη του ελέγχου ενός τηλερομπότ. Χρησιμοποιώντας εικονικά περιβάλλοντα μπορούμε να προσομοιώσουμε τις ενέργειες του τηλερομπότ και να απαντήσουμε σε ερωτήματα του τύπου: "τι θα γινόταν αν.." εκτελώντας την προσομοίωση και παρατηρώντας τα αποτελέσματα. Παρ΄ όλ΄ αυτά υπάρχουν αρκετά ψεγάδια, μερικά από τα οποία αναφέρονται παρακάτω:

- Ο τύπος και η φύση ενός εικονικού περιβάλλοντος είναι απολύτως ελεγχόμενα και η αλληλεπίδραση του με το τηλερομπότ μπορεί να αναλυθεί πλήρως. Από την άλλη μεριά, ένα πραγματικό περιβάλλον δεν έχει σταθερή και σίγουρη δομή και

δε μπορεί να αναπαρασταθεί και να εξομοιωθεί, παρά μόνο σε μερικό βαθμό.

- Σε ένα εικονικό περιβάλλον, οι εικονικοί αισθητήρες μπορούν να λαμβάνουν ακριβείς τιμές και να μην επηρεάζονται από διαταραχές και θορύβους του περιβάλλοντος, σε αντίθεση με το πραγματικό περιβάλλον για τις διαταραχές του οποίου έχουμε μερική, αν όχι μηδενική γνώση.

- Στο εικονικό περιβάλλον, το τηλερομπότ μπορεί να κινείται και να εκτελεί εργασίες χωρίς να καταναλώνει ενέργεια, κάτι το οποίο σε πραγματικά περιβάλλοντα είναι αδύνατο.

2.2.5 Καταμερισμός εργασίας μεταξύ ανθρώπων και ρομπότ

Τις τελευταίες δύο δεκαετίες τα βιομηχανικά ρομπότ χρησιμοποιούνται ευρέως, ειδικά σε κατασκευές. Έχουν μεγάλο βαθμό αυτονομίας και μειώνουν την ανάγκη ανθρώπινης ενασχόλησης. Αντίθετα, τα τηλερομπότ χρειάζονται έναν ανθρώπινο χειριστή.

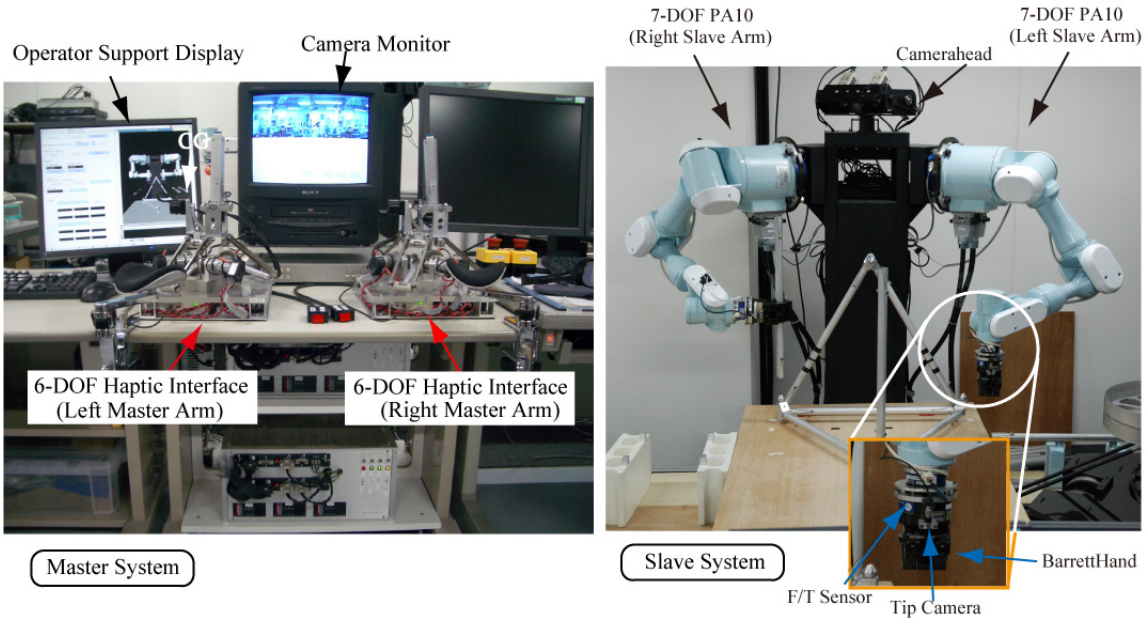
Γενικά, ο τηλεχειρισμός ως όρος χρησιμοποιείται, για να περιγράψει όλες τις μεθοδολογίες και τις τεχνικές, που επιτρέπουν σε έναν ανθρώπινο χειριστή να εκτελέσει μια ενέργεια χειρίζοντας ένα ενδιάμεσο μηχατρονικό(μηχανολογικό + ηλεκτρονικό) σύστημα. Οι βασικές διαφορές μεταξύ των τηλερομπότ και των βιομηχανικών ρομπότ έγκεινται στη φύση της εργασίας και στην ανάγκη από ανθρώπινο χειρισμό. Τα βιομηχανικά ρομπότ εκτελούν κυρίως προβλέψιμες εργασίες σε ελεγχόμενα περιβάλλοντα με ταχύτητα και ακρίβεια, σε αντίθεση με τα τηλερομπότ στα οποία σπανίως υπάρχει επανάληψη της ίδιας εργασίας.

Το όλο σύστημα του τηλεχειρισμού είναι ένα σύστημα αφέντη-σκλάβου (master-slave system). Το πρώτο μέρος του συστήματος, επονομαζόμενο ώς αφέντης, είναι αυτό που ελέγχεται από τον ανθρώπινο χειριστή και το δεύτερο μέρος (σκλάβος) εκτελεί την

εργασία.

Η ολοένα αναπτυσσόμενη τεχνολογία των τηλερομπότ τους επιτρέπει να εκτελούν πολύπλοκες υποεργασίες ελαχιστοποιώντας την ανάγκη ανθρώπινης επίβλεψης. Ωστόσο έως σήμερα υπάρχουν αυτόνομα και έξυπνα ρομπότ τα οποία μπορούν να δρουν σε άγνωστο ή μεταβαλλόμενο περιβάλλον μόνο για την εκτέλεση λίγων συγκεκριμένων εργασιών.

2.2.6 Έλεγχος με επίβλεψη



Τα απομακρυσμένα ρομπότ είναι μηχανήματα σχεδιασμένα να λειτουργούν σύμφωνα με παραλλαγές της αρχιτεκτονικής ελέγχου "αισθάνσου-σκέψου-σχεδίασε-δράσε". Ωστόσο, εξαιτίας της αυξημένης πολυπλοκότητας τέτοιων εργασιών η παρουσία του ανθρώπινου παράγοντα ως επιβλέπων είναι απαραίτητη. Σε ένα σύστημα ελέγχου με επίβλεψη, ο επιβλέπων πρέπει να ενεργεί σύμφωνα με τα ακόλουθα:

- Σχεδιάζει ποια εργασία θα εκτελεστεί και πώς
- Διδάσκει τον υπολογιστή αυτό που σχεδιάστηκε

- Παρακολουθεί την (αυτόματη) κίνηση σε κάποια οθόνη, ώστε να σιγουρευτεί ότι όλα πάνε σύμφωνα με το σχέδιο, και προσπαθεί να εντοπίσει αποτυχίες
- Παρεμβαίνει, δηλαδή επιβοηθάει την ενεργή διεργασία, είτε παίρνοντας ολοκληρωτικά τον έλεγχο, εφόσον επιτεύχθηκε η επιθυμητή εργασία, είτε την διακόπτει, ώστε να σχεδιάσει μια καινούρια εργασία
- Αποκτά εμπειρία, ώστε να ενεργήσει καλύτερα στο μέλλον

Ο ρόλος των υπολογιστών στην τηλερομποτική μπορεί να κατηγοριοποιηθεί ανάλογα με το πόση εργασία αναλαμβάνεται σχετικά με την εργασία που μπορεί να αναλάβει από μόνος του ο άνθρωπος. Οι δυο τους μοιράζονται εργασίες για δύο κυρίως λόγους:

- Για να αποφορτωθεί ο άνθρωπος από συγκεκριμένες εργασίες. Στην περίπτωση αυτή εκτελούνται εργασίες σύμφωνα με προγραμματισμένο τρόπο, ορισμένο από τον χειριστή
- Για να επεκταθούν οι ανθρώπινες ικανότητες, όταν χρειάζεται μεγάλη ακρίβεια στις κινήσεις ή εφαρμογή μεγάλων δυνάμεων.

2.2.7 Γραφική διεπαφή ανθρώπου-υπολογιστή (GUI)

Η γραφική διεπαφή ανθρώπου υπολογιστή (Graphical User Interface - GUI) είναι ένα σύνολο γραφικών στοιχείων, τα οποία εμφανίζονται στην οθόνη κάποιας ψηφιακής συσκευής (πχ στην οθόνη ενός ηλεκτρονικού υπολογιστή) και χρησιμοποιούνται για την αλληλεπίδραση του χρήστη με τη συσκευή αυτή. Παρέχουν σ΄ αυτόν, μέσω γραφικών, ενδείξεις και εργαλεία προκειμένου να φέρει εις πέρας κάποιες επιθυμητές λειτουργίες. Για το λόγο αυτό δέχονται και είσοδο από το χρήστη και αντιδρούν ανάλογα σε συμβάντα που αυτός προκαλεί με τη βοήθεια κάποιας συσκευής εισόδου (πχ πληκτρολόγιο ή ποντίκι).

2.2.7.1 Ιστορία των GUI Όπως γίνεται με πολλές εφευρέσεις και νέα προϊόντα, μερικές από τις ιδέες ενός συστήματος με *GUI* υπήρχαν στο μυαλό ανθρώπων αρκετά προτού μπορέσει η τεχνολογία να φτιάξει μηχανήματα που θα μπορούν να τις υποστηρίζουν. Ένας από τους πρώτους που εξέφρασαν αυτές τις ιδέες ήταν ο Vannevar Bush στις αρχές της δεκαετίας του 1930. Έγραψε ένα βιβλίο σχετικά με ένα μηχάνημα, το οποίο ονόμαζε "Memex", το οποίο οραματιζόταν σαν ένα γραφείο με δύο οθόνες αφής, ένα πληκτρολόγιο και ένα σκάνερ. Αυτό επέτρεπε στους χρήστες του να έχουν πρόσβαση σε όλη την ανθρώπινη γνώση χρησιμοποιώντας συνδέσμους παρόμοιους με τους υπερσυνδέσμους (hyperlinks). Ωστόσο εκείνη την εποχή δεν υπήρχε τρόπος να λειτουργήσει ένα τέτοιο μηχάνημα κι έτσι οι ιδέες του δεν ακούστηκαν όσο θα έπρεπε. Αυτός όμως που θεωρείται ο πατέρας του *GUI* ήταν ο Douglas Englebart, ο οποίος ξεκίνησε να σκέφτεται πώς θα υλοποιήσει τις ιδέες του Bush. Το 1962 δημοσίευσε τις ιδέες του στο έργο του "Augmenting Human Intellect". Σε αυτήν την εργασία υποστήριξε ότι θα πρέπει ένας υπολογιστής όχι να αντικαταστήσει τους ανθρώπους, αλλά να τους υποβοηθήσει αυξάνοντας την ικανότητα των ανθρώπων να προσεγγίσουν μια κατάσταση κατανοώντας την καλύτερα, ώστε να καταλήγουν σε λύσεις στα αρχικά τους προβλήματα. Δημιούργησε και το σύστημα NLS(oNLine System), το οποίο χρησιμοποιούσε έναν κέρσορα οδηγούμενο από ποντίκι και αρκετά παράθυρα, ώστε να δουλεύει με υπερκείμενα (hypertext). Κάποια χρόνια αργότερα μια ομάδα ερευνητών της Xerox παρήγαγε δύο μοντέλα με GUI, το Alto και το Star. Το Star δόθηκε σε διάθεση προς το κοινό το 1981, όμως η τιμή του ήταν αρκετά περιοριστική κι έτσι κατάφερε να πουλήσει μόνο 25.000 κομμάτια. Όμως το λειτουργικό σύστημα ήταν αρκατά "βαρύ" κι ο υπολογιστής δεν ήταν, όσο δυνατός θα έπρεπε, ώστε να χειριστεί τις απαιτήσεις του λειτουργικού

συστήματος με τα επιθυμητά αποτελέσματα. Το 1984 η Apple εισήγαγε στην αγορά τα Macintosh, ένα οικονομικά προσιτό GUIσε ηλεκτρονικό υπολογιστή με ποντίκι.

2.2.7.2 Πλεονεκτήματα των GUI Το σημαντικότερο πλεονέκτημα των GUIs είναι ότι κάνουν τη λειτουργία του υπολογιστή πιο διαισθητική και επομένως ευκολότερη στη χρήση. Για παράδειγμα, είναι πολύ ευκολότερο για ένα νέο χρήστη να πατήσει ένα κουμπί, για να ξεκινήσει η εκτέλεση μιας εντολής, από το να χρειάζεται να θυμάται και να πληκτρολογεί ακαταλαβίστικες εντολές προκειμένου να πετύχει το ίδιο αποτέλεσμα. Επιπλέον, οι γραφικές διεπαφές παρέχουν στο χρήστη άμεση οπτική ανάδραση σχετική με το αποτέλεσμα κάθε ενέργειας. Για παράδειγμα, όταν διαγράφουμε ένα αντικείμενο μέσω του GUI, βλέπουμε το εικονίδιό του να εξαφανίζεται, ενώ, αν το διαγράψουμε από τη γραμμή εντολών, δεν θα λάβουμε ανάδραση που να μας πληροφορεί ότι το αρχείο πραγματικά διαγράφηκε. Ένα ακόμη πλεονέκτημα είναι ότι δίνεται η δυνατότητα στο χρήστη να εκτελεί πολλές ενέργειες ταυτόχρονα, αλλά και να βλέπει την ίδια στιγμή τα αποτελέσματά τους. Μία σωστά σχεδιασμένη γραφική διεπαφή προσφέρει στο χρήστη ένα όμορφο, εύχρηστο και λειτουργικό περιβάλλον εργασίας. Οι γραφικές διεπαφές πρέπει να περιλαμβάνουν στοιχεία όπως παράθυρα, μενού, κουμπιά, μπάρες κύλισης και εικόνες, τα οποία σε συνδυασμό με τη χρήση ήχου φωνής και κινούμενων ή μη εικόνων, διευκολύνουν το χρήστη.

2.2.8 Επικοινωνία ανθρώπου-υπολογιστή μέσω χειρονομιών

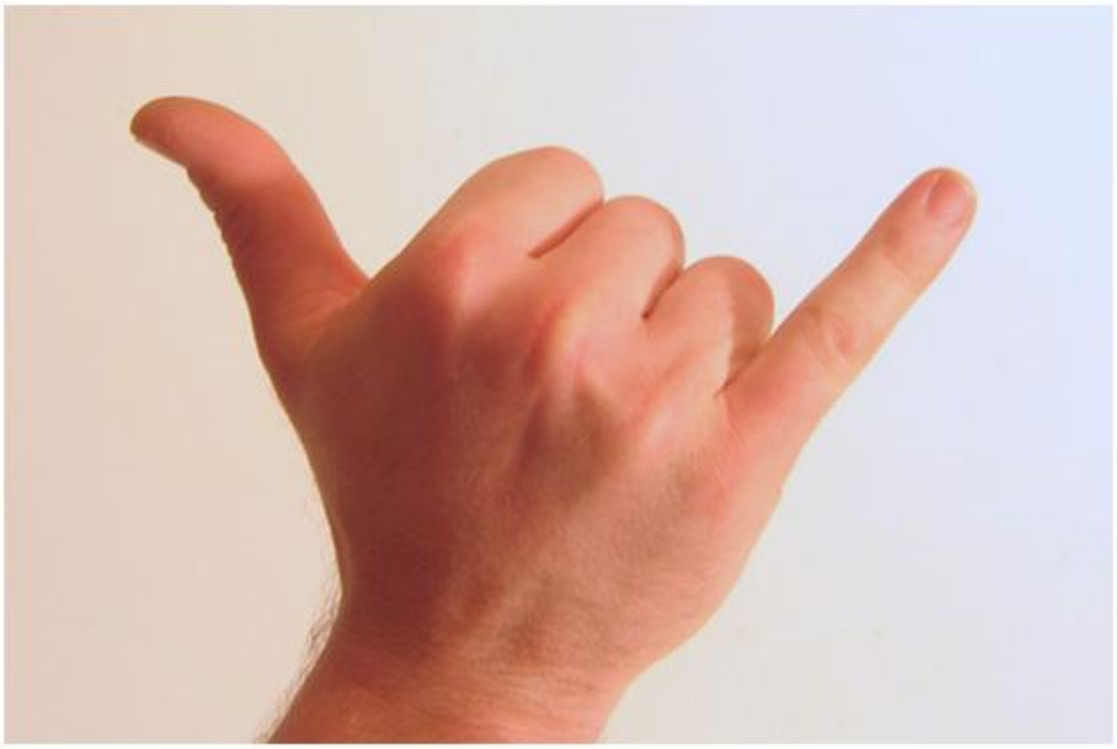
Οι χειρονομίες αποτελούν έναν τρόπο διάδρασης ανάμεσα στον άνθρωπο και έναν υπολογιστή. Ο τρόπος αυτός είναι χρήσιμος σε περιπτώσεις που δεν είναι δυνατή η χρήση του πληκτρολογίου, αλλά κυρίως σε περιπτώσεις χρήσης από άτομα με διάφορες ειδικές

ανάγκες. Το μειονέκτημα της επικοινωνίας με χειρονομίες είναι ότι περιλαμβάνει την κατανόηση της ανθρώπινης κίνησης, η οποία είναι από μόνη της ένα περίπλοκο αντικείμενο, αφού το σχήμα του κάθε ανθρώπου δε μπορεί να είναι σαφώς καθορισμένο εξαιτίας της διαφορετικής δομής του ή της πιθανής ύπαρξης ενδυμάτων. Επιπλέον ακόμη και οι χειρονομίες είναι ενέργειες οι οποίες διαφέρουν από άνθρωπο σε άνθρωπο.

Δύο είναι οι συνηθισμένοι τρόποι επικοινωνίας ανθρώπου-υπολογιστή μέσω χειρονομιών. Ο πρώτος τρόπος είναι φορώντας ένα ειδικό γάντι. Το γάντι αυτό περιλαμβάνει αισθητήρες κίνησης και μεταβολής γωνίας ωστε να μπορεί να μεταφέρει στον υπολογιστή τις κινήσεις του χεριού μας και με τον τρόπο αυτόν να δώσουμε μια εντολή. Το πλεονέκτημα αυτού του είδους της διάδρασης ανθρώπου-υπολογιστή είναι ότι έχουμε αρκετή ακρίβεια στη μετάδοση της πληροφορίας. Από την άλλη όμως είναι μια τεχνολογία εξαιρετικά άβολη, αφού ο χρήστης είναι υποχρεωμένος να φοράει το ειδικό αυτό γάντι το οποίο είναι φτιαγμένο από συνθετικό υλικό.

