ΑΝΩΤΑΤΟ ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΚΡΗΤΗΣ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΝΙΚΗΣ

ΑΝΑΠΤΥΞΗ ΕΚΠΑΙΔΕΥΤΙΚΟΥ ΛΟΓΙΣΜΙΚΟΥ ΓΙΑ ΠΡΟΣΟΜΟΙΩΣΗ ΚΑΙ ΕΛΕΓΧΟ ΠΡΑΓΜΑΤΙΚΟΥ ΡΟΜΠΟΤΙΚΟΥ ΒΡΑΧΙΟΝΑ ΤΕΣΣΑΡΩΝ ΒΑΘΜΩΝ ΕΛΕΥΘΕΡΙΑΣ

Τριμελής Επιτροπή:

Ε. Δοϊτσίδης	Εργ. Συνεργάτης (Επιβλέπων)
Γ. Φουσκιτάκης	Επίκουρος Καθηγητής
Ν. Φραγκιαδάκης	Καθηγητής Εφαρμογών

υπό

Δημήτριου Θ. Τσόντου

Χανιά, 2009

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΕΦΑΛΑΙΟ 1
ΕΙΣΑΓΩΓΗ

1.1 ΕΙΣΑΓΩΓΗ

Η κατανόηση βασικών εννοιών που σχετίζονται με τους αυτοματισμούς, την μηχανολογία, την ηλεκτρονική και τη θεωρία της ρομποτικής είναι βασικό στοιχείο της εκπαίδευσης ενός τεχνολόγου μηχανικού, ανεξαρτήτως της ειδικότητάς του. Ιδανική πλατφόρμα για την κατανόηση των παραπάνω εννοιών αποτελούν οι ρομποτικές συσκευές.

Παρά την τεχνολογική εξέλιξη και τη μείωση του κόστους τους, αυτό παραμένει αρκετά υψηλό έτσι ώστε δεν είναι δυνατή η ευρεία χρήση τους σε εργαστηριακό περιβάλλον. Μια εναλλακτική λύση είναι η χρήση εργαλείων λογισμικού, που επιτρέπουν την ρεαλιστική τους μελέτη και αναπαράσταση, μέσω προσομοίωσης. Υπάρχει μια ποικιλία εργαλείων προσομοίωσης ικανά να ανταποκριθούν στις ολοένα αυξανόμενες απαιτήσεις για την αναπαράσταση πραγματικών συσκευών.

Στην παρούσα πτυχιακή εργασία, παρουσιάζεται η μοντελοποίηση ενός πραγματικού βραχίονα και η ανάπτυξη ενός μοντέλου σε περιβάλλον προσομοίωσης εικονικής πραγματικότητας. Στην συνέχεια παρουσιάζεται γραφικό περιβάλλον διεπαφής, που αναπτύχθηκε για την θεωρητική μελέτη του ευθέως και του αντίστροφου κινηματικού προβλήματος για τον βραχίονα, καθώς και για τον έλεγχο κίνησης τόσο του προσομοιωμένου όσο και του πραγματικού συστήματος.

Η πτυχιακή εργασία δομείται ως εξής:

Στο $2^{ο}$ κεφάλαιο, παρουσιάζεται μια σύντομη επισκόπηση διαθέσιμων ρομποτικών συσκευών, καθώς και εργαλείων προσομοίωσης για τη διδασκαλία θεμάτων που σχετίζονται με τη ρομποτική και τους αυτοματισμούς.

Στο $3^{ο}$ κεφάλαιο, περιγράφεται ο τρόπος ανάπτυξης αντικειμένων και εικονικών κόσμων με τη βοήθεια της γλώσσας VRML, καθώς και ο τρόπος σύνδεσής τους με το λογισμικό MATLAB.

Στο $4^{ο}$ κεφάλαιο, περιγράφεται αναλυτικά το μοντέλο προσομοίωσης που αναπτύχθηκε για την μελέτη ενός πραγματικού, κατακόρυφου, αρθρωτού βραχίονα, τεσσάρων βαθμών ελευθερίας, καθώς και ο τρόπος διασύνδεσής του με το λογισμικό MATLAB.

Στο 5^{ο} κεφάλαιο, περιγράφεται αναλυτικά το περιβάλλον διεπαφής που αναπτύχθηκε για την προσομοίωση και τη μελέτη του ευθέως και του αντίστροφου κινηματικού προβλήματος και τον έλεγχο με ταυτόχρονη εικονική παρουσίαση ενός πραγματικού, κατακόρυφου, αρθρωτού βραχίονα, τεσσάρων βαθμών ελευθερίας. Τέλος, παρατίθενται παραδείγματα, που επιδεικνύουν την σωστή λειτουργία του λογισμικού που αναπτύχθηκε.

ΚΕΦΑΛΑΙΟ 2

ΡΟΜΠΟΤΙΚΟΙ ΒΡΑΧΙΟΝΕΣ ΚΑΙ ΕΚΠΑΙΔΕΥΤΙΚΑ ΛΟΓΙΣΜΙΚΑ ΕΛΕΓΧΟΥ

2.1 ΕΙΣΑΓΩΓΗ

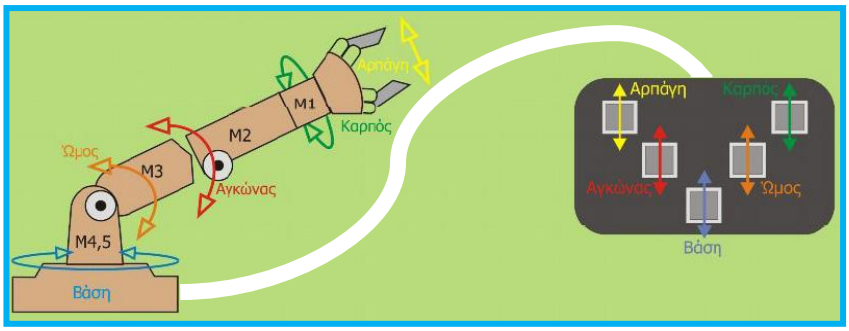
Για την κατανόηση θεμάτων που σχετίζονται με την ρομποτική, τον έλεγχο συστημάτων, τον προγραμματισμό, κλπ, υπάρχουν διαθέσιμες στο εμπόριο, πολλές διαφορετικές επιλογές τόσο για πραγματικά ρομποτικά συστήματα, όσο και για εφαρμογή και προσομοίωση σε επίπεδο λογισμικού. Στο κεφάλαιο αυτό, θα περιγραφούν ενδεικτικές διαθέσιμες λύσεις, τόσο σε επίπεδο υλικού (hardware) όσο και σε επίπεδο λογισμικού (software).

2.2 ΕΚΠΑΙΔΕΥΤΙΚΟΙ ΒΡΑΧΙΟΝΕΣ

Πολλών ειδών βραχίονες, είναι διαθέσιμοι στο εμπόριο για εκπαιδευτική χρήση. Οι διαφορές τους επικεντρώνονται στο τύπο, στους βαθμούς ελευθερίας, στα υλικά κατασκευής, στο μέγεθος και στις δυνατότητες ελέγχου τους. Ενδεικτικά στην συνέχεια παρουσιάζονται οι ακόλουθοι:

Ο ρομποτικός βραχίονας “Elekit” [1] [2] (Σχήμα 2.1), έχει πέντε βαθμούς ελευθερίας και είναι κατασκευαστικά απλός. Ο έλεγχός του πραγματοποιείται από έναν ενσύρματο ελεγκτή (Σχήμα 2.2), με πέντε διακόπτες κίνησης, ένα για κάθε τμήμα (βάση, ώμος, αγκώνας, καρπός και αρπάγη) και έχει σχεδιαστεί για εκπαιδευτικούς σκοπούς με κόστος που δεν ξεπερνά τα 70 ευρώ.

Σχήμα 2.1 Εκπαιδευτικός ρομποτικός βραχίονας “Elekit”



Σχήμα 2.2 Ανάλυση ελεγκτή του εκπαιδευτικού ρομποτικού βραχίονα "Elekit"

Προδιαγραφές ρομποτικού βραχίονα "Elekit"	
Βαθμοί ελευθερίας	5
Περιστροφή βάσης (δεξιά και αριστερά)	περίπου 360°
Ανύψωση ώμου	περίπου 120°
Ανύψωση αγκώνα	περίπου 135°
Περιστροφή καρπού CW και CCW	περίπου 340°
Άνοιγμα και κλείσιμο αρπάγης	περίπου 50 mm
Μέγιστο ύψος βραχίονα	510 mm
Μέγιστο μήκος βραχίονα	360 mm
Μέγιστο βάρος ανύψωσης	περίπου 130 g
Ενσύρματος ελεγκτής 5 διακοπτών κίνησης	

Πίνακας 2.1 Προδιαγραφές εκπαιδευτικού ρομποτικού βραχίονα "Elekit"

Ο ρομποτικός βραχίονας "EduBot 100 GP" [3] [4] (Σχήμα 2.2), έχει πέντε βαθμούς ελευθερίας, συνδέεται με την σειριακή θύρα του υπολογιστή και ελέγχεται, μέσω ειδικού λογισμικού ελέγχου (Robotica). Έχει κατασκευαστεί από αντιστατικό πλαστικό PVC (PolyVinyl Chloride) και έχει περισσότερες δυνατότητες ελέγχου, από τον "Elekit" .

Σχήμα 2.3 Εκπαιδευτικός ρομποτικός βραχίονας "EduBot 100 GP"

Έχει σχεδιαστεί για την εκμάθηση της ρομποτικής και δίνεται η δυνατότητα στο χρήστη, να μπορεί να προγραμματίζει και να ελέγχει το βραχίονα, με απλό τρόπο, μέσω του λογισμικού ελέγχου Robotica. Ακόμα, μπορεί να χρησιμοποιηθεί και σε ασύρματες κινητές εφαρμογές. Έχει πάρα πολλές εφαρμογές για εκπαιδευτικούς σκοπούς και αυτό οφείλεται, στο πολύ καλό λογισμικό ελέγχου και στην ιδιαίτερη κατασκευή του, αφού μπορούν να τοποθετηθούν και επιπρόσθετα εξαρτήματα (όπως αισθητήρες, διακόπτες, ρελέ, φωτοδίοδοι, κ.ά). Το κόστος είναι 1.200 ευρώ.

Προδιαγραφές ρομποτικού βραχίονα "EduBot"	
Βαθμοί ελευθερίας	5
Περιστροφή βάσης	180° max 170° nominal
Ανύψωση ώμου	180° max 170° nominal
Ανύψωση αγκώνα	180° max 170° nominal
Περιστροφή καρπού Roll και Pitch	180° max 170° nominal
Βάρος βραχίονα	1.5 kg
Μέγιστο ύψος βραχίονα	485 mm
Μέγιστο μήκος βραχίονα	372.5 mm
Μέγιστο βάρος ανύψωσης	περίπου 100 g
Σειριακή επικοινωνία 9-pin RS232	

Πίνακας 2.2 Προδιαγραφές εκπαιδευτικού ρομποτικού βραχίονα "EduBot 100 GP"

Εκτός από τους παραπάνω, στο εμπόριο υπάρχουν και άλλοι βραχίονες με αντίστοιχα χαρακτηριστικά όπως: Scorbot-ER 9Pro [5], Lynx 5 [6] και Gridbots [7].

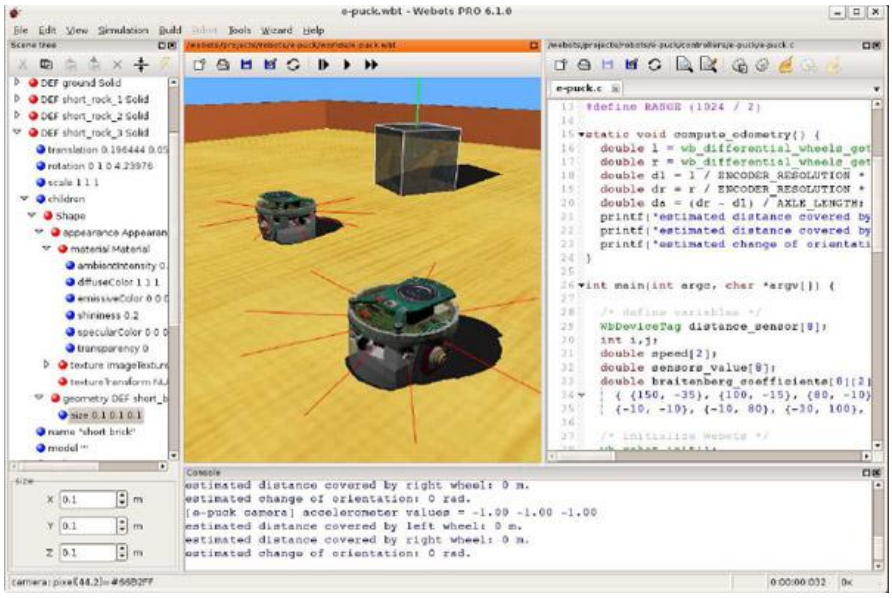
2.3 ΕΚΠΑΙΔΕΥΤΙΚΑ ΛΟΓΙΣΜΙΚΑ ΡΟΜΠΟΤΙΚΗΣ

Πολλοί διαφορετικοί τύποι λογισμικών, είναι διαθέσιμοι για εκπαιδευτικούς και ερευνητικούς σκοπούς. Ενδεικτικά αναφέρονται τα παρακάτω:

Το Webots [8], είναι ένα πακέτο λογισμικού που επιτρέπει την προσομοίωση ρομποτικών συστημάτων. Προσφέρει ένα γρήγορο περιβάλλον διαμόρφωσης πρωτοτύπων, που επιτρέπει στο χρήστη να δημιουργεί τρισδιάστατους εικονικούς κόσμους με ιδιότητες φυσικής, όπως η μάζα, οι ενώσεις, ο συντελεστής τριβής, κλπ. Ο χρήστης μπορεί να προσθέσει ενεργά αντικείμενα και συγκεκριμένα ρομποτικές συσκευές διαφόρων τύπων.

Αυτές, μπορούν να εξοπλιστούν με διαφορετικές συσκευές αισθητήρων και επενεργητών, όπως αισθητήρες απόστασης, ρόδες, κάμερες, servos, αισθητήρες αφής, κλπ. Ο χρήστης μπορεί να προγραμματίσει ρομποτικές συσκευές σε περιβάλλον

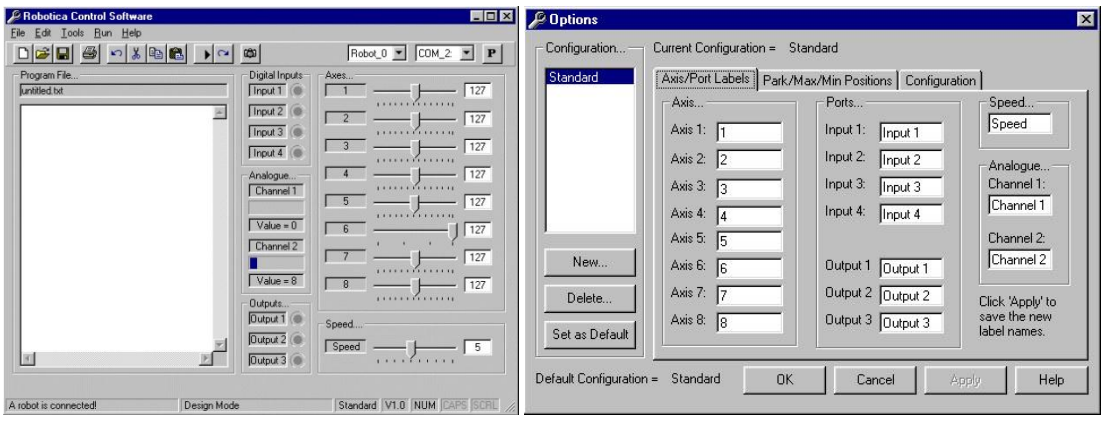
προσομοίωσης και στη συνέχεια να μεταφέρει το λογισμικό στις πραγματικές συσκευές όπως το Khepera, Hemisson, LEGO Mindstorms, Aibo, κλπ.



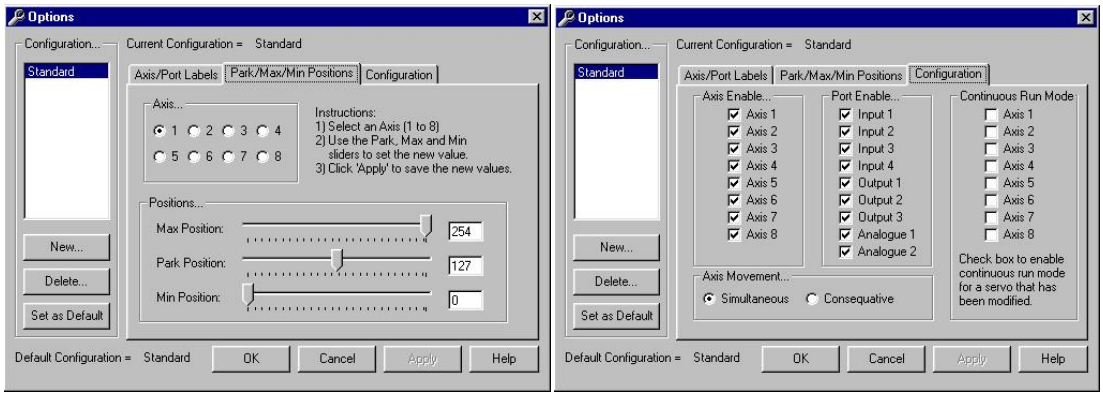
Σχήμα 2.4 Γραφικό περιβάλλον του λογισμικού ελέγχου ρομπότ Webots

Το Robotica [9], είναι ένα λογισμικό πακέτο, για έλεγχο ρομποτικών βραχιόνων. Χρησιμοποιεί μια σειρά από λειτουργίες και ορισμούς, του προγράμματος μαθηματικών Mathematicasymbolic. Το Robotica, μπορεί να λειτουργήσει σε περιβάλλον Microsoft Windows και συνεργάζεται με το πρόγραμμα Mathematica. Το κύριο χαρακτηριστικό του Robotica, είναι η δυνατότητα να υπολογίζει, συμβολικά ή αριθμητικά, τις κινηματικές και δυναμικές εξισώσεις, ενός ρομποτικού συστήματος, χρησιμοποιώντας την μέθοδο Denevit-Hartenburg.

Στο Σχήμα 2.4, παρουσιάζονται μερικές ενδεικτικές εικόνες, από το γραφικό περιβάλλον του Robotica [10].



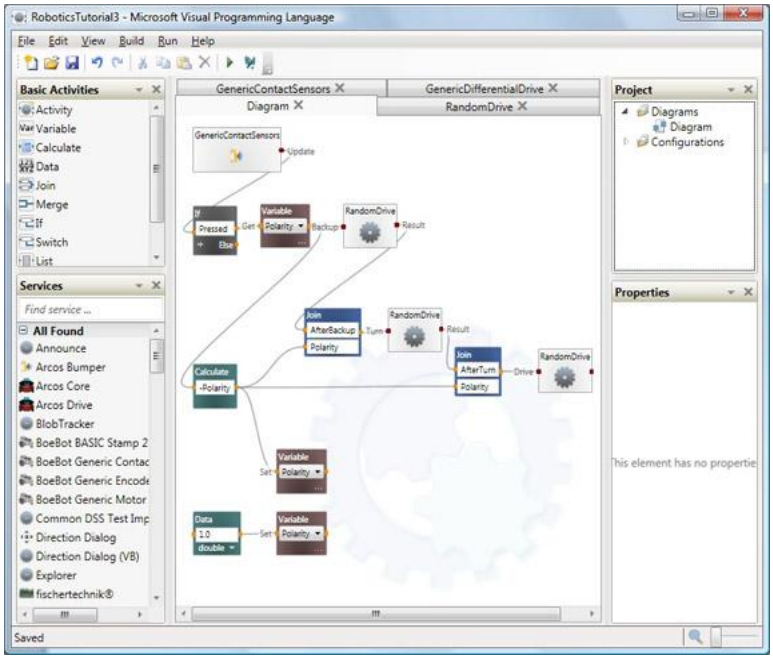
Σχήμα 2.5.α Γραφικό περιβάλλον του λογισμικού ελέγχου βραχίονα Robotica



Σχήμα 2.5.β Γραφικό περιβάλλον του λογισμικού ελέγχου βραχίονα Robotica

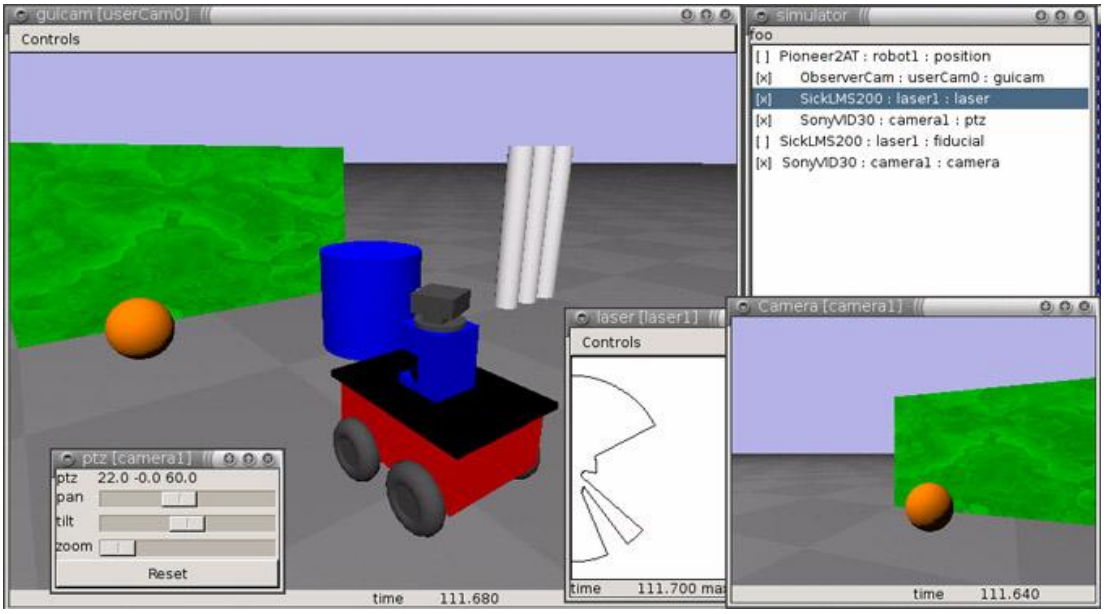
Το Microsoft Robotics Developer Studio [11], είναι ένα λογισμικό πακέτο, για έλεγχο και προσομοίωση ρομπότ. Το πακέτο αυτό είναι βασισμένο στο περιβάλλον των Microsoft Windows και σχεδιάστηκε έτσι ώστε να επιτρέπει σε οποιονδήποτε χρήστη να δημιουργεί εφαρμογές για των έλεγχο ρομποτικών συστημάτων. Χρησιμοποιεί, ένα γραφικό εργαλείο προγραμματισμού (visual programming tool) και καθιστά εύκολη την πρόσβαση σε αισθητήρες και μηχανισμούς κίνησης. Υποστηρίζει ένα μεγάλο αριθμό από γλώσσες όπως C#, Visual Basic, .NET, Jscript και IronPython. Ακόμα, εμπεριέχει κάποια λογισμικά όπως το "Soccer Simulation" και το "Sumo Competition" της Microsoft.

Σχήμα 2.6 Λογισμικό πακέτο ελέγχου ρομπότ Microsoft Robotics Developer Studio



Σχήμα 2.7 Γραφικό περιβάλλον του λογισμικού πακέτου ελέγχου και προσομοίωσης ρομπότ Microsoft Robotics Developer Studio

Το "Player Project" [12], είχε σαν σκοπό να αποτελέσει μια πλατφόρμα ανάπτυξης λογισμικών για ρομποτικές συσκευές και αισθητήρες. Το "Stage" του Player Project (Player/Stage), είναι ένας προσομοιωτής πολλαπλών ρομποτικών συσκευών (multiple robot simulator). Προσομοιώνει ομάδες ρομποτικών συσκευών που κινούνται και αισθάνονται σε ένα 3D χαρτογραφημένο περιβάλλον. Ποικίλα μοντέλα αισθητήρων διατίθενται, όπως σόναρ (sonar), σαρωτές απόστασης λέιζερ (scanning laser rangefinder), κάμερα και οδόμετρο.



Σχήμα 2.8 Γραφικό περιβάλλον του λογισμικού εφαρμογών Project (Player/Stage)

ΚΕΦΑΛΑΙΟ 3

ΓΛΩΣΣΑ ΜΟΝΤΕΛΟΠΟΙΗΣΗΣ ΕΙΚΟΝΙΚΗΣ ΠΡΑΓΜΑΤΙΚΟΤΗΤΑΣ (VRML – VIRTUAL REALITY MODELING LANGUAGE)

3.1 Εισαγωγη

Στο κεφάλαιο αυτό θα περιγραφεί αναλυτικά ο τρόπος ανάπτυξης εικονικών κόσμων με την βοήθεια της VRML (Virtual Reality Modeling Language - Γλώσσα Μοντελοποίησης Εικονικής Πραγματικότητας), τα βασικά της χαρακτηριστικά, καθώς και ο τρόπος αλληλεπίδρασης των εικονικών κόσμων με το λογισμικό MATLAB.

3.2 Γενικα για την VRML

Η γλώσσα VRML, είναι ένα πρότυπο δημιουργίας τρισδιάστατων γραφικών για το παγκόσμιο ιστό (World Wide Web). Πρόκειται για τον ορισμό ενός συγκεκριμένου τύπου αρχείου (file format), που περιέχει μια ASCII περιγραφή των τρισδιάστατων σκηνών. Η VRML αποτέλεσε την πρώτη ουσιαστική προσπάθεια για τη δημιουργία τρισδιάστατης (3D) γλώσσας, για χρήση, κυρίως στο διαδίκτυο. [13]

Με τη χρήση της VRML, ο προγραμματιστής είναι σε θέση να υλοποίει μια ακολουθία εικόνων και να αλληλεπιδρά με αυτές. Για παράδειγμα, μπορεί κανείς να παρατηρεί ένα δωμάτιο ενός κτιρίου και χρησιμοποιώντας κατάλληλα χειριστήρια να μετακινείται στο δωμάτιο σαν να επρόκειτο για πραγματική περιπλάνησή του μέσα στο χώρο. Για την ορθή πλοήγηση μέσα σε έναν τέτοιο τρισδιάστατο κόσμο και για την πρακτική ανάγνωση ενός VRML αρχείου από τον υπολογιστή, είναι απαραίτητη η χρήση ενός φυλλομετρητή (browser) για εφαρμογές της VRML. [14]

Οι εφαρμογές της VRML είναι πολλές: επιχειρήσεις, διαφημίσεις στον παγκόσμιο ιστό (e-commerce), ψυχαγωγία, εκπαίδευση, επικοινωνία, αρχιτεκτονική κλπ. Μπορεί να χρησιμοποιηθεί για τη δημιουργία τρισδιάστατων αναπαραστάσεων πολύπλοκων σκηνών, όπως εικονογραφήσεις, ορισμοί προϊόντων και παρουσιάσεις εικονικής πραγματικότητας. Η VRML αποτελεί ένα ανεξάρτητο πρότυπο, που μπορεί να χρησιμοποιηθεί στο διαδίκτυο ανεξάρτητα από την υπολογιστική πλατφόρμα, καθώς και για τον ορισμό τρισδιάστατων αντικειμένων και την παραμετροποίηση των χαρακτηριστικών τους (σχήμα, χρώμα, μέγεθος, κλπ). [15]

Η πρώτη έκδοση της VRML παρουσιάστηκε το Μάιο του 1995 και βασίστηκε πάνω σε ένα υποσύνολο του 3D μοντέλου της Silicon Graphics, με την ονομασία Ανοικτός Εφευρέτης (Open Inventor). Τον Ιανουάριο του 1996 παρουσιάστηκε η βελτιωμένη έκδοση VRML 1.0c. Το 1997 η έκδοση 2.0 της VRML προτυποποιήθηκε κατά ISO (ISO/IEC 14772-1:1997) και με ελάχιστες διαφορές από την έκδοση 2.0 ονομάστηκε VRML 97. Μεταξύ της πρώτης και της τελευταίας έκδοσης μεσολάβησαν τουλάχιστον 47 πρόχειρες εκδόσεις (drafts).

Η VRML97 είναι ασύμβατη με την VRML 1 και έχει τις εξής βελτιώσεις:

- αυξημένες δυνατότητες διαδραστικότητας και αλληλεπίδρασης με τα μέρη του εικονικού κόσμου,
- υποστήριξη JAVA και Javascript,
- εντολές για ήχο,
- κίνηση και υποστήριξη video.

Το 1999 παρουσιάστηκε το νέο πρότυπο X3D (ISO/IEC 19775-1) που αποτελεί υπερσύνολο της VRML και ενδεικτικά περιλαμβάνει:

- χρήση XML,
- περισσότερες εντολές, άρα και περισσότερες δυνατότητες,
- πιο αυστηρή δομή και κανονικοποίηση,
- δυνατότητα παραγωγής κώδικα σε δυαδική μορφή και ταυτόχρονη συμπίεση (αφού η VRML είναι σε μορφή κειμένου).

Η VRML ποτέ δεν εξαπλώθηκε όσο αναμενόταν διότι:

- όταν εμφανίστηκε απαιτούσε αρκετή επεξεργαστική ισχύ από τις κάρτες γραφικών (πλέον οι απλές σύγχρονες κάρτες γραφικών είναι αρκετά ισχυρές για να αποδώσουν πολύ ικανοποιητικά έναν σύνθετο εικονικό κόσμο)
- ήταν αρκετά πρωτοποριακή ιδέα για το γενικότερο επίπεδο χρηστών του διαδικτύου οι οποίοι, την περίοδο που παρουσιάστηκε η VRML, ζητούσαν την εύκολη και γρήγορη εύρεση πληροφοριών και την απλότητα στην επικοινωνία
- τα εμπορικά προϊόντα πλοήγησης σε εικονικούς κόσμους και απεικόνισης 3D γραφικών δεν βρήκαν απήχηση στο αγοραστικό κοινό με αποτέλεσμα οι εταιρίες να στραφούν σε άλλες μορφές προϊόντων λογισμικού και υλικού.

Ωστόσο, η VRML είναι μια γλώσσα με την οποία μπορούν πολύ εύκολα να δημιουργηθούν απλά τρισδιάστατα αντικείμενα (κύβος, κώνος, κύλινδρος και σφαίρα), να καλυφθούν με χρωματιστές υφές (textures) και να ενσωματωθούν σε άλλα αντικείμενα. Το ίδιο απλή παραμένει και η μεταφορά, η περιστροφή, η μεγέθυνση-σμίκρυνση, ο χρωματισμός και η επικάλυψη των αντικειμένων με υφές.

Οι λόγοι για τους οποίους επιλέχθηκε η VRML για την συγκεκριμένη εργασία, αντί του νέου προτύπου X3D είναι οι εξής:

- Η περιγραφή 3D αντικειμένων με το πρότυπο X3D, βασίζεται στη VRML, οπότε η γνώση της VRML είναι προαπαιτούμενη.

- Για το πρότυπο X3D απαιτείται χρήση της XML με αποτέλεσμα να εισάγεται ακόμη ένα επίπεδο δυσκολίας για τον μέσο χρήστη.
- Η ευκολία διασύνδεσης μοντέλων VRML με προηγμένα λογισμικά προσομοίωσης και ελέγχου σε πραγματικό χρόνο (π.χ. MATLAB). [16]

Τα αρχεία της VRML ονομάζονται εικονικοί κόσμοι και έχουν κατάληξη *".wrl"* (ή ακόμα "*.wrz*" για να δηλώνεται ότι είναι συμπιεσμένα). Τα αρχεία αυτά μπορούν να συγγραφούν από ένα οποιονδήποτε κειμενογράφο (text editor) ή μπορούν να εξαχθούν από κάποια εφαρμογή τρισδιάστατης μοντελοποίησης όπως 3D Studio Max, V-Realm Builder, FormZ, κλπ.. Η δομή τους είναι δενδρική και τους επιτρέπει την αλληλεπίδραση και την αλληλεξάρτηση των επιμέρους τμημάτων τους.

Ένα αρχείο VRML αποτελείται από ένα δένδρο κόμβων (VRML Tree). Οι επιμέρους κόμβοι μπορεί να είναι:

i. Κόμβοι Σχήμα (Shape node), όπως ένα σύνολο από πολύγωνα, μία σφαίρα, ένας κύβος, κ.α.
ii. Κόμβοι Ιδιότητας (Property node), όπως ένα σύνολο από Normals, από Materials, ένα Light ή ένας Transformation.
iii. Κόμβοι τύπου Ομάδας (Group node), που περιέχει κόμβους παιδιά κ.ο.κ.

Σε ένα αρχείο τύπου VRML, ένας κόμβος αποτελείται από ένα προαιρετικό όνομα, ένα τύπο κόμβου και μια λίστα από πεδία (fields) ή ιδιότητες (properties). Προφανώς, όταν δοθεί ένα όνομα σε κάποιο κόμβο, αυτός μπορεί να χρησιμοποιηθεί στη συνέχεια ως έχει. [17]

3.3 ΕΝΑΛΛΑΚΤΙΚΑ ΕΡΓΑΛΕΙΑ ΜΟΝΤΕΛΟΠΟΙΗΣΗΣ ΤΡΙΣΔΙΑΣΤΑΤΩΝ ΑΝΤΙΚΕΙΜΕΝΩΝ

Για αναπαράσταση 3D αντικειμένων, εκτός από την VRML είναι δυνατόν να χρησιμοποιηθούν και εναλλακτικές γλώσσες ή εργαλεία. Ενδεικτικά αναφέρονται τα παρακάτω:

- Η **3DMLW** (**3D M**ark-up **L**anguage for **W**eb) [18] είναι μια γλώσσα ανοιχτού κώδικα (open-source), που βασίζεται σε αρχεία τύπου XML και χρησιμοποιείται για την αναπαράσταση τρισδιάστατων και δισδιάστατων αντικειμένων, με δυνατότητες αλληλεπίδρασης, για τον παγκόσμιο ιστό.
- Η **COLLADA** (**COLLA**borative **D**esign **A**ctivity) [19] είναι μια γλώσσα προγραμματισμού, που βασίζεται σε ένα XML σχήμα και έχει κατοχυρωθεί ως ένα διεθνές πρότυπο, για τη δημιουργία 3D εφαρμογών στο Διαδίκτυο.
- Η **O3D** [20] είναι μια γλώσσα ανοιχτού κώδικα, τύπου JavaScript API. Αναπτύχθηκε από την Google, για την δημιουργία 3D εφαρμογών, με δυνατότητες αλληλεπίδρασης, έτσι ώστε οι τρισδιάστατες εφαρμογές, να λειτουργούν σε ένα φυλλομετρητή ιστού (web browser) ή σε μια XUL εφαρμογή. Η O3D αναπτύσσεται σαν ένα πειραματικό πρόσθετο (plug-in) για φυλλομετρητές ιστού.