Ο δεύτερος τρόπος γίνεται μέσω κάμερας. Ο χρήστης κάνει διάφορες χειρομορφές μπροστά σε μία κάμερα, η οποία μεταφέρει το σήμα αυτό στον υπολογιστή. Ο υπολογιστής με τη σειρά του επεξεργάζεται τις εικόνες, τις αναγνωρίζει και τις αντιστοιχίζει με κάποια προεπιλεγμένη εντολή για το συγκεκριμένο σήμα. Η αντιστοίχιση αυτή μπορεί να γίνει προσαρμόζοντας ένα δισδιάστατο ή τρισδιάστατο μοντέλο πάνω στην εικόνα που λαμβάνουμε από την

κάμερα.



3 Περιγραφή συστήματος

Το σύστημά μας περιλαμβάνει δύο αυτοκινούμενα τροχοφόρα οχήματα της Activemedia Robotics, και συγκεκριμένα το Pioneer P3-DX και το Pioneer 3AT. Και τα δύο περιλαμβάνουν ενσωματωμένη αυτόνομη υπολογιστική μονάδα και υποστηρίζουν ασύρματη επικοινωνία τύπου Ethernet. Επίσης περιλαμβάνουν αισθητήρες laser, 8 εμπρόσθιους Sonar αισθητήρες και 8 οπίσθιους. Επιπλέον υπάρχει δυνατότητα προγραμματισμού μέσω της ARIA πλατφόρμας (Advanded Robotics Interface for Applications). Το Pioneer P3-DXαποτελείται από τρεις ρόδες (δύο ελεγχόμενες και μία ελεύθερα περιστρεφόμενη), μπορεί να μεταφέρει μέχρι $23kgr$ και μπορεί να φτάσει ταχύτητες μέχρι και $1.6m/sec$, σε αντίθεση με το Pioneer 3AT που αποτελείται από τέσσερις ελεγχόμενες ρόδες, μπορεί να μεταφέρει έως $30kgr$ και η ταχύτητα του μπορεί να φτάσει τα $8m/sec$.

Πάνω στη ρομποτική πλατφόρμα βρίσκεται η RoHS 1024X768 Color 6mm Bumblebee2 System (Point Grey), μια στερεοσκοπική κάμερα η οποία παρέχει υψηλή ποιότητα video σε συνδυασμό με υψηλές ταχύτητες επεξεργασίας αυτών. Διαθέτει φακούς $3.8mm$ ή $6mm$ και η ανάλυση της είναι $640x480$ $pixels$ σε $48FPS$ ή $1024x768$ σε $18FPS$. Η κάμερα αυτή χρησιμοποιείται για την πλοήγηση. Το σύστημα αποτελείται από δύο ξεχωριστά υποσυστήματα: το υποσύστημα του υπολογιστή-ρομπότ και το υποσύστημα του χειριστή. Η επικοινωνία μεταξύ των δύο υποσυστημάτων γίνεται μέσω ενός wireless δικτύου. Ο έλεγχος πλοήγησης του κινούμενου ρομπότ μπορεί να γίνει με τρεις τρόπους. Ο πρώτος είναι ο τηλεχειρισμός μέσω του GUI. Ο χρήστης μπορεί να καθοδηγεί το ρομπότ με απλές εντολές της μορφής “πήγαινε ευθεία”, “στρίψε 90^o μοίρες αριστερά ή δεξιά”, “στρίψε κατά άλλη γωνία” κλπ. Η καθοδήγηση μπορεί να γίνει με βάση είτε τη θέση και τον

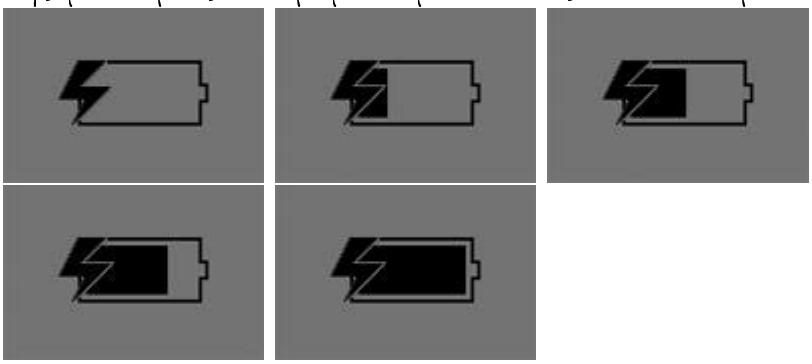
προσανατολισμό του ρομπότ στο χάρτη, είτε τις εικόνες από την web camera του ρομπότ είτε τις τιμές των αισθητήρων. Ένας δεύτερος τρόπος τηλεχειρισμού του ρομπότ είναι μέσω κάποιας απτικής συσκευής, την οποία ο χρήστης χρησιμοποιεί ως joystick ,ώστε να αποστείλει τις εντολές κίνησης. Σε αυτήν την περίπτωση, ο χειριστής μπορεί να αντιληφθεί το περιβάλλον και μέσω της απτικής ανάδρασης στο χέρι του. Για παράδειγμα, όταν μια εντολή οδηγεί το ρομπότ προς κάποιο εμπόδιο, μια δύναμη αντιστρόφως ανάλογη της απόστασης του ρομπότ από το εμπόδιο ασκείται στον χειριστή. Έτσι μπορούμε να έχουμε καλύτερη αίσθηση του περιβάλλοντος, αλλά και λιγότερες περιπτώσεις συγκρούσεων. Ωστόσο συνήθως σε τέτοιες περιπτώσεις υπάρχει το μειονέκτημα της αύξησης του χρόνου εκτέλεσης μιας διαδρομής. Τέλος μπορούμε να έχουμε αυτόνομη πλοήγηση, η οποία έγκειται στην κίνηση του ρομπότ μεταξύ σημείων στόχων. Η σχεδίαση της τροχιάς γίνεται μέσω του αλγορίθμου Dijkstra, ο οποίος εξασφαλίζει την εύρεση τροχιάς, αν υπάρχει, και βρίσκει τη συντομότερη διαδρομή μεταξύ των σημείων αρχής και στόχου. Το ρομπότ πρέπει να γνωρίζει ανά πάσα χρονική στιγμή τη θέση του στο περιβάλλον. Για να το πετύχει αυτό, χρησιμοποιεί τις τιμές από τους encoders στις δύο ρόδες και τις τιμές των αισθητήρων Ultrasonic. Όμως οι encoders συσσωρεύουν σφάλματα και μετά από κάποιο μήκος διαδρομής οι μετρήσεις τους γίνονται αναξιόπιστες. Επίσης και τα σοναρς δίνουν τη δυνατότητα μέτρησης μόνο οκτώ εμποδίων σε εύρος 180^o, οι οποίες συχνά είναι λανθασμένες (όταν το κύμα προσπίπτει σε σκληρή επιφάνεια υπό γωνία), δε μπορούν να ανιχνεύσουν συρματόπλεγμα ή μαλακά υλικά που απορροφούν τον ήχο, επηρεάζονται από ηχητικά κύματα προηγούμενων μετρήσεων κλπ. Η ένωση των δύο εκτιμήσεων γίνεται με τη χρήση ενός αλγορίθμου Localization ο οποίος χρησιμοποιεί φίλτρα Kalman και ένα μεγάλο αριθμό από εκτιμήσεις (Particle Filters). Η γραφική εφαρμογή (GUI) είναι ουσιαστικά

το πρόγραμμα διεπαφής του χειριστή με το ρομπότ. Μέσω αυτού ο χειριστής μπορεί να στέλνει εντολές στο ρομπότ (πχ πήγαινε στο σημείο (X,Υ), στρίψε ϑ^o μοίρες, προχώρα ευθεία) και να λαμβάνει πληροφορίες για την κατάσταση του ρομπότ (πχ θέση, τιμές αισθητήρων).

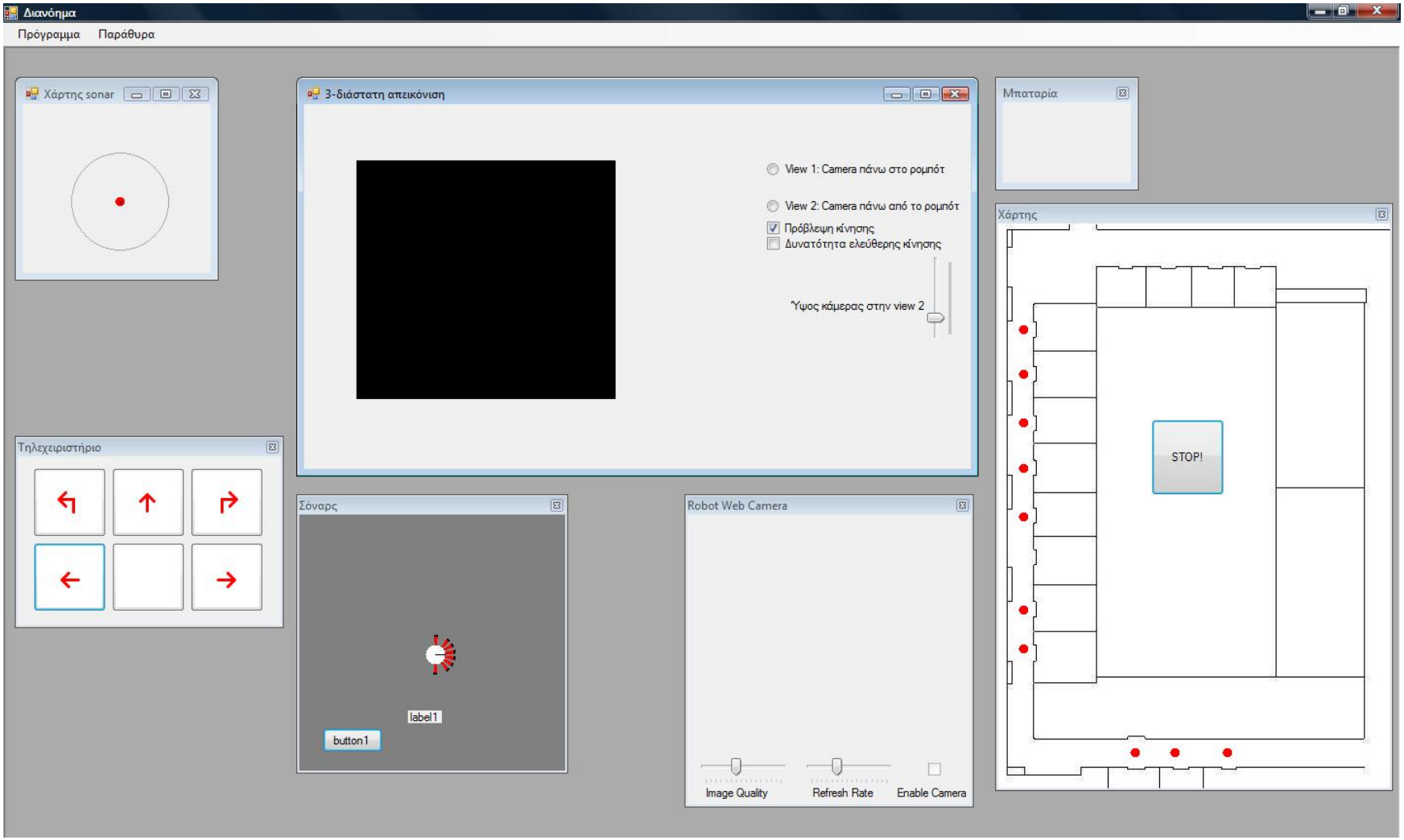
3.1 Πριν την εκτέλεση

Το βασικό παράθυρο του GUI πριν ξεκινήσει να εκτελείται το πρόγραμμα φαίνεται στο παρακάτω σχήμα.

Στο πάνω αριστερά μέρος της οθόνης βλέπουμε τον χάρτη sonar . Ο χάρτης αυτός όπως έχουμε περιγράψει και παραπάνω δημιουργείται καθώς το πρόγραμμα εκτελείται και μας δείχνει τι βρίσκεται σε μια περιοχή γύρωθεν του ρομπότ. Από κάτω ακριβώς βρίσκεται το τηλεχειριστήριο. Υπάρχει δυνατότητα αποστολής έξι βασικών εντολών: τρεις εντολές μετατόπισης (μπροστά, δεξιά και αριστερα), δύο εντολές περιστροφής (90^o δεξιά ή 90^o αριστερά) και μια εντολή σταματήματος. Στην πάνω δεξιά μεριά της διεπαφής βρίσκεται ένα παράθυρο το οποίο μας ενημερώνει για την κατάσταση της μπαταρίας του ρομπότ μέσω ενός από τα παρακάτω εικονίδια.



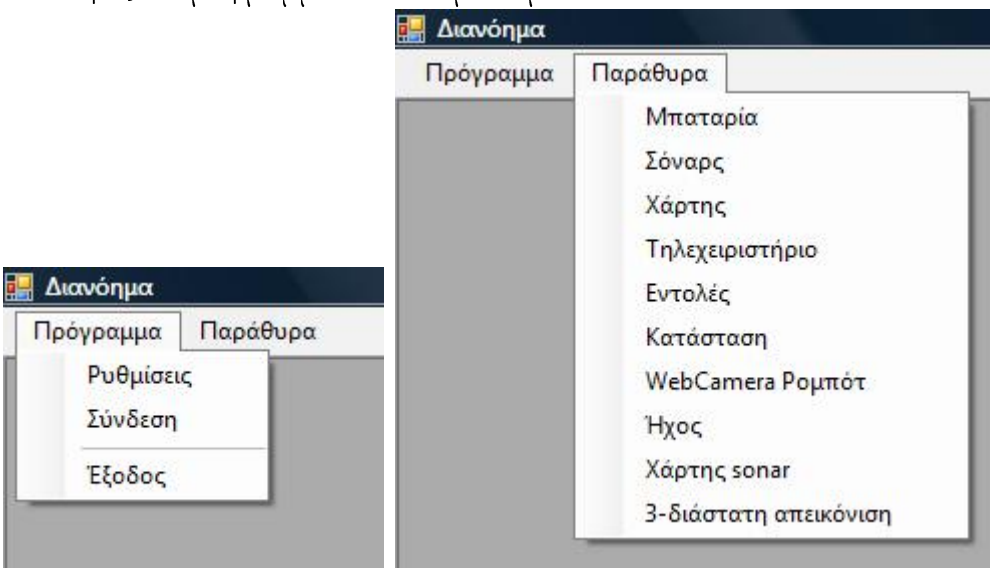
Στο μέσο του κάτω μέρους της οθόνης βρίσκονται δύο παράθυρα. Το αριστερό από αυτά είναι το παράθυρο των Ultrasonars. Εκεί βλέπουμε σχηματικά τις τιμές που στέλνει καθένας από τους οκτώ αισθητήρες της αισθητήριας διάταξης. Δεξιά βρίσκεται το παράθυρο το οποίο δείχνει το βίντεο από την κάμερα που βρίσκεται πάνω


Διανόημα
Πρόγραμμα
Παράθυρα
Χάρτης sonar
3-διάστατη απεικόνιση
View 1: Camera πάνω στο ρομπότ
View 2: Camera πάνω από το ρομπότ
Πρόβλεψη κίνησης
Δυνατότητα ελεύθερης κίνησης
Ύψος κάμερας στην view 2
Μπαταρία
Χάρτης
STOP!
Τηλεχειριστήριο
Σόναρς
label1
button1
Robot Web Camera
Image Quality
Refresh Rate
Enable Camera

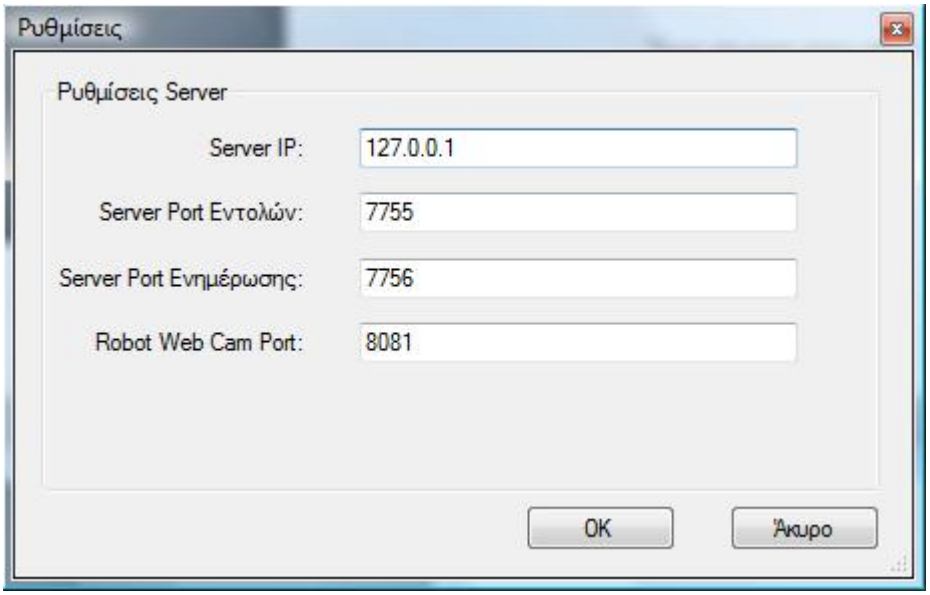
στο ρομπότ. Στη δεξιά μεριά του GUI υπάρχει ένας δισδιάστατος χάρτης του περιβάλλοντος στο οποίο κινείται το ρομπότ. Σ' αυτόν φαίνονται προς το παρόν μόνο η θέση των τοίχων και οι πιθανές θέσεις – στόχοι προς τις οποίες θα κινηθεί το ρομπότ. Η αποστολή ενός σημείου-στόχου στο ρομπότ μπορεί να ακυρωθεί από ένα κουμπί "STOP", το οποίο βρίσκεται στο κέντρο του παραθύρου. Τέλος το μεγαλύτερο κομμάτι της διεφαπής αποτελείται από τον τρισδιάστατο χάρτη, στον οποίο προς το παρόν φάινονται κάποιες από τις δυνατότητες προβολής που είναι διαθέσιμες.

3.2 Προς την εκτέλεση

Στο πάνω μέρος της οθόνης βρίσκεται ένα μενού με δύο κυρίως επιλογές: Πρόγραμμα και Παράθυρα.



Από την πρώτη καρτέλα και συγκεκριμένα από την επιλογή "Ρυθμίσεις", ρυθμίζουμε τις παραμέτρους που είναι σχετικές με την επικοινωνία της διεπαφής με το ρομπότ.

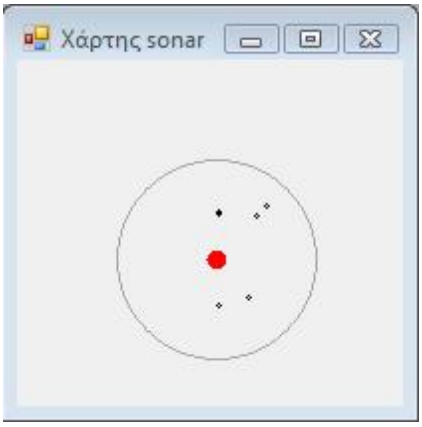


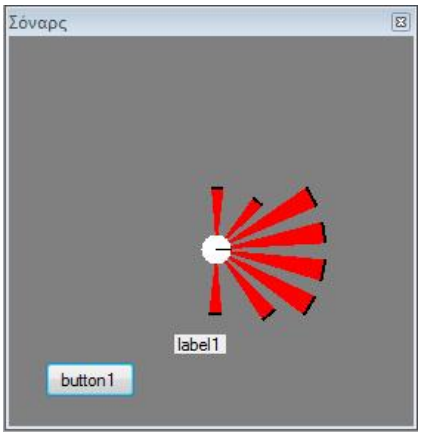
Επιλέγοντας "Σύνδεση" γίνεται η σύνδεση με το ρομπότ και ξεκινάει ουσιαστικά η εκτέλεση της εφαρμογης, ενώ τέλος η επιλογή "Εξοδος" τερματίζει το πρόγραμμα. Από το μενού "Παράθυρα" μπορούμε να επιλέξουμε κάποιο συγκεκριμένο παράθυρο για χρήση, ή να το ξαναδημιουργήσουμε σε περίπτωση που το έχουμε κλείσει.

3.3 Εκτέλεση του Προγράμματος

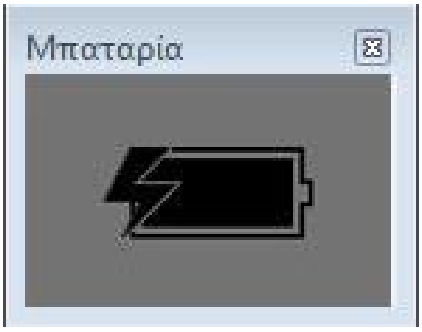
Με το που πατάμε "Σύνδεση" παρατηρούμε ότι:

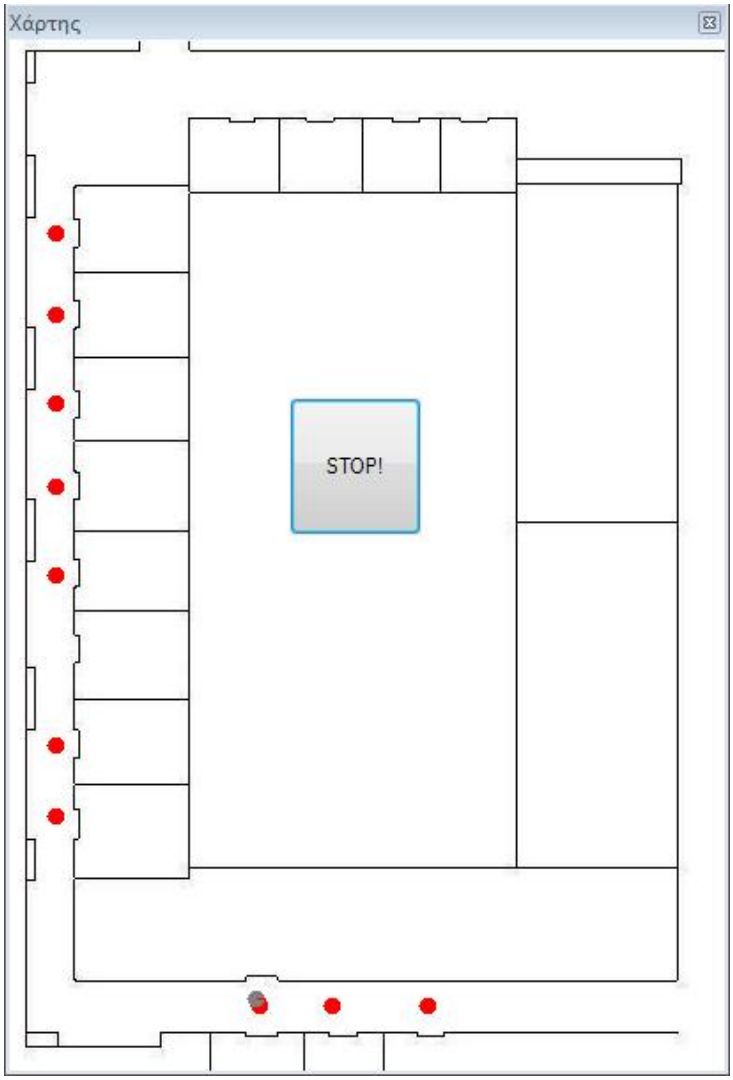
- Ο χάρτης sonar αρχίζει να σχηματίζεται
- Οι αισθητήρες sonars αρχίζουν να αποκτούν τιμή
- Εμφανίζεται η εικόνα της μπαταρίας
- Στο δισδιάστατο χάρτη βλέπουμε τη θέση και τον προσανατολισμό του ρομπότ


Χάρτης sonar

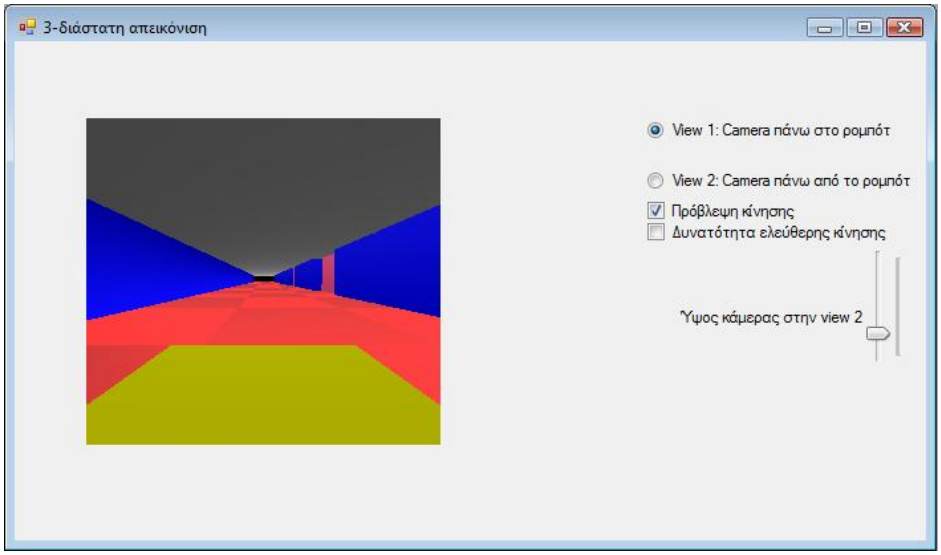

Σόναρς
label1
button1

.

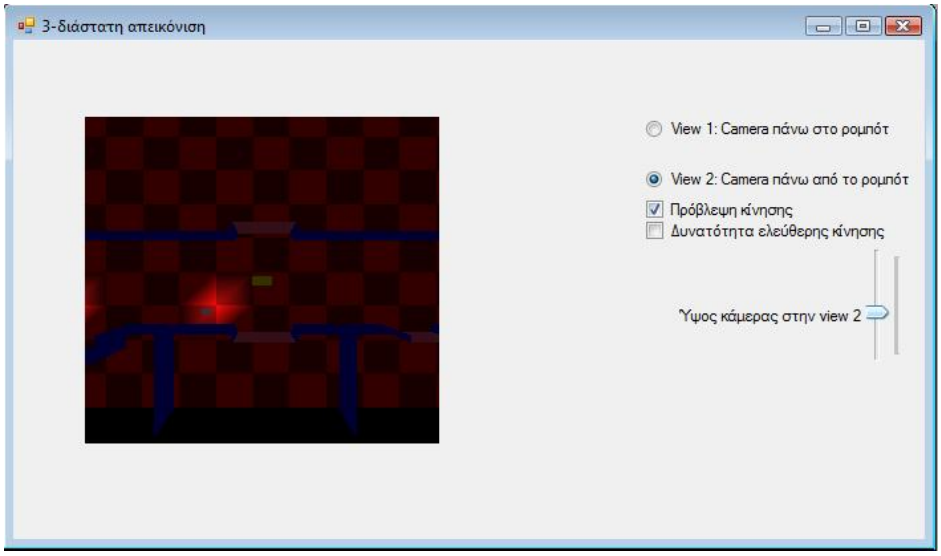

Μπαταρία



Σχετικά με το παράθυρο του τρισδιάστατου χάρτη, για να εμφανιστεί η εικόνα, θα πρέπει να επιλέξουμε έναν από τους δύο εναλλακτικούς τύπους προβολής. Ο πρώτος (view 1) μας δείχνει την εικόνα που θεωρητικά θα φαινόταν από την κάμερα που βρίσκεται πάνω στο ρομπότ. Ο δεύτερος (view 2) μας δείχνει την εικόνα από μια κάμερα που βρίσκεται στις συντεταγμένες (x,y) που βρίσκεται το ρομπότ, αλλά σε κάποιο ύψος πάνω από αυτό. Είναι ουσιαστικά μια κάτοψη, στης οποίας το κέντρο βρίσκεται κάθε φορά το ρομπότ.



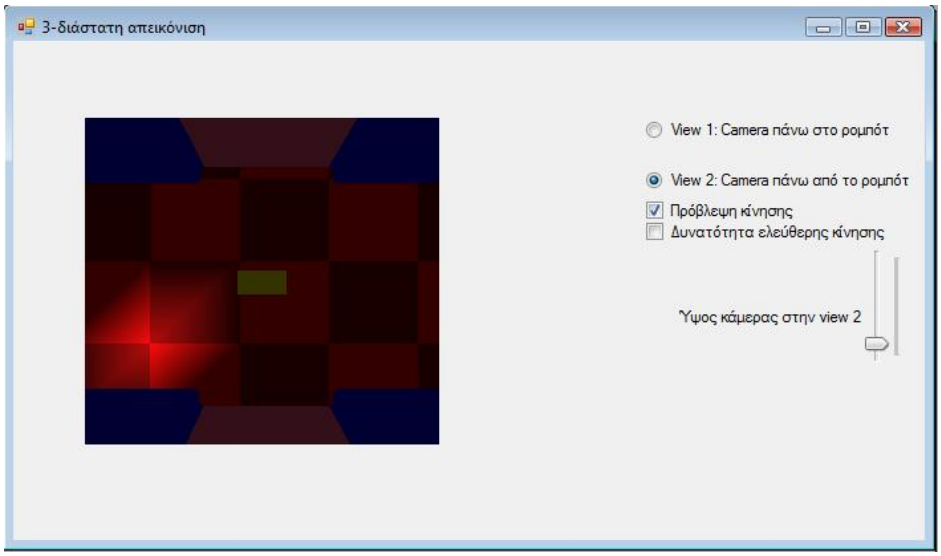
view 1: onboard camera



view 2: Κάτοψη

Όταν έχουμε επιλέξει τον δεύτερο τύπο προβολής, μπορούμε μέσω μίας μπάρας κύλισης, να επιλέξουμε το ύψος στο οποίο θα βρίσκεται η κάμερα. Δύο παραδείγματα για τις ακραίες τιμές ύψους φαίνονται στις δύο εικόνες παρακάτω:

view 2: μεγάλο ύψος



view 2: μικρό ύψος

Ένα ακόμη σημαντικό πλεονέκτημα που μας παρέχει η γραφική διεπαφή είναι η δυνατότητα κίνησης της κάμερας στο χώρο. Αυτό γίνεται επιλέγοντας το κουτάκι με την ένδειξη "Δυνατότητα ελεύθερης κίνησης". Με το που κάνουμε αυτήν την επιλογή θα εμφανιστεί στη διεπαφή ένα βοήθημα σχετικά με το πώς μπορούμε να μετακινήσουμε την κάμερα. Οι δυνατότητες κίνησης είναι:

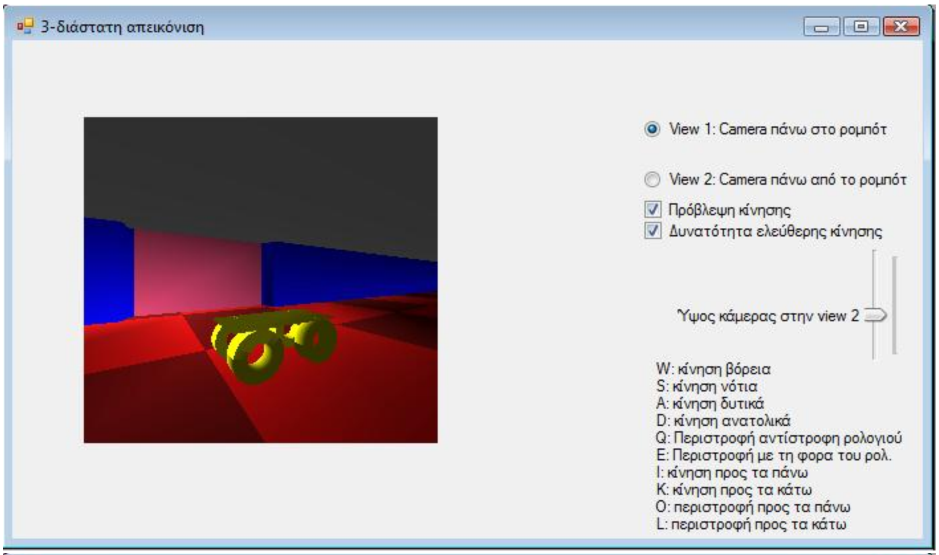
- Μετατόπιση σε τρεις διευθύνσεις (δύο διευθύνσεις στο επίπεδο που είναι παράλληλο με το έδαφος και μία κάθετη σε αυτό)

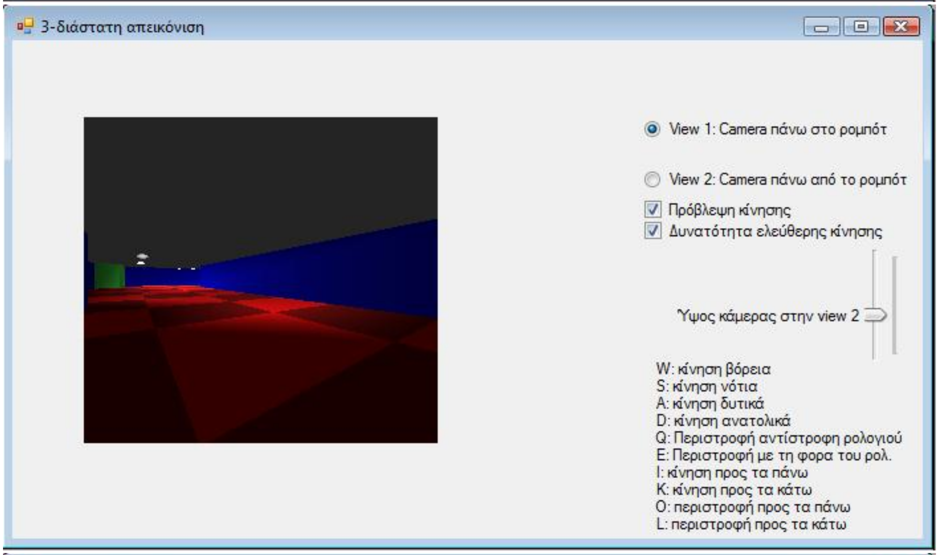
- Περιστροφική κίνηση γύρω από άξονα κάθετο στο έδαφος
- Περιστροφική κίνηση γύρω από άξονα παράλληλο στο έδαφος και κάθετο στη διεύθυνση που "κοιτάει" το ρομπότ.

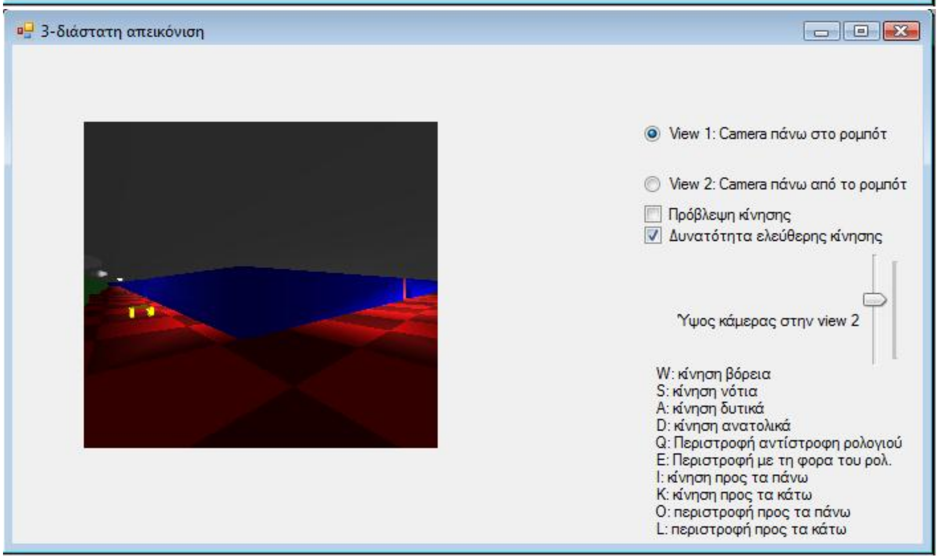
Αναλυτικά τα πλήκτρα εντολών φαίνονται στον παρακάτω πίνακα:

Πλήκτρο	Αποτέλεσμα
W	Κίνηση βόρεια
S	Κίνηση νότια
A	Κίνηση δυτικά
D	Κίνηση ανατολικά
Q	Περιστροφή αντίθετη της φοράς των δεικτών του ρολογιού
E	Περιστροφή σύμφωνα με τη φορά των δεικτών του ρολογιού
I	Κίνηση προς τα πάνω
K	Κίνηση προς τα κάτω
O	Περιστροφή προς τα πάνω
L	Περιστροφή προς τα κάτω

Μερικές εικόνες του τρισδιάστατου χάρτη στην επιλογή ελεύθερης κίνησης φαίνονται παρακάτω:


3-διάστατη απεικόνιση
View 1: Camera πάνω στο ρομπότ
View 2: Camera πάνω από το ρομπότ
Πρόβλεψη κίνησης
Δυνατότητα ελεύθερης κίνησης
Ύψος κάμερας στην view 2
W: κίνηση βόρεια
S: κίνηση νότια
A: κίνηση δυτικά
D: κίνηση ανατολικά
Q: Περιστροφή αντίστροφη ρολογιού
E: Περιστροφή με τη φορα του ρολ.
I: κίνηση προς τα πάνω
K: κίνηση προς τα κάτω
O: περιστροφή προς τα πάνω
L: περιστροφή προς τα κάτω


3-διάστατη απεικόνιση
View 1: Camera πάνω στο ρομπότ
View 2: Camera πάνω από το ρομπότ
Πρόβλεψη κίνησης
Δυνατότητα ελεύθερης κίνησης
Ύψος κάμερας στην view 2
W: κίνηση βόρεια
S: κίνηση νότια
A: κίνηση δυτικά
D: κίνηση ανατολικά
Q: Περιστροφή αντίστροφη ρολογιού
E: Περιστροφή με τη φορα του ρολ.
I: κίνηση προς τα πάνω
K: κίνηση προς τα κάτω
O: περιστροφή προς τα πάνω
L: περιστροφή προς τα κάτω


3-διάστατη απεικόνιση
View 1: Camera πάνω στο ρομπότ
View 2: Camera πάνω από το ρομπότ
Πρόβλεψη κίνησης
Δυνατότητα ελεύθερης κίνησης
Ύψος κάμερας στην view 2
W: κίνηση βόρεια
S: κίνηση νότια
A: κίνηση δυτικά
D: κίνηση ανατολικά
Q: Περιστροφή αντίστροφη ρολογιού
E: Περιστροφή με τη φορα του ρολ.
I: κίνηση προς τα πάνω
K: κίνηση προς τα κάτω
O: περιστροφή προς τα πάνω
L: περιστροφή προς τα κάτω

4 Χάρτης sonar

4.1 Αισθητήριες διατάξεις

Οι αισθητήρες διακρίνονται σε δύο κατηγορίες: Τους αισθητήρες εσωτερικής κατάστασης, οι οποίοι παρέχουν πληροφορίες σχετικά με τις εσωτερικές παραμέτρους του ρομπότ (επίπεδο μπαταρίας, θέση των τροχών) και τους αισθητήρες εξωτερικής κατάστασης, οι οποίοι παρέχουν πληροφορίες σχετικά με το περιβάλλον (θερμοκρασία, υγρασία). Ανάλογα με τη μέθοδο την οποία χρησιμοποιούν, για να "αισθανθούν", χωρίζονται στις εξής κατηγορίες:

- Αισθητήρες αφής: Οι αισθητήρες αυτοί είναι κυρίως αισθητήρες επαφής οι οποίοι στηριζόμενοι σε ιδιότητες μαγνητικής αντίστασης, πιεζοηλεκτρικών υλικών, ηλεκτρικής αντίστασης, μαγνητοελαστικότητας κλπ, μπορούν να δημιουργήσουν μια αίσθηση αφής - δύναμης, ώστε να παράξουν στο ρομπότ τα χαρακτηριστικά των επιφανειών με τις οποίες βρίσκεται σε επαφή. Υπάρχουν βέβαια και αισθητήρες οι οποίοι περνώντας αρκετά κοντά από μια επιφάνεια, χωρίς να υπάρχει επαφή, εκμεταλλευόμενοι μεγέθη, όπως η χωρητικότητα, μπορούν να προσδώσουν παρόμοιο αποτέλεσμα.