- Η **Universal 3D** (**U3D**) [21] είναι ένα συμπιεσμένο πρότυπο αρχείου, που χρησιμοποιείται για δεδομένα 3D γραφικών. Η "3D Industry Forum" είχε ορίσει το πρότυπο αυτό, για να διευκολύνει την ανταλλαγή δεδομένων. Το πρότυπο αυτό κατοχυρώθηκε από την "Ecma International" τον Αύγουστο του 2005 ως ECMA-363, με στόχο να γίνει ένα διεθνές πρότυπο για 3D δεδομένα όλων των ειδών. Στην U3D μορφή, μπορούν να εισαχθούν 3D αντικείμενα με δυνατότητες αλληλεπίδρασης, σε κείμενα τύπου PDF και να παρατηρηθούν μέσω του Acrobat Reader.
- Η **X3D** [22] είναι μια γλώσσα προγραμματισμού, που έχει κατοχυρωθεί ως πρότυπο ISO (ISO/IEC 19775-1), θεωρείται υπερσύνολο της VRML και για την αναπαράσταση εικονικών κόσμων, χρησιμοποιεί αρχεία τύπου XML. Τα χαρακτηριστικά της X3D επεκτείνουν τη VRML (π.χ. Humanoid Animation, NURBS, GeoVRML, κλπ) και της δίνουν την ικανότητα να κωδικοποιεί την εικόνα, χρησιμοποιώντας ένα XML συντακτικό αντίστοιχο, του Ανοιχτού Εφευρέτη της VRML97. Ακόμα, εμπλουτίζεται και το περιβάλλον προγραμματισμού των εφαρμογών (Application Programming Interface - APIs).

3.4 Σχεδιασμος εικονικων κοσμων με χρηση του V-Realm Builder

Για τον σχεδιασμό VRML αντικειμένων, είναι δυνατόν να χρησιμοποιηθεί το πρόγραμμα μοντελοποίησης εικονικών κόσμων, V-Realm Builder (Version 2.0).

Το V-Realm Builder είναι ένα ισχυρό προγραμματιστικό εργαλείο, με δυνατότητα δημιουργίας 3D αντικειμένων και "κόσμων", που στην συνέχεια μπορούν να παρατηρηθούν, από ένα V-Realm φυλλομετρητή ή άλλο φυλλομετρητή συμβατό με VRML 2.0. Το V-Realm Builder και η VRML, δε δημιουργήθηκαν για να αντικαταστήσουν τα σημερινά εργαλεία μοντελοποίησης, που μπορούν να δημιουργήσουν ρεαλιστικά αντικείμενα, με 5 ή 10 εκατομμύρια πολύγωνα. Το V-Realm Builder, χρησιμοποιώντας VRML, έχει το πλεονέκτημα, να ελαχιστοποιεί το μέγεθος των αρχείων και να παρέχει ένα μέσο μοντελοποίησης σύνθετων αντικειμένων, που χρησιμοποιούν μικρό μέγεθος αρχείων.

Το V-Realm Builder έχει ένα φιλικό για τον χρήστη, γραφικό παραθυρικό περιβάλλον, που του δίνει τη δυνατότητα σχεδιασμού και ελέγχου ενός 3D κόσμου, χρησιμοποιώντας μόνο το ποντίκι, αφού δεν είναι απαραίτητη η γραφή κώδικα (hand-coding). Οι περισσότερες συναρτήσεις ολοκληρώνονται με το ποντίκι και η χρήση του πληκτρολογίου είναι περιορισμένη, αφού πραγματοποιείται μόνο για τις συναρτήσεις που απαιτούν είσοδο. Το γραφικό περιβάλλον (Graphical User Interface - GUI) που διαθέτει το V-Realm Builder, είναι προσαρμοσμένο ειδικά στη VRML, με ισχυρές επεμβατικές ικανότητες και λόγω των άμεσων στιγμιαίων οπτικών αναδράσεων, απλοποιεί τη διαδικασία δημιουργίας 3D κόσμων. [23]

3.5 ΕΓΚΑΤΑΣΤΑΣΗ ΚΑΙ ΑΡΧΙΚΟΠΟΙΗΣΗ ΤΟΥ V-REALM BUILDER

Οι εικονικοί κόσμοι που δημιουργούνται με την βοήθεια του V-Realm Builder, έχουν την δυνατότητα αλληλεπίδρασης και ελέγχου, από το λογισμικό MATLAB. Το V-Realm Builder συνοδεύει το λογισμικό MATLAB και μπορεί να εγκατασταθεί μέσω αυτού. Προκειμένου να γίνει αυτό, στο Command Window του λογισμικού MATLAB πληκτρολογούνται οι εντολές: **vrinstall** και **-install editor** ή για λόγους συντομίας μπορεί από την αρχή να πληκτρολογηθεί η εντολή: **vrinstall('-install','editor')**. Στην συνέχεια εμφανίζονται στο Command Window του λογισμικού MATLAB τα εξής:

Starting editor installation…

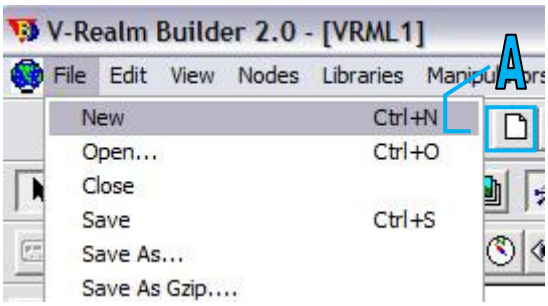
Done.

Για να ελεγχθεί εάν είναι επιτυχής η εγκατάσταση, στο Command Window του λογισμικού MATLAB πληκτρολογούνται οι εντολές: **vrinstall** και **-check** ή για λόγους συντομίας μπορεί από την αρχή να δοθεί η εντολή: **vrinstall('-check')**. Εάν η εγκατάσταση είναι επιτυχής, εμφανίζονται στο Command Window του λογισμικού MATLAB τα εξής:

VRML editor: installed

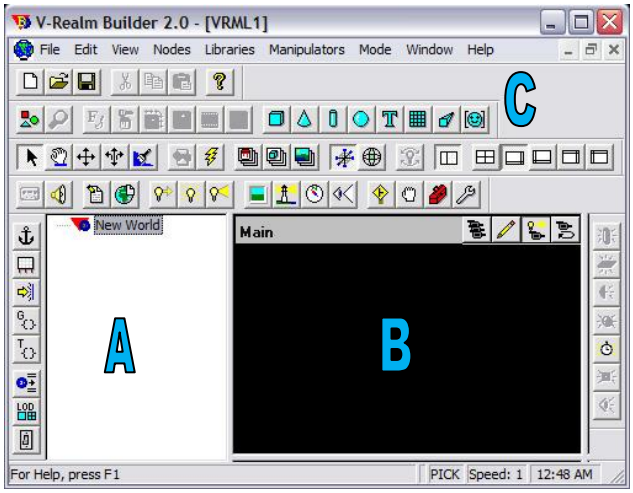
Για την εκκίνηση του V-Realm Builder στα Microsoft Windows, πρέπει να εκτελεστεί το αρχείο “vrbuild2.exe”. [24]

Για να σχεδιαστεί ένας νέος κόσμος με το V-Realm Builder, θα πρέπει στο μενού του V-Realm Builder να επιλεγεί το File και μετά το New ή να πληκτρολογηθεί Ctrl+N ή επιλογή του εικονίδιο New (Σχήμα 3.1, [A]).



Σχήμα 3.1 Μενού έναρξης του V-Realm Builder

Στο αριστερό παράθυρο του V-Realm Builder, υπάρχει ένα δέντρο κόμβων (Σχήμα 3.2, [A]) που καθορίζει την ιεραρχία τον αντικειμένων και τις ιδιότητές τους, όπως τη θέση τους ως προς τον κόσμο και ως προς άλλα αντικείμενα, μέγεθος, χρώμα, περιστροφή κλπ. Στα δεξιά που είναι το βασικό παράθυρο του εικονικού κόσμου (Σχήμα 3.2, [B]), εμφανίζεται ένας άδειος εικονικός κόσμος που θα γεμίσει με τα διάφορα αντικείμενα που θα εισαχθούν σε αυτόν. Ακόμα, υπάρχουν και διάφορες εργαλειοθήκες για την επεξεργασία του εικονικού κόσμου (Σχήμα 3.2, [C]).



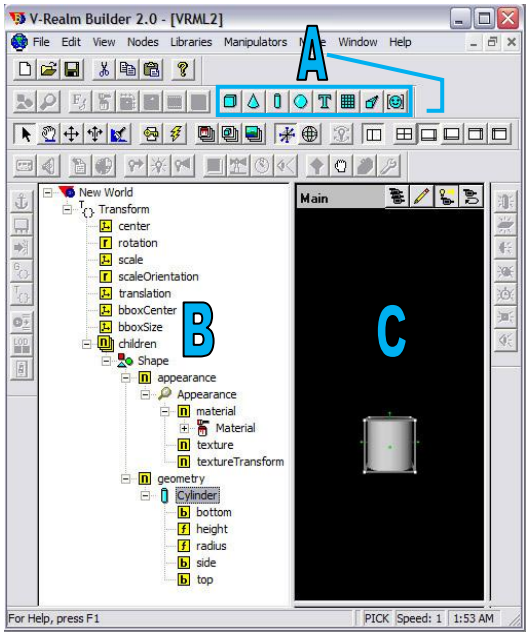
Σχήμα 3.2 Το γραφικό περιβάλλον του V-Realm Builder

Η ιεραρχία ενός τυπικού VRML δέντρου έχει την ακόλουθη δομή. Στην κορυφή βρίσκεται ο εικονικός κόσμος και στο αμέσως χαμηλότερο επίπεδο βρίσκονται διάφοροι κόμβοι (nodes). Κάθε κόμβος (node) αντιπροσωπεύει ένα από τα αντικείμενα που έχουν εισαχθεί και απαρτίζουν τον εικονικό κόσμο, μαζί με τις ιδιότητες που το χαρακτηρίζουν. Τα ομαδοποιημένα στοιχεία κάθε κόμβου λέγονται παιδιά (children). Για παράδειγμα, ένα κτίριο μιας πόλης, μπορεί να θεωρηθεί σαν μια ομάδα κόμβων που έχει ως παιδιά, παράθυρα, πόρτες και τοίχους. Αφού αυτά τα ομαδοποιημένα στοιχεία του κτιρίου, είναι ένα μέρος από την μεγαλύτερη δομή ενός κόσμου που έχει οριστεί, δηλαδή την πόλη. Δεν υπάρχει κάποιο όριο, στον αριθμό τον παιδιών που μπορεί να έχει μια ομάδα, αλλά όλα τα παιδιά μοιράζονται υποχρεωτικά τα χαρακτηριστικά του γονέα (group node).

3.6 Εισαγωγη και επεξεργασια αντικειμενων στο V-Realm Builder

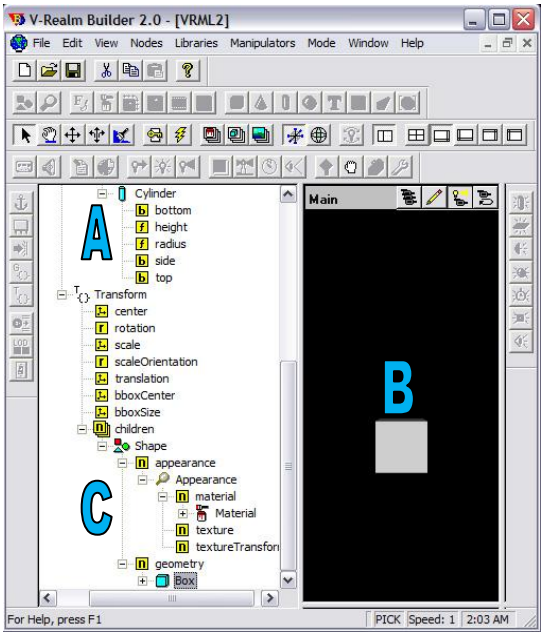
Για την εισαγωγή ενός τρισδιάστατου γεωμετρικού αντικειμένου, στον εικονικό κόσμο, όπως π.χ. ενός κυλίνδρου, πρέπει στην Geometery Node ToolBar (Σχήμα 3.3, [A]) να επιλεγεί, το αντίστοιχο επιθυμητό εικονίδιο.

Μετά την εισαγωγή του, στα αριστερά φαίνεται το δέντρο που περιγράφει τον κύλινδρο (Σχήμα 3.3, [B]). Περιέχει πληροφορίες του αντικειμένου (Σχήμα 3.3, [C]) για τη θέση του στον χώρο, το σχήμα, την εμφάνιση, το υλικό και τη γεωμετρία του (Transform, Shape, Appearance, Material και geometry αντίστοιχα).



Σχήμα 3.3 Εισαγωγή αντικειμένου στο V-Realm Builder

Για να ενωθεί ένα αντικείμενο με ένα άλλο, όπως ένας κύλινδρος με ένα κύβο, θα πρέπει να επιλεγεί το children (παιδιά), από το δέντρο του πρώτου αντικειμένου, δηλαδή το κυλίνδρου (Σχήμα 3.4, [A]) και με τον ίδιο τρόπο, να εισαχθεί από την Geometery Node ToolBar το δεύτερο αντικείμενο, δηλαδή ο κύβος. Όπως φαίνεται παρακάτω, έχει εισαχθεί ο κύβος (Σχήμα 3.4, [B]), που είναι πλέον συνδεδεμένος, με τον κύλινδρο. Με την εισαγωγή του κύβου, εμφανίζεται αυτόματα και το VRML δέντρο του (Σχήμα 3.4, [C]), που εμπεριέχει τις πληροφορίες και τα χαρακτηριστικά του.

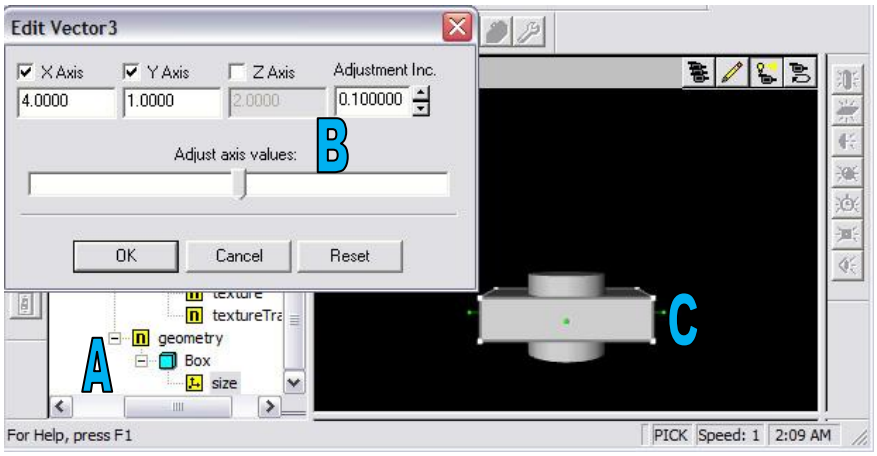


Σχήμα 3.4 Ένωση αντικειμένων στο V-Realm Builder

Για την επεξεργασία των διαστάσεων του κύβου, πρέπει στο δέντρο του κύβου (Σχήμα 3.5, [A]) να ανοίξει:

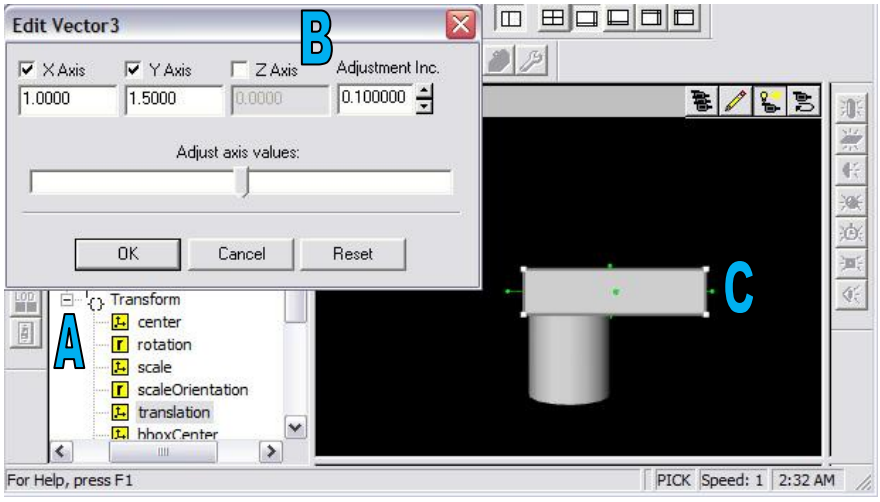
i. Ο κόμβος children, που έχει τα χαρακτηριστικά του σχήματος του αντικειμένου.
ii. Ο κόμβος Shape, που έχει τα γεωμετρικά χαρακτηριστικά και τα χαρακτηριστικά του υλικού.
iii. Ο κόμβος geometry που έχει το γεωμετρικό σχήμα.
iv. Ο κόμβος Box που είναι το σχήμα που είχε επιλεγεί παραπάνω.
v. Το size που ορίζει το μέγεθος του γεωμετρικού σχήματος ενός αντικειμένου.

Στη συνέχεια εμφανίζεται ένα παράθυρο με όνομα "Edit Vector 3" (Σχήμα 3.5, [B]), για να αλλάξουν οι τιμές κατά τους άξονες X, Y και Z. Όπως φαίνεται παρακάτω, ο κύβος έχει μετατραπεί σε ένα ορθογώνιο παραλληλόγραμμο (Σχήμα 3.5, [C]), λόγω των αλλαγών που έχει υποστεί στις διαστάσεις του.



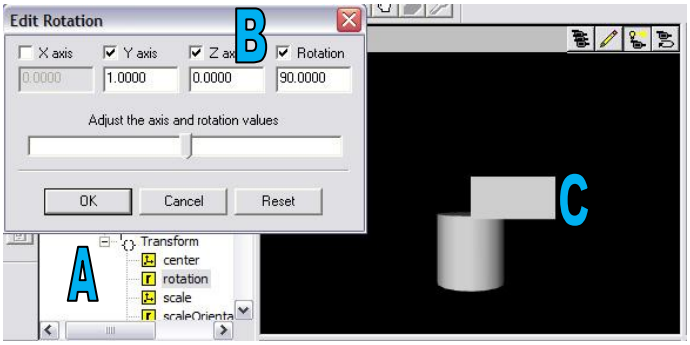
Σχήμα 3.5 Επεξεργασία διαστάσεων αντικειμένων στο V-Realm Builder

Για την επεξεργασία των σχετικών θέσεων των αντικειμένων και συγκεκριμένα, της θέσης του δεύτερου αντικειμένου σε σχέση με το πρώτο, πρέπει να μεταβληθούν οι τιμές της παραμέτρου Transform, στο δέντρο του δεύτερου αντικειμένου (Σχήμα 3.6, [A]). Δηλαδή, στο Transform του δέντρου, του ορθογώνιου παραλληλογράμμου, πρέπει να επιλεγεί το translation, που ορίζει τη θέση των αντικειμένων (Transform → translation). Στο παράθυρο που θα εμφανιστεί (Σχήμα 3.6, [B]) πρέπει να αλλαχθούν οι τιμές κατά τους άξονες X, Y και Z. Όπως φαίνεται, το αντικείμενο έχει τοποθετηθεί στην κορυφή του κυλίνδρου και προς τα δεξιά (Σχήμα 3.6, [C]).



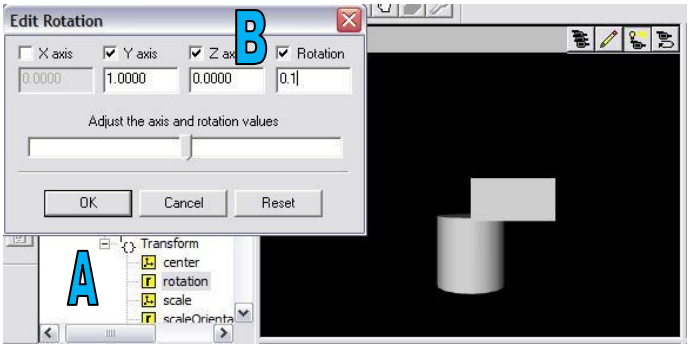
Σχήμα 3.6 Επεξεργασία θέσης αντικειμένων στο V-Realm Builder

Για την επεξεργασία της γωνίας και του άξονα περιστροφής των αντικειμένων, συγκεκριμένα του δεύτερου αντικειμένου σε σχέση με το πρώτο, θα πρέπει στο Transform του δέντρου, του ορθογώνιου παραλληλογράμμου (Σχήμα 3.7.α, [A]), να επιλεγεί το rotation που ορίζει τις γωνίες και τον άξονα περιστροφής (Transform→rotation). Θα εμφανιστεί ένα παράθυρο με όνομα "Edit Rotation" (Σχήμα 3.7.α, [B]), για να επιλεγεί η γωνία και ο άξονας περιστροφής του αντικειμένου. Όπως φαίνετε παρακάτω, το αντικείμενό έχει περιστραφεί ως προς άλλο άξονα (Σχήμα 3.7.α, [C]). Συγκεκριμένα, ο διαμορφωμένος κύβος έχει περιστραφεί 90° ως προς τον Y άξονα.



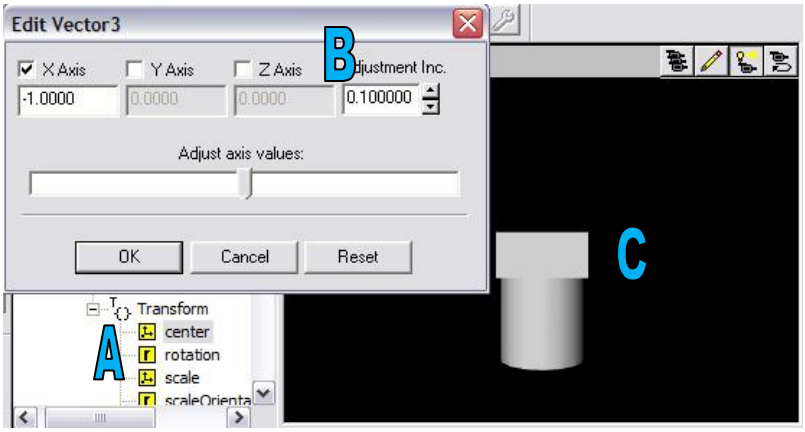
Σχήμα 3.7.α Επεξεργασία γωνίας και άξονα περιστροφής αντικειμένων στο V-Realm Builder

Για να επιλεγεί ο ίδιος άξονας περιστροφής και για τα δυο αντικείμενα, θα πρέπει στο Transform του δέντρου, του κυλίνδρου (Σχήμα 3.7.β, [A]), να επιλεγεί το rotation που ορίζει τις γωνίες και τον άξονα περιστροφής (Transform→rotation) και στο παράθυρο που θα εμφανιστεί με όνομα "Edit Rotation" (Σχήμα 3.7.β, [B]) να επιλεγεί ο επιθυμητός άξονας (Y άξονας). Αν δεν επιλεγεί κοινός άξονας περιστροφής, το αντικείμενο θα περιστρέφεται ως προς δύο άξονες και η κίνηση δεν θα είναι ομαλή.



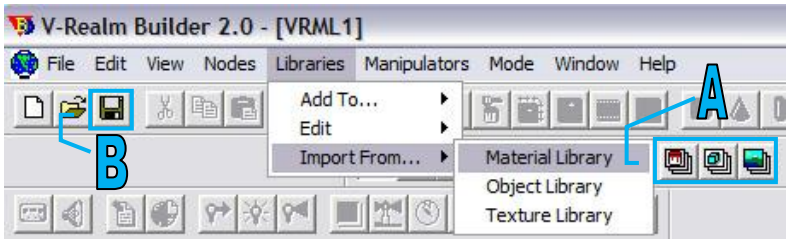
Σχήμα 3.7.β Επεξεργασία γωνίας και άξονα περιστροφής αντικειμένων στο V-Realm Builder

Για να επιλεγεί το ίδιο κέντρο και για τα δυο αντικείμενα, πρέπει στο Transform του δέντρου, του ορθογώνιου παραλληλογράμμου (Σχήμα 3.8, [A]), να επιλεγεί το center, που ορίζει το κέντρο ενός αντικειμένου και στο παράθυρο που εμφανίζεται (Σχήμα 3.7, [B]), να μεταβληθούν οι τιμές του κέντρου κατά τους άξονες X, Y και Z. Όπως φαίνεται, το κέντρο του αντικείμενου έχει μετατοπιστεί (Σχήμα 3.8, [C]).



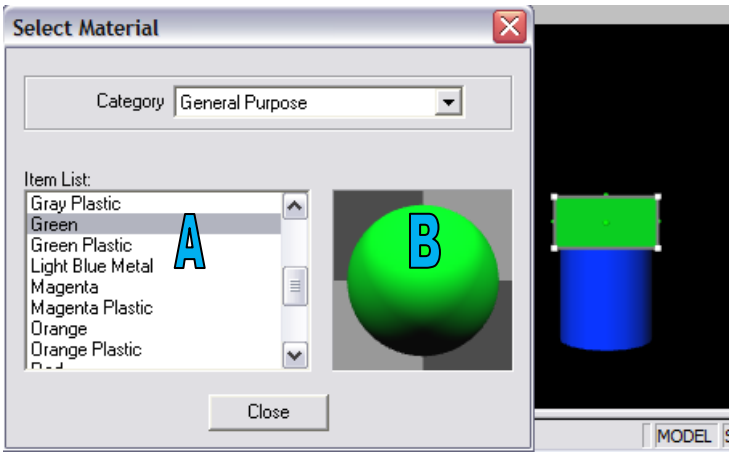
Σχήμα 3.8 Επεξεργασία κέντρου αντικειμένων στο V-Realm Builder

Τα αντικείμενα εμφανίζονται με μια απόχρωση του γκρι, αυτό συμβαίνει γιατί δεν έχουν ορισθεί ακόμα χρώματα για τα αντικείμενα. Για την εισαγωγή χρωμάτων, θα πρέπει στη Mode View ToolBar (Σχήμα 3.9, [A]), να επιλεγεί το πρώτο εικονίδιο που είναι η Βιβλιοθήκη Υλικών (Material Library), για την επιλογή ενός μονόχρωμου υλικού. Το τρίτο εικονίδιο είναι η Βιβλιοθήκη Υφής (Texture Library) και έχει τη δυνατότητα επιλογής έτοιμων χρωματιστών σχεδίων υφής (Texture).



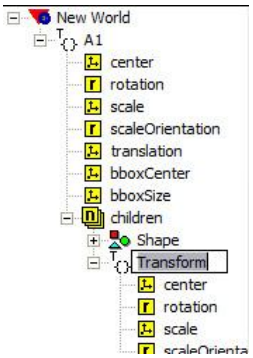
Σχήμα 3.9 Ορισμός χρωμάτων από το Mode View ToolBar *του V-Realm Builder*

Στο παράθυρο που θα εμφανιστεί πρέπει να επιλεγεί χρώμα από την "Item List:" (Σχήμα 3.10, [A]) και στη συνέχεια από την σφαίρα-δειγματολόγιο (Σχήμα 3.10, [B]), να τοποθετηθεί στο αντικείμενο (drag and drop).



Σχήμα 3.10 Εισαγωγή χρωμάτων αντικειμένων στο V-Realm Builder

Προκειμένου να ελεγχθούν τα αντικείμενα ενός εικονικού κόσμου, είναι αναγκαίο να χαρακτηρίζονται από κάποιο όνομα αναφοράς. Για να γίνει η μετονομασία των αντικειμένων αυτών, πρέπει στο δέντρο τους, να επιλεγεί το Transform και μετά να επιλεγεί ξανά, για να μπορεί να ονομαστεί από το πληκτρολόγιο. Θέτω A1 το όνομα του κύβου και A2 το όνομα του παραλληλογράμμου. Στο δέντρο δεν χρειάζεται να ονομαστεί και ο εικονικός κόσμος, δηλαδή το New World καθώς θα ονομαστεί αυτόματα μόλις αυτός αποθηκευτεί.



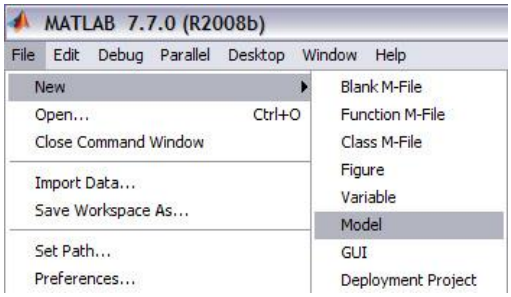
Σχήμα 3.11 Μετονομασία αντικειμένων στο V-Realm Builder

Για να αποθηκευτεί ο εικονικός κόσμος που έχει δημιουργηθεί, πρέπει στο μενού του V-Realm Builder, να επιλεγεί το File και μετά στο Save ή Ctrl+S ή να επιλεγεί το εικονίδιο Save (Σχήμα 3.9, [B]). Το όνομα μπορεί να αλλαχθεί ή να μείνει αυτό που ήδη υπάρχει ως προεπιλογή (vrml1). Έτσι, δημιουργείτε ένα αρχείο με κατάληξη .wrl (π.χ. vrml1.wrl) από τη λέξη world.

3.7 ΕΛΕΓΧΟΣ ΕΙΚΟΝΙΚΩΝ ΚΟΣΜΩΝ ΜΕΣΩ SIMULINK

Οι εικονικοί κόσμοι που δημιουργούνται με τη χρήση του V-Realm Builder, είναι δυνατόν να ελεγχθούν μέσω του λογισμικού Simulink. Το Simulink είναι ένα γραφικό περιβάλλον μοντελοποίησης και προσομοίωσης γραμμικών και μη γραμμικών συστημάτων. Αποτελεί τμήμα του λογισμικού MATLAB και διαφοροποιείται από αυτό, στο ότι διαθέτει, ένα παραθυρικό γραφικό περιβάλλον (GUI) που με την βοήθεια αυτού, ένας χρήστης μπορεί να δημιουργήσει δικά του μοντέλα. Για την αλληλεπίδρασή του με τους εικονικούς κόσμους, χρησιμοποιείται η βιβλιοθήκη εικονικής πραγματικότητας (Virtual Reality Toolbox library) του Simulink. Η βιβλιοθήκη εικονικής πραγματικότητας, παρέχει λογικά (block) διαγράμματα για απευθείας σύνδεση των σημάτων του λογισμικού προσομοίωσης με εικονικούς κόσμους.

Για την δημιουργία ενός μοντέλου προσομοίωσης (Simulink Model) το οποίο έχει κατάληξη .mdl (από την λέξη model), θα πρέπει στο κεντρικό μενού του λογισμικού MATLAB να επιλεγεί File→New→Model και να αποθηκευτεί με ένα επιθυμητό όνομα. Συγκεκριμένα, "demo1.mdl".

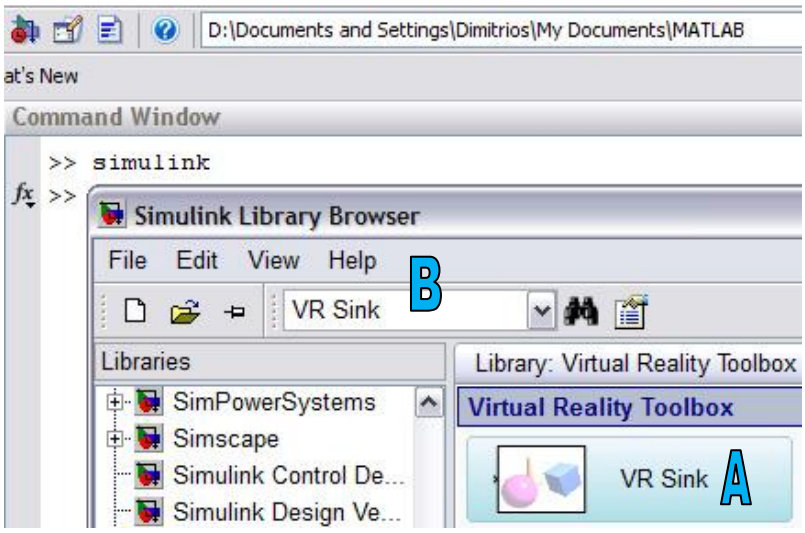


Σχήμα 3.12 Δημιουργία Μοντέλου προσομοίωσης στο λογισμικό MATLAB

Για να γίνει κατανοητός ο τρόπος σύνδεσης εικονικών κόσμων, με μοντέλα προσομοίωσης που έχουν αναπτυχθεί στο λογισμικό Simulink, θα περιγραφεί αναλυτικά ο τρόπος σύνδεσης του κόσμου που αναπτύχθηκε στις προηγούμενες παραγράφους, με ένα μοντέλο αυτού του τύπου.

Με την χρήση του διαγράμματος VR Sink (Σχήμα 3.13, [A]), είναι δυνατή η εγγραφή δεδομένων από το μοντέλο προσομοίωσης, στον εικονικό κόσμο. Για να εντοπιστεί το διάγραμμα αυτό, τοποθετείται στο φυλλομετρητή της Βιβλιοθήκης του λογισμικού Simulink (Simulink Library Browser), η λέξη κλειδί "VR Sink" (Σχήμα

3.13, [B]). Γίνεται η αναζήτηση του VR Sink και όταν αυτό βρεθεί, τοποθετείται στο μοντέλο προσομοίωσης.