- Αισθητήρες μέτρησης απόλυτης θέσης: Μετρούν την απόλυτη διεύθυνση κίνησης σε οριζόντιο επίπεδο και τη γωνία κλίσης του ρομπότικού συστήματος σε σχέση με το οριζόντιο επίπεδο.

- Επιταχυνσιονόμετρα - Γυροσκόπια: Τα επιταχυνσιονόμετρα είναι συστήματα αναρτημένης μάζας σε ελατήρια και εκμεταλλεύονται τους νόμους του Νεύτωνα για την εύρεση της επιτάχυνσης.

$$\{F = ma \qquad F = kx^2\} \tag{1}$$

$$\Rightarrow a = \frac{kx^2}{m} \tag{2}$$

Ανάλογα δουλεύουν και τα γυροσκόπια τα οποία χρησιμοποιούν μια γρήγορα στρεφόμενη μάζα αναρτημένη σε αντίζυγα πυξίδας, ώστε να υπολογιστεί η γωνιακή επιτάχυνση.

- Αισθητήρες υπέρυθρων ακτίνων: Οι αισθητήρες αυτοί διακρίνονται σε ενεργητικούς και παθητικούς. Οι ενεργητικοί στέλνουν μια υπέρυθρη δέσμη και μετρούν το χρόνο που χρειάζεται να επιστρέψει αφού ανακλαστεί σε ένα αντικείμενο. Έτσι εκτός από την παρουσία/απουσία του αντικειμένου προσδιορίζουμε και τη θέση του. Οι παθητικοί αισθητήρες ανιχνεύουν αντικείμενα λαμβάνοντας την υπέρυθρη ακτινοβολία την οποία αυτά εκπέμπουν.

- Αισθητήρες υπερήχων: Οι αισθητήρες αυτοί εκπέμπουν ένα ηχητικό σήμα και μετρούν το χρόνο επιστροφής του ανακλόμενου σήματος. Η απόσταση βρίσκεται από τον τύπο:

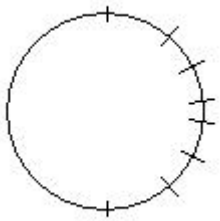
$$2d = c\Delta t \Leftrightarrow d = 1/2c\Delta t \tag{3}$$

,όπου d είναι η ζητούμενη απόσταση, Δt ο χρόνος από την εκπομπή ως τη λήψη του σήματος και c η ταχύτητα του ήχου στο συγκεκριμένο μέσο που βρισκόμαστε.

4.2 Υλοποίηση του χάρτη

Το πρώτο πράγμα που έπρεπε να υλοποιηθεί στην εφαρμογή μας ήταν η χαρτογράφηση του χώρου. Για να γίνει αυτό, πρέπει να αντλήσουμε πληροφορίες από τους αισθητήρες sonar του ρομπότ. Οι αισθητήρες είναι τα εξαρτήματα που πληροφορούν τη μονάδα ελέγχου για την κατάσταση του περιβάλλοντος του αλλά και για τη δική του κατάσταση. Ο πιο ακριβής τρόπος αίσθησης του περιβάλλοντος είναι η μηχανική όραση. Όμως είναι και ο

πιο ακριβός αφού απαιτείται μεγάλη υπολογιστική ισχύς για την επεξεργασία των εικόνων που λαμβάνονται από τις κάμερες. Έτσι τα τελευταία χρόνια χρησιμοποιείται η τεχνολογία sonar (SOund NAvigation and Ranging). Τα ηχοεντοπιστικά αυτά συστήματα χρησιμοποιούνται για τον εντοπισμό αντικειμένων στην περιοχή τους και ο κύριος λόγος ύπαρξης τους είναι η αποφυγή συγκρούσεων με αντικείμενα του κοντινού περιβάλλοντος. Η λειτουργία τους στηρίζεται στη μέτρηση του χρόνου που χρειάζεται ένα ηχητικό σήμα που εκπέμπεται από τη συσκευή να ανακλαστεί σε ένα εμπόδιο και να επιστρέψει πίσω. Το ρομπότ της εφαρμογής μας είναι εφοδιασμένο με 8 αισθητήρες σε περιμετρική διάταξη, οι οποίοι σχηματίζουν ένα ημικύκλιο. Πιο συγκεκριμένα, αν θεωρήσουμε ότι η ευθεία που κοιτάει το ρομπότ είναι 0^o, οι αισθητήρες αυτοί βρίσκονται σε γωνίες -90^o, -50^o, -30^o, -10^o, 10^o, 30^o, 50^o και 90^o. Σχηματικά βλέπουμε τη διάταξη αυτή στο παρακάτω σχήμα:



Η δουλειά τους είναι να ανιχνεύουν εμπόδια σε απόσταση μικρότερη των $2,75m$, καθώς και την απόσταση στην οποία αυτά βρίσκονται. Εφόσον στο χώρο κίνησης του ρομπότ δεν υπάρχουν εξωτερικογενή εμπόδια, κάθε εμπόδιο θα αντιπροσωπεύει ένα σημείο του τοίχου. Εμείς ανά τακτά χρονικά διαστήματα λαμβάνουμε τις 8 αυτές τιμές των αισθητήρων μαζί με τη θέση στην οποία βρίσκεται την εκάστοτε στιγμή το ρομπότ. Έτσι μπορούμε να βρούμε το σημείο του χάρτη στο οποίο υπάρχει εμπόδιο από τον τύπο:

$$f(L,\theta) = (Lcos\theta, Lsin\theta) \tag{4}$$

$$\theta = \theta_R + \theta_i \tag{5}$$

,όπου (x, y) είναι η θέση του εμποδίου σε καρτεσιανές συντεταγμένες, (Rx, Ry) είναι η θέση του ρομπότ σε καρτεσιανές συντεταγμένες και $f(L, \theta)$ η συνάρτηση που μετατρέπει τη θέση του σημείου σε σχέση με το ρομπότ από πολικές σε καρτεσιανές συντεταγμένες. Ουσιαστικά ισχύει ότι

$$(x, y) = (Rx, Ry) + f(L, \theta) \tag{6}$$

όπου L είναι η τιμή του σοναρ i και θ_R η γωνία του ρομπότ και θ_i η γωνία του αισθητήρα. Έτσι κάθε στιγμή που γίνεται ενημέρωση γνωρίζουμε έως 8 (εφόσον αρκετές φορές δεν υπάρχει εμπόδιο στην απόσταση των $2,75m$) επιπλέον σημεία του χάρτη. Φτιάχνοντας λοιπόν έναν πίνακα θα μπορούσαμε να του προσθέτουμε συνεχώς καινούρια σημεία και τελικά μετά από κάποιο χρόνο περιήγησης του ρομπότ να γνωρίζουμε μεγάλη περιοχή του χάρτη.

4.3 Προβλήματα και λύσεις

Θεωρητικά αυτή η μέθοδος φαντάζει καλή. Στην πράξη όμως υπάρχουν αρκετά μειονεκτήματα. Το σημαντικότερο απ΄ όλα είναι το θέμα της οικονομίας του χώρου. Αν κάθε περίοδο χρόνου (έστω κάθε δευτερόλεπτο) προσθέτουμε στον πίνακα μας (έως) 8 σημεία, τότε θα πρέπει να έχουμε στη διάθεσή μας έναν τεράστιο πίνακα, κάτι που σε μια υπολογιστική εφαρμογή δεν είναι οικονομικό. Αυτό που κάνουμε, για να επιλύσουμε αυτό το πρόβλημα, είναι κάθε φορά που λαμβάνουμε ένα καινούριο σημείο, να ελέγχουμε τα προηγούμενα αποθηκευμένα σημεία και να κοιτάμε πόσα από αυτά βρίσκονται σε μια κοντινή περιοχή του. Θεωρούμε ότι σε κάθε περιοχή $50cm^2$ είναι αρκετό να γνωρίζουμε 5 σημεία. Έτσι αν δεχτούμε μια τιμή για την οποία γνωρίζουμε ήδη 5 ακόμη σημεία στην γύρω περιοχή, τότε την απορρίπτουμε χωρίς αυτό να έχει ιδιαίτερη επίδραση στο τελικό αποτέλεσμα. Με τον τρόπο αυτόν αποφεύγουμε να επιβαρύνουμε τη μνήμη του προγράμματος

με επιπρόσθετες ανούσιες πληροφορίες.

Αφού επιλύσαμε το προηγούμενο πρόβλημα που προέκυψε, έπρεπε να σκεφτούμε πως θα γίνει η χαρτογράφηση περισσότερο οικονομική, χωρίς όμως να χάσουμε σημαντικές πληροφορίες. Αυτό που σκεφτήκαμε ήταν ότι θα μας ήταν αρκετό για τη συγκεκριμένη εφαρμογή να γνωρίζουμε μόνο τον χάρτη σε μια ευρεία περιοχή γύρω από το ρομπότ, χωρίς να είναι απαραίτητο να θυμόμαστε την περιοχή από την οποία περάσαμε πριν από αρκετή ώρα και η οποία βρίσκεται πλέον μακριά από το ρομπότ. Έτσι θα μπορούσαμε να ξεχνάμε τα σημεία τα οποία βρίσκονται μακριά από την τρέχουσα θέση του ρομπότ και να γνωρίζουμε μόνο τα σημεία που βρίσκονται σε μία περιοχή έστω $3,5m^2$ γύρω από το ρομπότ. Με τον τρόπο αυτό, το αποτέλεσμα είναι να έχουμε στον πίνακα μας κάθε φόρα όχι περισσότερα από 100 σημεία κατά μέσο όρο καθιστώντας το πρόγραμμά μας έτσι περισσότερο αποδοτικό. Αυτό βέβαια δε σημαίνει ότι ανά πάσα στιγμή δε μπορούμε να αλλάξουμε τις παραμέτρους, ώστε να μπορεί το ρομπότ να θυμάται μεγαλύτερη περιοχή του χάρτη.

5 3d χάρτης

5.1 Η Γλώσσα προγραμματισμού OpenGL

Η γλώσσα προγραμματισμού την οποία επιλέξαμε, για να σχεδιάσουμε τον τρισδιάστατο χάρτη μας, είναι η opengl. Η OpenGL (Open Graphics Library) είναι μια γλώσσα που χρησιμοποιείται ευρέως για τον σχεδιασμό 2-διάστατων και 3-διάστατων γραφικών σε υπολογιστές. Για το σκοπό αυτό περιλαμβάνει πάνω από 250 εντολές μέσω των οποίων μπορούμε να σχεδιάσουμε πολύπλοκα τρισδιάστατα σκηνικά από απλά πρωταρχικά στοιχεία. Παράλληλα χρησιμοποιούμε και τη βιβλιοθήκη Glut (OpenGL Utility Toolkit), η οποία είναι μια βιβλιοθήκη με χρήσιμες συναρτήσεις για την OpenGL. Τέλος χρειαζόμαστε και το Tao Framework, ώστε να μπορέσουμε να χρησιμοποιήσουμε την OpenGL στην Visual C# στην οποία αναπτύσσουμε την εφαρμογή μας. Οι σημαντικότερες συναρτήσεις που χρησιμοποιούμε φαίνονται παρακάτω μαζί με το αποτέλεσμά τους:

Συνάρτηση	Αποτέλεσμα
$glLightfv(...)$	Ορίζονται παράμετροι της πηγής φωτισμού (πχ θέση, προσανατολισμός, χρώμα, ένταση, εξασθένηση)
$gluPerspective(...)$	Χρησιμοποιούμε προοπτική προβολή.
$glMaterialfv(...)$	Ορίζονται παράμετροι των αντικειμένων η μπροστινή επιφάνεια, το χρώμα, η λάμψη, η αντανάκλαση.
$glutsolidCube(...)$	Σχεδιάζεται ένας κύβος. Χρησιμοποιώντας επιπλέον συναρτήσεις μπορούμε να αλλάξουμε το μέγεθος του και το σχήμα του, ώστε να σχηματίσουμε ορθογώνια παραλληλεπίπεδα.
$gluLookAt(...)$	Ορίζουμε που βρίσκεται το "μάτι" της κάμερας και προς ποια μεριά κοιτάει.

Αρχικά δημιουργούμε το πλάισιο μέσα στο οποίο θα βρίσκεται

το σχέδιό μας, ορίζοντας παράλληλα μερικές βασικές παραμέτρους σχετικές με τα χρώματα και το βάθος προβολής.

5.2 Ο εξομοιωτής *SRIsim*

Ο εξομοιωτής *SRIsim* έχει αποθηκευμένο τον χάρτη σε ένα τύπου *.wld* αρχείο. Εμείς καλούμαστε να επεξεργαστούμε το αρχείο, ώστε να φτιάξουμε τον δικό μας χάρτη. Το αρχείο έχει τη μορφή:

```
;; <filename>
;; <Comments>
width  <number>
height <number>
origin <number> <number>
position <number> <number> <number>
;; line
<number> <number> <number> <number>
```

όπου οι δύο τελευταίες γραμμές επαναλαμβάνονται όσες φορές χρειάζεται προκειμένου να συμπληρωθεί ο χάρτης. Συγκεκριμένα οι τέσσερις αριθμοί ορίζουν τις συντεταγμένες των άκρων του ευθύγραμμου τμήματος που σχεδιάζουμε κάθε φορά. Όσο για τους αριθμούς στην αρχή του αρχείου, αυτοί προσδιορίζουν το μέγεθος του χάρτη καθώς και πληροφορίες σχετικές με τη θέση και τον προσανατολισμό του ρομπότ.

Εμείς τοποθετούμε τις τιμές των άκρων του ευθύγραμμου τμήματος σε μια συνάρτηση που έχουμε δημιουργήσει, την

```
public void CreateWall3d(float x1, float y1, float x2,
float y2, float height, float R, float G, float B)
```

η οποία δέχεται ως ορίσματα:

- τις 4 συντεταγμένες δύο σημείων του επιπέδου $A(x1, y1)$ και $B(x2, y2)$

- μία τιμή για ύψος και
- 3 τιμές οι οποίες ορίζουν το χρώμα στη μορφή RGB

Το αποτέλεσμα είναι η δημιουργία ενός ορθογωνίου παραλληλεπιπέδου με δοσμένο ύψος και χρώμα στη θέση, ώστε το ευθύγραμμο τμήμα AB να αποτελεί τη διαγώνιο της βάσης του. Επιλέγουμε ως ύψος την τιμή 900 και χρώμα μπλε για τους τοίχους, κοκκινωπό για τις πόρτες και γκρι για τα σώματα θέρμανσης. Όσον αφορά το χρώμα του εδάφους, για να είναι ευκολότερος ο υπολογισμός της σχετικής απόστασης δύο αντικειμένων με το μάτι, έχουμε χρησιμοποιήσει διχρωμία σε σχηματισμό σκακιέρας με δύο αποχρώσεις του κόκκινου.

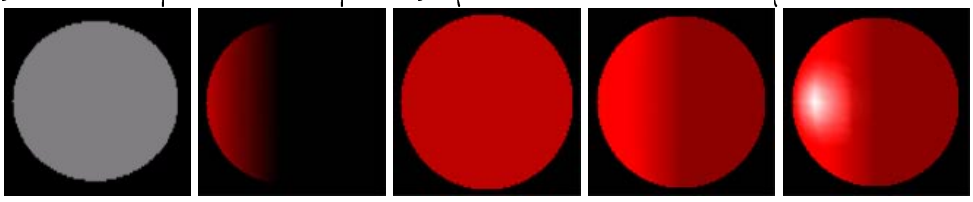
5.3 Φωτισμός

5.3.1 Φωτισμός στην OpenGL

Η OpenGL υποστηρίζει σημειακές, κατευθυντικές, μακρινές πηγές και περιβάλλοντα φωτισμό. Μπορούμε να χρησιμοποιήσουμε τουλάχιστον οκτώ πηγές φωτισμού (GL_LIGHT0, GL_LIGHT1, ..., GL_LIGHT7). Ο φωτισμός πρέπει να ενεργοποιηθεί συνολικά, κι αυτό γίνεται με τη χρήση της εντολής glEnable(GL_LIGHTING). Όμως και κάθε πηγή πρέπει να οριστεί και να ενεργοποιηθεί χωριστά. Η ενεργοποίηση γίνεται με εντολές τύπου glEnable(GL_LIGHTi) και η απενεργοποίηση με εντολές τύπου glDisable(GL_LIGHTi), όπου i είναι ο αριθμός της επιθυμητής πηγής φωτισμού. Στη συνέχεια θα πρέπει να προσδιορίσουμε ξεχωριστά τις παραμέτρους της κάθε πηγής. Ο προσδιορισμός τους γίνεται με εντολές δύο τύπων:

- glLightfv(πηγή, παράμετρος, διάνυσμα)
- glLightf(πηγή, παράμετρος, τιμή)

ανάλογα αν οι παράμετροι είναι διανυσματικές ή βαθμωτές. Η πρώτη παράμετρος που πρέπει να οριστεί είναι η θέση της πηγής (GL_POSITION), η οποία είναι διανυσματική παράμετρος. Εδώ έχουμε δύο επιλογές. Μπορούμε είτε να ορίσουμε τη φωτεινή πηγή σε κάποια συγκεκριμένη θέση είτε να θεωρήσουμε ότι η φωτεινή πηγή βρίσκεται πολύ μακριά και να ορίσουμε απλά την κατεύθυνση του φωτός. Στην πρώτη περίπτωση το διάνυσμα μας είναι της μορφής (x, y, z, 1), όπου x, y, και z είναι οι συντεταγμένες της θέσης της φωτεινής πηγής, ενώ στη δεύτερη περίπτωση το διάνυσμα έχει τη μορφή (x, y, z, 0),όπου x, y, z είναι πλέον ένα διάνυσμα που δείχνει την κατεύθυνση του φωτός. Στη συνέχεια πρέπει να ορίσουμε τα χαρακτηριστικά του φωτός. Υπάρχουν τρεις συνιστώσες φωτός. Το περιβάλλον φως (ambient light) είναι το φως που παρέχει ομοιόμορφο φωτισμό σε ένα χώρο. Δηλαδή η ένταση του φωτός σε όλα τα σημεία του χώρου είναι η ίδια. Το φως διαχυτικής ανάκλασης (diffuse light) είναι το φως που ανακλάται προς όλες τις κατευθύνσεις, όταν προσπίπτει σε κάποια επιφάνεια και το αποτέλεσμα του εξαρτάται από την κάθε γωνία. Τέλος υπάρχει και το φως κατευθυντικής ανάκλασης (specular light) , το οποίο είναι σαν το φως διαχυτικής ανάκλασης, αλλά ανακλάται με πιο έντονο και ομοιόμορφο τρόπο. Και τα τρία είδη φωτός μαζί μας δίνουν την εντύπωση ενός τρισδιάστατου αντικειμένου.