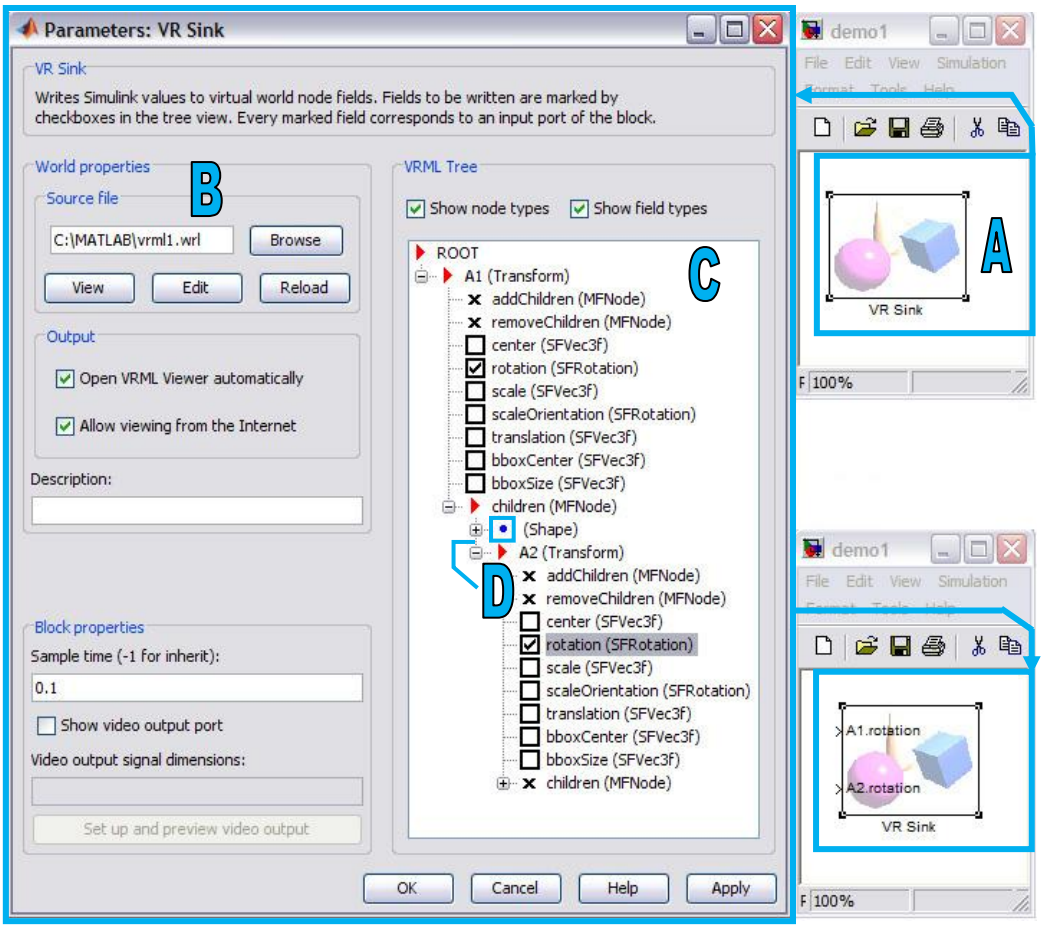
Σχήμα 3.13 Άνοιγμα του φυλλομετρητή της βιβλιοθήκης του λογισμικού Simulink από το λογισμικό MATLAB

Οι παράμετροι ελέγχου, μπορούν να επιλεχθούν, μέσα από το περιβάλλον προσομοίωσης. Μετά από την επιλογή των παραμέτρων, το λογισμικό Simulink ενημερώνει το διάγραμμα (VR Sink) με εισόδους και εξόδους, που αντιστοιχούν στους επιλεγμένους κόμβους του εικονικού κόσμου. Αφού συνδεθούν στις εισόδους τα κατάλληλα σήματα, ο χρήστης μπορεί να παρακολουθεί τη προσομοίωση με ένα λογισμικό αναπαράστασης VRML (VRML viewer).

Όλες οι ιδιότητες των VRML κόμβων, παρατίθενται ιεραρχικά σε μορφή δέντρου. Όταν εισαχθεί στο VR Sink ένας εικονικός κόσμος, σαρώνεται αυτόματα για διαθέσιμους VRML κόμβους, που μπορούν να οριστούν ως είσοδοι του διαγράμματος και να οδηγηθούν σε αυτές, τα σήματα που παράγονται από το λογισμικό Simulink.

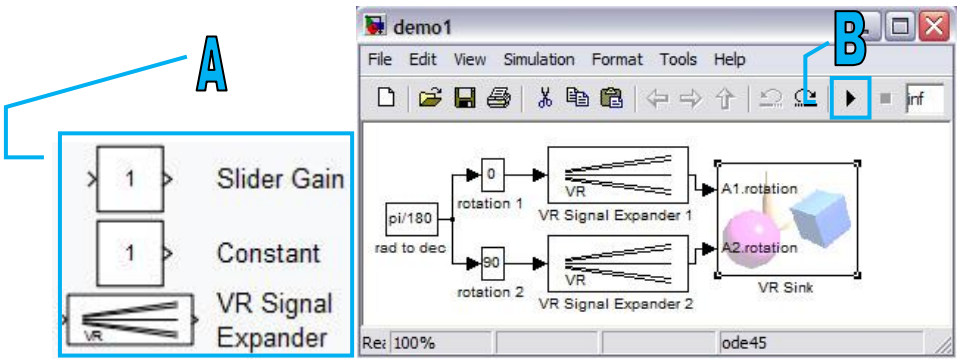
Συγκεκριμένα για τον έλεγχο των δυο αντικειμένων που περιγράφονται στις προηγούμενες παραγράφους, τοποθετείται ένα VR Sink. Για να εμφανιστεί το παράθυρο ιδιοτήτων “Parameters: VR Sink”, θα πρέπει να ανοιχτεί το VR Sink (Σχήμα 3.14, [A]). Στα αριστερά του παραθύρου “Parameters: VR Sink”, υπάρχει ένα πλαίσιο (panel) με το όνομα “World properties”. Μέσα σε αυτό, υπάρχει άλλο ένα, με το όνομα “Source file” (Σχήμα 3.14, [B]). Στο Source file, πρέπει να επιλεγεί το “Browse”, για να αναζητηθεί και να βρεθεί, το αρχείο του εικονικού κόσμου που θα χρησιμοποιηθεί (vrml1.wrl). Στα δεξιά του παραθύρου “Parameters: VR Sink”, υπάρχει το πλαίσιο “VRML Tree”, στο οποίο εμφανίζεται, το VRML δέντρο του εικονικού κόσμου (Σχήμα 3.14, [C]). Για να επιλεχθούν οι παράμετροι ελέγχου του κόσμου, θα πρέπει να επεκταθούν (expand) τα Transform επιλέγοντας τα εικονίδια με

το "+" (Σχήμα 3.14, [D]) και από εκεί να επιλεχθούν οι είσοδοι, συγκεκριμένα, τα "rotation (SFRotation)".



Σχήμα 3.14 Παράθυρο ιδιοτήτων του VR Sink διαγράμματος

Για να επιτευχθεί η κίνηση των αντικειμένων, πρέπει να τοποθετηθούν δομικά τμήματα (Σχήμα 3.15, [A]), που να συνδέονται με τις εισόδους του VR Sink και δίνουν τις τιμές για την κίνηση των αντικειμένων.

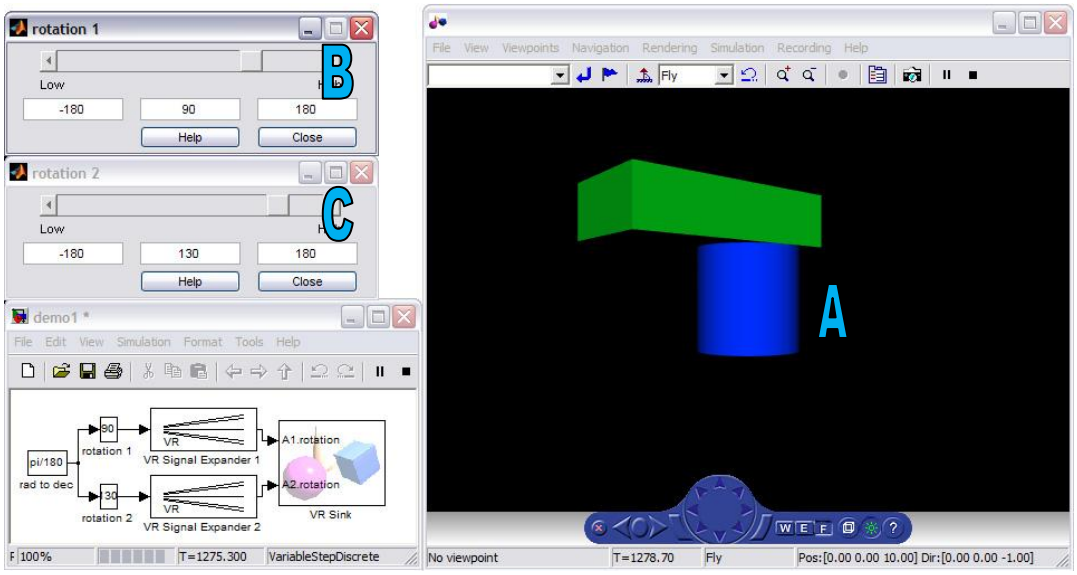


Σχήμα 3.15 Διαγράμματα και το Μοντέλο προσομοίωσης στο λογισμικό MATLAB

Για να κινηθεί περιστροφικά ολόκληρο το αντικείμενο ή μόνο το ορθογώνιο παραλληλόγραμμο, δίνονται τιμές από δύο “Slider Gain” διαγράμματα (ένα για την κάθε περιστροφική κίνηση), που έχουν τη δυνατότητα να πολλαπλασιάζουν το σήμα εισόδου, ανάλογα με την θέση που βρίσκεται η μπάρα ολίσθησης (Slider). Για τον έλεγχο της πλήρους περιστροφής (360°) του κάθε αντικειμένου, τα Slider Gain ρυθμίζονται με όριο από -180 έως 180. Οι τιμές των Slider Gain είναι σε ακτίνια. Για την μετατροπή των ακτινίων σε μοίρες, τοποθετείται στην είσοδο των Slider Gain διαγραμμάτων, ένα “Constant” διάγραμμα ίσο με π/180, το οποίο έχει την ιδιότητα, να δίνει ένα σήμα εξόδου. Αυτό στην συνέχεια πολλαπλασιάζεται με το αποτέλεσμα των Slider Gain διαγραμμάτων και οδηγείται στην έξοδό τους. Πριν οδηγηθεί το σήμα στο VR Sink, τοποθετείται ανάμεσα στην έξοδο των Slider Gain διαγραμμάτων και την είσοδο του VR Sink διαγράμματος, ένα “VR Signal Expander” για την επέκταση του σήματος

Για την έναρξη της προσομοίωσης, πρέπει στο μενού του Simulink Model (μοντέλου προσομοίωσης) του λογισμικού MATLAB, να επιλεγεί το Simulation και μετά το Start ή Ctrl+T ή να επιλεγεί το εικονίδιο Start simulation (Σχήμα 3.15, [B]).

Για να εμφανιστεί ο εικονικός κόσμος (Σχήμα 3.16, [A]), πρέπει να επιλεχθεί το VR Sink. Για τον έλεγχο του εικονικού κόσμου υπάρχουν τα Slider Gain, που ανοίγοντάς τα εμφανίζονται οι μπάρες ολίσθησης. Η περιστροφή του κάθε αντικειμένου στον εικονικό κόσμο, είναι ανάλογη των τιμών της μπάρας ολίσθησης. Με την μπάρα ολίσθησης του παραθύρου που έχει ονομαστεί “rotation 1”, μπορεί να ελεγχθεί η περιστροφή ολόκληρου του αντικειμένου (Σχήμα 3.16, [B]). Με την μπάρα ολίσθησής του παραθύρου που έχει ονομαστεί “rotation 2”, μπορεί να ελεγχθεί η περιστροφή μόνο του ορθογώνιου παραλληλογράμμου, του αντικειμένου (Σχήμα 3.16, [C]).



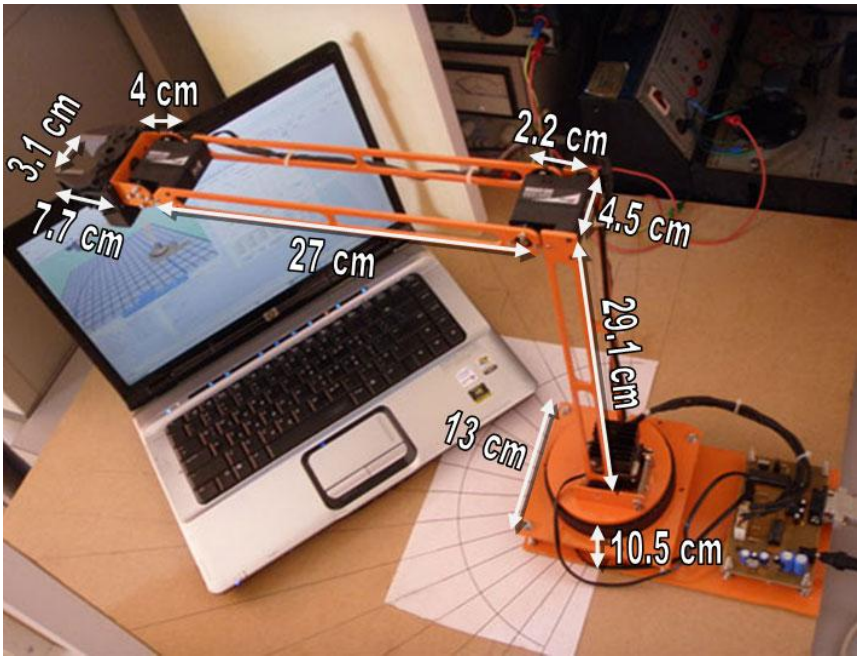
Σχήμα 3.16 Μπάρες ολίσθησης του Μοντέλου προσομοίωσης και προβολή από ειδικό λογισμικό προσομοίωσης του λογισμικού MATLAB

ΚΕΦΑΛΑΙΟ 4

ΣΧΕΔΙΑΣΜΟΣ ΠΡΩΤΟΤΥΠΟΥ ΡΟΜΠΟΤΙΚΟΥ ΒΡΑΧΙΟΝΑ ΣΕ ΕΙΚΟΝΙΚΟ ΠΕΡΙΒΑΛΛΟΝ

4.1 Εισαγωγη

Στο κεφάλαιο αυτό, θα περιγραφεί αναλυτικά ο τρόπος με τον οποίο μοντελοποιήθηκε ένας πραγματικός ρομποτικός βραχίονας, που έχει αναπτυχθεί και κατασκευαστεί στο Τμήμα Ηλεκτρονικής του Α.Τ.Ε.Ι. Κρήτης [25]. Πρόκειται για ένα κατακόρυφο, αρθρωτό βραχίονα, τεσσάρων βαθμών ελευθερίας. Έχει κατασκευαστεί από αλουμίνιο και φέρει πέντε κινητήρες τύπου σέρβο (servo) που είναι υπεύθυνοι για την κίνησή του. Αναλυτικά, υπάρχει ένας κινητήρας στη βάση, ένας σε κάθε άρθρωση και ένας στο άκρο εργασίας. Η κίνηση του βραχίονα, ελέγχεται από ένα ελεγκτή τύπου "SSC-32 Ver 2.0". Ο ελεγκτής των σερβοκινητήρων, δέχεται εντολές μέσω της σειριακής θύρας του υπολογιστή από ένα ειδικά σχεδιασμένο λογισμικό, που έχει δημιουργηθεί με το λογισμικό MATLAB. Ο βραχίονας και οι διαστάσεις του παρουσιάζονται παρακάτω (Σχήμα 4.1).



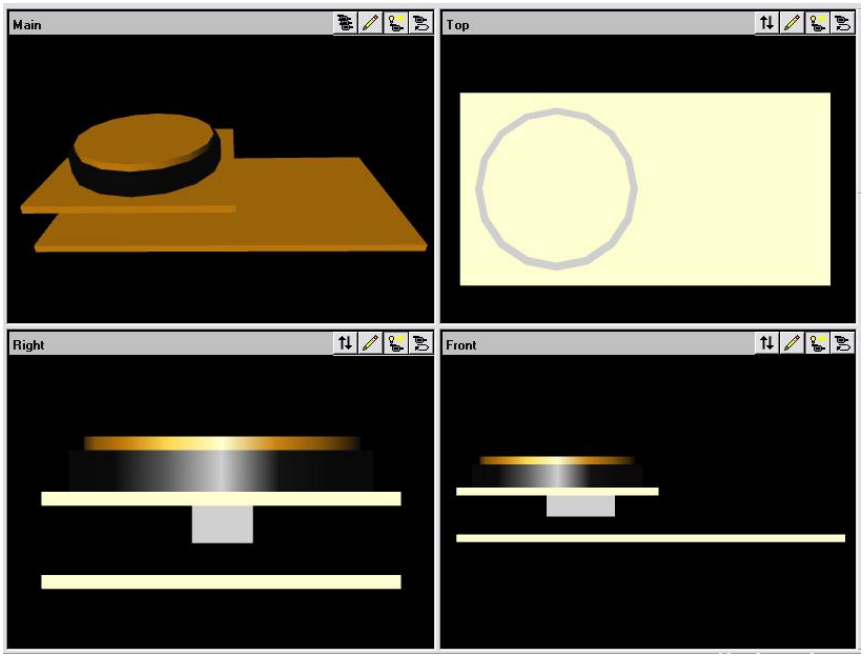
Σχήμα 4.1 Ο Ρομποτικός Βραχίονας

4.2 ΑΝΑΠΤΥΞΗ ΕΙΚΟΝΙΚΟΥ ΜΟΝΤΕΛΟΥ

Για την ανάπτυξη του εικονικού μοντέλου του βραχίονα, χρησιμοποιήθηκε το λογισμικό V-Realm Builder. Έγινε προσπάθεια για την ακριβέστερη μεταφορά του βραχίονα στον εικονικό κόσμο, λαμβάνοντας υπόψη τις δυσκολίες αναπαράστασης ορισμένων σχεδιαστικών λεπτομερειών. Για να είναι πιο εύκολος ο σχεδιασμός, έγιναν οι κατάλληλες απλοποιήσεις και αφαιρέθηκαν σχεδιαστικές λεπτομέρειες, όπως τα καλώδια και τα εξαρτήματα του ελεγκτή των σερβοκινητήρων.

4.2.1 Σχεδιασμός βάσης του βραχίονα με το V-Realm Builder

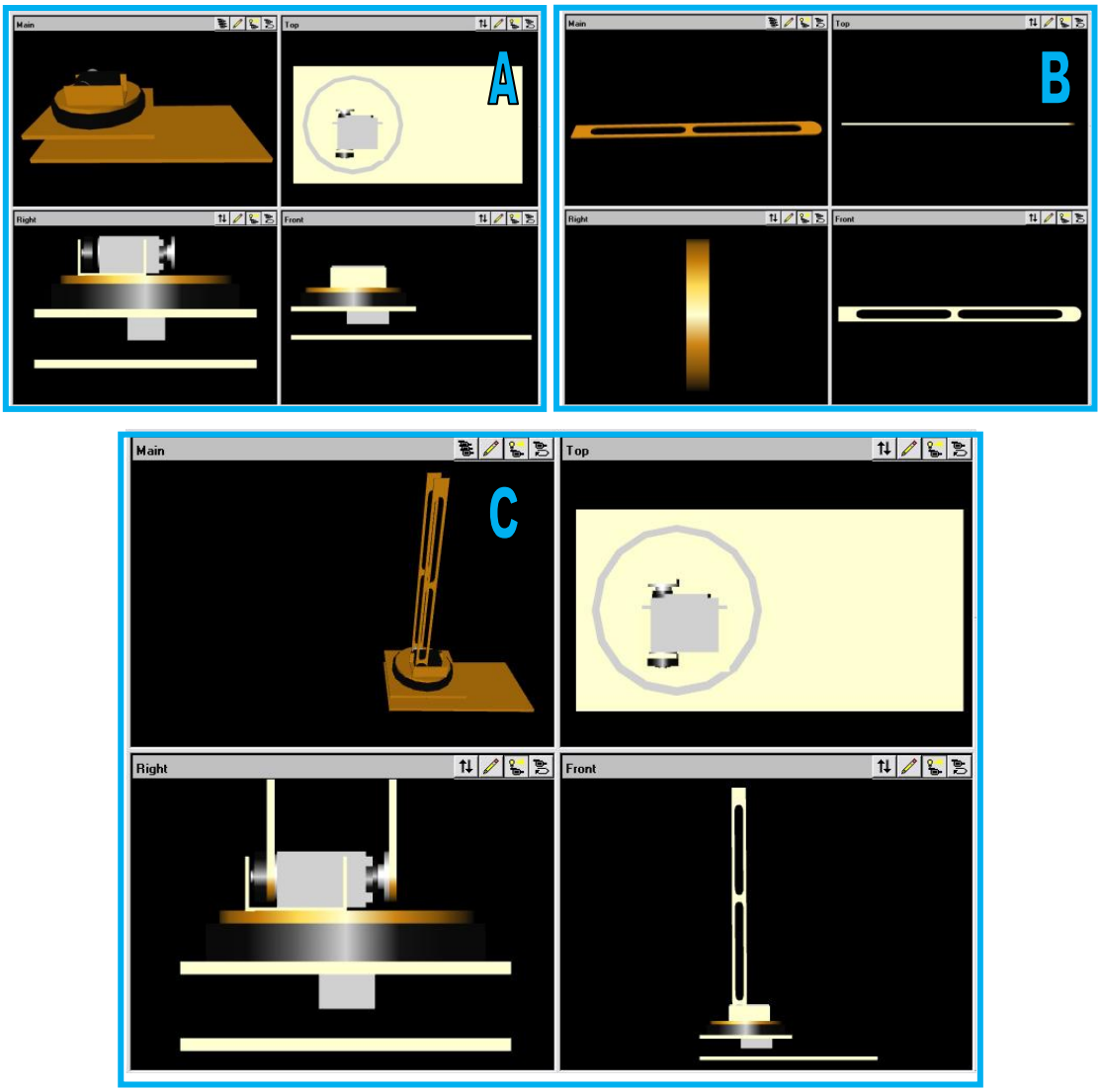
Για το σχεδιασμό της βάσης του βραχίονα, επιλέγονται από την εργαλειομπάρα γεωμετρικών αντικειμένων (Geometry Node ToolBar), αντικείμενα σχήματος κουτιού (box) και κυλίνδρου (cylinder). Τα αντικείμενα αυτά διαμορφώνονται εισάγοντας τις ανάλογες διαστάσεις.



Σχήμα 4.2 Σχεδιασμός βάσης εικονικού βραχίονα

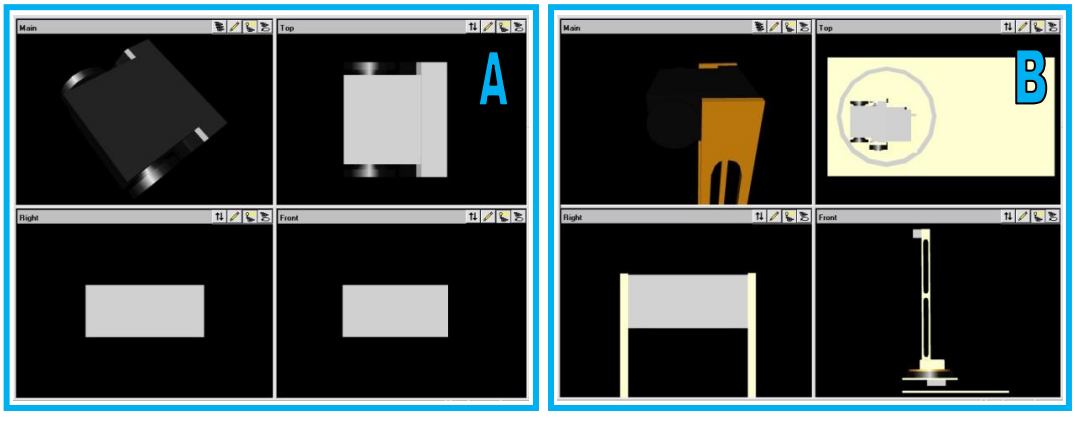
4.2.2 Σχεδιασμός αρθρώσεων του βραχίονα με το V-Realm Builder

Για το σχεδιασμό της πρώτης άρθρωσης του βραχίονα, δημιουργείται αρχικά μια βάση στήριξης, στην οποία τοποθετείται ο σερβοκινητήρας (Σχήμα 4.3, [A]) και στα άκρα αυτού, συνδέονται δυο ειδικά κομμένες λάμες (Σχήμα 4.3, [B]). Όλα τα αντικείμενα που χρησιμοποιήθηκαν για το σχεδιασμό του εικονικού βραχίονα, αποτελούνται από ένα σύνολο μικρότερων γεωμετρικών σχημάτων. Όπως φαίνεται, συνδέεται η πρώτη άρθρωση, με την βάση (Σχήμα 4.3, [C]).



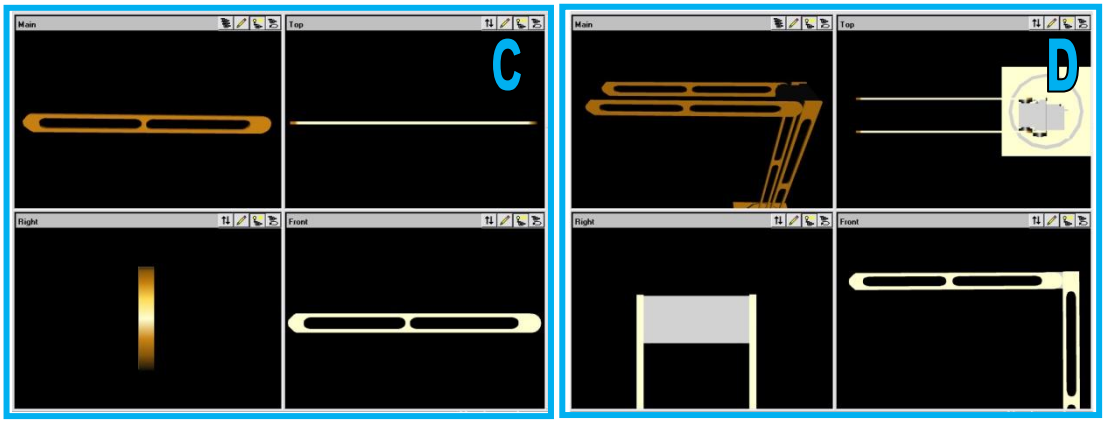
Σχήμα 4.3 Σχεδιασμός πρώτης άρθρωσης του εικονικού βραχίονα

Για το σχεδιασμό της δεύτερης άρθρωσης του βραχίονα, δημιουργείται αρχικά ένας σερβοκινητήρας (Σχήμα 4.4.α, [A]), ο οποίος τοποθετείται στο άκρο της πρώτης άρθρωσης (Σχήμα 4.4.α, [B]).



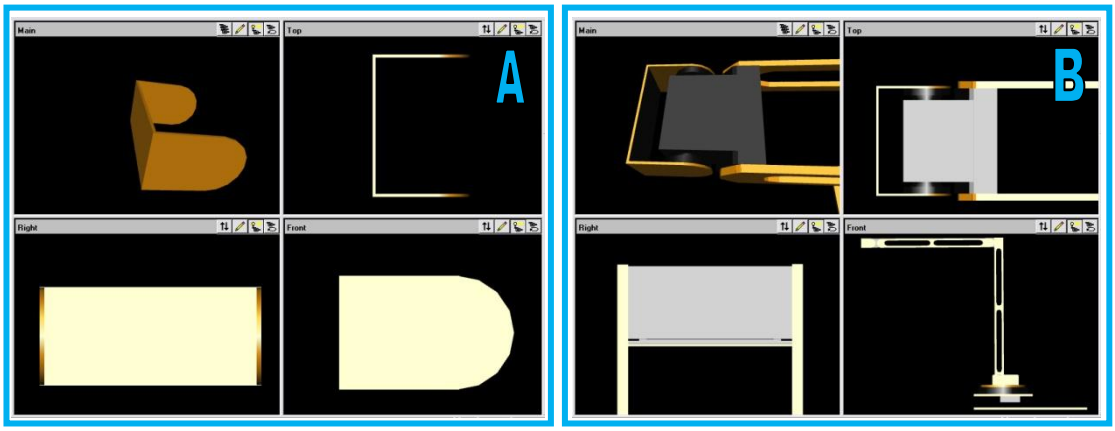
Σχήμα 4.4.α Σχεδιασμός δεύτερης άρθρωσης του εικονικού βραχίονα

Στα άκρα του σερβοκινητήρα, συνδέονται δυο ειδικά κομμένες λάμες (Σχήμα 4.4.β, [C]). Όπως φαίνεται, συνδέεται η δεύτερη άρθρωση, με την πρώτη (Σχήμα 4.4.β, [D]).



Σχήμα 4.4.β Σχεδιασμός δεύτερης άρθρωσης του εικονικού βραχίονα

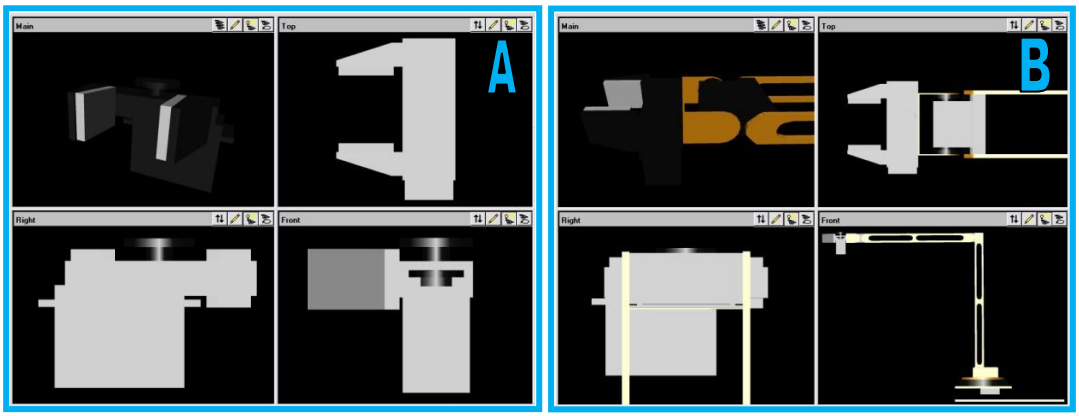
Για το σχεδιασμό της τρίτης άρθρωσης του βραχίονα, τοποθετείται στο άκρο της δεύτερης άρθρωσης ένας σερβοκινητήρας. Στα άκρα του σερβοκινητήρα, συνδέεται μια ειδικά κομμένη λάμα, σε σχήμα "Π" (Σχήμα 4.5, [A]). Όπως φαίνεται, συνδέεται η τρίτη άρθρωση, με την δεύτερη (Σχήμα 4.5, [B]).



Σχήμα 4.5 Σχεδιασμός τρίτης άρθρωσης του εικονικού βραχίονα

4.2.3 Σχεδιασμός άκρου εργασίας του βραχίονα με το V-Realm Builder

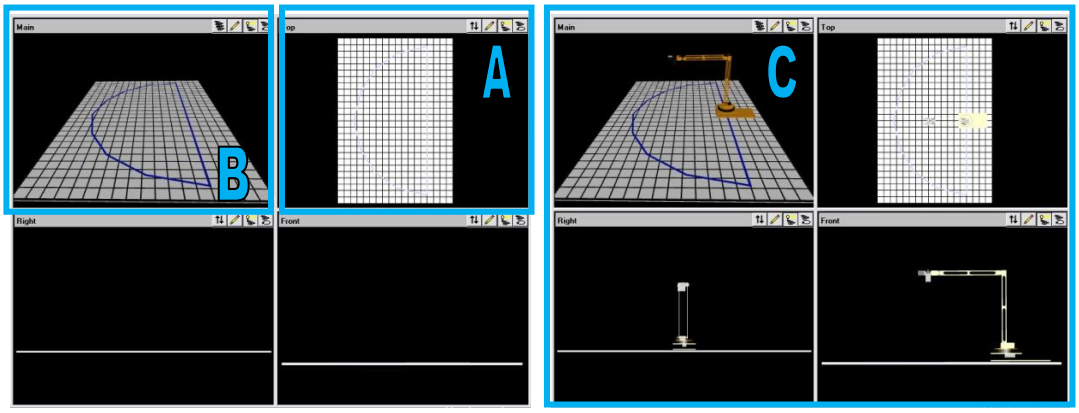
Για το σχεδιασμό του άκρου εργασίας του βραχίονα, τοποθετείται στο άκρο της τρίτης άρθρωσης, μια αρπάγη με έναν ενσωματωμένο σερβοκινητήρα (Σχήμα 4.6, [A]). Όπως φαίνεται, συνδέεται η αρπάγη, με την τρίτη άρθρωση (Σχήμα 4.6, [B]).



Σχήμα 4.6 Σχεδιασμός άκρου εργασίας του εικονικού βραχίονα

4.2.4 Σχεδιασμός δαπέδου του βραχίονα με το V-Realm Builder

Για το σχεδιασμό του δαπέδου του βραχίονα, δημιουργείται μια επιφάνεια που χωρίζεται σε τετράγωνα πέντε εκατοστών (Σχήμα 4.7, [A]), για να αναπαριστάται με μεγαλύτερη ακρίβεια, η κίνηση του βραχίονα πάνω στο δάπεδο. Στο δάπεδο τοποθετήθηκε και ένα ημικύκλιο (Σχήμα 4.7, [B]) που δείχνει τα όρια τις έκτασης του βραχίονα. Το δάπεδο αυτό τοποθετείται στον βραχίονα (Σχήμα 4.7, [C]).

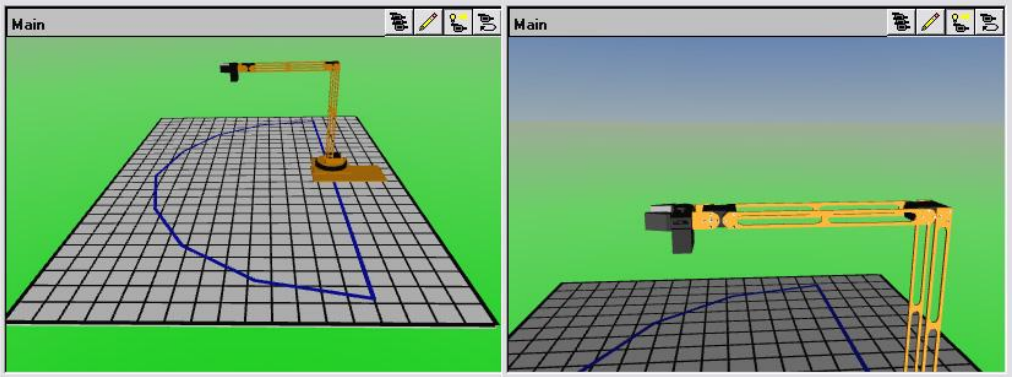


Σχήμα 4.7 Σχεδιασμός δαπέδου του εικονικού βραχίονα

4.2.5 Ορισμός φόντου του βραχίονα με το V-Realm Builder

Για τον ορισμό φόντου του βραχίονα, επιλέγεται από την γενική εργαλειομπάρα (Common Node ToolBar), το εικονίδιο “Insert Background” (εισαγωγή φόντου) (Σχήμα 4.8, [A]) και αυτόματα εισάγεται φόντο, με αποχρώσεις ουρανού και Γής (μπλε και πράσινο).

Σχήμα 4.8 Εισαγωγή φόντου με το V-Realm Builder



Σχήμα 4.9 Ορισμός φόντου του εικονικού βραχίονα

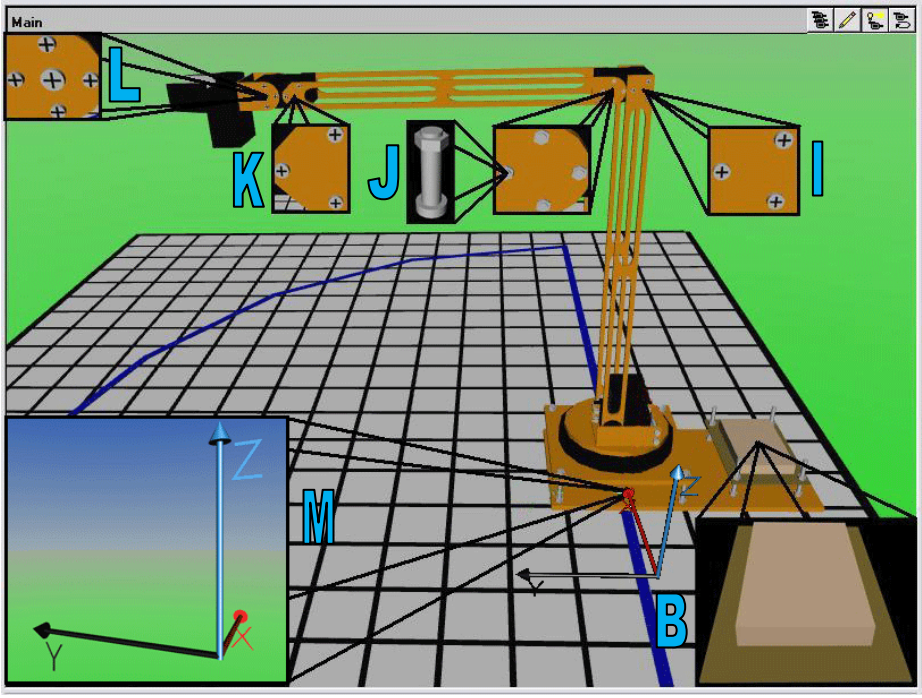
4.2.6 Προσθήκη σχεδιαστικών λεπτομερειών του βραχίονα με το V-Realm Builder

Για να είναι το μοντέλο του εικονικού βραχίονα ρεαλιστικό, αναπτύχθηκαν και προστέθηκαν, μια σειρά από σχεδιαστικές λεπτομέρειες (Πίνακας 4.1), όπως αυτές υπάρχουν στο πραγματικό σύστημα.

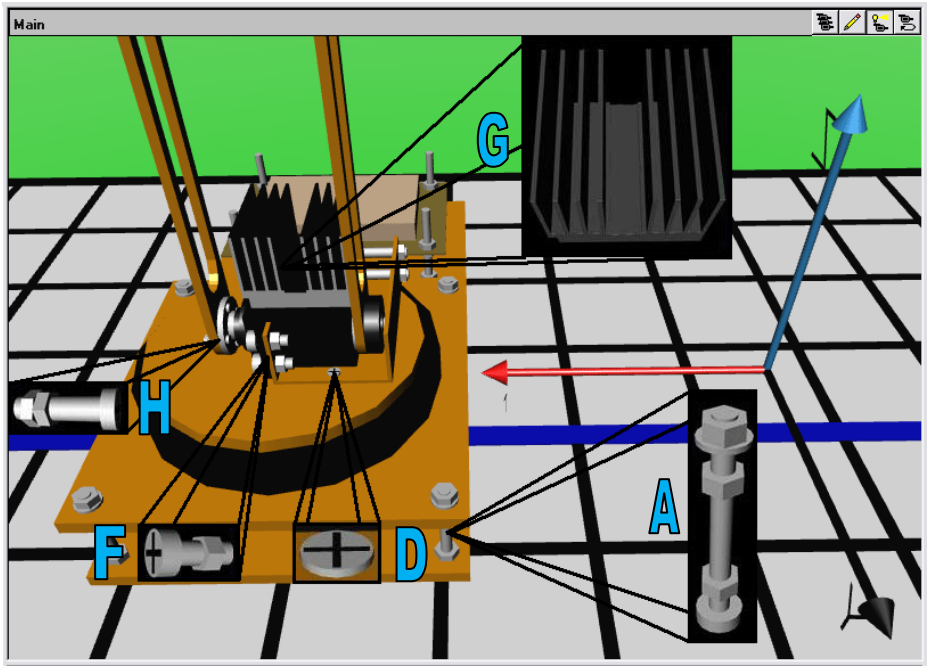
	Σχεδιαστικές λεπτομέρειες	**Σχήματα**
A	Βίδες στήριξης της βάσης του βραχίονα	(Σχήμα 4.10.β, [A])
B	Ελεγκτής των σερβοκινητήρων του βραχίονα	(Σχήμα 4.10.α, [B])
C	Βίδες στήριξης του ελεγκτή του βραχίονα	(Σχήμα 4.10.γ, [C])
D	Βίδες στήριξης της βάσης του σερβοκινητήρα, της πρώτης άρθρωσης με τη βάση του βραχίονα	(Σχήμα 4.10.β & γ, [D])
E	Μεγάλες βίδες στήριξης της βάσης του σερβοκινητήρα, της πρώτης άρθρωσης του βραχίονα	(Σχήμα 4.10.γ, [E])
F	Μικρές βίδες στήριξης της βάσης του σερβοκινητήρα, της πρώτης άρθρωσης του βραχίονα	(Σχήμα 4.10.β, [F])
G	Ψήκτρα του σερβοκινητήρα, της πρώτης άρθρωσης του βραχίονα	(Σχήμα 4.10.β, [G])

H	Βίδες στήριξης του άκρου του σερβοκινητήρα της πρώτης άρθρωσης του βραχίονα με τη λάμα	(Σχήμα 4.10.β, [H])
I	Βίδες στήριξης του σερβοκινητήρα της δεύτερης άρθρωσης του βραχίονα, με το άκρο της πρώτης	(Σχήμα 4.10.α, [I])
J	Βίδες στήριξης των άκρων του σερβοκινητήρα, της δεύτερης άρθρωσης του βραχίονα με τις λάμες	(Σχήμα 4.10.α, [J])
K	Βίδες στήριξης του σερβοκινητήρα της τρίτης άρθρωσης του βραχίονα, με το άκρο της δεύτερης	(Σχήμα 4.10.α, [K])
L	Βίδες στήριξης των άκρων του σερβοκινητήρα της τρίτης άρθρωσης του βραχίονα, με τη λάμα σχήματος "Π"	(Σχήμα 4.10.α, [L])
M	Άξονες X, Y και Z για την καλύτερη κατανόηση, της κίνησης του βραχίονα	(Σχήμα 4.10.α, [M])

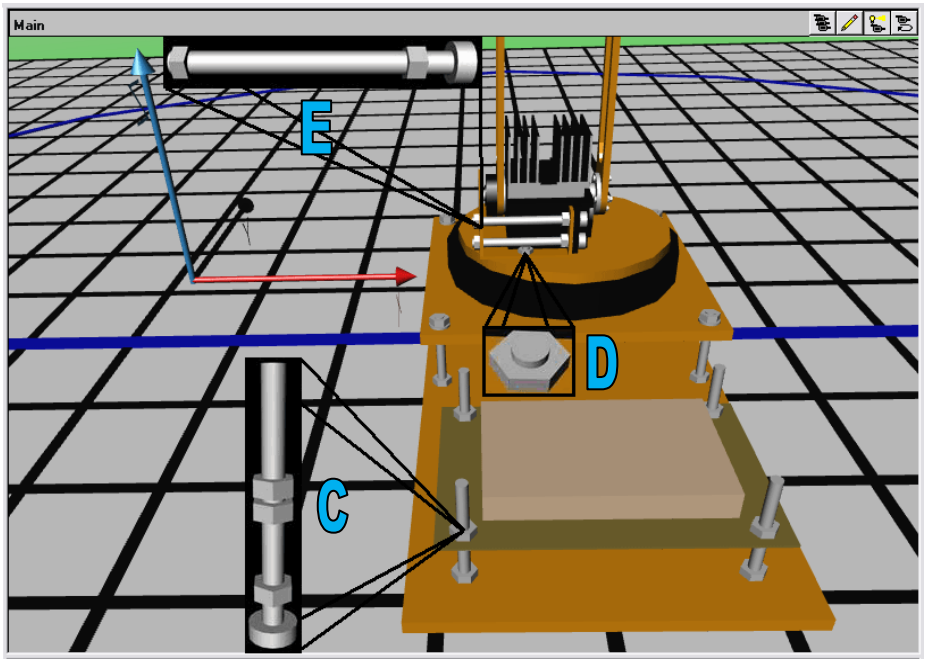
Πίνακας 4.1 Σχεδιαστικές Λεπτομέρειες Εικονικού Βραχίονα



Σχήμα 4.10.α Προσθήκη σχεδιαστικών λεπτομερειών του εικονικού βραχίονα



Σχήμα 4.10.β Προσθήκη σχεδιαστικών λεπτομερειών του εικονικού βραχίονα

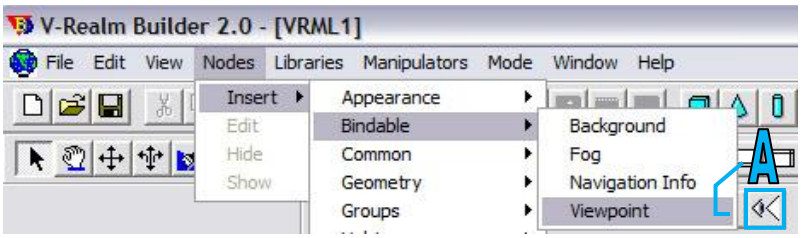


Σχήμα 4.10.γ Προσθήκη σχεδιαστικών λεπτομερειών του εικονικού βραχίονα

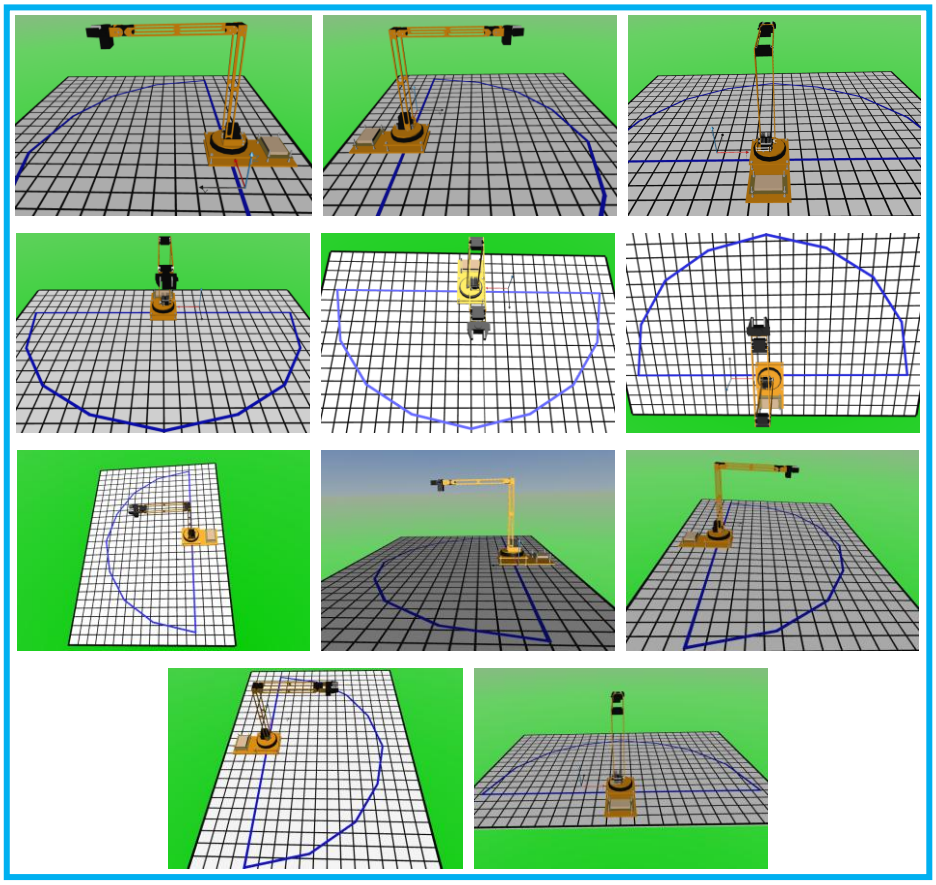
4.2.7 Ορισμός Viewpoint του βραχίονα με το V-Realm Builder

Για την καλύτερη παρακολούθηση του εικονικού βραχίονα, ορίζονται στον εικονικό κόσμο, διαφορετικά σημεία παρατήρησης (Viewpoints). Για την επιλογή τους, έγινε αρχικά περιήγηση με χρήση της εργαλειομπάρας πλοήγησης (Mode View ToolBar) και επιλέχθηκαν τα καλύτερα. Για τον ορισμό τους, επιλέγεται από την

γενική εργαλειομπάρα, το εικονίδιο “Access/Edit Viewpoint” (πρόσβαση/εισαγωγή σημείου παρατήρησης) (Σχήμα 4.11, [A]).



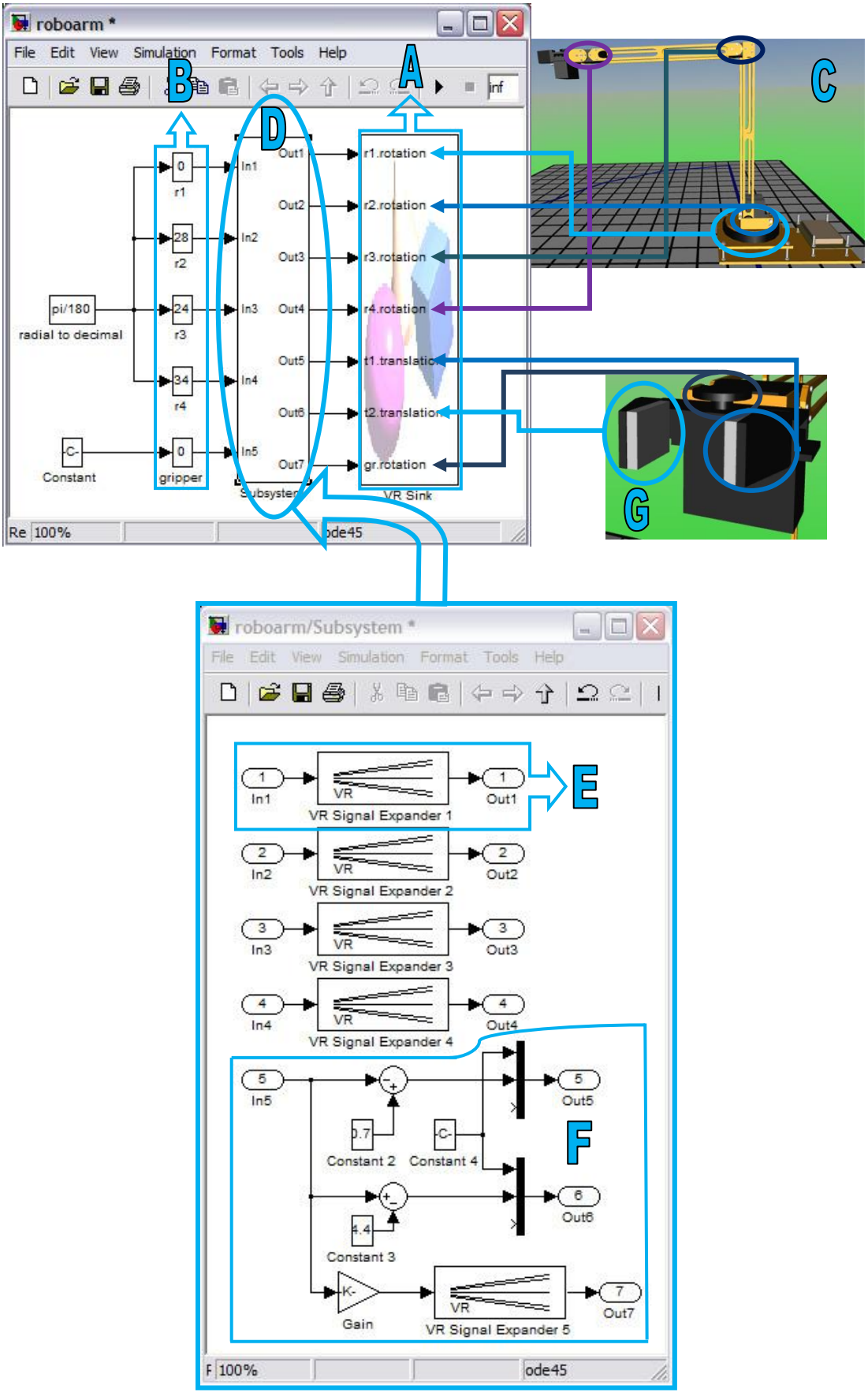
Σχήμα 4.11 Εισαγωγή Viewpoint



Σχήμα 4.12 Viewpoint του εικονικού βραχίονα με το V-Realm Builder

4.3 Σύνδεση του εικονικού κόσμου με το Simulink

Προκειμένου να κινηθεί ο βραχίονας στον εικονικό κόσμο με τρόπο ανάλογο του πραγματικού, δημιουργήθηκε ένα μοντέλο προσομοίωσης με χρήση Simulink, με τρόπο ανάλογο αυτού που περιγράφηκε στην παράγραφο 3.6.



Σχήμα 4.13 Το Μοντέλο προσομοίωσης και το υποσύστημά του για τον εικονικό βραχίονα

Το μοντέλο προσομοίωσης που δημιουργήθηκε, έχει τη δυνατότητα να κινεί ρεαλιστικά, όλα τα τμήματα του εικονικού βραχίονα (βάση, αρθρώσεις και αρπάγη) (Σχήμα 4.13). Για να συνδεθεί ο εικονικός βραχίονας με το μοντέλο προσομοίωσης, χρησιμοποιείται ένα VR Sink διάγραμμα (Σχήμα 4.13, [A]), που έχει τη δυνατότητα εγγραφής δεδομένων, από το μοντέλο προσομοίωσης στον εικονικό κόσμο.

Ο έλεγχος των τμημάτων του εικονικού βραχίονα, γίνεται μέσω των Slider Gain (Σχήμα 4.13, [B]) διαγραμμάτων, που έχουν την ιδιότητα να πολλαπλασιάζουν το σήμα εισόδου τους, ανάλογα με την θέση που βρίσκεται η μπάρα ολίσθησής (Slider) τους και να το οδηγούν στην έξοδό τους.

Για τον έλεγχο περιστροφής, της βάσης και των αρθρώσεων του εικονικού βραχίονα (Σχήμα 4.13, [C]), ρυθμίστηκαν αντίστοιχα τα τέσσερα πρώτα Slider Gain, με όρια περιστροφής ίδια με του πραγματικού βραχίονα. Για τον έλεγχο της κίνησης, των άκρων της αρπάγης, το πέμπτο Slider Gain ρυθμίστηκε, με όριο από 0 έως 100.

Οι έξοδοι των Slider Gain οδηγούνται στις εισόδους του υποσυστήματος (Σχήμα 4.13, [D]) και οι έξοδοι του υποσυστήματος στις εισόδους του VR Sink. Οι τέσσερις πρώτες είσοδοι του υποσυστήματος είναι συνδεδεμένες με ένα VR Signal Expander (Σχήμα 4.13, [E]), που έχει την ιδιότητα να επεκτείνει το σήμα εισόδου και να οδηγεί το επεκταμένο σήμα στην έξοδο, δηλαδή στις τέσσερεις πρώτες αντίστοιχες εξόδους του υποσυστήματος. Είναι σημαντικό να αναφερθεί ότι η τιμή του σήματος για την περιστροφή ενός αντικειμένου, πρέπει να είναι σε μοίρες και να επεκταθεί, πριν οδηγηθεί στο VR Sink.

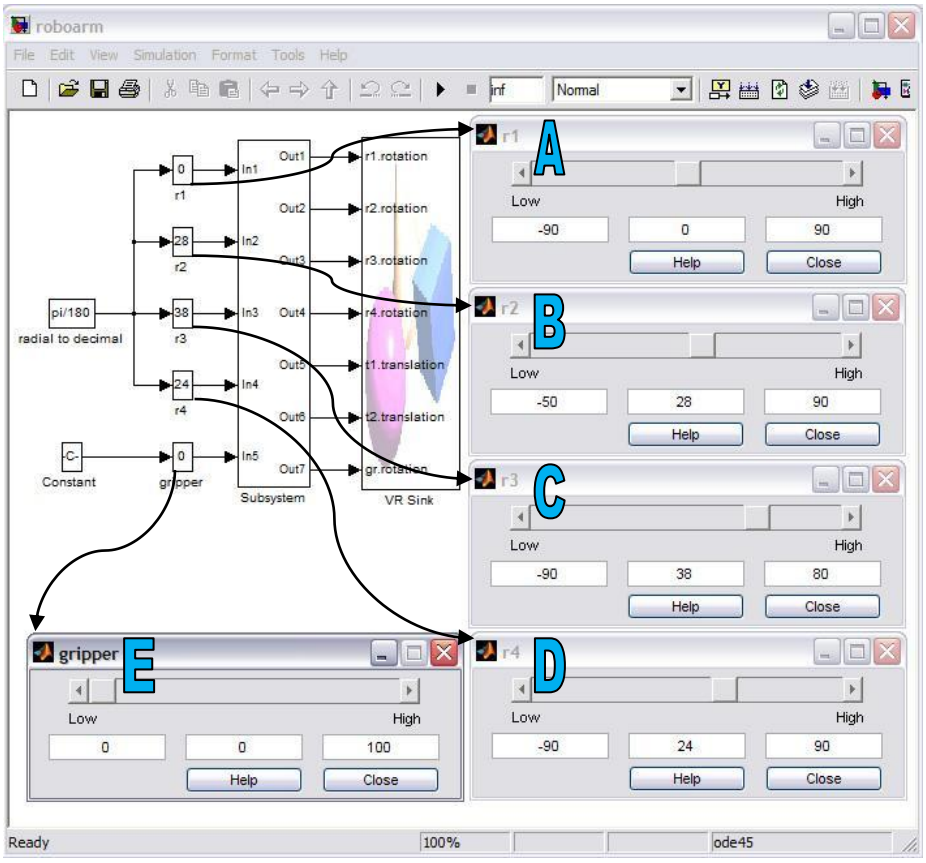
Η πέμπτη είσοδος του υποσυστήματος είναι συνδεδεμένη με μια σειρά λογικών διαγραμμάτων (Constant, Sum, Mux, Gain και VR Signal Expander) (Σχήμα 4.13, [F]), που είναι υπεύθυνα για την κίνηση των άκρων της αρπάγης και για την περιστροφή ενός μικρού τμήματος, για αισθητικούς λόγους και ρεαλιστικότερη κίνηση (Σχήμα 4.13, [G]). Οι έξοδοι αυτής της σειράς των λογικών διαγραμμάτων, είναι οι τρεις τελευταίες έξοδοι του υποσυστήματος. Είναι σημαντικό να αναφερθεί ότι το σήμα για την κίνηση ενός αντικειμένου, αποτελείται από τρεις τιμές, οι οποίες αναπαριστούν τις συντεταγμένες του αντικειμένου στον εικονικό κόσμο, ως προς τους άξονες X, Y και Z. Για αυτό τοποθετείται ένας πολυπλέκτης (Mux) τριών εισόδων, ο οποίος δέχεται τις τιμές αυτές και αφού κάνει την πολυπλεξία, οδηγεί το σήμα εξόδου του, στις εισόδους του VR Sink.

Για την κίνηση των άκρων της αρπάγης, έχουν τοποθετηθεί εσωτερικά του υποσυστήματος, πριν την πέμπτη και την έκτη έξοδο, από ένας πολυπλέκτης. Οι έξοδοι αυτοί συνδέονται με την πέμπτη και την έκτη αντίστοιχη είσοδο του VR Sink. Ο κάθε πολυπλέκτης έχει μια μόνο μεταβλητή τιμή, γιατί η κίνηση των άκρων της αρπάγης είναι ως προς ένα μόνο άξονα. Η τιμή αυτή ορίζεται με την πρόσθεση του σήματος εξόδου του πέμπτου Slider Gain και ενός συντελεστή για τη θέση του αντικειμένου σε σχέση με τον άξονα κατεύθυνσής του. Η πρόσθεση αυτή γίνεται μέσω του Sum διαγράμματος, που έχει την ιδιότητα να προσθέτει τις τιμές που δέχεται στις εισόδους του και να οδηγεί το άθροισμά τους στην έξοδό του. Οι

υπόλοιπες δυο τιμές του πολυπλέκτη είναι σταθερές και δίνονται μέσω διαγραμμάτων τύπου Constant.

Για την περιστροφή του μικρού τμήματος, έχει τοποθετηθεί ένα Signal Expander στο εσωτερικό του υποσυστήματος, πριν την έβδομη έξοδο. Η έξοδος αυτή είναι συνδεδεμένη με την έβδομη είσοδο του VR Sink. Για να οριστεί η ευαισθησία περιστροφής του μικρού αυτού τμήματος, έχει τοποθετηθεί μπροστά από το Signal Expander ένα Gain, που έχει την ιδιότητα να πολλαπλασιάζει το σήμα εισόδου του, με μια τιμή που έχει επιλεγεί. Το Gain αυτό έχει σαν είσοδο την έξοδο του πέμπτου Slider Gain και η τιμή που έχει επιλεγεί είναι μια μεταβλητή η οποία έχει υπολογιστεί εμπειρικά.

Για τον έλεγχο της κίνησης του βραχίονα χρησιμοποιούνται οι μπάρες ολίσθησης των πέντε Slider gain. Το r1 (Σχήμα 4.14, [A]) είναι υπεύθυνο για την περιστροφή της βάσης, το r2 (Σχήμα 4.14, [B]) για την περιστροφή της πρώτης άρθρωσης, το r3 (Σχήμα 4.14, [C]) για την περιστροφή της δεύτερης άρθρωσης, το r4 (Σχήμα 4.14, [D]) για την περιστροφή της τρίτης άρθρωσης και το gripper (Σχήμα 4.14, [E]) για την κίνηση του άκρου εργασίας.



Σχήμα 4.14 Το Μοντέλο προσομοίωσης και τα Slider για την κίνηση του εικονικού βραχίονα μέσω του Simulink

ΚΕΦΑΛΑΙΟ 5

Έλεγχος Βραχίονα από Υπολογιστή

5.1 Εισαγωγη

Στο κεφάλαιο αυτό, θα μελετηθεί και θα αναλυθεί, το πρόβλημα προσδιορισμού του τελικού σημείου δράσης, ως προς το σύστημα βάσης, δεδομένων των τιμών, των μεταβλητών και των αρθρώσεων (ευθύ κινηματικό), καθώς και το πρόβλημα εύρεσης των κατάλληλων τιμών, των μεταβλητών των αρθρώσεων, που επιτυγχάνουν, μια δεδομένη τοποθέτηση του ρομποτικού εργαλείου, ως προς ένα σύστημα αναφοράς (αντίστροφο κινηματικό) [26], για το ρομποτικό βραχίονα που έχει περιγραφεί αναλυτικά, στα προηγούμενα κεφάλαια. Με βάση την ανάλυση αυτή, αναπτύχθηκε και περιγράφετε ένα γραφικό περιβάλλον διεπαφής που επιτρέπει στον χρήστη:

i. Να μελετήσει το βραχίονα σε προσομοίωση για το ευθύ και το αντίστροφο κινηματικό πρόβλημα.
ii. Να ελέγξει ένα πραγματικό ρομποτικό βραχίονα για το ευθύ και το αντίστροφο κινηματικό πρόβλημα.

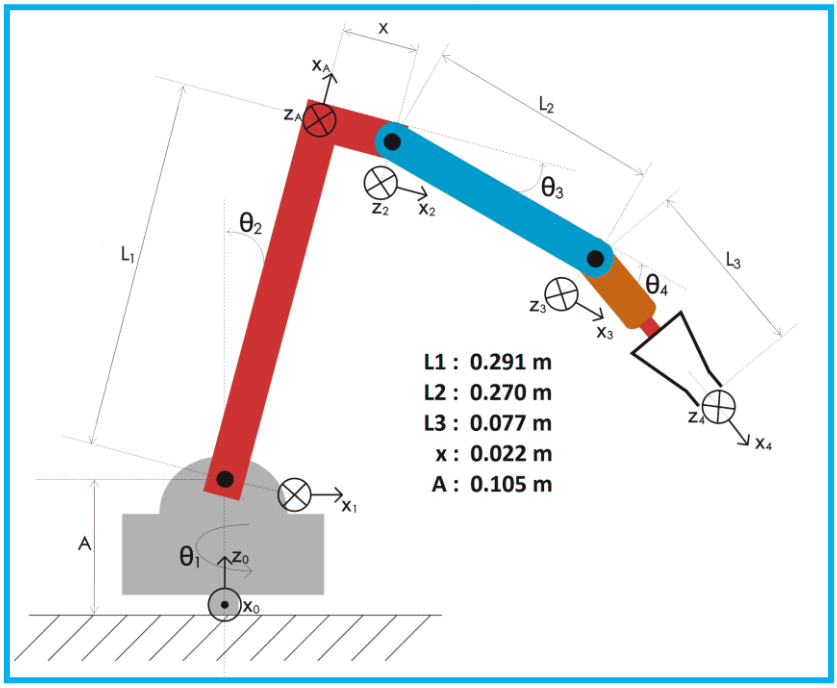
5.2 Ευθυ Κινηματικο Προβλημα

Το ευθύ κινηματικό πρόβλημα, συνίσταται στον υπολογισμό της θέσης και του προσανατολισμού του άκρου εργασίας, ως προς ένα σταθερό σύστημα αναφοράς με συγκεκριμένη τιμή. Η κίνηση και ο έλεγχος του βραχίονα και συγκεκριμένα του άκρου εργασίας, επιτυγχάνεται με την τοποθέτηση τοπικών συστημάτων συντεταγμένων σε κάθε άρθρωση του ρομπότ. Η αρχή κάθε τοπικού συστήματος συντεταγμένων, θεωρείται πακτωμένη στην αντίστοιχη άρθρωση και το σύστημα συντεταγμένων κινείται όπως κινείται ο επόμενος σύνδεσμος. Η διαδικασία με την οποία τοποθετούνται τα τοπικά συστήματα συντεταγμένων, είναι γνωστή ως αλγόριθμος Denavit-Hartenberg. Βάσει του αλγόριθμου αυτού, κάθε τοπικό σύστημα συντεταγμένων, είναι τοποθετημένο και προσανατολισμένο με συγκεκριμένο τρόπο σε σχέση με το προηγούμενο σύστημα συντεταγμένων. Για το λόγο αυτό, μετά την τοποθέτηση των τοπικών συστημάτων συντεταγμένων, κατασκευάζεται ο πίνακας τιμών, γνωστός ως πίνακας παραμέτρων Denavit-Hartenberg. Κάθε σύστημα συντεταγμένων, θεωρείται (σχετικώς) κινούμενο σε σχέση με το προηγούμενό του, το οποίο λαμβάνεται (στιγμιαία) ως ακίνητο σύστημα συντεταγμένων (στιγμιαίο σύστημα αναφοράς). Συνεπώς, για κάθε τοπικό σύστημα συντεταγμένων, υφίσταται ένας ομογενής πίνακας μετασχηματισμού (διαστάσεων 4×4) που μετασχηματίζει

(και ταυτίζει) το τρέχων τοπικό (κινούμενο) σύστημα, σε σχέση με το (αμέσως προηγούμενο) και (στιγμιαία) ακίνητο σύστημα συντεταγμένων. Ο μετασχηματισμός αυτός δίνεται από τη σχέση:

$$^{i-1}T_i = trsl\begin{pmatrix}0\\0\\r_i\end{pmatrix}.rotz(\theta_i).trsl\begin{pmatrix}d_i\\0\\0\end{pmatrix}.rotx(a_i)$$

όπου οι τιμές r_i, θ_i, d_i και a_i, λαμβάνονται από την i γραμμή του πίνακα τιμών Denavit-Hartenberg.