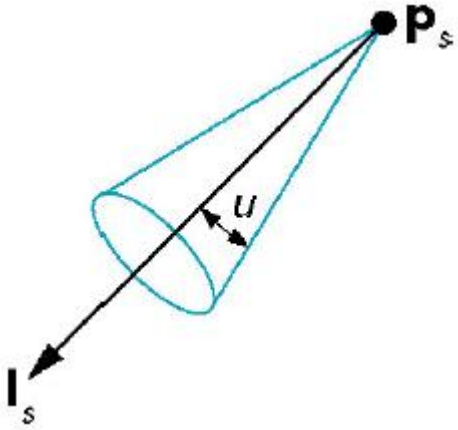


Στην πρώτη εικόνα έχουμε μόνο περιβάλλον φως και το αντικείμενο φαίνεται δισδιάστατο. Στη δεύτερη εικόνα, ένα κόκκινο φως διαχυτικής ανάκλασης φωτίζει ένα μαύρο αντικείμενο. Εδώ αρχίζει και φαίνεται η τρισδιάστατη μορφή του. Στην τρίτη εικόνα βλέπουμε μια τρισδιάστατη σφαίρα να φωτίζεται μόνο από περιβάλλον φως. Στην επόμενη εικόνα προσθέτουμε και φως

διαχυτικής ανάκλασης και τέλος στην τελευταία εικόνα έχουμε προσθέσει και φως κατευθυντικής ανάκλασης.

Οι παράμετροι για τις παραπάνω συνιστώσες του φωτός είναι GL_AMBIENT, GL_DIFFUSE και GL_SPECULAR αντίστοιχα. Σαν διάνυσμα χρησιμοποιούμε έναν πίνακα της μορφής (r, g, b, 1) το οποίο μας δείχνει ποιο χρώμα έχει η κάθε συνιστώσα φωτισμού.

Το φως έχει την ιδιότητα να εξασθενεί ανάλογα με την απόσταση. Η εξασθένηση αυτή μπορεί να είναι είτε σταθερή, είτε γραμμική είτε τετραγωνική. Δηλαδή είναι της μορφής $(a + bd + cd^2)^{-1}$,όπου d είναι η απόσταση και οι συντελεστές a,b,c ορίζονται χρησιμοποιώντας τις παραμέτρους GL_CONSTANT_ATTENUATION, GL_LINEAR_ATTENUATION και GL_QUADRATIC_ATTENUATION αντίστοιχα.



Έχουμε επίσης τη δυνατότητα να ορίσουμε κατευθυντικές πηγές φωτισμού, όπως στο σχήμα δίπλα, οι οπόίες εκπέμπουν φως σε στενή περιοχή γωνιών. Η κορυφή Ps είναι η θέση του φωτός, το διάνυσμα Is ορίζεται από την παράμετρο GL_SPOT_DIRECTION και η γωνία u από την παράμετρο GL_SPOT_CUTOFF. Τέλος οι φωτεινές πηγές δεν είναι ορατές. Για να μπορέσουμε να "δούμε" μια φωτεινή πηγή που ακτινοβολεί (πχ λάμπα), αυτό που μπορούμε να κάνουμε είναι να τοποθετήσουμε ένα αντίστοιχο αντικείμενο στην ίδια θέση με την πηγή.

5.3.2 Υλοποίηση του φωτισμού

Στο πρόγραμμά μας έχουμε φτιάξει μια συνάρτηση, την InitLights(), μέσα στην οποία χειριζόμαστε τις πηγές φωτισμού. Συγκεκριμένα

δημιουργούμε έξι κατευθυντικές φωτεινές πηγές στο ταβάνι πάνω από τους διαδρόμους. Οι παράμετροι για τις συνιστώσες του φωτός ορίζονται ως:

```
float[] lightAmbient = { 0.1f, 0.1f, 0.1f, 1.0f };
float[] lightDiffuse = { 1.0f, 1.0f, 1.0f, 1.0f };
float[] lightSpecular = { .1f, .1f, .1f, 1.0f };
```

τις οποίες περνάμε σε κάθε μια πηγή. Ενδεικτικά φαίνονται παρακάτω οι εντολές που χρησιμοποιούμε για την φωτεινή πηγή GL_LIGHT1

```
Gl.glLightfv(Gl.GL_LIGHT1, Gl.GL_AMBIENT, lightAmbient);
Gl.glLightfv(Gl.GL_LIGHT1, Gl.GL_DIFFUSE, lightDiffuse);
Gl.glLightfv(Gl.GL_LIGHT1, Gl.GL_SPECULAR, lightSpecular);
Gl.glLightfv(Gl.GL_LIGHT1, Gl.GL_POSITION, lightPosition1);
```

ενώ επιλέγουμε ως γωνία αποκοπής τις 40^o με την παρακάτω εντολή

```
Gl.glLightf(Gl.GL_LIGHT1, Gl.GL_SPOT_CUTOFF, 40f);
```

Τέλος ενεργοποιούμε τον φωτισμό και κάθε πηγή χωριστά:

```
Gl.glEnable(Gl.GL_LIGHTING);
Gl.glEnable(Gl.GL_LIGHT1);
Gl.glEnable(Gl.GL_LIGHT2);
Gl.glEnable(Gl.GL_LIGHT3);
Gl.glEnable(Gl.GL_LIGHT4);
Gl.glEnable(Gl.GL_LIGHT5);
```

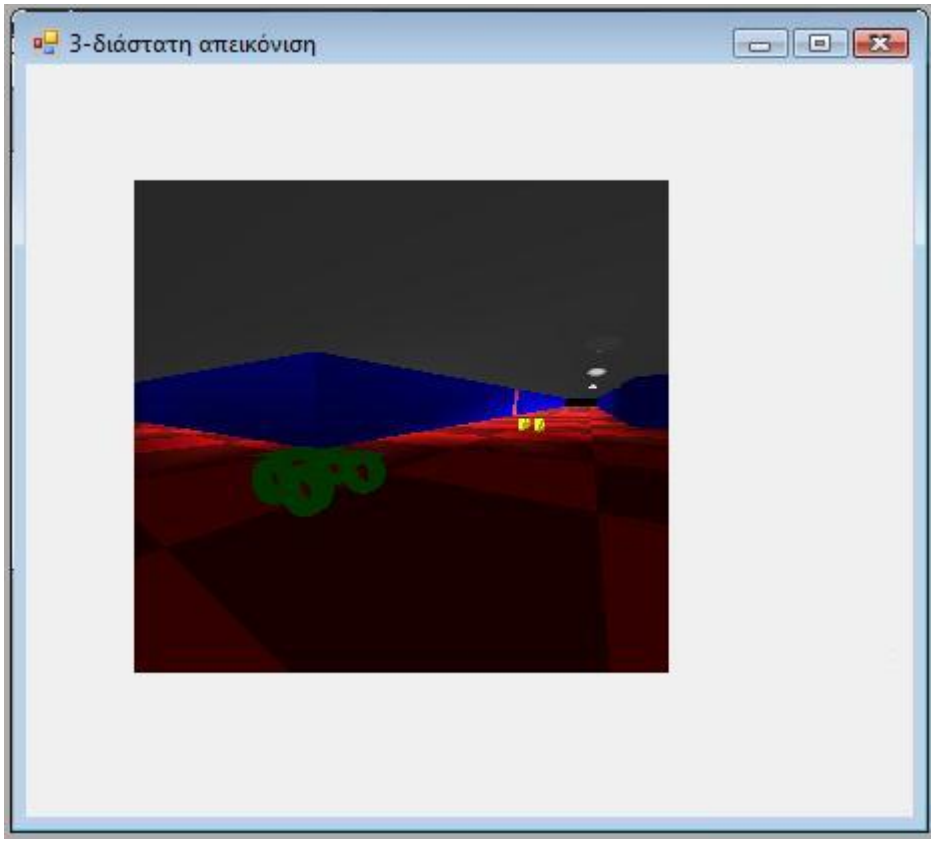
Τη συνάρτηση αυτή την καλούμε μέσα στη συνάρτηση Paint(), δηλαδή κάθε φορά που πρόκειται να επανασχεδιάσουμε την εικόνα μας.

5.4 Πρόβλεψη κίνησης

Απαραίτητη προϋπόθεση, για να μπορέσουμε να σχεδιάσουμε την τρισδιάστατη απεικόνιση του ρομποτικού μας συστήματος μαζί με το περιβάλλον εργασίας του, είναι να γνωρίζουμε κάθε στιγμή τις συντεταγμένες και τον προσανατολισμό του ρομπότ. Κάτι τέτοιο θα μπορούσε να συμβαίνει μόνο ιδανικά καθώς, όπως έχουμε προαναφέρει, η χρονακαθυστέρηση στην επικοινωνία του χειριστή με το ρομπότ δε μας επιτρέπει να λαμβάνουμε τις τιμές αυτές, όσο συχνά θα θέλαμε. Ο εξομοιωτής μας στέλνει συνεχώς τιμές κι εμείς τις λαμβάνουμε αμέσως, αφού στην εξομοίωση ο χρόνος μεταφοράς των πληροφοριών είναι πρακτικά μηδενικός. Αυτό που θα μπορούσαμε να κάνουμε είναι να προσομοιώσουμε κάποια καθυστέρηση. Για να το πετύχουμε, δημιουργούμε μία ουρά (queue) μέσα από την οποία περνάνε οι πληροφορίες, πριν καταλήξουν στο πρόγραμμά μας για επεξεργασία. Σχηματικά βλέπουμε τη διάταξη στην παρακάτω εικόνα:

Έτσι η πληροφορία της θέσης και του προσανατολισμού του ρομποτ έχει καθυστερήσει να έρθει και ουσιαστικά αυτό που βλέπουμε στη γραφική διεπαφή είναι το πού βρισκόταν το ρομπότ 2 ή 3 δευτερόλεπτα νωρίτερα. Βλέπουμε δηλαδή κατά κάποιον τρόπο το παρελθόν. Εμείς όμως, γνωρίζοντας την εντολή που έχουμε δώσει στο ρομπότ, ξέρουμε σε ποιο σημείο προσπαθεί να πάει και

ξέρουμε και την πορεία που πρέπει να ακολουθήσει. Έτσι μπορούμε να σχεδιάσουμε τη θέση στην οποία θα έπρεπε να βρίσκεται το ρομπότ, η οποία είναι η θέση στην οποία βρίσκεται, εφόσον δεν έχουν παρουσιαστεί προβλήματα και επομένως η θέση στην οποία θα δούμε το ρομπότ σε 2-3 δευτερόλεπτα. Για να διαχωρίζεται από το πραγματικό ρομπότ, συνηθίζεται μια τέτοια απεικόνιση να σχεδιάζεται με διακεκομμένες γραμμές. Στην εφαρμογή μας, εφόσον είναι ενεργοποιημένη η δυνατότητα πρόβλεψης κίνησης, η θέση αυτή του ρομπότ φαίνεται με πράσινο χρώμα και η θέση, της οποίας τις τιμές λάβαμε τελευταία, απεικονίζεται με κίτρινο χρώμα.



Έτσι στο χρόνο που μεσολαβεί μέχρι να λάβουμε τις νέες τιμές της θέσης του ρομπότ, μπορούμε να δούμε μια πρόβλεψη της κίνησης του ρομπότ.

5.5 Σχεδιασμός της απεικόνισης του ρομπότ

5.5.1 Το πολυγωνικό μοντέλο

Η πιο διαδεδομένη παράσταση αντικειμένων είναι αυτή που γίνεται χρησιμοποιώντας πολύγωνα. Τα πολύγωνα είναι ένα εύχρηστο συστατικό για την παράσταση των επιφανειών των αντικειμένων, και ειδικά όταν οι επιφάνειες αυτές είναι επίπεδες. Όμως ακόμα και στην περίπτωση που έχουμε να σχεδιάσουμε επιφάνειες μη επίπεδων πολύπλοκων αντικειμένων, το πολυγωνικό μοντέλο είναι ένα χρήσιμο εργαλείο. Ο σκοπός του πολυγωνικού μοντέλου είναι να προσεγγιστούν οι μη-επίπεδες επιφάνειες από ένα σύνολο πολυγώνων. Καθώς προσπαθούμε όμως να υλοποιήσουμε αυτήν την προσέγγιση δημιουργείται το σημαντικότερο ερώτημα που απασχολεί αυτό το μοντέλο: "Πόσα πολύγωνα πρέπει να χρησιμοποιηθούν·" Η απάντηση σ΄ αυτό το ερώτημα δεν είναι μονοδιάστατη. Συνήθως επιλέγουμε τον αριθμό των πολυγώνων διαισθητικά. Δύο πράγματα όμως που πρέπει να έχουμε στο μυαλό μας είναι ότι:

- περιοχές με μεγάλη καμπυλότητα πρέπει να αποτελούνται από μεγαλύτερο αριθμό πολυγώνων ανά μονάδα επιφάνειας σε σχέση με τις περισσότερο επίπεδες περιοχές
- ο αριθμός των πολυγώνων ενός αντικειμένου πρέπει να εξαρτάται από το τελικό μέγεθος του αντικειμένου στην εικόνα.

Δηλαδή, σε ένα αντικείμενο που (παρόλο που μπορεί να έχει μεγάλο μέγεθος) βρίσκεται μακριά από τον παρατηρητή και στην οθόνη θα καταλαμβάνει μικρό μέγεθος (μικρό αριθμό εικονοστοιχείων) δεν πρέπει να χρησιμοποιήσουμε πολύ μεγάλο αριθμό πολυγώνων. Επειδή όμως δε μπορούμε να ξέρουμε εκ των προτέρων για όλα τα αντικείμενα αν στην προβολή θα βρίσκονται κοντά ή μακριά από τον παρατηρητή, αυτό που μπορούμε να κάνουμε είναι να έχουμε αποθηκευμένα δύο μοντέλα για το ίδιο αντικείμενο, ένα με υψηλή

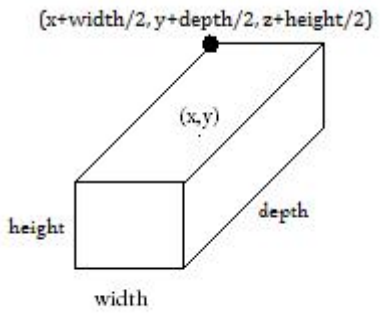
ανάλυση και ένα με χαμηλή και να χρησιμοποιούμε κάθε φορά το ενδεικνύμενο συναρτήσει της απόστασης.

5.5.2 Το ρομπότ

Θεωρούμε ότι το μοντέλο του ρομπότ μας αποτελείται από ένα ορθογώνιο παραλληλεπίπεδο στερεωμένο σε τέσσερις ρόδες. Για να σχεδιάσουμε κάθε κομμάτι, εργαζόμαστε ως εξής:

5.5.2.1 Σχεδιασμός του σώματος του ρομπότ Εφόσον χρησιμοποιώντας το πολυγωνικό μοντέλο μπορούμε να σχεδιάσουμε μόνο πολύγωνα, ο σχεδιασμός του ορθογωνίου παραλληλεπιπέδου θα γίνει σχεδιάζοντας έξι ορθογώνια παραλληλεπίπεδα με 2 κοινές κορυφές ανά δύο, τα οποία τα αποτελούν τις πλευρές του παραλληλεπιπέδου.

Εφόσον γνωρίζουμε κάθε φορά τη θέση του ρομπότ (x, y), το ύψος του κέντρου του πολυγώνου z και τις διαστάσεις του (width, depth, height), μπορούμε να υπολογίσουμε τις θέσεις των κορυφών του.

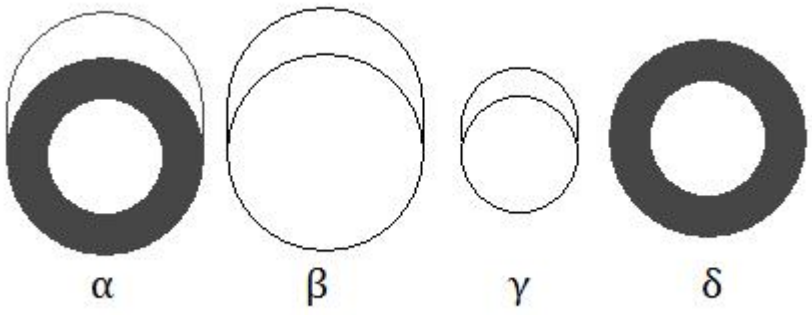


Οι οκτώ κορυφές είναι τα σημεία

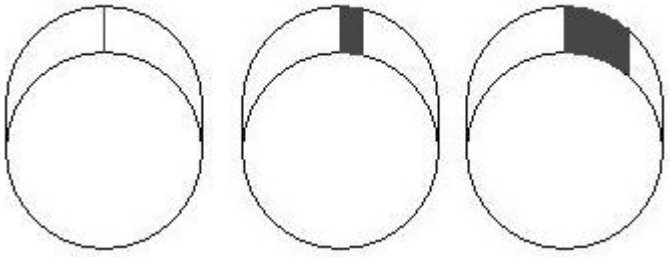
$$(x \pm width/2, y \pm depth/2, z \pm height/2) \tag{7}$$

για τους οκτώ δυνατούς συνδυασμούς + και -. Στο δίπλα σχήμα απεικονίζεται ενδεικτικά η μία κορυφή.

5.5.2.2 Σχεδιασμός των ροδών Φανταζόμαστε την κάθε ρόδα σαν ένα δαχτυλίδι (α), το οποίο αποτελείται από την εξωτερική επιφάνεια (β), την εσωτερική επιφάνεια (γ) και τα κομμάτια που ενώνουν τα άκρα των δύο αυτών επιφανειών μεταξύ τους (δ).