Σχήμα 5.1 Τοποθέτηση συστημάτων αξόνων του ρομποτικού βραχίονα

Ο συνολικός πίνακας ομογενούς μετασχηματισμού προκύπτει από το γινόμενο των μετασχηματισμών κάθε άρθρωσης και υπολογίζεται σύμφωνα με την παρακάτω σχέση:

$$^0T_m = {}^0T_1 * {}^1T_2 * \ldots * {}^{m-2}T_{m-1} * {}^{m-1}T_m$$

όπου m ο αριθμός της τελευταίας άρθρωσης.

$$trsl(\theta) = \begin{pmatrix}1 & 0 & 0 & x\\0 & 1 & 0 & y\\0 & 0 & 1 & z\\0 & 0 & 0 & 1\end{pmatrix}$$

$$rotz(\theta) = \begin{pmatrix}\cos(\theta) & -\sin(\theta) & 0 & 0\\\sin(\theta) & \cos(\theta) & 0 & 0\\0 & 0 & 1 & 0\\0 & 0 & 0 & 1\end{pmatrix}$$

$$rotx(\theta) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) & 0 \\ 0 & \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Πίνακες μετασχηματισμού κάθε άρθρωσης του Αρθρωτού Ρομποτικού Βραχίονα (Α.Ρ.Β.):

$${}^{0}T_{1} = \begin{pmatrix} -\sin(\theta 1) & 0 & -\cos(\theta 1) & 0 \\ \cos(\theta 1) & 0 & -\sin(\theta 1) & 0 \\ 0 & -1 & 0 & A \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$${}^{1}T_{A} = \begin{pmatrix} \sin(\theta 2) & \cos(\theta 2) & 0 & \sin(\theta 2) * L1 \\ -\cos(\theta 2) & \sin(\theta 2) & 0 & -\cos(\theta 2) * L1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$${}^{A}T_{2} = \begin{pmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & x \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

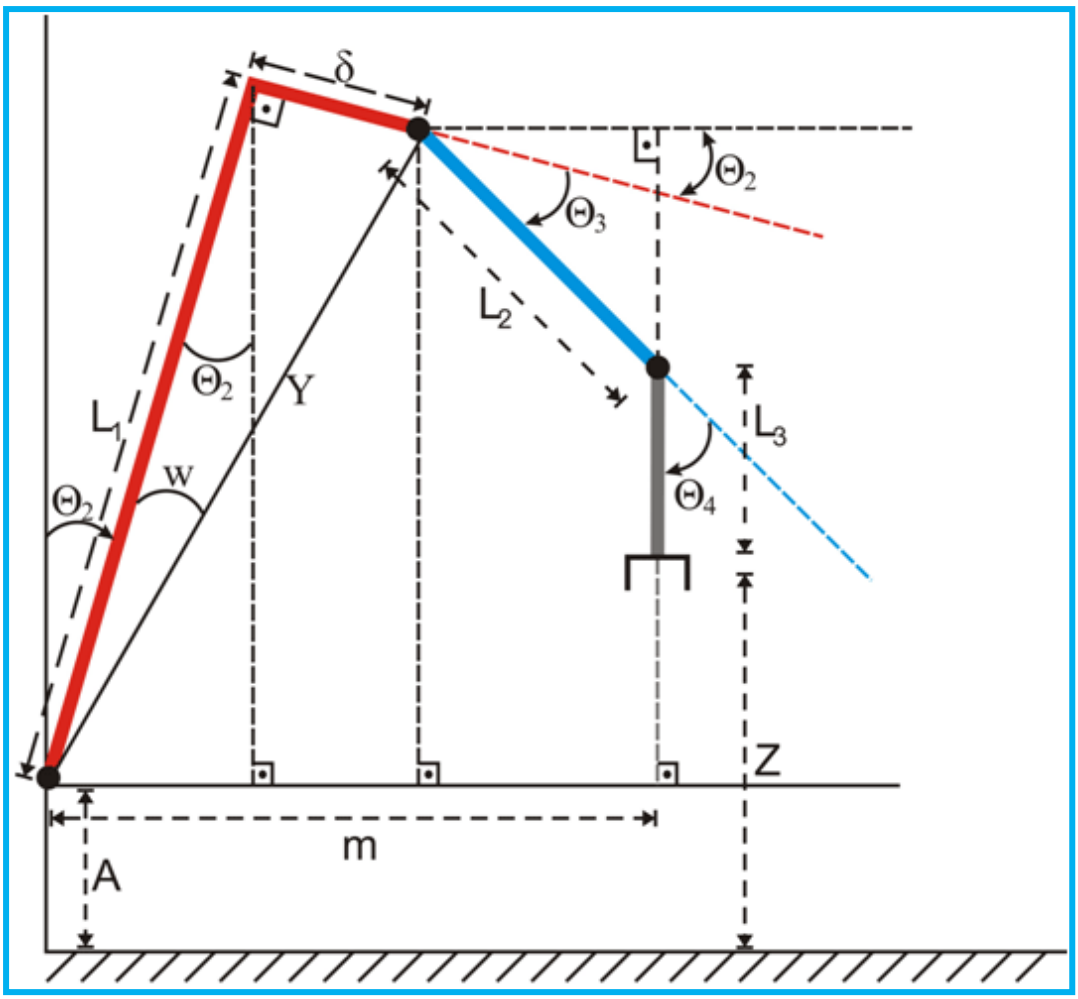
$${}^{2}T_{3} = \begin{pmatrix} \cos(\theta 3) & -\sin(\theta 3) & 0 & \cos(\theta 3) * L2 \\ \sin(\theta 3) & \cos(\theta 3) & 0 & \sin(\theta 3) * L2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$${}^{3}T_{4} = \begin{pmatrix} \cos(\theta 4) & -\sin(\theta 4) & 0 & \cos(\theta 4) * L3 \\ \sin(\theta 4) & \cos(\theta 4) & 0 & \sin(\theta 4) * L3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Συνολικός πίνακας ομογενούς μετασχηματισμού: $T = {}^{0}T_{1} * {}^{1}T_{A} * {}^{A}T_{2} * {}^{2}T_{3} * {}^{3}T_{4}$

5.3 ΑΝΤΙΣΤΡΟΦΟ ΚΙΝΗΜΑΤΙΚΟ ΠΡΟΒΛΗΜΑ

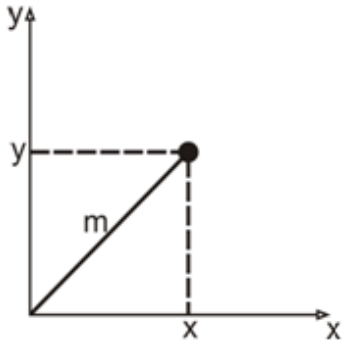
Το αντίστροφο κινηματικό πρόβλημα, συνίσταται στον υπολογισμό των τιμών των παραμέτρων των αρθρώσεων του βραχίονα, για δοσμένη θέση και προσανατολισμό του άκρου εργασίας. Ο υπολογισμός του έγινε με γεωμετρική μέθοδο, χρησιμοποιώντας τριγωνομετρικές εξισώσεις, για την ταχύτερη εκτέλεση του λογισμικού επίλυσης του αντίστροφου κινηματικού προβλήματος.



Σχήμα 5.2 Πλάγια όψη του βραχίονα

Τμήμα	Τιμή
A	0.1047 m
L_1	0.2910 m
L_2	0.2720 m
L_3	0.0770 m
δ	0.0220 m

Πίνακας 5.1 Γνωστές τιμές



Σχήμα 5.3 Κάτοψη του βραχίονα

Για τις γωνίες θ_2, θ_3, θ_4 έχουμε: $\theta_2 + 90^o + \theta_3 + \theta_4 = 180^o \Rightarrow \theta_2 + \theta_3 + \theta_4 = 90^o$

Για το Y έχουμε: $Y = \sqrt{L_1^2 + \delta^2} = 0.2918m$

$$Y\cos(w) = L_1 \Rightarrow w = a\cos(L_1 / Y) = 4.3234^o$$

Για το m έχουμε: $m = \sqrt{x^2 + y^2}$

Όμως: $Y\sin(\theta_2 + w) + L_2\cos(\theta_2 + \theta_3) = m$

Για κάθε γνωστό x, y

Για το κατακόρυφο τμήμα έχουμε:

$$A + Y\cos(\theta_2 + w) - L_2\sin(\theta_2 + \theta_3) - L_3 = Z \Rightarrow$$
$$Y\cos(\theta_2 + w) - L_2\sin(\theta_2 + \theta_3) = Z + L_3 - A = n$$

Όπου n γνωστό για κάθε Z, τα (L_3 και A δίνονται στον πίνακα 3.3.1)

Σπάμε τα ημίτονα και συνημίτονα των αθροισμάτων:

$$\left.\begin{aligned} Y\sin(\theta_2 + w) + L_2\cos(\theta_2 + \theta_3) = m \\ Y\cos(\theta_2 + w) - L_2\sin(\theta_2 + \theta_3) = n \end{aligned}\right\} \Rightarrow$$

Έχοντας τις εκφράσεις m και n, τις υψώνουμε στο τετράγωνο και μετά τις προσθέτουμε για να βρούμε το θ_3:

$$\begin{pmatrix} Y^2\left[\sin(\theta_2+w)\right]^2 \\ + \\ Y^2\left[\cos(\theta_2+w)\right]^2 \end{pmatrix}\begin{pmatrix} +L_2\left[\cos(\theta_2+\theta_3)\right]^2 \\ + \\ +L_2\left[\sin(\theta_2+\theta_3)\right]^2 \end{pmatrix}\begin{pmatrix} +2\cdot Y\cdot L_2\sin(\theta_2+w)\cdot\cos(\theta_2+\theta_3) \\ + \\ -2\cdot Y\cdot L_2\cos(\theta_2+w)\cdot\sin(\theta_2+\theta_3) \end{pmatrix} = m^2+n^2$$

$$\Rightarrow Y^2+L_2^2+2\cdot Y\cdot L_2\cdot\left[\sin(\theta_2+w)\cdot\cos(\theta_2+\theta_3)-\cos(\theta_2+w)\cdot\sin(\theta_2+\theta_3)\right]=m^2+n^2\Rightarrow$$

$$Y^2+L_2^2+2\cdot Y\cdot L_2\cdot\sin(w-\theta_3)=m^2+n^2\Rightarrow\sin(w-\theta_3)=\frac{m^2+n^2-Y^2-L_2^2}{2\cdot Y\cdot L_2}\Rightarrow$$

$$w-\theta_3=a\sin\left[\frac{m^2+n^2-Y^2-L_2^2}{2\cdot Y\cdot L_2}\right]\Rightarrow\theta_3=w-a\sin\left[\frac{m^2+n^2-Y^2-L_2^2}{2\cdot Y\cdot L_2}\right]$$

Συνεχίζουμε για τη γωνία θ_2:

$$\left.\begin{aligned} Y\cdot S_2\cdot C_w+Y\cdot C_2\cdot S_w+L_2\cdot C_2\cdot C_3-L_2\cdot S_2\cdot S_3=m \\ Y\cdot C_2\cdot C_w-Y\cdot S_2\cdot S_w-L_2\cdot S_2\cdot C_3-L_2\cdot C_2\cdot S_3=n \end{aligned}\right\}\Rightarrow$$

$$\left.\begin{aligned} S_2(Y\cdot C_w-L_2\cdot S_3)+C_2(Y\cdot S_w+L_2\cdot C_3)=m \\ S_2(-Y\cdot S_w-L_2\cdot C_3)+C_2(Y\cdot C_w+L_2\cdot S_3)=n \end{aligned}\right\}\Rightarrow$$

Όπου $C_2=\cos(\theta_2), S_w=\sin(x)$ κτλ.

Λύνουμε την πρώτη ως προς C_2 και αντικαθιστούμε στη δεύτερη:

$$C_2=\frac{m-S_2(Y\cdot Cw-L_2\cdot S_3)}{Y\cdot S_w+L_2\cdot C_3}$$

$$S_2\cdot(-Y\cdot S_w-L_2\cdot C_3)+\frac{m-S_2\cdot(Y\cdot C_w-L_2\cdot S_3)}{Y\cdot S_w+L_2\cdot C_3}\cdot(Y\cdot C_w-L_2\cdot S_3)=n\Rightarrow$$

$$-S_2\cdot(Y\cdot S_w+L_2\cdot C_3)^2+[m-S_2\cdot(Y\cdot C_w-L_2\cdot S_3)]\cdot(Y\cdot C_w-L_2\cdot S_3)=n\cdot(Y\cdot S_w+L_3\cdot C_3)\Rightarrow$$

$$-S_2\cdot(Y\cdot S_w+L_2\cdot C_3)^2+m\cdot(Y\cdot C_w-L_2\cdot S_3)-S_2\cdot(Y\cdot C_w-L_2\cdot S_3)^2=n\cdot(Y\cdot S_w+L_2\cdot C_3)\Rightarrow$$

$$m\cdot\underbrace{(Y\cdot C_w-L_2\cdot S_3)}_{k}-n\cdot\underbrace{(Y\cdot S_w+L_2\cdot C_3)}_{L}=S_2\cdot[\overbrace{(Y\cdot S_w+L_2\cdot C_3)}^{L}{}^2+\overbrace{(Y\cdot C_w-L_2\cdot S_3)}^{k}{}^2]\Rightarrow$$

$$S_2 = \frac{m \cdot k - n \cdot L}{L^2 + k^2} \Rightarrow \theta_2 = a\sin\left(\frac{m \cdot k - n \cdot L}{k^2 + L^2}\right)$$

Για τη γωνία Θ_1 έχουμε:

Αν $x = 0 \Rightarrow \theta_1 = 0^o$

Αν $y = 0$ και $x > 0 \Rightarrow \theta_1 = -90^o$

Αν $y = 0$ και $x < 0 \Rightarrow \theta_1 = +90^o$

Για όλες τις άλλες περιπτώσεις: $\theta_1 = -a\tan\left(\frac{x}{y}\right)$

Στο σημείο αυτό έχουμε βρει όλες τις απαραίτητες γωνίες για τη λύση του αντίστροφου κινηματικού προβλήματος.

5.4 ΠΡΟΣΟΜΟΙΩΣΗ ΚΑΙ ΕΛΕΓΧΟΣ ΒΡΑΧΙΟΝΑ ΑΠΟ ΥΠΟΛΟΓΙΣΤΗ

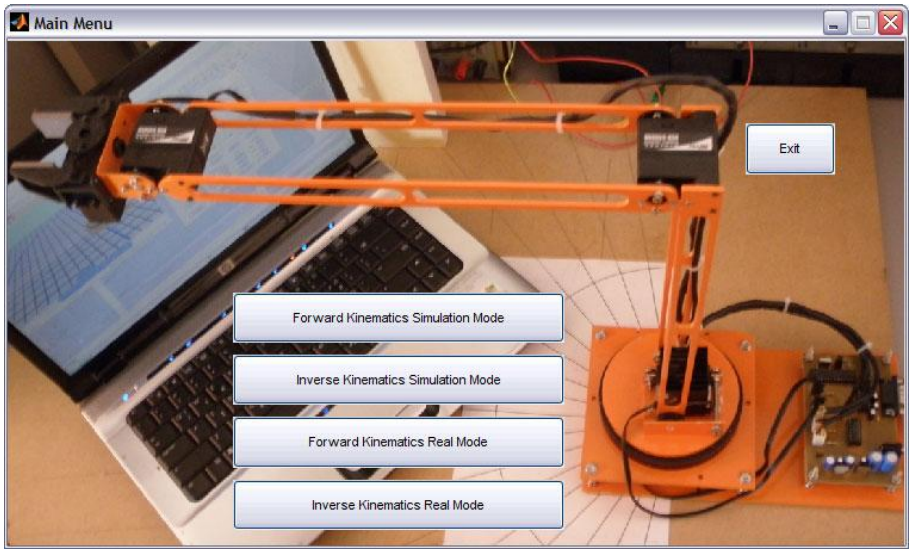
Για να μελετηθεί ο τρόπος λειτουργίας του βραχίονα, αναπτύχθηκε γραφικό περιβάλλον διεπαφής (Graphical User Interface - GUI), με χρήση του λογισμικού MATLAB. Το γραφικό περιβάλλον επιτρέπει στο χρήστη να ελέγχει το βραχίονα σε πραγματικές συνθήκες ή σε συνθήκες προσομοίωσης, χρησιμοποιώντας το ευθύ ή το αντίστροφο κινηματικό πρόβλημα.

5.4.1 Κεντρικό Μενού Επιλογής σε Γραφικό Περιβάλλον Διεπαφής

Για την επιλογή προσομοίωσης ή πραγματικής λειτουργίας, αρχικά, αναπτύχθηκε ένα γραφικό περιβάλλον διεπαφής όπως αυτό παρουσιάζεται στο Σχήμα 5.4.

Για την επιλογή προσομοίωσης του ευθέως κινηματικού προβλήματος, επιλέγεται το "Forward Kinematics Simulation Mode" και για την προσομοίωση του αντίστροφου κινηματικού προβλήματος, επιλέγεται το "Inverse Kinematics Simulation Mode". Οι αντίστοιχες επιλογές υπάρχουν και για την πραγματική λειτουργία.

Για την έξοδο από το κεντρικό μενού επιλογών, επιλέγεται το κουμπί "Exit".



Σχήμα 5.4 Κεντρική γραφική διεπαφή

5.4.2 Ανάπτυξη γραφικού περιβάλλοντος διεπαφής για την μελέτη του ευθέως κινηματικού προβλήματος σε περιβάλλον προσομοίωσης

Για την μελέτη του ευθέως κινηματικού προβλήματος σε περιβάλλον προσομοίωσης, αναπτύχθηκε το γραφικό περιβάλλον διεπαφής που παρουσιάζεται παρακάτω (Σχήμα 5.5).

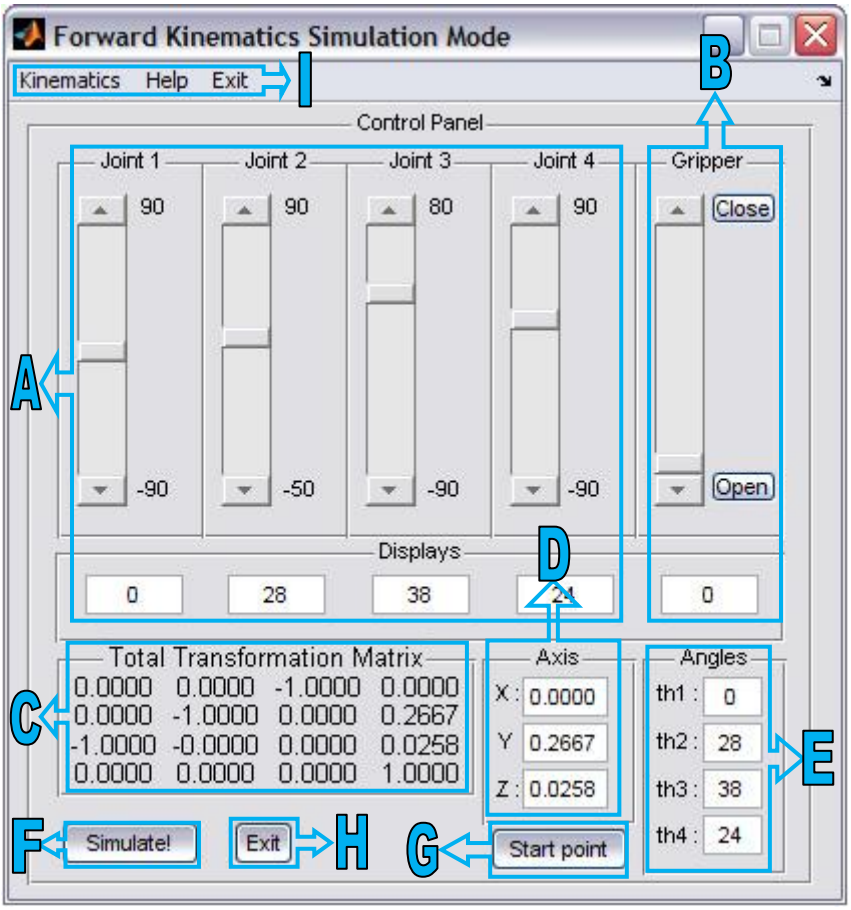
Με την χρήση του γραφικού περιβάλλοντος διεπαφής, ο χρήστης έχει την δυνατότητα να εισάγει τις τιμές των αρθρώσεων και να υπολογίζει τις συντεταγμένες του άκρου εργασίας. Ακόμα, υπολογίζεται ο συνολικός πίνακας ομογενούς μετασχηματισμού.

Το γραφικό περιβάλλον διεπαφής για την μελέτη του ευθέως κινηματικού προβλήματος, σε περιβάλλον προσομοίωσης, αναλύεται παρακάτω (Πίνακας 5.2):

	Ανάλυση γραφικού περιβάλλοντος διεπαφής
A	Στην περιοχή [A], ο χρήστης έχει τη δυνατότητα να πληκτρολογεί τις επιθυμητές τιμές των γωνιών, για την κίνηση του εικονικού βραχίονα ή μπορεί να δώσει τις τιμές αυτές, από τις μπάρες ολίσθησης (sliders).
B	Στην περιοχή [B], ο χρήστης έχει τη δυνατότητα να ελέγξει το άνοιγμα ή το κλείσιμο της αρπάγης του άκρου εργασίας του εικονικού βραχίονα, πληκτρολογώντας την επιθυμητή τιμή (0-100) ή κινώντας την μπάρα ολίσθησης. Ακόμα, διαθέτει κουμπιά “Open” και “Close” για την πλήρη έκταση και το κλείσιμο αντίστοιχα.
C	Στην περιοχή [C], εμφανίζεται ο συνολικός πίνακας ομογενούς μετασχηματισμού, του εικονικού βραχίονα.

D	Στην περιοχή [D], εμφανίζονται οι τιμές των συντεταγμένων του εικονικού βραχίονα.
E	Στην περιοχή [E], εμφανίζονται οι τιμές των γωνιών του εικονικού βραχίονα.
F	Στην περιοχή [F], ο χρήστης έχει την δυνατότητα μέσω του κουμπιού "Simulate!", να δώσει εντολή για την προσομοίωση του εικονικού βραχίονα.
G	Στην περιοχή [G], ο χρήστης έχει την δυνατότητα μέσω του κουμπιού "Start point", να δώσει εντολή στον εικονικού βραχίονα να επιστρέψει στην αρχική του θέση (Home position).
H	Στην περιοχή [H], ο χρήστης έχει την δυνατότητα μέσω του κουμπιού "Exit", να εξέλθει από τη γραφική διεπαφή χρήστη.
I	Τέλος, στην περιοχή [I], υπάρχει μενού για προσομοίωση ή πραγματική κίνηση του βραχίονα, επιλέγοντας το ευθύ ή το ανάστροφο κινηματικό πρόβλημα (Kinematics), μενού πληροφοριών (Help) και μενού εξόδου.

Πίνακας 5.2 Ανάλυση γραφικού περιβάλλοντος διεπαφής

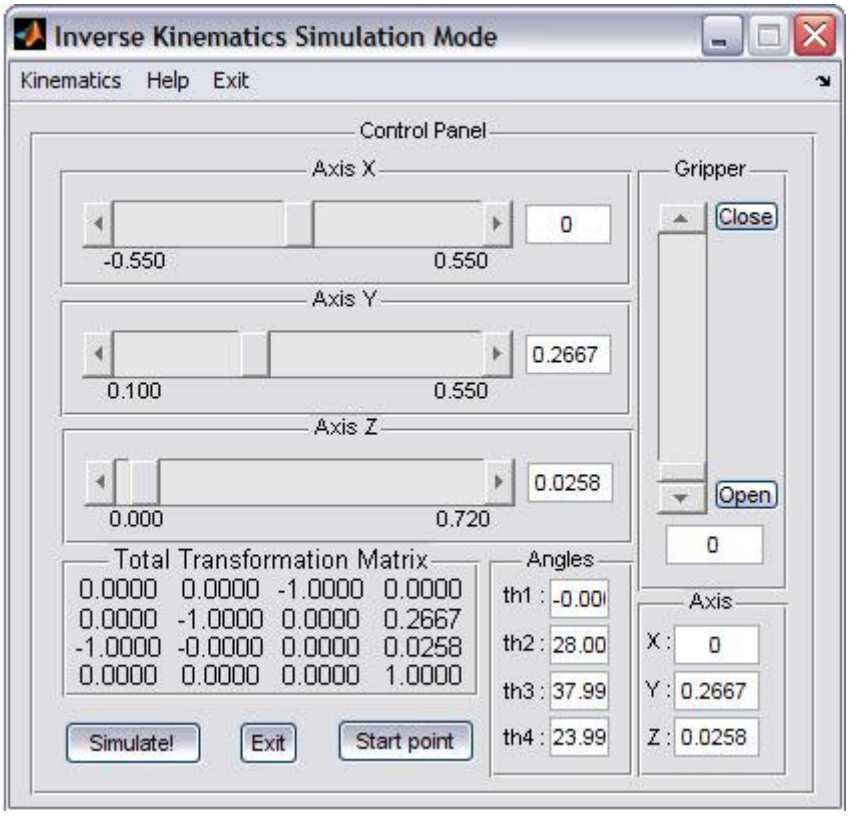


Σχήμα 5.5 Γραφική διεπαφή χρήστη προσομοίωσης ευθέως κινηματικού προβλήματος

5.4.3 Ανάπτυξη γραφικού περιβάλλοντος διεπαφής για την μελέτη του αντίστροφου κινηματικού προβλήματος σε περιβάλλον προσομοίωσης

Για την μελέτη του αντίστροφου κινηματικού προβλήματος σε περιβάλλον προσομοίωσης, αναπτύχθηκε το γραφικό περιβάλλον διεπαφής που παρουσιάζεται παρακάτω (Σχήμα 5.6).

Με την χρήση του γραφικού περιβάλλοντος διεπαφής, ο χρήστης έχει τη δυνατότητα να εισάγει τις τιμές των συντεταγμένων του άκρου εργασίας. Για κάθε τιμή συντεταγμένων (X, Y και Z), υπολογίζονται οι τιμές των αρθρώσεων, του εικονικού βραχίονα και ο συνολικός πίνακας ομογενούς μετασχηματισμού.



Σχήμα 5.6 Γραφική διεπαφή χρήστη προσομοίωσης αντίστροφου κινηματικού προβλήματος

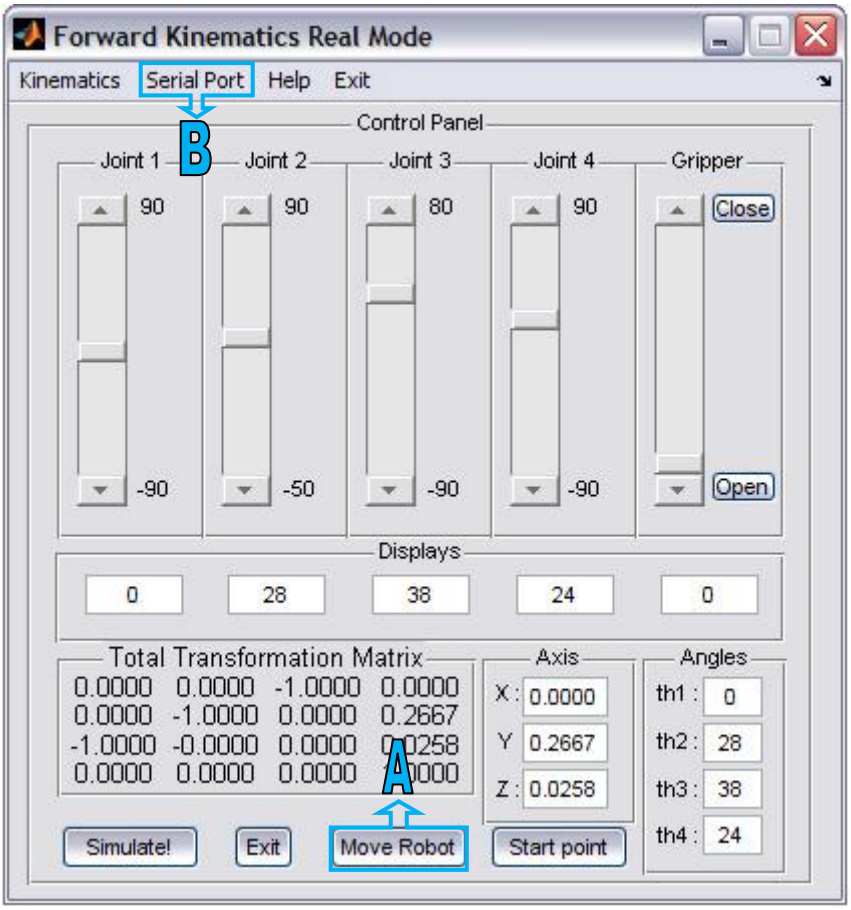
5.4.4 Ανάπτυξη γραφικού περιβάλλοντος διεπαφής για την μελέτη του ευθέως κινηματικού προβλήματος σε πραγματικό περιβάλλον

Για την μελέτη του ευθέως κινηματικού προβλήματος σε πραγματικό περιβάλλον, αναπτύχθηκε το γραφικό περιβάλλον διεπαφής, που παρουσιάζεται στο Σχήμα 5.7.

Οι διαφορές του γραφικού περιβάλλοντος διεπαφής, για την μελέτη του ευθέως κινηματικού προβλήματος σε πραγματικό περιβάλλον, από ότι σε περιβάλλον προσομοίωσης, είναι η προσθήκη του κουμπιού “Move Robot” (Σχήμα 5.5, [A]) και η προσθήκη στο μενού, της επιλογής “Serial Port” (Σχήμα 5.7, [B]) για την επιλογή της σειριακής θύρας.

Σε αυτό το περιβάλλον διεπαφής, ο χρήστης εισάγει τις τιμές των γωνιών που επιθυμεί και πατώντας το κουμπί “Move Robot” γίνονται οι απαραίτητες ενέργειες για την πραγματοποίηση της κίνησης του πραγματικού βραχίονα. Δηλαδή, ορίζεται μια σειριακή θύρα στην συνέχεια ενεργοποιείται και μέσω αυτής, δίνονται οι εντολές κίνησης του βραχίονα. Όταν ολοκληρωθεί η κίνηση του βραχίονα, η σειριακή απενεργοποιείται και διαγράφεται μέχρι την επόμενη φορά που θα γίνει επιλογή κίνησης.

Ο χρήστης έχει τη δυνατότητα παρατήρησης της λειτουργίας του βραχίονα μέσω του λογισμικού αναπαράστασης σε VRML. Ελέγχει τη θέση στην οποία θα κινηθεί ο βραχίονας και αν αυτή είναι η επιθυμητή τότε επιλέγει την εντολή κίνησης “Move Robot”.

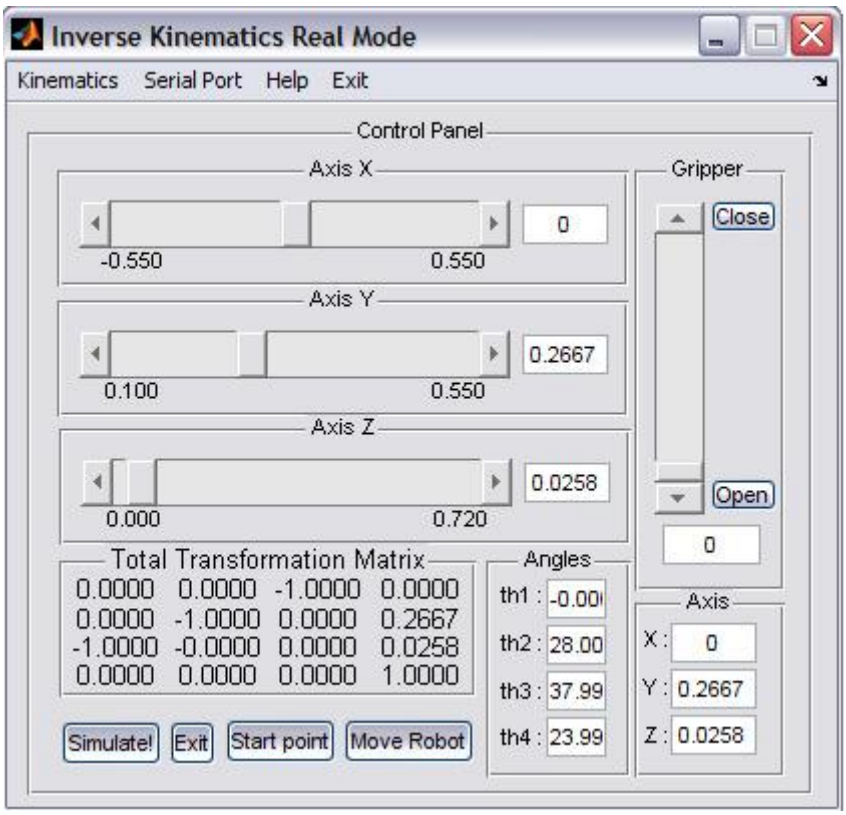


Σχήμα 5.7 Γραφική διεπαφή χρήστη ευθέως κινηματικού προβλήματος

5.4.5 Ανάπτυξη γραφικού περιβάλλοντος διεπαφής για την μελέτη του αντίστροφου κινηματικού προβλήματος σε πραγματικό περιβάλλον

Για την μελέτη του αντίστροφου κινηματικού προβλήματος σε πραγματικό περιβάλλον, αναπτύχθηκε το γραφικό περιβάλλον διεπαφής που παρουσιάζεται παρακάτω (Σχήμα 5.8).

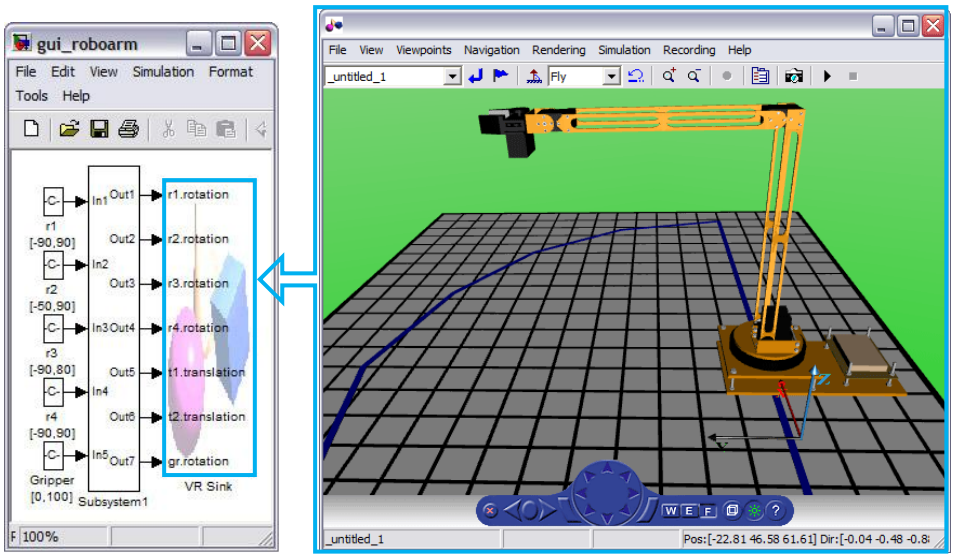
Σε αυτό το περιβάλλον διεπαφής, ο χρήστης εισάγει τις τιμές των συντεταγμένων που επιθυμεί να κινηθεί το άκρο εργασίας του πραγματικού βραχίονα και πατώντας το κουμπί “Move Robot”, γίνονται οι απαραίτητες ενέργειες, για την πραγματοποίηση της κίνησής του, αντίστοιχα με την διαδικασία για το ευθύ κινηματικό πρόβλημα.