Είναι γνωστό ότι περιστρέφοντας έναν κύκλο γύρω από το κέντρο του δημιουργούμε μία σφαίρα. Γενικότερα, περιστρέφοντας μια ευθεία γύρω από ένα σημείο δημιουργούμε μια επιφάνεια, ενώ περιστρέφοντας μια κλειστή επιφάνεια γύρω από ένα σημείο μπορούμε να δημιουργήσουμε στερεά. Με αυτόν τον τρόπο δημιουργούμε τις επιφάνειες β, γ και δ. Συγκεκριμένα για τις επιφάνειες β και γ περιστρέφουμε το ευθύγραμμο τμήμα της παρακάτω εικόνας,



ενώ για την επιφάνεια δ

Η περιστροφή γίνεται για διακριτές τιμές της γωνίας. Χρησιμοποιούμε ως κβάντο τη 1^o και γωνία περιστροφής από 0 ώς 359^o. Ο σχεδιασμός του κάθε τμήματος κάθε επιφάνειας γίνεται χρησιμοποιώντας την εντολή GL_QUADS η οποία σχεδιάζει τετράπλευρα, δοθέντων των κορυφών.

5.6 Προβολή

Ο κόσμος μας, όπως και το μαθηματικό μοντέλο με το οποίο προσπαθούμε να τον αναπαραστήσουμε, είναι τρισδιάστατος. Παρ' όλ' αυτά, εμείς καλούμαστε να τον απεικονίσουμε σε συσκευές απεικόνισης (οθόνη, εκτυπωτής) οι οποίες είναι δισδιάστατες. Συνεπώς θα πρέπει με κάποιον τρόπο να προβάλουμε τα τρισδιάστατα αντικείμενα σε ένα επίπεδο προβολής. Οι προβολές διακρίνονται σε δύο βασικές κατηγορίες:

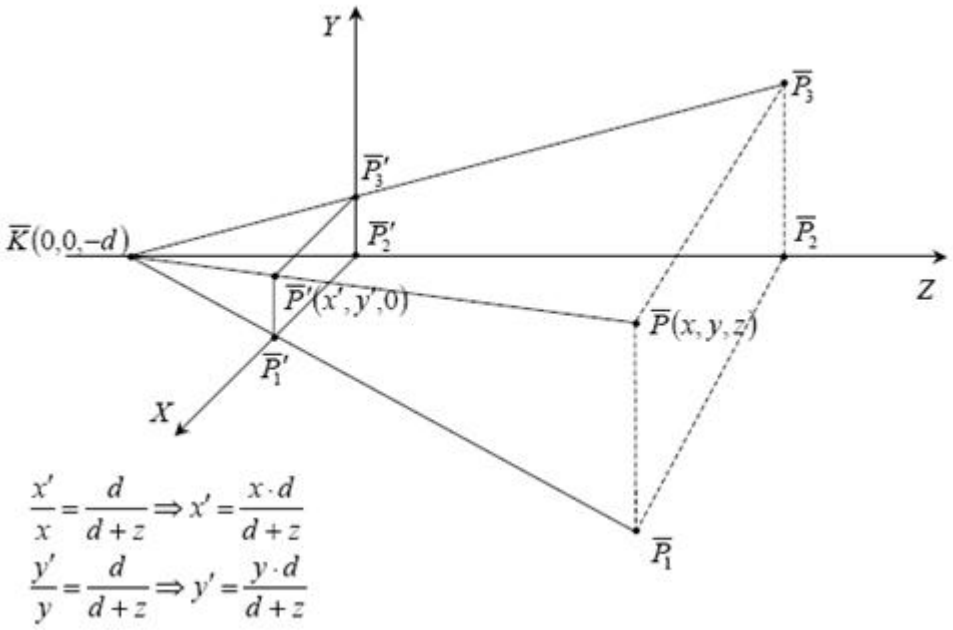
- Προοπτική προβολή. Σε αυτό το είδος προβολής, η απόσταση του κέντρου προβολής από το επίπεδο προβολής είναι πεπερασμένη
- Παράλληλη προβολή. Εδώ η απόσταση του κέντρου προβολής από το επίπεδο προβολής είναι άπειρη.

Οι δύο αυτές κατηγορίες προβολής έχουν κάποιες κοινές ιδιότητες. Μερικές από αυτές αναφέρονται παρακάτω:

- Οι ευθείες προβάλλονται σε ευθείες
- Οι αποστάσεις, εν γένει, αλλάζουν
- Ευθείες που στις τρεις διαστάσεις είναι παράλληλες (αλλά όχι παράλληλες με το επίπεδο προβολής) δεν προβάλλονται σε παράλληλες ευθείες αλλά σε ευθείες που τέμνονται σε κάποιο σημείο φυγής
- Η γωνία μεταξύ δύο ευθειών αλλάζει, εκτός αν το επίπεδο των ευθειών είναι παράλληλο προς το επίπεδο προβολής.

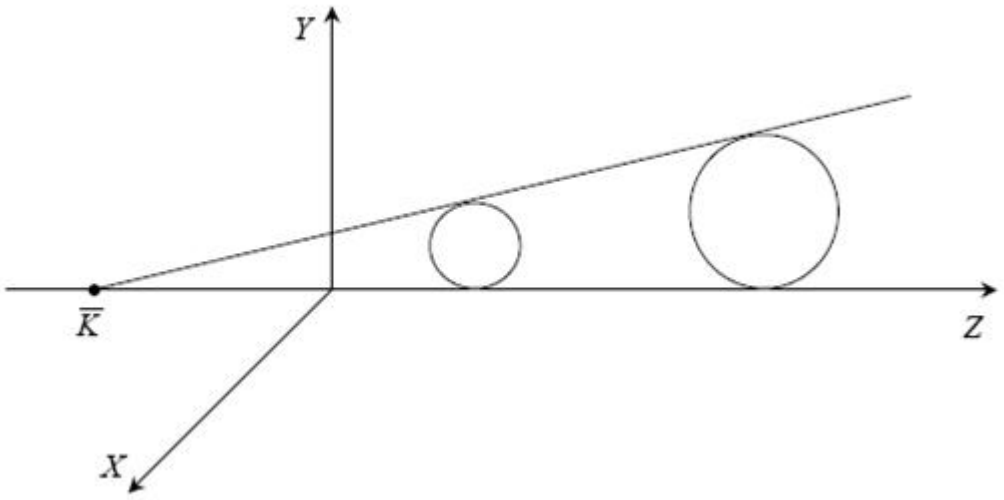
5.6.1 Προοπτική προβολή

Θεωρούμε ότι το κέντρο προβολής βρίσκεται πάνω στον αρνητικό z-άξονα, στο σημείο $\bar{K}(0, 0, -d)$ και ότι το $X\Upsilon$ είναι το επίπεδο προβολής όπως στο παρακάτω σχήμα.



$$\frac{x'}{x} = \frac{d}{d+z} \Rightarrow x' = \frac{x \cdot d}{d+z}$$
$$\frac{y'}{y} = \frac{d}{d+z} \Rightarrow y' = \frac{y \cdot d}{d+z}$$

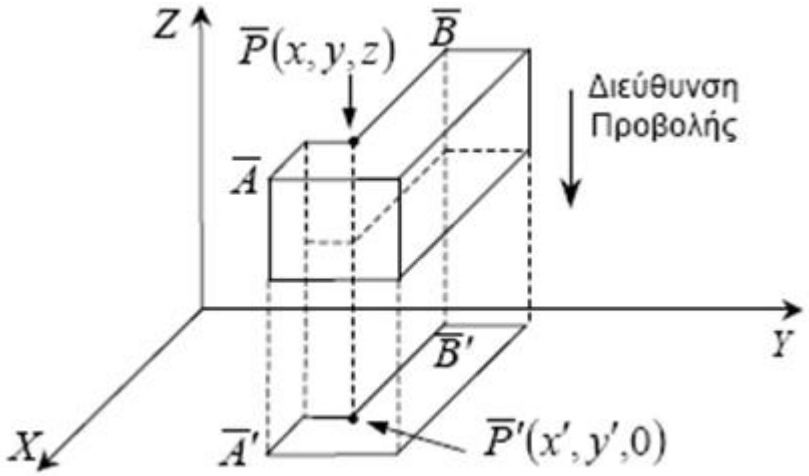
Για να υπολογίσουμε την προβολή ενός σημείου $\bar{P}(x, y, z)$ του τρισδιάστατου χώρου πάνω στο επίπεδο $\mathrm{X\Upsilon}$, φέρνουμε το ευθύγραμμο τμήμα $\bar{K}\bar{P}$. Το σημείο τομής αυτού του ευθύγραμμου τμήματος με το $\mathrm{X\Upsilon}$ επίπεδο είναι το ζητούμενο σημείο $\bar{P}'$. Οι συντεταγμένες x' και y' του σημείου προβολής βρίσκονται χρησιμοποιώντας την ομοιότητα των τριγώνων $\bar{K}\bar{P}_1'\bar{P}_2' \sim \bar{K}\bar{P}_1\bar{P}_2$ και $\bar{K}\bar{P}_2'\bar{P}_3' \sim \bar{K}\bar{P}_2\bar{P}_3$ Ένα σημαντικό χαρακτηριστικό της προοπτικής προβολής είναι ότι το μέγεθος του αντικειμένου είναι αντιστρόφως ανάλογο με την απόστασή του από το επίπεδο προβολής, όπως φαίνεται και στο παρακάτω σχήμα.



Παρ΄ όλ΄ αυτά η προοπτική προβολή προσομοιάζει καλά τον τρόπο με τον οποίο το ανθρώπινο μάτι αντιλαμβάνεται το βάθος. Αν τώρα θέλουμε να διατηρήσουμε τις αποστάσεις αναλλοίωτες, μπορούμε να χρησιμοποιήσουμε την παράλληλη προβολή.

5.6.2 Παράλληλη προβολή

Όπως προείπαμε, στην παράλληλη προβολή η απόσταση του κέντρου προβολής από το επίπεδο προβολής είναι άπειρη. Θα μπορούσαμε λοιπόν να θεωρήσουμε ότι η παράλληλη προβολή είναι μια υποπερίπτωση της προοπτικής προβολής, όταν το σημείο K βρίσκεται στο $(0, 0, -\infty)$. Σε αυτήν την περίπτωση, χρησιμοποιώντας τους ίδιους τύπους πάλι, υπολογίζουμε ότι η παράλληλη προβολή ενός σημείου $\bar{P}(x, y, z)$ πάνω στο επίπεδο XΥ είναι το σημείο $\bar{P}'(x, y, 0)$. Αυτή είναι η ορθογώνια προβολή, στην οποία η διεύθυνση προβολής είναι κάθετη στο επίπεδο XΥ.



Γενικότερα, ορθογώνια παράλληλη προβολή έχουμε όταν η δέσμη των ακτίνων προβολής είναι κάθετη σε κάποιο από τα επίπεδα ΧΥ, ΥΧ ή ΧΖ, δηλαδή όταν η δέσμη των ακτίνων έχει τη διεύθυνση κάποιου από τους άξονες συντεταγμένων. Υπάρχει και η αξονομετρική παράλληλη προβολή, στην οποία θα αναφερθούμε μόνο ονομαστικά, στην οποία το επίπεδο προβολής δεν είναι κάθετο σε κανέναν άξονα συντεταγμένων.

Στην εφαρμογή μας δίνεται η δυνατότητα να δούμε το περιβάλλον:

- Από μία κάμερα η οποία βρίσκεται πάνω στο ρομπότ (προοπτική προβολή)
- Από ένα σημείο ψηλά πάνω από το ρομπότ (κάτοψη)

6 Συμπεράσματα - Μελλοντικές εργασίες

Στη διπλωματική αυτή εργασία περιγράψαμε στοιχεία της εικονικής και της επαυξημένης πραγματικότητας και δώσαμε έμφαση στις εφαρμογές της στην τηλερομποτική. Πιο συγκεκριμένα, είδαμε εφαρμογές στις οποίες η χρήση τηλερομποτικής επιβοηθάει τον άνθρωπο στο να φέρει εις πέρας ορισμένες εργασίες.

Στη συνέχεια σχεδιάσαμε μερικά κομμάτια μιας γραφικής διεπαφής ανθρώπου - υπολογιστή η οποία χρησιμοποιείται στον τηλεπρογραμματισμό ενός mobile robot. Συγκεκριμένα δημιουργήσαμε ένα δισδιάστατο χάρτη γύρω από την θέση του ρομπότ, ο οποίος δημιουργείται παίρνοντας πληροφορίες από τις ενδείξεις των sonars του ρομπότ. Όμως, τουλάχιστον στο πρόγραμμα του εξομοιωτή του ρομπότ, οι ενδείξεις αυτές δεν είναι πολύ ακριβείς. Συγκεκριμένα, κάποιοι διαγώνιοι αισθητήρες, κατά άτακτα χρονικά διαστήματα, δίνουν τιμές μεγαλύτερες από τις πραγματικές. Έτσι ο χάρτης που δημιουργείται δεν είναι όσο ακριβής θα θέλαμε να είναι. Επίσης, λαμβάνοντας τη θέση και τον προσανατολισμό του ρομπότ κάθε στιγμή και γνωρίζοντας το χάρτη, μπορέσαμε να φτιάξουμε μια μεταβαλλόμενη εικόνα, η οποία μας δείχνει κάθε στιγμή τι θα βλέπαμε θεωρητικά από μια κάμερα η οποία βρίσκεται πάνω στο ρομπότ.

Σε μια περαιτέρω μελέτη θα μπορούσαν να χρησιμοποιηθούν και άλλα χαρακτηριστικά του περιβάλλοντος κίνησης του ρομπότ, ώστε να αποκτούμε μια ακριβέστερη εκτίμηση του χώρου αυτού. Εναλλακτικά θα μπορούσαμε να χρησιμοποιήσουμε αποστασιόμετρα laser με στόχο να μειώσουμε σε μεγάλο βαθμό τα σφάλματα και το θόρυβο των μετρήσεων. Έτσι η εκτίμηση της θέσης του ρομπότ θα ήταν πιο αξιόπιστη και οι χάρτες μας και η εικόνα θα αντικατόπτριζε περισσότερο τις πραγματικές εικόνες. Μια ακόμη εργασία που θα μπορούσε να γίνει, θα ήταν μια εφαρμογή η

οποία θα συγκρίνει την τρισδιάστατη εικόνα που δημιουργείται, με την εικόνα την οποία λαμβάνουμε από την κάμερα που βρίσκεται πάνω στο ρομπότ. Από τη σύγκριση αυτή θα μπορούσαμε να αντιληφθούμε αν υπάρχουν σφάλματα στις τιμές της θέσης που λαμβάνουμε. Επίσης θα μπορούσαμε να διαπιστώσουμε τυχόν αλλαγές που έχουν γίνει στο περιβάλλον κίνησης του ρομπότ, όπως για παράδειγμα διάφορα αντικείμενα που έχουν τοποθετηθεί στο χώρο ή ακόμη και δομικές αλλαγές. Τέλος, θα μπορούσε η κίνηση της κάμερας της εφαρμογής στην επιλογή "Δυνατότητα ελεύθερης κίνησης" να γίνεται μέσω ενός spaceball, αντί να γίνεται μέσω του πληκτρολογίου. Το spaceball είναι ένα μηχάνημα σαν mouse, με το οποίο μπορούμε να δώσουμε εντολές στον υπολογιστή. Ένας τύπος spaceball φαίνεται στην παρακάτω εικόνα.

7 Αναφορές

- Harald Friz, Design of an augmented reality user interface for an internet based telerobot using multiple monoscopic views
- Nathaniel I. Durlach, Anne S. Mavor, National Research Council (U.S.), Virtual Reality
- Clive Akass, The men who really invented the GUI
- Jeremy Reimer, A History of the GUI
- D. C. Engelbart, AUGMENTING HUMAN INTELLECT: A Conceptual Framework
- Costas S. Tzafestas, Virtual and Mixed Reality in Telerobotics: A Survey, Chapter in Industrial Robotics: Programming, Simulation and Applications.
- Κωνσταντίνος Σ. Τζαφέστας, Συμπληρωματικές σημειώσεις του μαθήματος: Ρομποτική ΙΙ: Ευφυή και επιδέξια ρομποτικά συστήματα
- Σπύρος Γ. Τζαφέστας, Ρομποτική: Ανάλυση - Έλεγχος - Σχεδιασμός - Προγραμματισμός - Αίσθηση
- Βλαχαβάς Ιωάννης, Κεφαλάς Πέτρος, Βασιλειάδης Νικόλαος, Κόκκορας Φώτης, Σακελλαρίου Ηλίας, Τεχνητη νοημοσύνη
- Σπύρος Γ. Τζαφέστας, Εικονική Πραγματικότητα. Απτικά Συστήματα και Εφαρμογές.

8 Παραρτηματα

8.1 Κώδικας C

8.1.1 FrmSonarMap

```
public partial class FrmSonarmap : Form
   {
       double[] mapx = new double[10000] ;
       double[] mapy = new double[10000] ;
       int mapsize=0;
       int writepose;
       int StatheriThesiX=100;
       int StatheriThesiY=100;
       int PosoVlepeiStotetragwno = 2500;
       double a;
       int i;
       double[] simeiox ={ 0, 0, 0, 0, 0, 0, 0, 0 };
       double[] simeioy ={ 0, 0, 0, 0, 0, 0, 0, 0 };

       private RobotPose _robotPose;
       public RobotPose robotPose
       {
           get { return _robotPose; }
           set { _robotPose = value; }
       }

       public FrmSonarmap()
       {
           InitializeComponent();
           _robotPose = new RobotPose(5, 583, 0);

       }

       private void Form1_Load(object sender, EventArgs e)
       {
       }

       private int [] _SonarAngles = { -90, -50, -30, -10, 10, 30, 50, 90 };
       private int[] _Sonars = new int[8];
       public void setSonars(int[] value)
       {
           for (int i = 0; i < _Sonars.Length; i++)
           {
               _Sonars[i] = value[i] / 55;
           }
           if (_Sonars[0] == 0)
```

```
        return;
    Refresh();
 }


private void Form1_Paint(object sender, PaintEventArgs e)
 {

        Pen pen = new Pen(Color.Red);
        e.Graphics.FillEllipse(pen.Brush, Convert.ToInt64(StatheriThesiX-5),
        Convert.ToInt64(StatheriThesiY-5), 10, 10);
        pen.Color = Color.DarkGray;
        e.Graphics.DrawEllipse(pen, Convert.ToInt64(StatheriThesiX - Math.Sqrt
        (PosoVlepeiStotetragwno)), Convert.ToInt64(StatheriThesiY - Math.Sqrt
        (PosoVlepeiStotetragwno)), Convert.ToInt64(Math.Sqrt(PosoVlepeiStotetragwno)
         * 2), Convert.ToInt64(Math.Sqrt(PosoVlepeiStotetragwno) * 2));
        pen.Color = Color.Black;
        for (i = 0; i < _Sonars.Length; i++)
            if (_Sonars[i]<2750/55)
        {
                writepose=mapsize;
                for(int j=1; j<mapsize;j++)
                    if (mapx[j] * mapx[j] + mapy[j] * mapy[j] == 0)
                    {
                        writepose = j;
                        break;
                    }

            simeiox[i] = _robotPose.X / 55 + _Sonars[i]  * Math.Cos(Math.PI / 180 *
             (_robotPose.Theta - _SonarAngles[i]));
            simeioy[i] = _robotPose.Y / 55 + _Sonars[i]  * Math.Sin(Math.PI / 180 *
             (_robotPose.Theta - _SonarAngles[i]));
            int count=0;
            for (int j = 1; j < mapsize; j++)
            {
                if ((mapx[j] - simeiox[i]) * (mapx[j] - simeiox[i]) + (mapy[j] -
                simeioy[i]) * (mapy[j] - simeioy[i]) < 30)
                    count++;
            }
            if (count < 5)
            {
                mapx[writepose] = simeiox[i];
                mapy[writepose] = simeioy[i];
                if (writepose == mapsize)
                    mapsize++;
            }
            Console.Write(mapsize);
        }
```

```
        for (i=1;i<mapsize;i++)
            if ((mapx[i] - _robotPose.X / 55) * (mapx[i] - _robotPose.X / 55) + (mapy[i]
            - _robotPose.Y / 55) * (mapy[i] - _robotPose.Y / 55) < PosoVlepeiStotetragwno)
                e.Graphics.DrawEllipse(pen, Convert.ToInt64(StatheriThesiX-_robotPose.X/55
                +mapx[i]), Convert.ToInt64(StatheriThesiY-_robotPose.Y/55+mapy[i]), 2, 2);
            else
            {
                mapx[i] = 0;
                mapy[i] = 0;
            }
    }
```