Σχήμα 5.8 Γραφική διεπαφή χρήστη αντίστροφου κινηματικού προβλήματος

5.5 ΠΑΡΑΔΕΙΓΜΑΤΑ

Για να επιδειχθεί η σωστή λειτουργία του λογισμικού που αναπτύχθηκε, εκτελέστηκαν διάφορα πειράματα που παρουσιάζονται αναλυτικά στην συνέχεια.

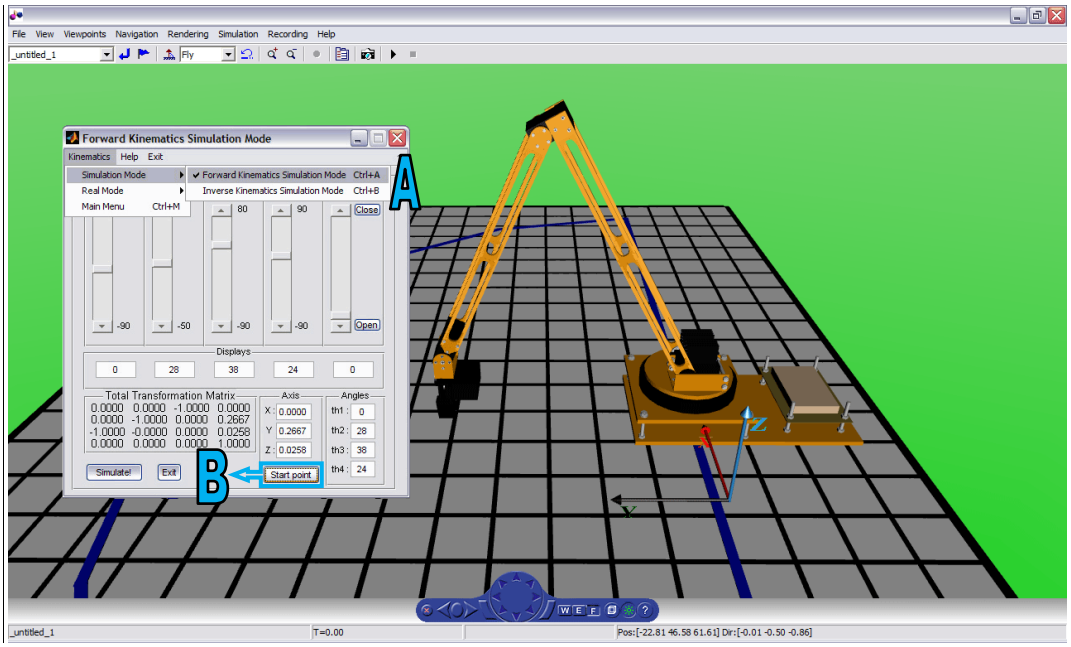


Σχήμα 5.7 Το Simulink Model και ο εικονικός κόσμος

5.5.1 Πειραμα 1: Μελέτη του ευθέως κινηματικού προβλήματος σε περιβάλλον προσομοίωσης

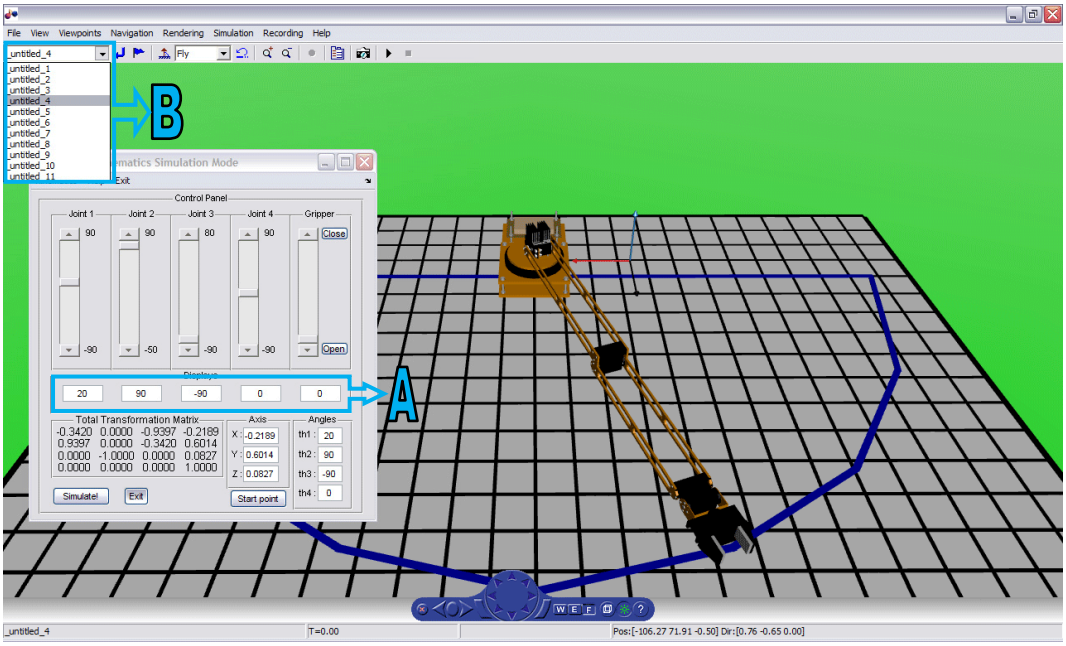
Για την προσομοίωση του ευθέως κινηματικού προβλήματος, πρέπει να επιλεγεί το "Forward Kinematics Simulation Mode" είτε από την κεντρική διεπαφή (Main Menu) είτε από το Kinematics→Simulation Mode→Forward Kinematics Simulation Mode του μενού της ήδη υπάρχουσας διεπαφής (Σχήμα 5.10, [A]).

Επιλέγοντας το Start point (Σχήμα 5.10, [B]), ο εικονικός βραχίονας πηγαίνει στη θέση εκκίνησης που του έχει οριστεί.



Σχήμα 5.11 Προσομοίωση του ευθέως κινηματικού με τον εικονικό βραχίονα στην αρχική του θέση

Αλλάζοντας τις τιμές των αρθρώσεων Joint1, Joint2, Joint3 και Joint4 σε 20,90,-90 και 0 αντίστοιχα (Σχήμα 5.11, [A]), ο εικονικός βραχίονας οδηγείται σε μια άλλη θέση. Για να αλλάξει το σημείο παρατήρησης του εικονικού βραχίονα πρέπει να επιλεγεί διαφορετικό Viewpoint (Σχήμα 5.11, [B]). Στο πείραμα αυτό επιλέχθηκε το τέταρτο Viewpoint.

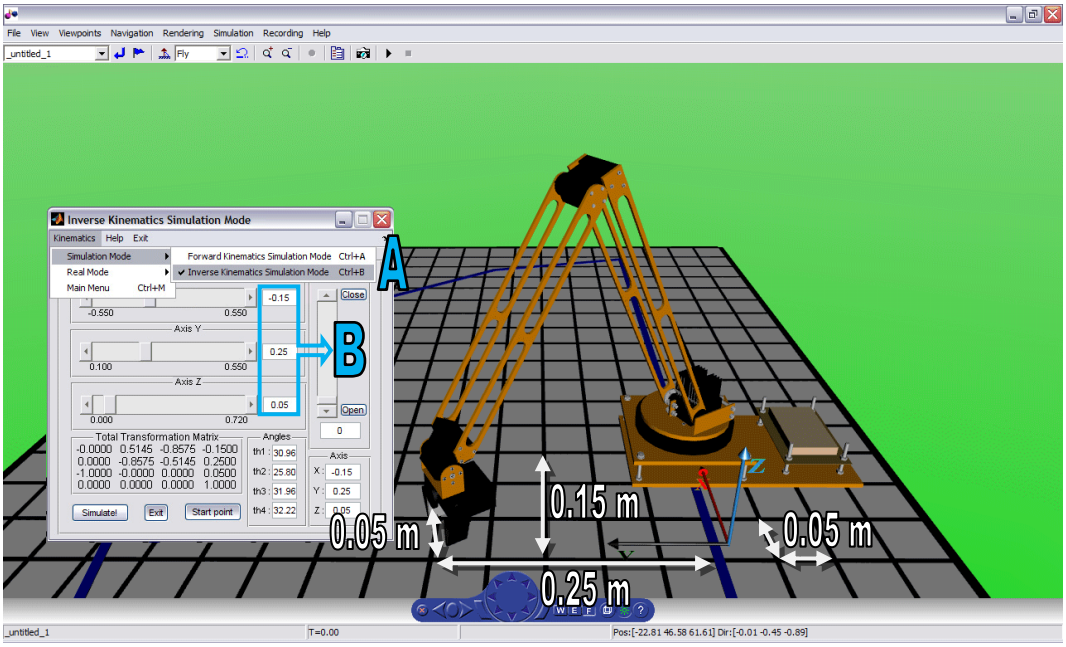


Σχήμα 5.11 Προσομοίωση του ευθέως κινηματικού με διαφορετικό σημείο παρατήρησης

5.5.2 Πείραμα 2: Μελέτη του αντίστροφου κινηματικού προβλήματος σε περιβάλλον προσομοίωσης

Για την προσομοίωση του αντίστροφου κινηματικού προβλήματος πρέπει να επιλεγεί το "Inverse Kinematics Simulation Mode" είτε από την κεντρική διεπαφή (Main Menu) είτε από το Kinematics→Simulation Mode→Inverse Kinematics Simulation Mode του μενού της ήδη υπάρχουσας διεπαφής (Σχήμα 5.12, [A]).

Αλλάζοντας τις τιμές των συντεταγμένων κατά άξονα X, Y και Z σε -0.15, 0.25 και 0.05 αντίστοιχα (Σχήμα 5.12, [B]), ο εικονικός βραχίονας οδηγείται σε μια άλλη θέση. Για την επιβεβαίωση της κίνησης του άκρου εργασίας του εικονικού βραχίονα σύμφωνα με τις συντεταγμένες που δόθηκαν, μπορεί να χρησιμοποιηθεί το δάπεδο το οποίο αποτελείται από τετράγωνα των 0.05 m.

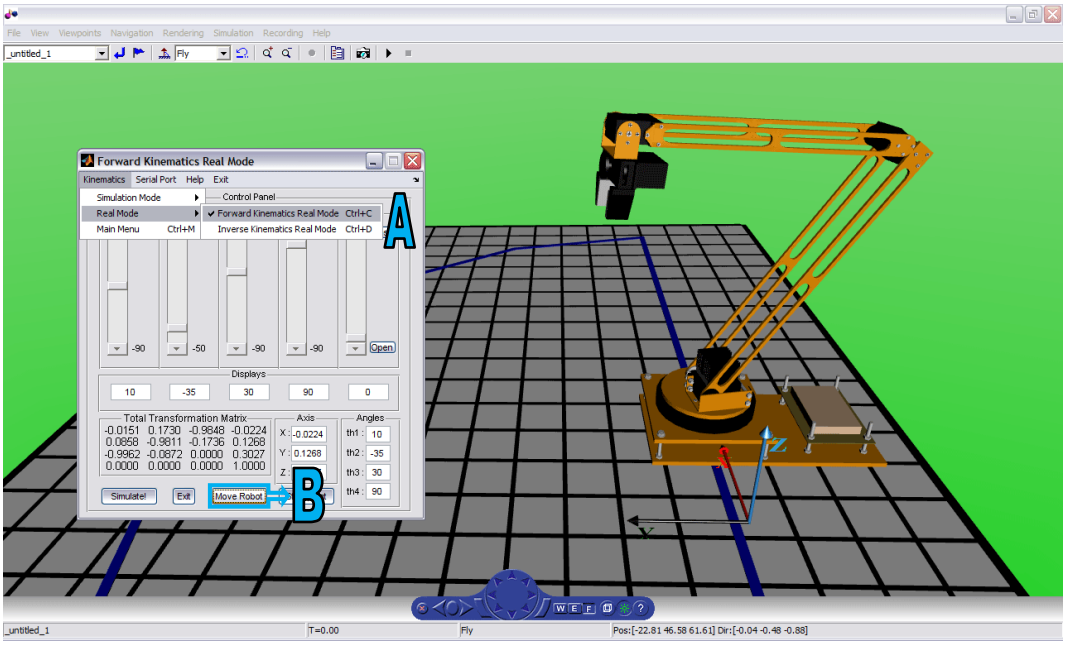


Σχήμα 5.12 Προσομοίωση του αντίστροφου κινηματικού προβλήματος

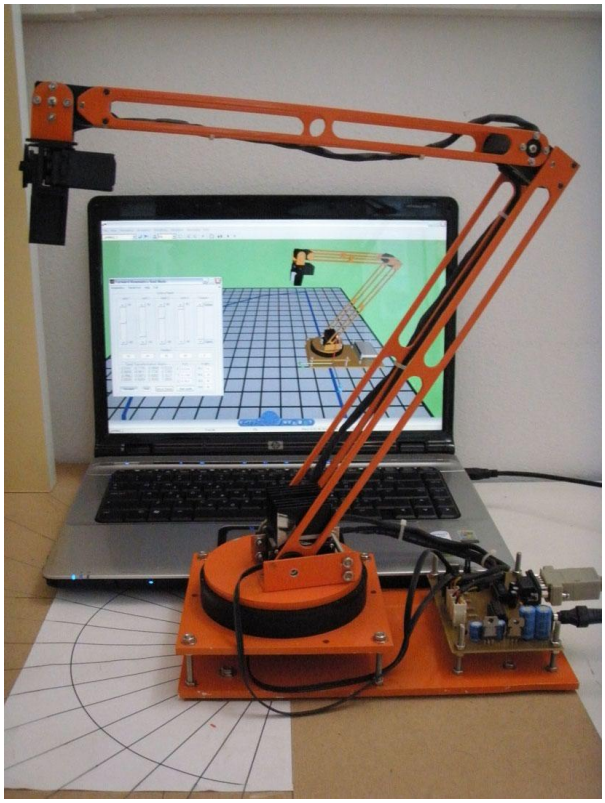
5.5.3 Πείραμα 3: Μελέτη του ευθέως κινηματικού προβλήματος σε πραγματικό περιβάλλον

Για την πραγματική κίνηση του βραχίονα χρησιμοποιώντας το ευθύ κινηματικό πρόβλημα πρέπει να επιλεγεί το "Forward Kinematics Real Mode" είτε από την κεντρική διεπαφή (Main Menu) είτε από το Kinematics→Real Mode→Forward Kinematics Real του μενού της ήδη υπάρχουσας διεπαφής (Σχήμα 5.13, [A]).

Αλλάζοντας τις τιμές των αρθρώσεων Joint1, Joint2, Joint3 και Joint4 σε 10,-35,30 και 90 αντίστοιχα, ο εικονικός βραχίονας οδηγείται σε μια άλλη θέση (Σχήμα 5.13). Για να κινηθεί και ο πραγματικός βραχίονας (Σχήμα 5.14) πρέπει να επιλεγεί το Move Robot (Σχήμα 5.13, [B]).

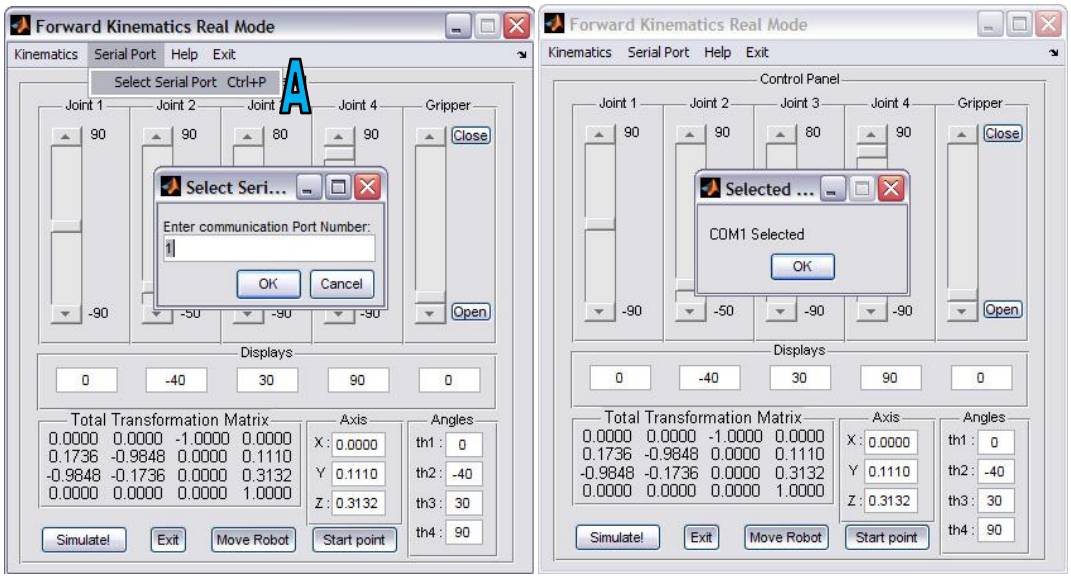


Σχήμα 5.13 Πραγματική κίνηση μέσω του ευθέως κινηματικού προβλήματος



Σχήμα 5.14 Κίνηση του πραγματικού βραχίονα

Πριν την πρώτη κίνηση του πραγματικού βραχίονα, πρέπει να γίνεται η επιλογή της σειριακής θύρας από το μενού της διεπαφής Serial Port→Select Serial Port (Σχήμα 5.13, [A]).

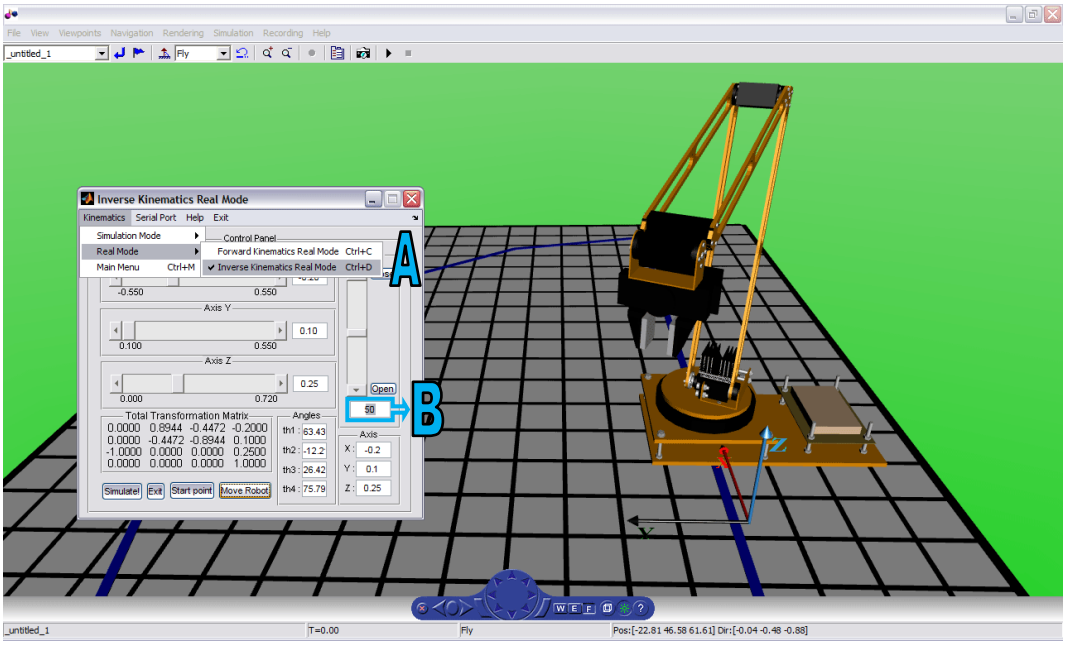


Σχήμα 5.15 Επιλογή σειριακής θύρας

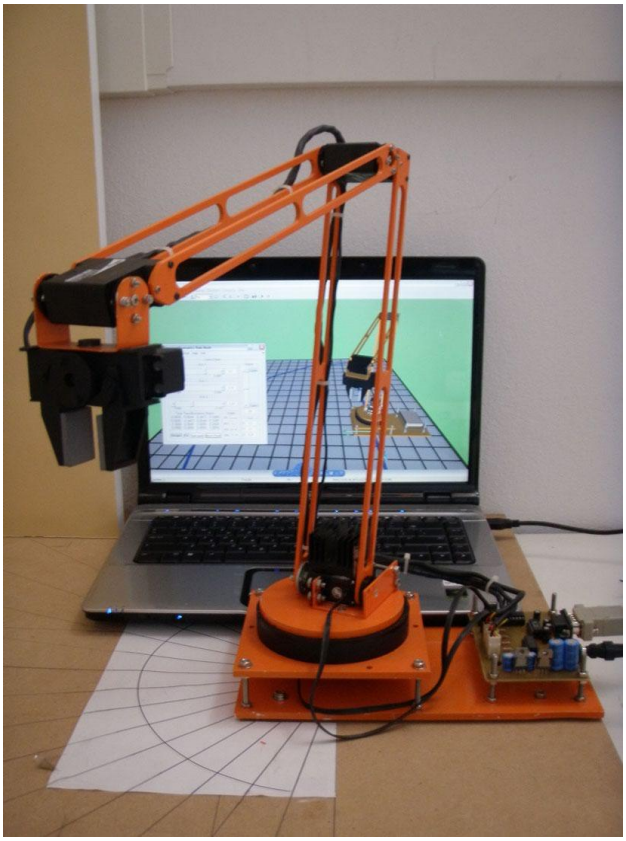
5.5.4 Πείραμα 4: Μελέτη του αντίστροφου κινηματικού προβλήματος σε πραγματικό περιβάλλον

Για την πραγματική κίνηση του βραχίονα χρησιμοποιώντας το αντίστροφο κινηματικό πρόβλημα πρέπει να επιλεγεί το "Inverse Kinematics Real Mode" είτε από την κεντρική διεπαφή (Main Menu) είτε από το Kinematics→Real Mode→Inverse Kinematics Real Mode του μενού της ήδη υπάρχουσας διεπαφής (Σχήμα 5.16, [A]).

Αλλάζοντας τις τιμές των συντεταγμένων κατά άξονα X, Y και Z σε -0.20, 0.10 και 0.25 αντίστοιχα, ο εικονικός βραχίονας οδηγείται σε μια άλλη θέση (Σχήμα 5.16). Ακόμα, για την κίνηση των άκρων της αρπάγης μπορούν να δοθούν τιμές από το 0 έως το 100, όπου στην τιμή 0 τα άκρα της αρπάγης είναι ανοιχτά και στην τιμή 100 τα άκρα είναι κλειστά. Συγκεκριμένα, επιλέχθηκε η τιμή 50 (Σχήμα 5.16, [B]). Για να κινηθεί και ο πραγματικός βραχίονας (Σχήμα 5.17) πρέπει να επιλεγεί το Move Robot.



Σχήμα 5.16 Πραγματική κίνηση μέσω του ευθέως κινηματικού προβλήματος



Σχήμα 5.17 Κίνηση του πραγματικού βραχίονα

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Elekit ROBOT ARM,

(url:http://www.robotstoreuk.com/ROBOTS-/pages/Robot-Arm/arm-details-01.htm)

[2] Εμμανουήλ Σαϊτάκη, "Μελέτη και κατασκευή ρομποτικού βραχίονα με ψηφιακά συστήματα", Πτυχιακή εργασία, Τ.Ε.Ι. Κοζάνης, 2003

[3] EduBot 100,

(url:http://www.robotica.co.uk/robotica/ramc/products/robotic_arms/edubot100.htm)

[4] EduBot 100 GP,

(url:http://www.robotica.co.uk/robotica/images/pdfs/edubot100_specs_gp.pdf)

[5] Scorbot-ER 9Pro,

(url:http://www.intelitek.com/admin/Products/uploads/File/File1_18.pdf)

[6] Lynx 5,

(url:http://www.lynxmotion.com/Category.aspx?CategoryID=2)

[7]Gridbots Robotic Arm,

(url:http://www.gridbots.com/roboticarm_brochure.pdf)

[8] Γκανάς Γιάννης, Βαρτζής Τριαντάφυλλος, "Ανάπτυξη ρομποτικών συστημάτων με το πακέτο εφαρμογής Webot", Πτυχιακή εργασία, Α.Τ.Ε.Ι. Κρήτης, 2009

[9] Robotics & Automation Magazine, IEEE, "Review of the Robotica software package for robotic manipulators", 1994

[10] Robotica Control Software,

(url:http://www.robotica.co.uk/robotica/ramc/products/servo_control/servo_software.htm)

[11] Robotics Studio, από Wikipedia,
(url: http://en.wikipedia.org/wiki/Robotics_Studio)
[12] The Player Project,

(url:http://playerstage.sourceforge.net/)

[13] Virtual Reality Modelling Language (VRML),

(url:http://www.it.uom.gr/project/MultimediaTechnologyNotes/extra/append9_3.htm)

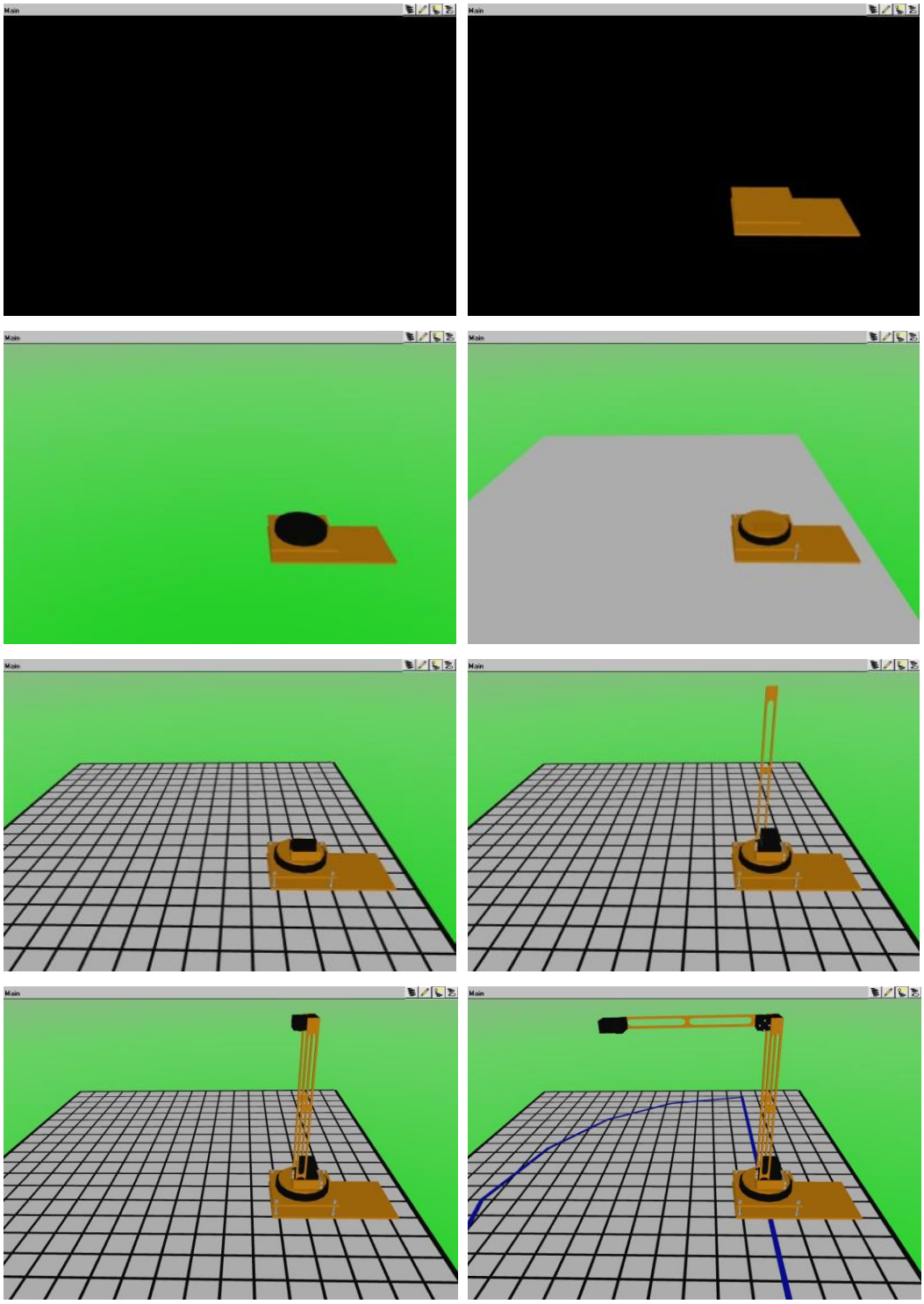
[14] Φοίβος-Απόστολος Μυλωνάς, "Σχεδιασμός & Υλοποίηση VRML Browser σε Java", Διπλωματική εργασία, Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών, 2003

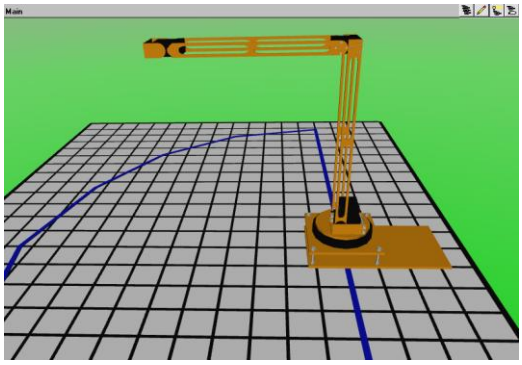
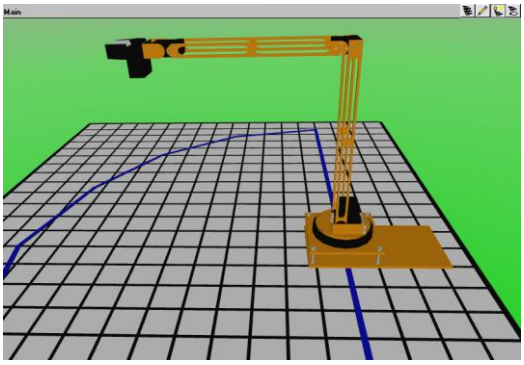
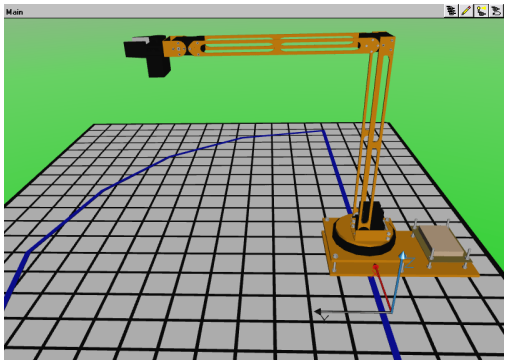
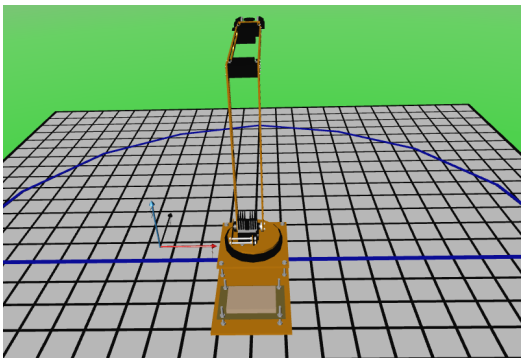
[15] Εισαγωγή στην Virtual Reality Modelling Language (VRML),
(url:http://alexandra.di.uoa.gr/mmtech/VirtualReality/eisagvgh.pdf)

[16] Μαυραντζάς Νικόλαος, "Οι τεχνολογίες 3D στην τάξη και παραδείγματα ενσωμάτωσης στη διδασκαλία χρησιμοποιώντας την γλώσσα VRML", 4ο Συνέδριο στη Σύρο - ΤΠΕ στην Εκπαίδευση, 2007
[17] VRML, από Wikipedia,
(url:http://en.wikipedia.org/wiki/VRML)
[18] 3DMLW, από Wikipedia,
(url:http://en.wikipedia.org/wiki/3DMLW)
[19] COLLADA, από Wikipedia,
(url:http://en.wikipedia.org/wiki/COLLADA)
[20] O3D, από Wikipedia,
(url:http://en.wikipedia.org/wiki/O3D)
[21] Universal 3D, από Wikipedia,
(url:http://en.wikipedia.org/wiki/Universal_3D)
[22] X3D, από Wikipedia,
(url:http://en.wikipedia.org/wiki/X3D)
[23] Introduction to V·Realm Builder Concepts, από το μενού help του V-Realm Builder 2.0
[24] Installing the VRML Editor on the Host Computer :: Installation (Virtual Reality Toolbox™), από το μενού help του λογισμικού MATLAB
[25] Θωμάς Σακάρος, "Κατασκευή και έλεγχος κατακόρυφου αρθρωτού ρομποτικού βραχίονα τεσσάρων βαθμών ελευθερίας", Πτυχιακή εργασία, Α.Τ.Ε.Ι Χανίων 2009
[26] Δ.Μ. Εμίρης, Δ.Ε. Κουλουριώτης, Ρομποτική 3η έκδοση, Αθήνα 2006

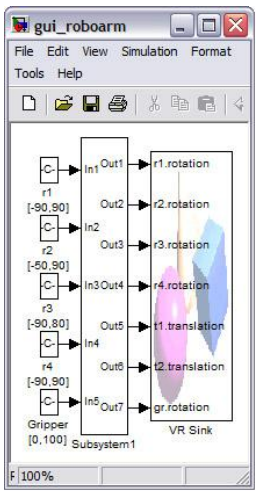
ΠΑΡΑΡΤΗΜΑ

ΚΑΤΑΣΚΕΥΗ ΕΙΚΟΝΙΚΟΥ ΒΡΑΧΙΟΝΑ

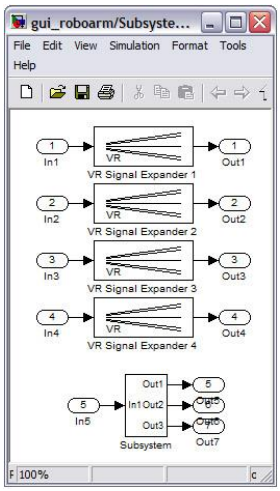


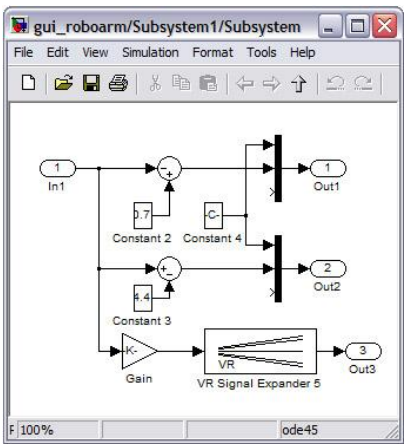
ΜΟΝΤΕΛΟ ΠΡΟΣΟΜΟΙΩΣΗΣ (SIMULINK MODEL)



Κεντρικό μοντέλο προσομοίωσης για τον έλεγχο του βραχίονα



Υποσύστημα κεντρικού μοντέλου προσομοίωσης για την επέκταση των σημάτων των περιστροφικών κινήσεων και τον έλεγχο του άκρου εργασίας



Υποσύστημα ελέγχου του άκρου εργασίας

ΚΩΔΙΚΕΣ

forward_simulation.m

Το forward_simulation.m δημιουργεί τη γραφική διεπαφή για την προσομοίωση του εικονικού βραχίονα, χρησιμοποιώντας το ευθύ κινηματικό πρόβλημα.

```
function varargout = forward_simulation(varargin)
% FORWARD_SIMULATION M-file for forward_simulation.fig
%      FORWARD_SIMULATION, by itself, creates a new
FORWARD_SIMULATION or raises the existing
%      singleton*.
%
%      H = FORWARD_SIMULATION returns the handle to a new
FORWARD_SIMULATION or the handle to
%      the existing singleton*.
%
%      FORWARD_SIMULATION('CALLBACK',hObject,eventData,handles,...)
calls the local
%      function named CALLBACK in FORWARD_SIMULATION.M with the given
input arguments.
%
%      FORWARD_SIMULATION('Property','Value',...) creates a new
FORWARD_SIMULATION or raises the
%      existing singleton*.  Starting from the left, property value
pairs are
%      applied to the GUI before forward_simulation_OpeningFcn gets
called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to forward_simulation_OpeningFcn
via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows
only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help
forward_simulation

% Last Modified by GUIDE v2.5 25-Sep-2009 19:34:15

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @forward_simulation_OpeningFcn,
...
                   'gui_OutputFcn',  @forward_simulation_OutputFcn,
...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
```

```
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before forward_simulation is made visible.
function forward_simulation_OpeningFcn(hObject, eventdata, handles,
varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to forward_simulation (see
VARARGIN)

% Choose default command line output for forward_simulation
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes forward_simulation wait for user response (see
UIRESUME)
% uiwait(handles.figure1

%Initialization of the robot's position

temp1=(get(handles.slider1,'value'));
set(handles.edit1,'String',temp1);
set(handles.edit9,'String',temp1);

temp2=(get(handles.slider2,'value'));
set(handles.edit2,'String',temp2);
set(handles.edit10,'String',temp2);

temp3=(get(handles.slider3,'value'));
set(handles.edit3,'String',temp3);
set(handles.edit11,'String',temp3);

temp4=(get(handles.slider4,'value'));
set(handles.edit4,'String',temp4);
set(handles.edit12,'String',temp4);

temp5=(get(handles.slider5,'value'));
set(handles.edit5,'String',temp5);

[handles] = Total_Transformation_Matrix(handles);


% --- Outputs from this function are returned to the command line.
function varargout = forward_simulation_OutputFcn(hObject, eventdata,
handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Total trasformation matrix.
function [handles] = Total_Transformation_Matrix(handles)

th1=(get(handles.slider1,'value'));
th1=round(th1);

th2=(get(handles.slider2,'value'));
th2=round(th2);

th3=(get(handles.slider3,'value'));
th3=round(th3);

th4=(get(handles.slider4,'value'));
th4=round(th4);

gripper=(get(handles.slider5,'value'));
gripper=round(gripper);

result=forward_calc([th1 th2 th3 th4],gripper);

set(handles.text17,'string',num2str(result.T(1,1),'%6.4f'));
set(handles.text18,'string',num2str(result.T(1,2),'%6.4f'));
set(handles.text19,'string',num2str(result.T(1,3),'%6.4f'));
set(handles.text20,'string',num2str(result.T(1,4),'%6.4f'));
set(handles.text21,'string',num2str(result.T(2,1),'%6.4f'));
set(handles.text22,'string',num2str(result.T(2,2),'%6.4f'));
set(handles.text23,'string',num2str(result.T(2,3),'%6.4f'));
set(handles.text24,'string',num2str(result.T(2,4),'%6.4f'));
set(handles.text25,'string',num2str(result.T(3,1),'%6.4f'));
set(handles.text26,'string',num2str(result.T(3,2),'%6.4f'));
set(handles.text27,'string',num2str(result.T(3,3),'%6.4f'));
set(handles.text28,'string',num2str(result.T(3,4),'%6.4f'));


% --- Joint 1 slider movement.
function slider1_Callback(hObject, eventdata, handles)
% hObject    handle to edit9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range
of slider

temp1=(get(handles.slider1,'value'));

set(handles.edit1,'String',temp1);
set(handles.edit9,'String',temp1);

[handles] = Total_Transformation_Matrix(handles);

pushbutton1_Callback(hObject, eventdata, handles);
```

```
% --- Executes during object creation, after setting all properties.
function slider1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end


% --- Joint 2 slider movement.
function slider2_Callback(hObject, eventdata, handles)
% hObject    handle to slider2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range
of slider

temp2=(get(handles.slider2,'value'));

set(handles.edit2,'String',temp2);
set(handles.edit10,'String',temp2);

[handles] = Total_Transformation_Matrix(handles);

pushbutton1_Callback(hObject, eventdata, handles)


% --- Executes during object creation, after setting all properties.
function slider2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end


% --- Joint 3 slider movement.
function slider3_Callback(hObject, eventdata, handles)
% hObject    handle to slider3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range
of slider
```

```
temp3=(get(handles.slider3,'value'));

set(handles.edit3,'String',temp3);
set(handles.edit11,'String',temp3);

[handles] = Total_Transformation_Matrix(handles);

pushbutton1_Callback(hObject, eventdata, handles)


% --- Executes during object creation, after setting all properties.
function slider3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end


% --- Joint 4 slider movement.
function slider4_Callback(hObject, eventdata, handles)
% hObject    handle to slider4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range
of slider

temp4=(get(handles.slider4,'value'));

set(handles.edit4,'String',temp4);
set(handles.edit12,'String',temp4);

[handles] = Total_Transformation_Matrix(handles);

pushbutton1_Callback(hObject, eventdata, handles)


% --- Executes during object creation, after setting all properties.
function slider4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end
```

```
% --- Gripper slider movement.
function slider5_Callback(hObject, eventdata, handles)
% hObject    handle to slider5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range
of slider

temp5=(get(handles.slider5,'value'));

set(handles.edit5,'String',temp5);

pushbutton1_Callback(hObject, eventdata, handles);


% --- Executes during object creation, after setting all properties.
function slider5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end


% --- Simulate! pushbutton.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

temp1=(get(handles.slider1,'value'));
temp2=(get(handles.slider2,'value'));
temp3=(get(handles.slider3,'value'));
temp4=(get(handles.slider4,'value'));
temp5=(get(handles.slider5,'value'));

set(handles.edit9,'String',temp1);
set(handles.edit10,'String',temp2);
set(handles.edit11,'String',temp3);
set(handles.edit12,'String',temp4);

opts = simset('SrcWorkspace','current');

value_1=temp1;
value_2=temp2;
value_3=temp3;
value_4=temp4;
value_5=temp5;

sim('gui_roboarm',[],opts);
```

```
[handles] = Total_Transformation_Matrix(handles);


% --- Start point pushbutton.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

set(handles.edit1,'string','0');
set(handles.edit9,'string','0');
set(handles.edit2,'string','28');
set(handles.edit10,'string','28');
set(handles.edit3,'string','38');
set(handles.edit11,'string','38');
set(handles.edit4,'string','24');
set(handles.edit12,'string','24');
set(handles.edit5,'string','0');
set(handles.slider1,'value',0);
set(handles.slider2,'value',28);
set(handles.slider3,'value',38);
set(handles.slider4,'value',24);
set(handles.slider5,'value',0);

pushbutton1_Callback(hObject, eventdata, handles);


% --- Exit pushbutton.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

delete(forward_simulation);


% --- Open gripper pushbutton.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

set(handles.edit5,'string','0');
set(handles.slider5,'value',0);

pushbutton1_Callback(hObject, eventdata, handles);


% --- Close gripper pushbutton.
function pushbutton5_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

set(handles.edit5,'string','100');
set(handles.slider5,'value',100);

pushbutton1_Callback(hObject, eventdata, handles);
```

```
% --- Joint 1 edit.
function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1
as a double

n=str2double(get(handles.edit1,'string'));

if isnan(n)
    errordlg('You must enter a numeric value between -90 and 90
degrees.','Invalid value','modal');
    temp1=(get(handles.slider1,'value'));
    set(handles.edit1,'String',temp1);
else
    if (n>90 || n<-90)
        errordlg('You must enter a value between -90 and 90
degrees.','Invalid value','modal');
        temp1=(get(handles.slider1,'value'));
        set(handles.edit1,'String',temp1);
    else
        set(handles.slider1,'value',n);
    end
end

pushbutton1_Callback(hObject, eventdata, handles);


% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Joint 2 edit.
function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2
as a double

n=str2double(get(handles.edit2,'string'));

if isnan(n)
```

```
    errordlg('You must enter a numeric value between -50 and 90
degrees.','Invalid value','modal');
    temp2=(get(handles.slider2,'value'));
    set(handles.edit2,'String',temp2);
else
    if (n>90 || n<-50)
        errordlg('You must enter a value between -50 and 90
degrees.','Invalid value','modal');
        temp2=(get(handles.slider2,'value'));
        set(handles.edit2,'String',temp2);
    else
        set(handles.slider2,'value',n);
    end
end

pushbutton1_Callback(hObject, eventdata, handles);


% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Joint 3 edit.
function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3
as a double

n=str2double(get(handles.edit3,'string'));

if isnan(n)
    errordlg('You must enter a numeric value between -90 and 80
degrees.','Invalid value','modal');
    temp3=(get(handles.slider3,'value'));
    set(handles.edit3,'String',temp3);
else
    if (n>80 || n<-90)
        errordlg('You must enter a value between -90 and 80
degrees.','Invalid value','modal');
        temp3=(get(handles.slider3,'value'));
        set(handles.edit3,'String',temp3);
    else
        set(handles.slider3,'value',n);
    end
```

```
end

pushbutton1_Callback(hObject, eventdata, handles);


% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



% --- Joint 4 edit.
function edit4_Callback(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit4 as text
%        str2double(get(hObject,'String')) returns contents of edit4
as a double

n=str2double(get(handles.edit4,'string'));

if isnan(n)
    errordlg('You must enter a numeric value between -90 and 90
degrees.','Invalid value','modal');
    temp4=(get(handles.slider4,'value'));
    set(handles.edit4,'String',temp4);
else
    if (n>90 || n<-90)
        errordlg('You must enter a value between -90 and 90
degrees.','Invalid value','modal');
        temp4=(get(handles.slider4,'value'));
        set(handles.edit4,'String',temp4);
    else
        set(handles.slider4,'value',n);
    end
end

pushbutton1_Callback(hObject, eventdata, handles);


% --- Executes during object creation, after setting all properties.
function edit4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
```

```
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



% --- Gripper edit.
function edit5_Callback(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit5 as text
%        str2double(get(hObject,'String')) returns contents of edit5
as a double

n=str2double(get(handles.edit5,'string'));

if isnan(n)
    errordlg('You must enter a numeric value between 0 and
100.','Invalid value','modal');
    temp5=(get(handles.slider5,'value'));
    set(handles.edit5,'String',temp5);
else
    if (n>100 || n<0)
        errordlg('You must enter a value between 0 and 100.','Invalid
value','modal');
        temp5=(get(handles.slider5,'value'));
        set(handles.edit5,'String',temp5);
    else
        set(handles.slider5,'value',n);
    end
end

pushbutton1_Callback(hObject, eventdata, handles);



% --- Executes during object creation, after setting all properties.
function edit5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



% --- Total transformation matrix X coordinate text.
function text20_Callback(hObject, eventdata, handles)
% hObject    handle to text20 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hints: get(hObject,'String') returns contents of text20 as text
%        str2double(get(hObject,'String')) returns contents of text20
as a double


% --- Executes during object creation, after setting all properties.
function text20_CreateFcn(hObject, eventdata, handles)
% hObject    handle to text20 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Total transformation matrix Y coordinate text.
function text24_Callback(hObject, eventdata, handles)
% hObject    handle to text24 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of text24 as text
%        str2double(get(hObject,'String')) returns contents of text24
as a double


% --- Executes during object creation, after setting all properties.
function text24_CreateFcn(hObject, eventdata, handles)
% hObject    handle to text24 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Total transformation matrix Z coordinate text.
function text28_Callback(hObject, eventdata, handles)
% hObject    handle to text28 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of text28 as text
%        str2double(get(hObject,'String')) returns contents of text28
as a double


% --- Executes during object creation, after setting all properties.
```

```
function text28_CreateFcn(hObject, eventdata, handles)
% hObject    handle to text28 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Angle th1 edit.
function edit9_Callback(hObject, eventdata, handles)
% hObject    handle to edit9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit9 as text
%        str2double(get(hObject,'String')) returns contents of edit9
as a double

    temp1=(get(handles.slider1,'value'));

    set(handles.edit9,'String',temp1);

% --- Executes during object creation, after setting all properties.
function edit9_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Angle th2 edit.
function edit10_Callback(hObject, eventdata, handles)
% hObject    handle to edit10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit10 as text
%        str2double(get(hObject,'String')) returns contents of edit10
as a double


% --- Executes during object creation, after setting all properties.
function edit10_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Angle th3 edit.
function edit11_Callback(hObject, eventdata, handles)
% hObject    handle to edit11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit11 as text
%        str2double(get(hObject,'String')) returns contents of edit11
as a double


% --- Executes during object creation, after setting all properties.
function edit11_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Angle th4 edit.
function edit12_Callback(hObject, eventdata, handles)
% hObject    handle to edit12 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit12 as text
%        str2double(get(hObject,'String')) returns contents of edit12
as a double


% --- Executes during object creation, after setting all properties.
function edit12_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit12 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
```

```
    set(hObject,'BackgroundColor','white');
end


% --------------------------------------------------------------------
-
function Help_Callback(hObject, eventdata, handles)
% hObject    handle to Help (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

open('user_guide.pdf');


% --------------------------------------------------------------------
-
function Kinematics_Callback(hObject, eventdata, handles)
% hObject    handle to Kinematics (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
-
function Simulation_Mode_Callback(hObject, eventdata, handles)
% hObject    handle to Simulation_Mode (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

forward_calc.m

Το forward_calc.m υπολογίζει το ευθύ κινηματικό πρόβλημα και το συνολικό πίνακα μετασχηματισμού.

```
function [result]=forward_calc(TH,gripper)

% Get the angle values.
th1=TH(1);, th2=TH(2);, th3=TH(3);, th4=TH(4);

% Lengths definition.
 A=0.1047;, L1=0.2910;, x=0.0220;, L2=0.2720;, L3=0.0770;

% Convert degrees to radians.
th1=th1*pi/180;, th2=th2*pi/180;, th3=th3*pi/180;, th4=th4*pi/180;

%% Direct kinematic problem formulation.

T01=[ -sin(th1)    0    -cos(th1)    0;
       cos(th1)    0    -sin(th1)    0;
       0          -1     0           A;
       0           0     0           1];

T1A=[  sin(th2)    cos(th2)      0    sin(th2)*L1;
      -cos(th2)    sin(th2)      0   -cos(th2)*L1;
       0           0             1    0;
       0           0             0    1];

TA2=[  0  -1    0    0;
       1   0    0    x;
       0   0    1    0;
       0   0    0    1];

T23=[  cos(th3)    -sin(th3)    0    cos(th3)*L2;
       sin(th3)     cos(th3)    0    sin(th3)*L2;
       0            0           1    0;
       0            0           0    1];

T34=[  cos(th4)   -sin(th4)     0    cos(th4)*L3;
       sin(th4)    cos(th4)     0    sin(th4)*L3;
       0           0            1    0;...
       0           0            0    1];

% Total transformation matrix.
T=T01*T1A*TA2*T23*T34;

% L1 vertical part.
T0A=T01*T1A;
% L1 horizontal part.
T02=T01*T1A*TA2;
% L2.
T03=T01*T1A*TA2*T23;
% L3.
T04=T01*T1A*TA2*T23*T34;

% These are the coordinates of the gripper points with respect to
coonrdinate system #4.
At=[ 0       0    0.02*(0.01*gripper)    1]';
```

```
Ab=[-0.03   0    0.02*(0.01*gripper)   1]';
B= [-0.03   0    0.03                  1]';
C= [-0.03   0   -0.03                  1]';
Dt=[ 0      0   -0.02*(0.01*gripper)   1]';
Db=[-0.03   0   -0.02*(0.01*gripper)   1]';

% The new coordinates with respect to coordinate system #0.
Atn=T04*At;
Abn=T04*Ab;
Bn =T04*B;
Cn =T04*C;
Dtn=T04*Dt;
Dbn=T04*Db;

result.T01=T01;
result.T0A=T0A;
result.T02=T02;
result.T03=T03;
result.T04=T04;
result.Atn=Atn;
result.Abn=Abn;
result.Bn=Bn;
result.Cn=Cn;
result.Dtn=Dtn;
result.Dbn=Dbn;

% Final total transformation matrix.
result.T=T;
```

forward_real.m

Το forward_real.m δημιουργεί τη γραφική διεπαφή για την πραγματική κίνηση του βραχίονα, χρησιμοποιώντας το ευθύ κινηματικό πρόβλημα.

```
 function varargout = forward_real(varargin)
% FORWARD_REAL M-file for forward_real.fig
%      FORWARD_REAL, by itself, creates a new FORWARD_REAL or raises
the existing
%      singleton*.
%
%      H = FORWARD_REAL returns the handle to a new FORWARD_REAL or
the handle to
%      the existing singleton*.
%
%      FORWARD_REAL('CALLBACK',hObject,eventData,handles,...) calls
the local
%      function named CALLBACK in FORWARD_REAL.M with the given input
arguments.
%
%      FORWARD_REAL('Property','Value',...) creates a new
FORWARD_REAL or raises the
%      existing singleton*.  Starting from the left, property value
pairs are
%      applied to the GUI before forward_real_OpeningFcn gets called.
An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to forward_real_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows
only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help forward_real

% Last Modified by GUIDE v2.5 28-Sep-2009 17:37:33

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @forward_real_OpeningFcn, ...
                   'gui_OutputFcn',  @forward_real_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
```

```
% --- Executes just before forward_real is made visible.
function forward_real_OpeningFcn(hObject, eventdata, handles,
varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to forward_real (see VARARGIN)

% Choose default command line output for forward_real
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes forward_real wait for user response (see UIRESUME)
% uiwait(handles.figure1

%Initialization of the robot's position

temp1=(get(handles.slider1,'value'));
set(handles.edit1,'String',temp1);
set(handles.edit9,'String',temp1);

temp2=(get(handles.slider2,'value'));
set(handles.edit2,'String',temp2);
set(handles.edit10,'String',temp2);

temp3=(get(handles.slider3,'value'));
set(handles.edit3,'String',temp3);
set(handles.edit11,'String',temp3);

temp4=(get(handles.slider4,'value'));
set(handles.edit4,'String',temp4);
set(handles.edit12,'String',temp4);

temp5=(get(handles.slider5,'value'));
set(handles.edit5,'String',temp5);

[handles] = Total_Transformation_Matrix(handles);


% --- Outputs from this function are returned to the command line.
function varargout = forward_real_OutputFcn(hObject, eventdata,
handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Total trasformation matrix.
function [handles] = Total_Transformation_Matrix(handles)

th1=(get(handles.slider1,'value'));
```

```
th1=round(th1);

th2=(get(handles.slider2,'value'));
th2=round(th2);

th3=(get(handles.slider3,'value'));
th3=round(th3);

th4=(get(handles.slider4,'value'));
th4=round(th4);

gripper=(get(handles.slider5,'value'));
gripper=round(gripper);

result=forward_calc([th1 th2 th3 th4],gripper);

set(handles.text17,'string',num2str(result.T(1,1),'%6.4f'));
set(handles.text18,'string',num2str(result.T(1,2),'%6.4f'));
set(handles.text19,'string',num2str(result.T(1,3),'%6.4f'));
set(handles.text20,'string',num2str(result.T(1,4),'%6.4f'));
set(handles.text21,'string',num2str(result.T(2,1),'%6.4f'));
set(handles.text22,'string',num2str(result.T(2,2),'%6.4f'));
set(handles.text23,'string',num2str(result.T(2,3),'%6.4f'));
set(handles.text24,'string',num2str(result.T(2,4),'%6.4f'));
set(handles.text25,'string',num2str(result.T(3,1),'%6.4f'));
set(handles.text26,'string',num2str(result.T(3,2),'%6.4f'));
set(handles.text27,'string',num2str(result.T(3,3),'%6.4f'));
set(handles.text28,'string',num2str(result.T(3,4),'%6.4f'));


% --- Joint 1 slider movement.
function slider1_Callback(hObject, eventdata, handles)
% hObject    handle to edit9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range
of slider

temp1=(get(handles.slider1,'value'));

set(handles.edit1,'String',temp1);
set(handles.edit9,'String',temp1);

[handles] = Total_Transformation_Matrix(handles);

pushbutton1_Callback(hObject, eventdata, handles);


% --- Executes during object creation, after setting all properties.
function slider1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: slider controls usually have a light gray background.
```

```
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end


% --- Joint 2 slider movement.
function slider2_Callback(hObject, eventdata, handles)
% hObject    handle to slider2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range
of slider

temp2=(get(handles.slider2,'value'));

set(handles.edit2,'String',temp2);
set(handles.edit10,'String',temp2);

[handles] = Total_Transformation_Matrix(handles);

pushbutton1_Callback(hObject, eventdata, handles)


% --- Executes during object creation, after setting all properties.
function slider2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end


% --- Joint 3 slider movement.
function slider3_Callback(hObject, eventdata, handles)
% hObject    handle to slider3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range
of slider

temp3=(get(handles.slider3,'value'));

set(handles.edit3,'String',temp3);
set(handles.edit11,'String',temp3);

[handles] = Total_Transformation_Matrix(handles);

pushbutton1_Callback(hObject, eventdata, handles)
```

```
% --- Executes during object creation, after setting all properties.
function slider3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end


% --- Joint 4 slider movement.
function slider4_Callback(hObject, eventdata, handles)
% hObject    handle to slider4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range
of slider

temp4=(get(handles.slider4,'value'));

set(handles.edit4,'String',temp4);
set(handles.edit12,'String',temp4);

[handles] = Total_Transformation_Matrix(handles);

pushbutton1_Callback(hObject, eventdata, handles)


% --- Executes during object creation, after setting all properties.
function slider4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end


% --- Gripper slider movement.
function slider5_Callback(hObject, eventdata, handles)
% hObject    handle to slider5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range
of slider
```

```
temp5=(get(handles.slider5,'value'));

set(handles.edit5,'String',temp5);

pushbutton1_Callback(hObject, eventdata, handles);


% --- Executes during object creation, after setting all properties.
function slider5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end


% --- Simulate! pushbutton.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

temp1=(get(handles.slider1,'value'));
temp2=(get(handles.slider2,'value'));
temp3=(get(handles.slider3,'value'));
temp4=(get(handles.slider4,'value'));
temp5=(get(handles.slider5,'value'));

set(handles.edit9,'String',temp1);
set(handles.edit10,'String',temp2);
set(handles.edit11,'String',temp3);
set(handles.edit12,'String',temp4);

opts = simset('SrcWorkspace','current');

value_1=temp1;
value_2=temp2;
value_3=temp3;
value_4=temp4;
value_5=temp5;

sim('gui_roboarm',[],opts);

[handles] = Total_Transformation_Matrix(handles);


% --- Start point pushbutton.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

set(handles.edit1,'string','0');
```

```
set(handles.edit9,'string','0');
set(handles.edit2,'string','28');
set(handles.edit10,'string','28');
set(handles.edit3,'string','38');
set(handles.edit11,'string','38');
set(handles.edit4,'string','24');
set(handles.edit12,'string','24');
set(handles.edit5,'string','0');
set(handles.slider1,'value',0);
set(handles.slider2,'value',28);
set(handles.slider3,'value',38);
set(handles.slider4,'value',24);
set(handles.slider5,'value',0);

pushbutton1_Callback(hObject, eventdata, handles);


% --- Exit pushbutton.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

delete(forward_real);


% --- Open gripper pushbutton.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

set(handles.edit5,'string','0');
set(handles.slider5,'value',0);

pushbutton1_Callback(hObject, eventdata, handles);


% --- Close gripper pushbutton.
function pushbutton5_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

set(handles.edit5,'string','100');
set(handles.slider5,'value',100);

pushbutton1_Callback(hObject, eventdata, handles);


% --- Move robot pushbutton.
function pushbutton6_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%% serial communication
[handles] = forward_serial(handles);
```

```
% --- Joint 1 edit.
function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1
as a double

n=str2double(get(handles.edit1,'string'));

if isnan(n)
    errordlg('You must enter a numeric value between -90 and 90
degrees.','Invalid value','modal');
    temp1=(get(handles.slider1,'value'));
    set(handles.edit1,'String',temp1);
else
    if (n>90 || n<-90)
        errordlg('You must enter a value between -90 and 90
degrees.','Invalid value','modal');
        temp1=(get(handles.slider1,'value'));
        set(handles.edit1,'String',temp1);
    else
        set(handles.slider1,'value',n);
    end
end

pushbutton1_Callback(hObject, eventdata, handles);


% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Joint 2 edit.
function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2
as a double

n=str2double(get(handles.edit2,'string'));

if isnan(n)
```

```
    errordlg('You must enter a numeric value between -50 and 90
degrees.','Invalid value','modal');
    temp2=(get(handles.slider2,'value'));
    set(handles.edit2,'String',temp2);
else
    if (n>90 || n<-50)
        errordlg('You must enter a value between -50 and 90
degrees.','Invalid value','modal');
        temp2=(get(handles.slider2,'value'));
        set(handles.edit2,'String',temp2);
    else
        set(handles.slider2,'value',n);
    end
end

pushbutton1_Callback(hObject, eventdata, handles);


% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Joint 3 edit.
function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3
as a double

n=str2double(get(handles.edit3,'string'));

if isnan(n)
    errordlg('You must enter a numeric value between -90 and 80
degrees.','Invalid value','modal');
    temp3=(get(handles.slider3,'value'));
    set(handles.edit3,'String',temp3);
else
    if (n>80 || n<-90)
        errordlg('You must enter a value between -90 and 80
degrees.','Invalid value','modal');
        temp3=(get(handles.slider3,'value'));
        set(handles.edit3,'String',temp3);
    else
        set(handles.slider3,'value',n);
    end
```

```
end

pushbutton1_Callback(hObject, eventdata, handles);


% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



% --- Joint 4 edit.
function edit4_Callback(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit4 as text
%        str2double(get(hObject,'String')) returns contents of edit4
as a double

n=str2double(get(handles.edit4,'string'));

if isnan(n)
    errordlg('You must enter a numeric value between -90 and 90
degrees.','Invalid value','modal');
    temp4=(get(handles.slider4,'value'));
    set(handles.edit4,'String',temp4);
else
    if (n>90 || n<-90)
        errordlg('You must enter a value between -90 and 90
degrees.','Invalid value','modal');
        temp4=(get(handles.slider4,'value'));
        set(handles.edit4,'String',temp4);
    else
        set(handles.slider4,'value',n);
    end
end

pushbutton1_Callback(hObject, eventdata, handles);


% --- Executes during object creation, after setting all properties.
function edit4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
```

```
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



% --- Gripper edit.
function edit5_Callback(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit5 as text
%        str2double(get(hObject,'String')) returns contents of edit5
as a double

n=str2double(get(handles.edit5,'string'));

if isnan(n)
    errordlg('You must enter a numeric value between 0 and
100.','Invalid value','modal');
    temp5=(get(handles.slider5,'value'));
    set(handles.edit5,'String',temp5);
else
    if (n>100 || n<0)
        errordlg('You must enter a value between 0 and 100.','Invalid
value','modal');
        temp5=(get(handles.slider5,'value'));
        set(handles.edit5,'String',temp5);
    else
        set(handles.slider5,'value',n);
    end
end

pushbutton1_Callback(hObject, eventdata, handles);



% --- Executes during object creation, after setting all properties.
function edit5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



% --- Total transformation matrix X coordinate text.
function text20_Callback(hObject, eventdata, handles)
% hObject    handle to text20 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hints: get(hObject,'String') returns contents of text20 as text
%        str2double(get(hObject,'String')) returns contents of text20
as a double


% --- Executes during object creation, after setting all properties.
function text20_CreateFcn(hObject, eventdata, handles)
% hObject    handle to text20 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Total transformation matrix Y coordinate text.
function text24_Callback(hObject, eventdata, handles)
% hObject    handle to text24 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of text24 as text
%        str2double(get(hObject,'String')) returns contents of text24
as a double


% --- Executes during object creation, after setting all properties.
function text24_CreateFcn(hObject, eventdata, handles)
% hObject    handle to text24 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Total transformation matrix Z coordinate text.
function text28_Callback(hObject, eventdata, handles)
% hObject    handle to text28 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of text28 as text
%        str2double(get(hObject,'String')) returns contents of text28
as a double


% --- Executes during object creation, after setting all properties.
```

```
function text28_CreateFcn(hObject, eventdata, handles)
% hObject    handle to text28 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Angle th1 edit.
function edit9_Callback(hObject, eventdata, handles)
% hObject    handle to edit9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit9 as text
%        str2double(get(hObject,'String')) returns contents of edit9
as a double

    temp1=(get(handles.slider1,'value'));

    set(handles.edit9,'String',temp1);


% --- Executes during object creation, after setting all properties.
function edit9_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function edit10_Callback(hObject, eventdata, handles)
% hObject    handle to edit10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit10 as text
%        str2double(get(hObject,'String')) returns contents of edit10
as a double


% --- Angle th2 edit.
function edit10_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Angle th3 edit.
function edit11_Callback(hObject, eventdata, handles)
% hObject    handle to edit11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit11 as text
%        str2double(get(hObject,'String')) returns contents of edit11
as a double


% --- Executes during object creation, after setting all properties.
function edit11_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Angle th4 edit.
function edit12_Callback(hObject, eventdata, handles)
% hObject    handle to edit12 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit12 as text
%        str2double(get(hObject,'String')) returns contents of edit12
as a double


% --- Executes during object creation, after setting all properties.
function edit12_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit12 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
```

```
    set(hObject,'BackgroundColor','white');
end


% --------------------------------------------------------------------
-
function Help_Callback(hObject, eventdata, handles)
% hObject    handle to Help (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

open('user_guide.pdf');