8.1.2 Fdm3dvision

```
public partial class Frm3dvision : Form
   {
       double navx,navy,navth; //metavlites gia to free navigation
       double navth2, navz; // metavlites gia to free navigation

       double oldrobotx = 0, oldroboty = 0, oldrobotth = 0, countb = 0,predx=0,predy=0,predth=0;
       double oldrobotx2 = 0, oldroboty2 = 0, oldrobotth2 = 0;

       Queue<double> bufferx = new Queue<double>();
       Queue<double> buffery = new Queue<double>();
       Queue<double> bufferth = new Queue<double>();



       int delay = 100;
       private RobotPose _robotPose;
       public RobotPose robotPose
       {
           get { return _robotPose; }
           set { _robotPose = value; }
       }

       private void InitLights()
       {
           float[] colorWhite ={ 1f, 1f, 1f, 1f };
           float[] lightAmbient = { 0.1f, 0.1f, 0.1f, 1.0f };
           float[] lightDiffuse = { 1.0f, 1.0f, 1.0f, 1.0f };
           float[] lightSpecular = { .1f, .1f, .1f, 1.0f };
           float[] lightPosition = { 3000.0f, 5000.0f, 800.0f, 1.0f };
           float[] lightPosition1 = { 3000.0f, 3000.0f, 800.0f, 1.0f };
           float[] lightPosition2 = { 5000.0f, 3000.0f, 800.0f, 1.0f };
           float[] lightPosition3 = { 9000.0f, 3000.0f, 800.0f, 1.0f };
           float[] lightPosition4 = { 1000.0f, 5000.0f, 800.0f, 1.0f };
           float[] lightPosition5 = { 1000.0f, 7000.0f, 800.0f, 1.0f };
```

```
float[] lightPosition6 = { 1380.0f, 12000.0f, 800.0f, 1.0f };

Gl.glLightfv(Gl.GL_LIGHT0, Gl.GL_AMBIENT, lightAmbient);
Gl.glLightfv(Gl.GL_LIGHT0, Gl.GL_DIFFUSE, lightDiffuse);
Gl.glLightfv(Gl.GL_LIGHT0, Gl.GL_SPECULAR, lightSpecular);
Gl.glLightfv(Gl.GL_LIGHT0, Gl.GL_POSITION, lightPosition);
Gl.glLightf(Gl.GL_LIGHT0, Gl.GL_SPOT_CUTOFF, 40f);

Gl.glLightfv(Gl.GL_LIGHT1, Gl.GL_AMBIENT, lightAmbient);
Gl.glLightfv(Gl.GL_LIGHT1, Gl.GL_DIFFUSE, lightDiffuse);
Gl.glLightfv(Gl.GL_LIGHT1, Gl.GL_SPECULAR, lightSpecular);
Gl.glLightfv(Gl.GL_LIGHT1, Gl.GL_POSITION, lightPosition1);
Gl.glLightf(Gl.GL_LIGHT1, Gl.GL_SPOT_CUTOFF, 40f);

Gl.glLightfv(Gl.GL_LIGHT2, Gl.GL_AMBIENT, lightAmbient);
Gl.glLightfv(Gl.GL_LIGHT2, Gl.GL_DIFFUSE, lightDiffuse);
Gl.glLightfv(Gl.GL_LIGHT2, Gl.GL_SPECULAR, lightSpecular);
Gl.glLightfv(Gl.GL_LIGHT2, Gl.GL_POSITION, lightPosition2);
Gl.glLightf(Gl.GL_LIGHT2, Gl.GL_SPOT_CUTOFF, 40f);

Gl.glLightfv(Gl.GL_LIGHT3, Gl.GL_AMBIENT, lightAmbient);
Gl.glLightfv(Gl.GL_LIGHT3, Gl.GL_DIFFUSE, lightDiffuse);
Gl.glLightfv(Gl.GL_LIGHT3, Gl.GL_SPECULAR, lightSpecular);
Gl.glLightfv(Gl.GL_LIGHT3, Gl.GL_POSITION, lightPosition3);
Gl.glLightf(Gl.GL_LIGHT3, Gl.GL_SPOT_CUTOFF, 40f);

Gl.glLightfv(Gl.GL_LIGHT4, Gl.GL_AMBIENT, lightAmbient);
Gl.glLightfv(Gl.GL_LIGHT4, Gl.GL_DIFFUSE, lightDiffuse);
Gl.glLightfv(Gl.GL_LIGHT4, Gl.GL_SPECULAR, lightSpecular);
Gl.glLightfv(Gl.GL_LIGHT4, Gl.GL_POSITION, lightPosition4);
Gl.glLightf(Gl.GL_LIGHT4, Gl.GL_SPOT_CUTOFF, 40f);

Gl.glLightfv(Gl.GL_LIGHT5, Gl.GL_AMBIENT, lightAmbient);
Gl.glLightfv(Gl.GL_LIGHT5, Gl.GL_DIFFUSE, lightDiffuse);
Gl.glLightfv(Gl.GL_LIGHT5, Gl.GL_SPECULAR, lightSpecular);
Gl.glLightfv(Gl.GL_LIGHT5, Gl.GL_POSITION, lightPosition5);
Gl.glLightf(Gl.GL_LIGHT5, Gl.GL_SPOT_CUTOFF, 40f);

Gl.glLightfv(Gl.GL_LIGHT6, Gl.GL_AMBIENT, lightAmbient);
Gl.glLightfv(Gl.GL_LIGHT6, Gl.GL_DIFFUSE, lightDiffuse);
Gl.glLightfv(Gl.GL_LIGHT6, Gl.GL_SPECULAR, lightSpecular);
Gl.glLightfv(Gl.GL_LIGHT6, Gl.GL_POSITION, lightPosition6);

Gl.glEnable(Gl.GL_LIGHTING);
Gl.glEnable(Gl.GL_LIGHT0);
Gl.glEnable(Gl.GL_LIGHT1);
Gl.glEnable(Gl.GL_LIGHT2);
Gl.glEnable(Gl.GL_LIGHT3);
Gl.glEnable(Gl.GL_LIGHT4);
Gl.glEnable(Gl.GL_LIGHT5);
Gl.glEnable(Gl.GL_LIGHT6);
```

```
Gl.glPushMatrix();
Gl.glTranslatef(lightPosition[0], lightPosition[1], 850f);
Gl.glScalef(100f, 100f, 50f);
Gl.glMaterialfv(Gl.GL_FRONT_AND_BACK, Gl.GL_DIFFUSE, colorWhite);
Gl.glMaterialfv(Gl.GL_FRONT_AND_BACK, Gl.GL_AMBIENT, colorWhite);
Glut.glutSolidSphere(1, 10, 10);
Gl.glPopMatrix();

Gl.glPushMatrix();
Gl.glTranslatef(lightPosition1[0], lightPosition1[1], 850f);
Gl.glScalef(100f, 100f, 50f);
Gl.glMaterialfv(Gl.GL_FRONT_AND_BACK, Gl.GL_DIFFUSE, colorWhite);
Gl.glMaterialfv(Gl.GL_FRONT_AND_BACK, Gl.GL_AMBIENT, colorWhite);
Glut.glutSolidSphere(1, 10, 10);
Gl.glPopMatrix();

Gl.glPushMatrix();
Gl.glTranslatef(lightPosition2[0], lightPosition2[1], 850f);
Gl.glScalef(100f, 100f, 50f);
Gl.glMaterialfv(Gl.GL_FRONT_AND_BACK, Gl.GL_DIFFUSE, colorWhite);
Gl.glMaterialfv(Gl.GL_FRONT_AND_BACK, Gl.GL_AMBIENT, colorWhite);
Glut.glutSolidSphere(1, 10, 10);
Gl.glPopMatrix();

Gl.glPushMatrix();
Gl.glTranslatef(lightPosition3[0], lightPosition3[1], 850f);
Gl.glScalef(100f, 100f, 50f);
Gl.glMaterialfv(Gl.GL_FRONT_AND_BACK, Gl.GL_DIFFUSE, colorWhite);
Gl.glMaterialfv(Gl.GL_FRONT_AND_BACK, Gl.GL_AMBIENT, colorWhite);
Glut.glutSolidSphere(1, 10, 10);
Gl.glPopMatrix();

Gl.glPushMatrix();
Gl.glTranslatef(lightPosition4[0], lightPosition4[1], 850f);
Gl.glScalef(100f, 100f, 50f);
Gl.glMaterialfv(Gl.GL_FRONT_AND_BACK, Gl.GL_DIFFUSE, colorWhite);
Gl.glMaterialfv(Gl.GL_FRONT_AND_BACK, Gl.GL_AMBIENT, colorWhite);
Glut.glutSolidSphere(1, 10, 10);
Gl.glPopMatrix();

Gl.glPushMatrix();
Gl.glTranslatef(lightPosition5[0], lightPosition5[1], 850f);
Gl.glScalef(100f, 100f, 50f);
Gl.glMaterialfv(Gl.GL_FRONT_AND_BACK, Gl.GL_DIFFUSE, colorWhite);
Gl.glMaterialfv(Gl.GL_FRONT_AND_BACK, Gl.GL_AMBIENT, colorWhite);
Glut.glutSolidSphere(1, 10, 10);
Gl.glPopMatrix();

Gl.glPushMatrix();
Gl.glTranslatef(lightPosition6[0], lightPosition6[1], 850f);
Gl.glScalef(100f, 100f, 50f);
Gl.glMaterialfv(Gl.GL_FRONT_AND_BACK, Gl.GL_DIFFUSE, colorWhite);
```

```
   Gl.glMaterialfv(Gl.GL_FRONT_AND_BACK, Gl.GL_AMBIENT, colorWhite);
   Glut.glutSolidSphere(1, 10, 10);
   Gl.glPopMatrix();

}



public Frm3dvision()
{
    _robotPose = new RobotPose(5, 583, 0);

    Glut.glutInit();
    Gl.glEnable(Gl.GL_DEPTH_TEST);
    Glut.glutInitDisplayMode(Glut.GLUT_SINGLE | Glut.GLUT_RGB | Glut.GLUT_DEPTH);
    InitializeComponent();
    simpleOpenGlControl1.InitializeContexts();
    Gl.glClearColor(0.0f, 0.0f, 0.0f, 0.0f);
    Gl.glClearDepth(1.0);
    Gl.glClear(Gl.GL_COLOR_BUFFER_BIT | Gl.GL_DEPTH_BUFFER_BIT);
    Gl.glMatrixMode(Gl.GL_PROJECTION);
    Gl.glLoadIdentity();
    Glu.gluPerspective(90, 1, 1, 100000);
    Glut.glutSpaceballMotionFunc(checkspaceball);
    InitLights();
}
void checkspaceball(int x,int y,int z)
{
    label3.Text = Convert.ToString(x);
    label2.Text = Convert.ToString(y);
    label1.Text = Convert.ToString("");
}
private void Frm3dvision_Load(object sender, EventArgs e)
{
    label1.Visible = false;
    label2.Visible = false;
    label3.Visible = false;
    label15.Visible = false;
    label16.Visible = false;
    label17.Visible = false;
    trackBar1.Visible = false;
    button1.Visible = false;
    button2.Visible = false;
    button3.Visible = false;
    button4.Visible = false;

}

public void CreateWall(float x1, float y1, float x2, float y2, float height, float R, float G, float B)
{
    float[] colorBlueW ={ 0f, 0f, 1f, 1f };
    float[] colorGreyW ={ 0.3f, 1f, 0.3f, 1f };
```

```
    float[] colorDoorW ={ 1f, .3f, .5f, 1f };
    float[] matSpecular = { .1f, .1f, .1f, 1.0f };
    float[] matAmbient = { .05f, .05f, .05f, 1.0f };

    float x, y, len;

    int ii,jj;
    x = x2 - x1;
    y = y2 - y1;

    len = Convert.ToInt64(Math.Sqrt(x * 2 + y * 2));

       if ((x1 > 950 - 900 && x2 < 1460 - 900) || (x1 > 950 - 900 && x2 < 2560 - 900 &&
       y1 > 6950 - 5200 && y2 < 7660 - 5200))
    {
        Gl.glMaterialfv(Gl.GL_FRONT_AND_BACK, Gl.GL_DIFFUSE, colorGreyW);
        Gl.glMaterialfv(Gl.GL_FRONT_AND_BACK, Gl.GL_SPECULAR, matSpecular);
        Gl.glMaterialfv(Gl.GL_FRONT_AND_BACK, Gl.GL_AMBIENT, colorGreyW);
    }
    else if ((y1 == 2250 && y2 == 2250) || (y1 == 5150 && y2 == 5150) || (x1 == 3350 - 900 &&
    x2 == 3350 - 900))
    {
        Gl.glMaterialfv(Gl.GL_FRONT_AND_BACK, Gl.GL_DIFFUSE, colorDoorW);
        Gl.glMaterialfv(Gl.GL_FRONT_AND_BACK, Gl.GL_SPECULAR, matSpecular);
        Gl.glMaterialfv(Gl.GL_FRONT_AND_BACK, Gl.GL_AMBIENT, colorDoorW);
    }
    else
    {
        Gl.glMaterialfv(Gl.GL_FRONT_AND_BACK, Gl.GL_DIFFUSE, colorBlueW);
        Gl.glMaterialfv(Gl.GL_FRONT_AND_BACK, Gl.GL_SPECULAR, matSpecular);
        Gl.glMaterialfv(Gl.GL_FRONT_AND_BACK, Gl.GL_AMBIENT, colorBlueW);
    }
        Gl.glBegin(Gl.GL_QUADS);
        Gl.glColor3f(R, G, B);
        Gl.glNormal3f(0f,0f,1f);
        Gl.glVertex3f(x1, y1, 0.0f);
        Gl.glVertex3f(x2, y2, 0.0f);
        Gl.glVertex3f(x2, y2, height);
        Gl.glVertex3f(x1, y1, height);
        Gl.glEnd();
}



string b;
string[] c = new string[10];
float DimWidth;
float DimHeight;


private void simpleOpenGlControl1_Paint(object sender, PaintEventArgs e)
{
```

```
    bufferx.Enqueue(_robotPose.X);
    buffery.Enqueue(_robotPose.Y);
    bufferth.Enqueue(_robotPose.Theta);



if (checkBox1.Checked == true)
{
    label5.Visible = true;
    label6.Visible = true;
    label7.Visible = true;
    label8.Visible = true;
    label9.Visible = true;
    label10.Visible = true;
    label11.Visible = true;
    label12.Visible = true;
    label13.Visible = true;
    label14.Visible = true;
}
else
{
    label5.Visible = false;
    label6.Visible = false;
    label7.Visible = false;
    label8.Visible = false;
    label9.Visible = false;
    label10.Visible = false;
    label11.Visible = false;
    label12.Visible = false;
    label13.Visible = false;
    label14.Visible = false;
}

Glut.glutSpaceballMotionFunc(checkspaceball);

Gl.glClearColor(0.0f, 0.0f, 0.0f, 0.0f);
Gl.glClearDepth(1.0);
Gl.glClear(Gl.GL_COLOR_BUFFER_BIT | Gl.GL_DEPTH_BUFFER_BIT);
Gl.glEnable(Gl.GL_DEPTH_TEST);
Gl.glLightModeli(Gl.GL_LIGHT_MODEL_TWO_SIDE, Gl.GL_TRUE);

InitLights();

float[] colorBlueP ={ 0f, 0f, 1f, 1f };
float[] colorRedP ={ 1f, 0f, 0f, 1f };
float[] colorRed2P ={ 0.5f, 0f, 0f, 1f };
float[] colorRobP ={ 0f, 1f, 1f, 1f };
float[] colorCeilP ={ 0.7f, 0.7f, 0.7f, 1f };
float[] colorDarkP ={ 0.1f, 0.1f, 0.1f, 1f };
```

```
//-------------------------------------------------------------------------------
int radi, iii, jjj;
double how1 = 100; //height of wheel              (x2)
double how2 = 50; //inner height of wheel         (x2)
double wow = 30; //width of wheel               (x2)
int dis1 = 150;  //apostasi apo rodes mikos  (x2)
int dis2 = 100;  //apostasi apo rodes platos (x2)
float[] colorwheelsR ={ 1f, 1f, 0f, 1f };
float[] matSpecular = { .1f, .1f, .1f, 1.0f };
float[] matAmbient = { .05f, .05f, .05f, 1.0f };
Gl.glMaterialfv(Gl.GL_FRONT_AND_BACK, Gl.GL_DIFFUSE, colorwheelsR);
Gl.glMaterialfv(Gl.GL_FRONT_AND_BACK, Gl.GL_SPECULAR, matSpecular);
Gl.glMaterialfv(Gl.GL_FRONT_AND_BACK, Gl.GL_AMBIENT, colorwheelsR);


//**********************start I****************
if (countb > delay)
{
    oldrobotx = bufferx.Dequeue();
    oldroboty = buffery.Dequeue();
    oldrobotth = bufferth.Dequeue();


    for (iii = -dis1; iii <= dis1; iii = iii + 2 * dis1)
        for (jjj = -dis2; jjj <= dis2; jjj = jjj + 2 * dis2)
        {
            Gl.glPushMatrix();
            Gl.glTranslatef((float)(oldrobotx + iii), (float)(oldroboty + jjj), 0f);
            Gl.glRotatef((float)(oldrobotth), 0f, 0f, 1f);
            Gl.glBegin(Gl.GL_QUADS);
            for (radi = 0; radi <= 360; radi++)
            {
                Gl.glColor3f(1f, 1f, 0f);
                Gl.glNormal3f((float)(Math.Cos(Math.PI / 180 * (radi))), 0f,
                (float)(Math.Sin(Math.PI / 180 * (radi))));
                Gl.glVertex3f((float)(how1 * (Math.Cos(Math.PI / 180 * radi))),
                (float)(-wow), (float)(how1 + how1 * (Math.Sin(Math.PI / 180 * radi))));
                Gl.glVertex3f((float)(how1 * (Math.Cos(Math.PI / 180 * radi))),
                (float)(+wow), (float)(how1 + how1 * (Math.Sin(Math.PI / 180 * radi))));
                Gl.glNormal3f((float)(Math.Cos(Math.PI / 180 * (radi - 1))), 0f,
                (float)(Math.Sin(Math.PI / 180 * (radi - 1))));
                Gl.glVertex3f((float)(how1 * (Math.Cos(Math.PI / 180 * (radi + 1)))),
                (float)(+wow), (float)(how1 + how1 * (Math.Sin(Math.PI / 180 * (radi + 1)))));
                Gl.glVertex3f((float)(how1 * (Math.Cos(Math.PI / 180 * (radi + 1)))),
                (float)(-wow), (float)(how1 + how1 * (Math.Sin(Math.PI / 180 * (radi + 1)))));
            }
            Gl.glEnd();

            Gl.glBegin(Gl.GL_QUADS);
            for (radi = 0; radi <= 360; radi++)
            {
                Gl.glColor3f(1f, 1f, 0f);
```

```
            Gl.glNormal3f((float)(Math.Cos(Math.PI / 180 * (radi))), 0f,
            (float)(Math.Sin(Math.PI / 180 * (radi))));
            Gl.glVertex3f((float)(how2 * (Math.Cos(Math.PI / 180 * radi))),
            (float)(-wow), (float)(how1 + how2 * (Math.Sin(Math.PI / 180 * radi))));
            Gl.glVertex3f((float)(how2 * (Math.Cos(Math.PI / 180 * radi))),
            (float)(+wow), (float)(how1 + how2 * (Math.Sin(Math.PI / 180 * radi))));
            Gl.glNormal3f((float)(Math.Cos(Math.PI / 180 * (radi - 1))), 0f,
            (float)(Math.Sin(Math.PI / 180 * (radi - 1))));
            Gl.glVertex3f((float)(how2 * (Math.Cos(Math.PI / 180 * (radi + 1)))),
            (float)(+wow), (float)(how1 + how2 * (Math.Sin(Math.PI / 180 * (radi + 1)))));
            Gl.glVertex3f((float)(how2 * (Math.Cos(Math.PI / 180 * (radi + 1)))),
            (float)(-wow), (float)(how1 + how2 * (Math.Sin(Math.PI / 180 * (radi + 1)))));
        }
        Gl.glEnd();

        Gl.glBegin(Gl.GL_QUADS);
        for (radi = 0; radi <= 360; radi++)
        {
            Gl.glColor3f(1f, 1f, 0f);
            Gl.glNormal3f(0f, 1f, 0f);
            Gl.glVertex3f((float)(how1 * (Math.Cos(Math.PI / 180 * radi))), (float)(wow),
            (float)(how1 + how1 * (Math.Sin(Math.PI / 180 * radi))));
            Gl.glVertex3f((float)(how2 * (Math.Cos(Math.PI / 180 * radi))), (float)(wow),
            (float)(how1 + how2 * (Math.Sin(Math.PI / 180 * radi))));
            Gl.glVertex3f((float)(how2 * (Math.Cos(Math.PI / 180 * (radi + 1)))),
            (float)(wow), (float)(how1 + how2 * (Math.Sin(Math.PI / 180 * (radi + 1)))));
            Gl.glVertex3f((float)(how1 * (Math.Cos(Math.PI / 180 * (radi + 1)))),
            (float)(wow), (float)(how1 + how1 * (Math.Sin(Math.PI / 180 * (radi + 1)))));
        }
        Gl.glEnd();

        Gl.glBegin(Gl.GL_QUADS);
        for (radi = 0; radi <= 360; radi++)
        {
            Gl.glColor3f(1f, 1f, 0f);
            Gl.glNormal3f(0f, -1f, 0f);
            Gl.glVertex3f((float)(how1 * (Math.Cos(Math.PI / 180 * radi))), (float)(-wow),
            (float)(how1 + how1 * (Math.Sin(Math.PI / 180 * radi))));
            Gl.glVertex3f((float)(how2 * (Math.Cos(Math.PI / 180 * radi))), (float)(-wow),
            (float)(how1 + how2 * (Math.Sin(Math.PI / 180 * radi))));
            Gl.glVertex3f((float)(how2 * (Math.Cos(Math.PI / 180 * (radi + 1)))),
            (float)(-wow), (float)(how1 + how2 * (Math.Sin(Math.PI / 180 * (radi + 1)))));
            Gl.glVertex3f((float)(how1 * (Math.Cos(Math.PI / 180 * (radi + 1)))),
            (float)(-wow), (float)(how1 + how1 * (Math.Sin(Math.PI / 180 * (radi + 1)))));
        }
        Gl.glEnd();
        Gl.glPopMatrix();
    }

Gl.glPushMatrix();
Gl.glTranslatef((float)(oldrobotx), (float)(oldroboty), 0f);
Gl.glRotatef((float)(oldrobotth), 0f, 0f, 1f);
```

```
    Gl.glBegin(Gl.GL_QUADS);
    Gl.glColor3f(1f, 1f, 0f);
    Gl.glNormal3f(0f, 0f, 1f);
    Gl.glVertex3f((float)(-dis1 - how1), (float)(-dis2 - wow), (float)(2 * how1));
    Gl.glVertex3f((float)(-dis1 - how1), (float)(+dis2 + wow), (float)(2 * how1));
    Gl.glVertex3f((float)(+dis1 + how1), (float)(+dis2 + wow), (float)(2 * how1));
    Gl.glVertex3f((float)(+dis1 + how1), (float)(-dis2 - wow), (float)(2 * how1));
    Gl.glEnd();
    Gl.glPopMatrix();


    //**********************stop I***************
}
//************************************************start*************************************************************
if (checkBox2.Checked == true)
{
    float[] colorwheelsZ ={ 0f, 1f, 0f, 1f };
    Gl.glPushMatrix();
    Gl.glMaterialfv(Gl.GL_FRONT_AND_BACK, Gl.GL_DIFFUSE, colorwheelsZ);
    Gl.glMaterialfv(Gl.GL_FRONT_AND_BACK, Gl.GL_SPECULAR, matSpecular);
    Gl.glMaterialfv(Gl.GL_FRONT_AND_BACK, Gl.GL_AMBIENT, colorwheelsZ);
    for (iii = -dis1; iii <= dis1; iii = iii + 2 * dis1)
        for (jjj = -dis2; jjj <= dis2; jjj = jjj + 2 * dis2)
        {
            Gl.glPushMatrix();
            Gl.glTranslatef((float)(_robotPose.X + iii), (float)(_robotPose.Y + jjj), 0f);
            Gl.glRotatef((float)(_robotPose.Theta), 0f, 0f, 1f);
            Gl.glBegin(Gl.GL_QUADS);
            for (radi = 0; radi <= 360; radi++)
            {
                Gl.glColor3f(1f, 1f, 0f);
                Gl.glNormal3f((float)(Math.Cos(Math.PI / 180 * (radi))), 0f,
                (float)(Math.Sin(Math.PI / 180 * (radi))));
                Gl.glVertex3f((float)(how1 * (Math.Cos(Math.PI / 180 * radi))),
                (float)(-wow), (float)(how1 + how1 * (Math.Sin(Math.PI / 180 * radi))));
                Gl.glVertex3f((float)(how1 * (Math.Cos(Math.PI / 180 * radi))),
                (float)(+wow), (float)(how1 + how1 * (Math.Sin(Math.PI / 180 * radi))));
                Gl.glNormal3f((float)(Math.Cos(Math.PI / 180 * (radi - 1))), 0f,
                (float)(Math.Sin(Math.PI / 180 * (radi - 1))));
                Gl.glVertex3f((float)(how1 * (Math.Cos(Math.PI / 180 * (radi + 1)))),
                (float)(+wow), (float)(how1 + how1 * (Math.Sin(Math.PI / 180 * (radi + 1)))));
                Gl.glVertex3f((float)(how1 * (Math.Cos(Math.PI / 180 * (radi + 1)))),
                (float)(-wow), (float)(how1 + how1 * (Math.Sin(Math.PI / 180 * (radi + 1)))));
            }
            Gl.glEnd();

            Gl.glBegin(Gl.GL_QUADS);
            for (radi = 0; radi <= 360; radi++)
            {
                Gl.glColor3f(1f, 1f, 0f);
                Gl.glNormal3f((float)(Math.Cos(Math.PI / 180 * (radi))), 0f,
                (float)(Math.Sin(Math.PI / 180 * (radi))));
```

```
            Gl.glVertex3f((float)(how2 * (Math.Cos(Math.PI / 180 * radi))),
            (float)(-wow), (float)(how1 + how2 * (Math.Sin(Math.PI / 180 * radi))));
            Gl.glVertex3f((float)(how2 * (Math.Cos(Math.PI / 180 * radi))),
            (float)(+wow), (float)(how1 + how2 * (Math.Sin(Math.PI / 180 * radi))));
            Gl.glNormal3f((float)(Math.Cos(Math.PI / 180 * (radi - 1))), 0f,
            (float)(Math.Sin(Math.PI / 180 * (radi - 1))));
            Gl.glVertex3f((float)(how2 * (Math.Cos(Math.PI / 180 * (radi + 1)))),
            (float)(+wow), (float)(how1 + how2 * (Math.Sin(Math.PI / 180 * (radi + 1)))));
            Gl.glVertex3f((float)(how2 * (Math.Cos(Math.PI / 180 * (radi + 1)))),
            (float)(-wow), (float)(how1 + how2 * (Math.Sin(Math.PI / 180 * (radi + 1)))));
        }
        Gl.glEnd();

        Gl.glBegin(Gl.GL_QUADS);
        for (radi = 0; radi <= 360; radi++)
        {
            Gl.glColor3f(1f, 1f, 0f);
            Gl.glNormal3f(0f, 1f, 0f);
            Gl.glVertex3f((float)(how1 * (Math.Cos(Math.PI / 180 * radi))), (float)(wow),
            (float)(how1 + how1 * (Math.Sin(Math.PI / 180 * radi))));
            Gl.glVertex3f((float)(how2 * (Math.Cos(Math.PI / 180 * radi))), (float)(wow),
            (float)(how1 + how2 * (Math.Sin(Math.PI / 180 * radi))));
            Gl.glVertex3f((float)(how2 * (Math.Cos(Math.PI / 180 * (radi + 1)))),
            (float)(wow), (float)(how1 + how2 * (Math.Sin(Math.PI / 180 * (radi + 1)))));
            Gl.glVertex3f((float)(how1 * (Math.Cos(Math.PI / 180 * (radi + 1)))),
            (float)(wow), (float)(how1 + how1 * (Math.Sin(Math.PI / 180 * (radi + 1)))));
        }
        Gl.glEnd();

        Gl.glBegin(Gl.GL_QUADS);
        for (radi = 0; radi <= 360; radi++)
        {
            Gl.glColor3f(1f, 1f, 0f);
            Gl.glNormal3f(0f, -1f, 0f);
            Gl.glVertex3f((float)(how1 * (Math.Cos(Math.PI / 180 * radi))), (float)(-wow),
            (float)(how1 + how1 * (Math.Sin(Math.PI / 180 * radi))));
            Gl.glVertex3f((float)(how2 * (Math.Cos(Math.PI / 180 * radi))), (float)(-wow),
            (float)(how1 + how2 * (Math.Sin(Math.PI / 180 * radi))));
            Gl.glVertex3f((float)(how2 * (Math.Cos(Math.PI / 180 * (radi + 1)))),
            (float)(-wow), (float)(how1 + how2 * (Math.Sin(Math.PI / 180 * (radi + 1)))));
            Gl.glVertex3f((float)(how1 * (Math.Cos(Math.PI / 180 * (radi + 1)))),
            (float)(-wow), (float)(how1 + how1 * (Math.Sin(Math.PI / 180 * (radi + 1)))));
        }
        Gl.glEnd();


        Gl.glPopMatrix();
    }

Gl.glPushMatrix();
Gl.glTranslatef((float)(_robotPose.X), (float)(_robotPose.Y), 0f);
Gl.glRotatef((float)(_robotPose.Theta), 0f, 0f, 1f);
```

```
        Gl.glBegin(Gl.GL_QUADS);
        Gl.glColor3f(1f, 1f, 0f);
        Gl.glNormal3f(0f, 0f, 1f);
        Gl.glVertex3f((float)(-dis1 - how1), (float)(-dis2 - wow), (float)(2 * how1));
        Gl.glVertex3f((float)(-dis1 - how1), (float)(+dis2 + wow), (float)(2 * how1));
        Gl.glVertex3f((float)(+dis1 + how1), (float)(+dis2 + wow), (float)(2 * how1));
        Gl.glVertex3f((float)(+dis1 + how1), (float)(-dis2 - wow), (float)(2 * how1));
        Gl.glEnd();
        Gl.glPopMatrix();
        Gl.glPopMatrix();
    }
    //*************************************************stop*************************************************



    StreamReader sr = new StreamReader(@"C:\Users\J'anos\diplomatiki\progr5\lab_ntua.wld", Encoding.Default);
    for (int i = 0; i < 2; i++)
    {
        b = sr.ReadLine();
    }

    b = sr.ReadLine();
    c = b.Split(' ');
    DimWidth = Convert.ToInt64(c[1]);
    b = sr.ReadLine();
    c = b.Split(' ');
    DimHeight = Convert.ToInt64(c[1]);
    for (int i = 4; i < 6; i++)
    {
        b = sr.ReadLine();
    }
    while (!sr.EndOfStream)
    {
        sr.ReadLine();
        b = sr.ReadLine();
        c = b.Split(' ');
        CreateWall(Convert.ToInt64(c[0]) - 900, Convert.ToInt64(c[1]) - 5200,
        Convert.ToInt64(c[2]) - 900, Convert.ToInt64(c[3]) - 5200, 900.0f, 0f, 1f, 0f);
    }

    //-----------------------------------------------------------------------------
    float[] lightAmbientP = { 0.2f, 0.2f, 0.2f, 1.0f };
    float[] lightDiffuseP = { 1.0f, 1.0f, 1.0f, 1.0f };
    float[] lightSpecularP = { .5f, .5f, .5f, 1.0f };

Gl.glShadeModel(Gl.GL_SMOOTH);

    // skakiera
    Gl.glNormal3f(0f, 0f, 1f);
    for (int i = 0; i < (int)(DimWidth / 1000); i++)
        for (int j = 0; j < (int)(DimHeight / 1000); j++)
```

```
    {
        Gl.glPushMatrix();
        Gl.glTranslatef(Convert.ToInt64(500 + 1000 * i), Convert.ToInt64(500 + 1000 * j), 0.0f);
        Gl.glScalef(Convert.ToInt64(1000), Convert.ToInt64(1000), 1f);

        double k;
        k = i + j;

        if ((Math.Ceiling(k / 2)) == (Math.Floor(k / 2)))
        {
            Gl.glMaterialfv(Gl.GL_FRONT_AND_BACK, Gl.GL_DIFFUSE, colorRed2P);
            Gl.glMaterialfv(Gl.GL_FRONT_AND_BACK, Gl.GL_AMBIENT, colorRed2P);
        }
        else
        {
            Gl.glMaterialfv(Gl.GL_FRONT_AND_BACK, Gl.GL_DIFFUSE, colorRedP);
            Gl.glMaterialfv(Gl.GL_FRONT_AND_BACK, Gl.GL_AMBIENT, colorRedP);
        }

        Gl.glMaterialfv(Gl.GL_FRONT_AND_BACK, Gl.GL_SPECULAR, lightSpecularP);
        Glut.glutSolidCube(1);
        Gl.glPopMatrix();
    }
//skakiera

if (radioButton1.Checked)
{
    Gl.glPushMatrix();
    Gl.glTranslatef(Convert.ToInt64(DimWidth / 2), Convert.ToInt64(DimHeight / 2), 900.0f);
    Gl.glScalef(Convert.ToInt64(DimWidth), Convert.ToInt64(DimHeight), 1f);

    Gl.glMaterialfv(Gl.GL_FRONT_AND_BACK, Gl.GL_DIFFUSE, colorCeilP);
    Gl.glMaterialfv(Gl.GL_FRONT_AND_BACK, Gl.GL_AMBIENT, colorCeilP);
    Glut.glutSolidCube(1);
    Gl.glPopMatrix();

    Gl.glMatrixMode(Gl.GL_MODELVIEW);
    Gl.glEnable(Gl.GL_RESCALE_NORMAL);
    Gl.glLoadIdentity();
    if (checkBox1.Checked == false)
        Glu.gluLookAt(oldrobotx, oldroboty, 300, oldrobotx + 1 * Math.Sin(Math.PI / 180
        * (90 - oldrobotth)), oldroboty + 1 * Math.Cos(Math.PI / 180 * (90 - oldrobotth)),
        299.99, Math.Sin(Math.PI / 180 * (90 - oldrobotth)), Math.Cos(Math.PI / 180
        * (90 - oldrobotth)), 0);
    else
        Glu.gluLookAt(navx, navy, navz, navx + 1 * Math.Sin(Math.PI / 180 * (90 - navth))
        * Math.Cos(Math.PI / 180 * navth2), navy + 1 * Math.Cos(Math.PI / 180 * (90 - navth))
        * Math.Cos(Math.PI / 180 * navth2), navz - 0.01 + Math.Sin(Math.PI / 180 * navth2),
        Math.Sin(Math.PI / 180 * (90 - navth)) * Math.Cos(Math.PI / 180 * navth2),
        Math.Cos(Math.PI / 180 * (90 - navth)) * Math.Cos(Math.PI / 180 * navth2),
        Math.Sin(Math.PI / 180 * navth2));
}
```

```
        else if (radioButton2.Checked)
        {
            Gl.glMatrixMode(Gl.GL_MODELVIEW);
            Gl.glEnable(Gl.GL_RESCALE_NORMAL);
            Gl.glLoadIdentity();
            if (checkBox1.Checked == false)
                Glu.gluLookAt(oldrobotx, oldroboty, trackBar2.Value, oldrobotx, oldroboty + 10, 0, 0, 1, 0);
            else
                Glu.gluLookAt(navx, navy, trackBar2.Value, navx, navy + 10, 0, 0, 1, 0);
        }
        if (countb < delay+2)
        {
            countb++;
        }

    }


    private void simpleOpenGlControl1_KeyDown(object sender, KeyEventArgs e)
    {
        if (e.KeyCode == Keys.A)
            navx=navx-50;
        else if (e.KeyCode == Keys.D)
            navx=navx+50;
        else if (e.KeyCode == Keys.W)
            navy = navy + 50;
        else if (e.KeyCode == Keys.S)
            navy = navy - 50;
        else if (e.KeyCode == Keys.Q)
            navth = navth + 5;
        else if (e.KeyCode == Keys.E)
            navth = navth - 5;
        else if (e.KeyCode == Keys.I)
            navz = navz + 10;
        else if (e.KeyCode == Keys.K)
            navz = navz - 10;
        else if (e.KeyCode == Keys.O)
            navth2 = navth2 + 5;
        else if (e.KeyCode == Keys.L)
            navth2 = navth2 - 5;

    }

    private void checkBox1_CheckedChanged(object sender, EventArgs e)
    {
        navx = _robotPose.X;
        navy = _robotPose.Y;
        navth = _robotPose.Theta;
        navz = 300;
        navth2 = 0;
    }
}
```