% --------------------------------------------------------------------
-
function Kinematics_Callback(hObject, eventdata, handles)
% hObject    handle to Kinematics (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
-
function Simulation_Mode_Callback(hObject, eventdata, handles)
% hObject    handle to Simulation_Mode (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

forward_serial.m

Το forward_serial.m χρησιμοποιείται για το ευθύ κινηματικό και είναι υπεύθυνο για τη σειριακή επικοινωνία του υπολογιστή, με τον πραγματικό βραχίονα και τον έλεγχο των πέντε σερβοκινητήρων του.

```
function [handles] = forward_serial(handles)

%% Create serial port

s=serial('COM1');
s.BaudRate = 115200;
s.Terminator = 'CR/LF';
fopen(s);
%Sport=instrfind

%% Servo move

% Servo motor #1
angle1 = (get(handles.edit1,'string'));
angle1 = str2double(angle1);

x= 0.1*(angle1)+10;
pwm1 = -93.04*x + 2347.9;
pwm1=round(pwm1)

servo1 = sprintf('#0 P%d T3000',pwm1);
fprintf(s,servo1);

% Servo motor #2
angle2 = (get(handles.edit2,'string'));
angle2 = str2double(angle2);

pwm2 = 8.7932*angle2 + 1375.1;
pwm2=round(pwm2)

servo2 = sprintf('#1 P%d T3000',pwm2);
fprintf(s,servo2);

% Servo motor #3
angle3 = (get(handles.edit3,'string'));
angle3 = str2double(angle3);

pwm3 = -9.765*angle3 + 1537.3;
pwm3=round(pwm3)

servo3 = sprintf('#2 P%d T3000',pwm3);
fprintf(s,servo3);

% Servo motor #4
angle4 = (get(handles.edit4,'string'));
angle4 = str2double(angle4);

pwm4 = -10.3*angle4 + 1508.8;
pwm4=round(pwm4)

servo4 = sprintf('#3 P%d T3000',pwm4);
```

```
fprintf(s,servo4);

% Servo motor #5
angle5 = (get(handles.edit5,'string'));
angle5 = str2double(angle5);

pwm5 = 11.65*angle5 + 1335.14;
pwm5=round(pwm5)

servo5 = sprintf('#4 P%d T3000',pwm5);
fprintf(s,servo5);

%% Delete serial port

fclose(s);
delete(s);
%Sport=instrfind
```

inverse_simulation.m

Το inverse_simulation.m δημιουργεί τη γραφική διεπαφή για την προσομοίωση του εικονικού βραχίονα, χρησιμοποιώντας το αντίστροφο κινηματικό πρόβλημα.

```
function varargout = inverse_simulation(varargin)
% INVERSE_SIMULATION M-file for inverse_simulation.fig
%      INVERSE_SIMULATION, by itself, creates a new
INVERSE_SIMULATION or raises the existing
%      singleton*.
%
%      H = INVERSE_SIMULATION returns the handle to a new
INVERSE_SIMULATION or the handle to
%      the existing singleton*.
%
%      INVERSE_SIMULATION('CALLBACK',hObject,eventData,handles,...)
calls the local
%      function named CALLBACK in INVERSE_SIMULATION.M with the given
input arguments.
%
%      INVERSE_SIMULATION('Property','Value',...) creates a new
INVERSE_SIMULATION or raises the
%      existing singleton*.  Starting from the left, property value
pairs are
%      applied to the GUI before inverse_simulation_OpeningFcn gets
called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to inverse_simulation_OpeningFcn
via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows
only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help
inverse_simulation

% Last Modified by GUIDE v2.5 25-Sep-2009 19:32:10

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @inverse_simulation_OpeningFcn,
...
                   'gui_OutputFcn',  @inverse_simulation_OutputFcn,
...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
```

```
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before inverse_simulation is made visible.
function inverse_simulation_OpeningFcn(hObject, eventdata, handles,
varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to inverse_simulation (see
VARARGIN)

% Choose default command line output for inverse_simulation
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes inverse_simulation wait for user response (see
UIRESUME)
% uiwait(handles.figure1

%Initialization of the robot's position
pushbutton2_Callback(hObject, eventdata, handles)


% --- Outputs from this function are returned to the command line.
function varargout = inverse_simulation_OutputFcn(hObject, eventdata,
handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Total trasformation matrix.
function [handles] = inverse(handles)
xx=(get(handles.slider1,'value'));
yy=(get(handles.slider2,'value'));
zz=(get(handles.slider3,'value'));

gripper=(get(handles.slider4,'value'));

angles=inverse_simple_calc(xx,yy,zz);
global result;
result=forward_calc([angles],gripper);

set(handles.text1,'string',num2str(result.T(1,1),'%6.4f'));
set(handles.text2,'string',num2str(result.T(1,2),'%6.4f'));
set(handles.text3,'string',num2str(result.T(1,3),'%6.4f'));
set(handles.text4,'string',num2str(result.T(1,4),'%6.4f'));
set(handles.text5,'string',num2str(result.T(2,1),'%6.4f'));
```

```
set(handles.text6,'string',num2str(result.T(2,2),'%6.4f'));
set(handles.text7,'string',num2str(result.T(2,3),'%6.4f'));
set(handles.text8,'string',num2str(result.T(2,4),'%6.4f'));
set(handles.text9,'string',num2str(result.T(3,1),'%6.4f'));
set(handles.text10,'string',num2str(result.T(3,2),'%6.4f'));
set(handles.text11,'string',num2str(result.T(3,3),'%6.4f'));
set(handles.text12,'string',num2str(result.T(3,4),'%6.4f'));

set(handles.edit5,'string',num2str(angles(1),'%6.4f'));
set(handles.edit6,'string',num2str(angles(2),'%6.4f'));
set(handles.edit7,'string',num2str(angles(3),'%6.4f'));
set(handles.edit8,'string',num2str(angles(4),'%6.4f'));

%%


% --- Axis X slider movement.
function slider1_Callback(hObject, eventdata, handles)
% hObject    handle to edit9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range
of slider

temp1=(get(handles.slider1,'value'));

set(handles.edit1,'String',temp1);
set(handles.edit9,'String',temp1);
set(handles.text4,'string',temp1);

pushbutton1_Callback(hObject, eventdata, handles);


% --- Executes during object creation, after setting all properties.
function slider1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end


% --- Axis Y slider movement.
function slider2_Callback(hObject, eventdata, handles)
% hObject    handle to slider2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range
of slider
```

```
temp2=(get(handles.slider2,'value'));

set(handles.edit2,'String',temp2);
set(handles.edit10,'String',temp2);
set(handles.text8,'string',temp2);

pushbutton1_Callback(hObject, eventdata, handles);


% --- Executes during object creation, after setting all properties.
function slider2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end


% --- Axis Z slider movement.
function slider3_Callback(hObject, eventdata, handles)
% hObject    handle to slider3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range
of slider

temp3=(get(handles.slider3,'value'));

set(handles.edit3,'String',temp3);
set(handles.edit11,'String',temp3);
set(handles.text12,'string',temp3);

pushbutton1_Callback(hObject, eventdata, handles)


% --- Executes during object creation, after setting all properties.
function slider3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end


% --- Gripper slider movement.
function slider4_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to slider4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range
of slider

temp4=(get(handles.slider4,'value'));
set(handles.edit4,'String',temp4);

pushbutton1_Callback(hObject, eventdata, handles)


% --- Executes during object creation, after setting all properties.
function slider4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end


% --- Simulate! pushbutton.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

[handles] = inverse(handles);

temp1=str2double(get(handles.edit5,'string'));
temp2=str2double(get(handles.edit6,'string'));
temp3=str2double(get(handles.edit7,'string'));
temp4=str2double(get(handles.edit8,'string'));
temp5=str2double(get(handles.edit4,'string'));

opts = simset('SrcWorkspace','current');

value_1=temp1;
value_2=temp2;
value_3=temp3;
value_4=temp4;
value_5=temp5;

sim('gui_roboarm',[],opts);


% --- Start point pushbutton.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
set(handles.edit1,'string','0');
set(handles.edit9,'string','0');
set(handles.edit2,'string','0.2667');
set(handles.edit10,'string','0.2667');
set(handles.edit3,'string','0.0258');
set(handles.edit11,'string','0.0258');
set(handles.edit4,'string','0');
set(handles.slider1,'value',0);
set(handles.slider2,'value',0.2667);
set(handles.slider3,'value',0.0258);
set(handles.slider4,'value',0);

set(handles.edit5,'string','0');
set(handles.edit6,'string','28');
set(handles.edit7,'string','38');
set(handles.edit8,'string','24');

set(handles.text1,'string','0.0000');
set(handles.text2,'string','0.0000');
set(handles.text3,'string','-1.0000');
set(handles.text4,'string','0.0000');
set(handles.text5,'string','0.0000');
set(handles.text6,'string','-1.0000');
set(handles.text7,'string','0.0000');
set(handles.text8,'string','0.2667');
set(handles.text9,'string','-1.0000');
set(handles.text10,'string','0.0000');
set(handles.text11,'string','0.0000');
set(handles.text12,'string','0.0258');

pushbutton1_Callback(hObject, eventdata, handles);


% --- Exit pushbutton.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

delete(inverse_simulation);


% --- Close gripper pushbutton.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

set(handles.edit4,'string','100');
set(handles.slider4,'value',100);

temp1=str2double(get(handles.edit5,'string'));
temp2=str2double(get(handles.edit6,'string'));
temp3=str2double(get(handles.edit7,'string'));
temp4=str2double(get(handles.edit8,'string'));
temp5=str2double(get(handles.edit4,'string'));

opts = simset('SrcWorkspace','current');
```

```
value_1=temp1;
value_2=temp2;
value_3=temp3;
value_4=temp4;
value_5=temp5;

sim('gui_roboarm',[],opts);


% --- Open gripper pushbutton.
function pushbutton5_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

set(handles.edit4,'string','0');
set(handles.slider4,'value',0);

temp1=str2double(get(handles.edit5,'string'));
temp2=str2double(get(handles.edit6,'string'));
temp3=str2double(get(handles.edit7,'string'));
temp4=str2double(get(handles.edit8,'string'));
temp5=str2double(get(handles.edit4,'string'));

opts = simset('SrcWorkspace','current');

value_1=temp1;
value_2=temp2;
value_3=temp3;
value_4=temp4;
value_5=temp5;

sim('gui_roboarm',[],opts);


% --- Axis X slider edit.
function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1
as a double

n=str2double(get(handles.edit1,'string'));

if isnan(n)
    errordlg('You must enter a numeric value between -0.55 and 0.55
degrees.','Invalid value','modal');
    temp1=(get(handles.slider1,'value'));
    set(handles.edit1,'String',temp1);
else
    if (n>0.55 || n<-0.55)
        errordlg('You must enter a value between -0.55 and 0.55
degrees.','Invalid value','modal');
        temp1=(get(handles.slider1,'value'));
        set(handles.edit1,'String',temp1);
    else
```

```
        set(handles.slider1,'value',n);
    end
end

temp1=(get(handles.slider1,'value'));

set(handles.edit9,'String',temp1);
set(handles.text4,'string',temp1);

pushbutton1_Callback(hObject, eventdata, handles);


% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Axis Y slider edit.
function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2
as a double

n=str2double(get(handles.edit2,'string'));

if isnan(n)
    errordlg('You must enter a numeric value between 0.1 and 0.55
degrees.','Invalid value','modal');
    temp2=(get(handles.slider2,'value'));
    set(handles.edit2,'String',temp2);
else
    if (n>0.55 || n<0.1)
        errordlg('You must enter a value between 0.1 and 0.55
degrees.','Invalid value','modal');
        temp2=(get(handles.slider2,'value'));
        set(handles.edit2,'String',temp2);
    else
        set(handles.slider2,'value',n);
    end
end

temp2=(get(handles.slider2,'value'));

set(handles.edit10,'String',temp2);
set(handles.text8,'string',temp2);
```

```
pushbutton1_Callback(hObject, eventdata, handles);


% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



% --- Axis Z slider edit.
function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3
as a double

n=str2double(get(handles.edit3,'string'));

if isnan(n)
    errordlg('You must enter a numeric value between 0 and 0.72
degrees.','Invalid value','modal');
    temp3=(get(handles.slider3,'value'));
    set(handles.edit3,'String',temp3);
else
    if (n>0.72 || n<0)
        errordlg('You must enter a value between 0 and 0.72
degrees.','Invalid value','modal');
        temp3=(get(handles.slider3,'value'));
        set(handles.edit3,'String',temp3);
    else
        set(handles.slider3,'value',n);
    end
end

temp3=(get(handles.slider3,'value'));

set(handles.edit11,'String',temp3);
set(handles.text12,'string',temp3);

pushbutton1_Callback(hObject, eventdata, handles);


% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Gripper edit.
function edit4_Callback(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit4 as text
%        str2double(get(hObject,'String')) returns contents of edit4
as a double
n=str2double(get(handles.edit4,'string'));
if isnan(n)
    errordlg('You must enter a numeric value between 0 and
100.','Invalid value','modal');
    temp4=(get(handles.slider4,'value'));
    set(handles.edit4,'String',temp4);
else
    if (n>100 || n<0)
        errordlg('You must enter a value between 0 and 100.','Invalid
value','modal');
        temp4=(get(handles.slider4,'value'));
        set(handles.edit4,'String',temp4);
    else
        set(handles.slider4,'value',n);
    end
end
pushbutton1_Callback(hObject, eventdata, handles);


% --- Executes during object creation, after setting all properties.
function edit4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Angle th1 edit.
function edit5_Callback(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hints: get(hObject,'String') returns contents of edit5 as text
%        str2double(get(hObject,'String')) returns contents of edit5
as a double


% --- Executes during object creation, after setting all properties.
function edit5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Angle th2 edit.
function edit6_Callback(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit6 as text
%        str2double(get(hObject,'String')) returns contents of edit6
as a double


% --- Executes during object creation, after setting all properties.
function edit6_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Angle th3 edit.
function edit7_Callback(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit7 as text
%        str2double(get(hObject,'String')) returns contents of edit7
as a double


% --- Executes during object creation, after setting all properties.
```

```
function edit7_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Angle th4 edit.
function edit8_Callback(hObject, eventdata, handles)
% hObject    handle to edit8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit8 as text
%        str2double(get(hObject,'String')) returns contents of edit8
as a double


% --- Executes during object creation, after setting all properties.
function edit8_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Axis X edit.
function edit9_Callback(hObject, eventdata, handles)
% hObject    handle to edit9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit9 as text
%        str2double(get(hObject,'String')) returns contents of edit9
as a double

    temp1=(get(handles.slider1,'value'));

    set(handles.edit9,'String',temp1);


% --- Executes during object creation, after setting all properties.
function edit9_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit9 (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Axis Y edit.
function edit10_Callback(hObject, eventdata, handles)
% hObject    handle to edit10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit10 as text
%        str2double(get(hObject,'String')) returns contents of edit10
as a double


% --- Executes during object creation, after setting all properties.
function edit10_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Axis Z edit.
function edit11_Callback(hObject, eventdata, handles)
% hObject    handle to edit11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit11 as text
%        str2double(get(hObject,'String')) returns contents of edit11
as a double


% --- Executes during object creation, after setting all properties.
function edit11_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
```

```
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --------------------------------------------------------------------
-
function Help_Callback(hObject, eventdata, handles)
% hObject    handle to Help (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

open('user_guide.pdf');


% --------------------------------------------------------------------
-
function Kinematics_Callback(hObject, eventdata, handles)
% hObject    handle to Kinematics (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
-
function Simulation_Mode_Callback(hObject, eventdata, handles)
% hObject    handle to Simulation_Mode (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
-
function Real_Mode_Callback(hObject, eventdata, handles)
% hObject    handle to Real_Mode (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
-
function Forward_Kinematics_Real_Mode_Callback(hObject, eventdata,
handles)
% hObject    handle to Forward_Kinematics_Real_Mode (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
-
function Inverse_Kinematics_Real_Mode_Callback(hObject, eventdata,
handles)
% hObject    handle to Inverse_Kinematics_Real_Mode (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% --------------------------------------------------------------------
-
function Forward_Kinematics_Simulation_Mode_Callback(hObject,
eventdata, handles)
% hObject    handle to Forward_Kinematics_Simulation_Mode (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
-
function Inverse_Kinematics_Simulation_Mode_Callback(hObject,
eventdata, handles)
% hObject    handle to Inverse_Kinematics_Simulation_Mode (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
-
function Exit_Callback(hObject, eventdata, handles)
% hObject    handle to Exit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

inverse_simple_calc.m

Το inverse_simple_calc.m υπολογίζει το αντίστροφο κινηματικό πρόβλημα.

```
function [angles]=inverse_simple_calc(xx,yy,zz)

% th1 evaluation.
if yy==0 & xx<0, th1= 90;,                  end
if yy==0 & xx>0, th1=-90;,                  end
if yy~=0,        th1=-atan(xx/yy)*180/pi;, end

% Lengths definition.

A=0.1047;, L1=0.2910;, d=0.0220;, L2=0.2720;, L3=0.0770;

Y=sqrt(L1^2+d^2);

w=acos(L1/Y)*180/pi;

px=sqrt(xx^2+yy^2);, py=zz+L3-A;

th3=w-asin((px^2+py^2-Y^2-L2^2)/2/Y/L2)*180/pi;

K=Y*cos(w*pi/180)-L2*sin(th3*pi/180);
L=Y*sin(w*pi/180)+L2*cos(th3*pi/180);

th2=asin((px*K-py*L)/(L^2+K^2))*180/pi;

th4=90-th2-th3;

angles=[th1 th2 th3 th4];
```

inverse_real.m

Το inverse_real.m δημιουργεί τη γραφική διεπαφή για την πραγματική κίνηση του βραχίονα, χρησιμοποιώντας το αντίστροφο κινηματικό πρόβλημα.

```
function varargout = inverse_real(varargin)
% INVERSE_REAL M-file for inverse_real.fig
%      INVERSE_REAL, by itself, creates a new INVERSE_REAL or raises
the existing
%      singleton*.
%
%      H = INVERSE_REAL returns the handle to a new INVERSE_REAL or
the handle to
%      the existing singleton*.
%
%      INVERSE_REAL('CALLBACK',hObject,eventData,handles,...) calls
the local
%      function named CALLBACK in INVERSE_REAL.M with the given input
arguments.
%
%      INVERSE_REAL('Property','Value',...) creates a new
INVERSE_REAL or raises the
%      existing singleton*.  Starting from the left, property value
pairs are
%      applied to the GUI before inverse_real_OpeningFcn gets called.
An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to inverse_real_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows
only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help inverse_real

% Last Modified by GUIDE v2.5 30-Sep-2009 22:58:31

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @inverse_real_OpeningFcn, ...
                   'gui_OutputFcn',  @inverse_real_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
```

```
% --- Executes just before inverse_real is made visible.
function inverse_real_OpeningFcn(hObject, eventdata, handles,
varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to inverse_real (see VARARGIN)

% Choose default command line output for inverse_real
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes inverse_real wait for user response (see UIRESUME)
% uiwait(handles.figure1

%Initialization of the robot's position
pushbutton2_Callback(hObject, eventdata, handles)


% --- Outputs from this function are returned to the command line.
function varargout = inverse_real_OutputFcn(hObject, eventdata,
handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Total trasformation matrix.
function [handles] = inverse(handles)
xx=(get(handles.slider1,'value'));
yy=(get(handles.slider2,'value'));
zz=(get(handles.slider3,'value'));

gripper=(get(handles.slider4,'value'));

angles=inverse_simple_calc(xx,yy,zz);
global result;
result=forward_calc([angles],gripper);

set(handles.text1,'string',num2str(result.T(1,1),'%6.4f'));
set(handles.text2,'string',num2str(result.T(1,2),'%6.4f'));
set(handles.text3,'string',num2str(result.T(1,3),'%6.4f'));
set(handles.text4,'string',num2str(result.T(1,4),'%6.4f'));
set(handles.text5,'string',num2str(result.T(2,1),'%6.4f'));
set(handles.text6,'string',num2str(result.T(2,2),'%6.4f'));
set(handles.text7,'string',num2str(result.T(2,3),'%6.4f'));
set(handles.text8,'string',num2str(result.T(2,4),'%6.4f'));
set(handles.text9,'string',num2str(result.T(3,1),'%6.4f'));
set(handles.text10,'string',num2str(result.T(3,2),'%6.4f'));
set(handles.text11,'string',num2str(result.T(3,3),'%6.4f'));
set(handles.text12,'string',num2str(result.T(3,4),'%6.4f'));
```

```
set(handles.edit5,'string',num2str(angles(1),'%6.4f'));
set(handles.edit6,'string',num2str(angles(2),'%6.4f'));
set(handles.edit7,'string',num2str(angles(3),'%6.4f'));
set(handles.edit8,'string',num2str(angles(4),'%6.4f'));

%%


% --- Axis X slider movement.
function slider1_Callback(hObject, eventdata, handles)
% hObject    handle to edit9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range
of slider

temp1=(get(handles.slider1,'value'));

set(handles.edit1,'String',temp1);
set(handles.edit9,'String',temp1);
set(handles.text4,'string',temp1);

pushbutton1_Callback(hObject, eventdata, handles);


% --- Executes during object creation, after setting all properties.
function slider1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end


% --- Axis Y slider movement.
function slider2_Callback(hObject, eventdata, handles)
% hObject    handle to slider2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range
of slider

temp2=(get(handles.slider2,'value'));

set(handles.edit2,'String',temp2);
set(handles.edit10,'String',temp2);
set(handles.text8,'string',temp2);

pushbutton1_Callback(hObject, eventdata, handles);
```

```
% --- Executes during object creation, after setting all properties.
function slider2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end


% --- Axis Z slider movement.
function slider3_Callback(hObject, eventdata, handles)
% hObject    handle to slider3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range
of slider

temp3=(get(handles.slider3,'value'));

set(handles.edit3,'String',temp3);
set(handles.edit11,'String',temp3);
set(handles.text12,'string',temp3);

pushbutton1_Callback(hObject, eventdata, handles)


% --- Executes during object creation, after setting all properties.
function slider3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end


% --- Executes on slider movement.
function slider4_Callback(hObject, eventdata, handles)
% hObject    handle to slider4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range
of slider

temp4=(get(handles.slider4,'value'));
```

```
set(handles.edit4,'String',temp4);

pushbutton1_Callback(hObject, eventdata, handles)


% --- Executes during object creation, after setting all properties.
function slider4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end


% --- Simulate! pushbutton.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

[handles] = inverse(handles);

temp1=str2double(get(handles.edit5,'string'));
temp2=str2double(get(handles.edit6,'string'));
temp3=str2double(get(handles.edit7,'string'));
temp4=str2double(get(handles.edit8,'string'));
temp5=str2double(get(handles.edit4,'string'));

opts = simset('SrcWorkspace','current');

value_1=temp1;
value_2=temp2;
value_3=temp3;
value_4=temp4;
value_5=temp5;

sim('gui_roboarm',[],opts);


% --- Start point pushbutton.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

set(handles.edit1,'string','0');
set(handles.edit9,'string','0');
set(handles.edit2,'string','0.2667');
set(handles.edit10,'string','0.2667');
set(handles.edit3,'string','0.0258');
set(handles.edit11,'string','0.0258');
set(handles.edit4,'string','0');
set(handles.slider1,'value',0);
```

```
set(handles.slider2,'value',0.2667);
set(handles.slider3,'value',0.0258);
set(handles.slider4,'value',0);

set(handles.edit5,'string','0');
set(handles.edit6,'string','28');
set(handles.edit7,'string','38');
set(handles.edit8,'string','24');

set(handles.text1,'string','0.0000');
set(handles.text2,'string','0.0000');
set(handles.text3,'string','-1.0000');
set(handles.text4,'string','0.0000');
set(handles.text5,'string','0.0000');
set(handles.text6,'string','-1.0000');
set(handles.text7,'string','0.0000');
set(handles.text8,'string','0.2667');
set(handles.text9,'string','-1.0000');
set(handles.text10,'string','0.0000');
set(handles.text11,'string','0.0000');
set(handles.text12,'string','0.0258');

pushbutton1_Callback(hObject, eventdata, handles);


% --- Exit pushbutton.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

delete(inverse_real);


% --- Close gripper pushbutton.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

set(handles.edit4,'string','100');
set(handles.slider4,'value',100);

temp1=str2double(get(handles.edit5,'string'));
temp2=str2double(get(handles.edit6,'string'));
temp3=str2double(get(handles.edit7,'string'));
temp4=str2double(get(handles.edit8,'string'));
temp5=str2double(get(handles.edit4,'string'));

opts = simset('SrcWorkspace','current');

value_1=temp1;
value_2=temp2;
value_3=temp3;
value_4=temp4;
value_5=temp5;

sim('gui_roboarm',[],opts);
```

```
% --- Open gripper pushbutton.
function pushbutton5_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

set(handles.edit4,'string','0');
set(handles.slider4,'value',0);

temp1=str2double(get(handles.edit5,'string'));
temp2=str2double(get(handles.edit6,'string'));
temp3=str2double(get(handles.edit7,'string'));
temp4=str2double(get(handles.edit8,'string'));
temp5=str2double(get(handles.edit4,'string'));

opts = simset('SrcWorkspace','current');

value_1=temp1;
value_2=temp2;
value_3=temp3;
value_4=temp4;
value_5=temp5;

sim('gui_roboarm',[],opts);

pushbutton1_Callback(hObject, eventdata, handles);


% --- Move robot pushbutton.
function pushbutton6_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Serial communication
[handles] = inverse_serial(handles);


% --- Axis X slider edit.
function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1
as a double

n=str2double(get(handles.edit1,'string'));

if isnan(n)
    errordlg('You must enter a numeric value between -0.55 and 0.55
degrees.','Invalid value','modal');
    temp1=(get(handles.slider1,'value'));
    set(handles.edit1,'String',temp1);
else
    if (n>0.55 || n<-0.55)
```

```
        errordlg('You must enter a value between -0.55 and 0.55
degrees.','Invalid value','modal');
        temp1=(get(handles.slider1,'value'));
        set(handles.edit1,'String',temp1);
    else
        set(handles.slider1,'value',n);
    end
end

temp1=(get(handles.slider1,'value'));

set(handles.edit9,'String',temp1);
set(handles.text4,'string',temp1);

pushbutton1_Callback(hObject, eventdata, handles);


% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Axis Y slider edit.
function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2
as a double

n=str2double(get(handles.edit2,'string'));

if isnan(n)
    errordlg('You must enter a numeric value between 0.1 and 0.55
degrees.','Invalid value','modal');
    temp2=(get(handles.slider2,'value'));
    set(handles.edit2,'String',temp2);
else
    if (n>0.55 || n<0.1)
        errordlg('You must enter a value between 0.1 and 0.55
degrees.','Invalid value','modal');
        temp2=(get(handles.slider2,'value'));
        set(handles.edit2,'String',temp2);
    else
        set(handles.slider2,'value',n);
    end
end
```

```
temp2=(get(handles.slider2,'value'));

set(handles.edit10,'String',temp2);
set(handles.text8,'string',temp2);

pushbutton1_Callback(hObject, eventdata, handles);


% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Axis Z slider edit.
function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3
as a double

n=str2double(get(handles.edit3,'string'));

if isnan(n)
    errordlg('You must enter a numeric value between 0 and 0.72
degrees.','Invalid value','modal');
    temp3=(get(handles.slider3,'value'));
    set(handles.edit3,'String',temp3);
else
    if (n>0.72 || n<0)
        errordlg('You must enter a value between 0 and 0.72
degrees.','Invalid value','modal');
        temp3=(get(handles.slider3,'value'));
        set(handles.edit3,'String',temp3);
    else
        set(handles.slider3,'value',n);
    end
end

temp3=(get(handles.slider3,'value'));

set(handles.edit11,'String',temp3);
set(handles.text12,'string',temp3);

pushbutton1_Callback(hObject, eventdata, handles);
```

```
% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



% --- Gripper edit.
function edit4_Callback(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit4 as text
%        str2double(get(hObject,'String')) returns contents of edit4
as a double

n=str2double(get(handles.edit4,'string'));

if isnan(n)
    errordlg('You must enter a numeric value between 0 and
100.','Invalid value','modal');
    temp4=(get(handles.slider4,'value'));
    set(handles.edit4,'String',temp4);
else
    if (n>100 || n<0)
        errordlg('You must enter a value between 0 and 100.','Invalid
value','modal');
        temp4=(get(handles.slider4,'value'));
        set(handles.edit4,'String',temp4);
    else
        set(handles.slider4,'value',n);
    end
end

pushbutton1_Callback(hObject, eventdata, handles);



% --- Executes during object creation, after setting all properties.
function edit4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
% --- Angle th1 edit.
function edit5_Callback(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit5 as text
%        str2double(get(hObject,'String')) returns contents of edit5
as a double


% --- Executes during object creation, after setting all properties.
function edit5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Angle th2 edit.
function edit6_Callback(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit6 as text
%        str2double(get(hObject,'String')) returns contents of edit6
as a double


% --- Executes during object creation, after setting all properties.
function edit6_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Angle th3 edit.
function edit7_Callback(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit7 as text
```

```
%        str2double(get(hObject,'String')) returns contents of edit7
as a double


% --- Executes during object creation, after setting all properties.
function edit7_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Angle th4 edit.
function edit8_Callback(hObject, eventdata, handles)
% hObject    handle to edit8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit8 as text
%        str2double(get(hObject,'String')) returns contents of edit8
as a double


% --- Executes during object creation, after setting all properties.
function edit8_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Axis X edit.
function edit9_Callback(hObject, eventdata, handles)
% hObject    handle to edit9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit9 as text
%        str2double(get(hObject,'String')) returns contents of edit9
as a double

    temp1=(get(handles.slider1,'value'));

    set(handles.edit9,'String',temp1);
```

```
% --- Executes during object creation, after setting all properties.
function edit9_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Axis Y edit.
function edit10_Callback(hObject, eventdata, handles)
% hObject    handle to edit10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit10 as text
%        str2double(get(hObject,'String')) returns contents of edit10
as a double


% --- Executes during object creation, after setting all properties.
function edit10_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Axis Z edit.
function edit11_Callback(hObject, eventdata, handles)
% hObject    handle to edit11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit11 as text
%        str2double(get(hObject,'String')) returns contents of edit11
as a double


% --- Executes during object creation, after setting all properties.
function edit11_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called
```

```
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --------------------------------------------------------------------
-
function Help_Callback(hObject, eventdata, handles)
% hObject    handle to Help (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

open('user_guide.pdf');


% --------------------------------------------------------------------
-
function Kinematics_Callback(hObject, eventdata, handles)
% hObject    handle to Kinematics (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
-
function Serial_Port_Callback(hObject, eventdata, handles)
% hObject    handle to Serial_Port (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
-
function Simulation_Mode_Callback(hObject, eventdata, handles)
% hObject    handle to Simulation_Mode (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
-
function Real_Mode_Callback(hObject, eventdata, handles)
% hObject    handle to Real_Mode (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
-
function Forward_Kinematics_Real_Mode_Callback(hObject, eventdata,
handles)
% hObject    handle to Forward_Kinematics_Real_Mode (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% --------------------------------------------------------------------
-
function Inverse_Kinematics_Real_Mode_Callback(hObject, eventdata,
handles)
% hObject    handle to Inverse_Kinematics_Real_Mode (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
-
function Forward_Kinematics_Simulation_Mode_Callback(hObject,
eventdata, handles)
% hObject    handle to Forward_Kinematics_Simulation_Mode (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
-
function Inverse_Kinematics_Simulation_Mode_Callback(hObject,
eventdata, handles)
% hObject    handle to Inverse_Kinematics_Simulation_Mode (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
-
function Exit_Callback(hObject, eventdata, handles)
% hObject    handle to Exit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

inverse_serial.m

Το forward_serial.m χρησιμοποιείται για το αντίστροφο κινηματικό και είναι υπεύθυνο, για τη σειριακή επικοινωνία του υπολογιστή με τον πραγματικό βραχίονα και τον έλεγχο των πέντε σερβοκινητήρων του.

```
function [handles] = inverse_serial(handles)

%% Create serial port

s=serial('COM1');
s.BaudRate = 115200;
s.Terminator = 'CR/LF';
fopen(s);
%Sport=instrfind

%% Servo move

% Servo motor #1
angle1 = (get(handles.edit5,'string'));
angle1 = str2double(angle1);

x= 0.1*(angle1)+10;
pwm1 = -93.04*x + 2347.9;
pwm1=round(pwm1)

servo1 = sprintf('#0 P%d T3000',pwm1);
fprintf(s,servo1);

% Servo motor #2
angle2 = (get(handles.edit6,'string'));
angle2 = str2double(angle2);

pwm2 = 8.7932*angle2 + 1375.1;
pwm2=round(pwm2)

servo2 = sprintf('#1 P%d T3000',pwm2);
fprintf(s,servo2);

% Servo motor #3
angle3 = (get(handles.edit7,'string'));
angle3 = str2double(angle3);

pwm3 = -9.765*angle3 + 1537.3;
pwm3=round(pwm3)

servo3 = sprintf('#2 P%d T3000',pwm3);
fprintf(s,servo3);

% Servo motor #4
angle4 = (get(handles.edit8,'string'));
angle4 = str2double(angle4);

pwm4 = -10.3*angle4 + 1508.8;
pwm4=round(pwm4)

servo4 = sprintf('#3 P%d T3000',pwm4);
```

```
fprintf(s,servo4);

% Servo motor #5
angle5 = (get(handles.edit4,'string'));
angle5 = str2double(angle5);

pwm5 = 11.65*angle5 + 1335.14;
pwm5=round(pwm5)

servo5 = sprintf('#4 P%d T3000',pwm5);
fprintf(s,servo5);

%% Delete serial port

fclose(s);
delete(s);
%Sport=instrfind
```

serial_port.m

Το serial_port.m χρησιμοποιείται για την επιλογή σειριακής θύρας στον υπολογιστή.

```
function serial_port(hObject, eventdata, handles)
prompt = {'Enter Communication Port Number:'};
dlg_title = 'Select Serial Port';
num_lines = 1;
def = {'1'};
options.Resize='on';

commport = inputdlg(prompt,dlg_title,num_lines,def,options);
commport = str2double(commport);

if isnan(commport)
    errordlg('Invalid Serial Port.','Bad Input','modal');
    commport = 1;
end

if (commport <1 || commport >16)
    errordlg('Enter serial between 1 and 16.','Bad Input','modal');
else
    commport = num2str(commport);
    msgbox(['COM',commport,' Selected'],'Selected serial','modal');
end
%%
```

gui_robot.m

Το gui_robot.m δημιουργεί τη γραφική διεπαφή του κεντρικού μενού επιλογής, για την προσομοίωση ή την πραγματική λειτουργία του εικονικού βραχίονα.

```
function varargout = gui_robot(varargin)
% GUI_ROBOT M-file for gui_robot.fig
%      GUI_ROBOT, by itself, creates a new GUI_ROBOT or raises the
existing
%      singleton*.
%
%      H = GUI_ROBOT returns the handle to a new GUI_ROBOT or the
handle to
%      the existing singleton*.
%
%      GUI_ROBOT('CALLBACK',hObject,eventData,handles,...) calls the
local
%      function named CALLBACK in GUI_ROBOT.M with the given input
arguments.
%
%      GUI_ROBOT('Property','Value',...) creates a new GUI_ROBOT or
raises the
%      existing singleton*.  Starting from the left, property value
pairs are
%      applied to the GUI before gui_robot_OpeningFcn gets called.
An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to gui_robot_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows
only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help gui_robot

% Last Modified by GUIDE v2.5 10-Nov-2009 13:17:38

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @gui_robot_OpeningFcn, ...
                   'gui_OutputFcn',  @gui_robot_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
```

```
% --- Executes just before gui_robot is made visible.
function gui_robot_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to gui_robot (see VARARGIN)

% Choose default command line output for gui_robot
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes gui_robot wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% Load the background image into Matlab
backgroundImage = importdata('robot.jpg');

% Select the axes
axes(handles.axes1);

% Place image onto the axes
image(backgroundImage);

% Remove the axis tick marks
axis off

%%


% --- Outputs from this function are returned to the command line.
function varargout = gui_robot_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

set(hObject,'Visible','on');

%%


% --- Forward kinematics simulation mode pushbutton.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
delete(gui_robot);
forward_simulation

%%
```

```
% --- Inverse kinematics simulation mode pushbutton.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
delete(gui_robot);
inverse_simulation

%%


% --- Forward kinematics real mode pushbutton.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
delete(gui_robot);
forward_real

%%


% --- Inverse kinematics real mode pushbutton.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
delete(gui_robot);
inverse_real

%%


% --- Exit pushbutton.
function pushbutton5_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
delete(gui_robot);

%%
```