ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΜΗΧΑΝΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΤΩΝ ΚΑΤΕΡΓΑΣΙΩΝ

Προγραμματισμός Τροχιάς και Τηλεπαρακολούθηση Λειτουργίας Ρομποτικού Βραχίονα σε Περιβάλλον Εικονικής Πραγματικότητας με χρήση Αισθητήρων Ηλεκτρονικών Συσκευών Ευρείας Κατανάλωσης

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΣΕΡΑΦΕΙΜ ΜΙΧΑΣ

Επιβλέπων : ΓΕΩΡΓΙΟΣ-ΧΡΙΣΤΟΦΟΡΟΣ ΒΟΣΝΙΑΚΟΣ

ΚΑΘΗΓΗΤΗΣ

Αθήνα, Οκτώβριος 2015

Περίληψη

Στην παρούσα διπλωματική εργασία μελετήθηκε η μεθοδολογία και τα εργαλεία ανάπτυξης του εικονικού ισοδύναμου ενός πραγματικού βιομηχανικού χώρου. Συγκεκριμένα, μοντελοποιήθηκε ο ρομποτικός βραχίονας Stäubli RX90L που βρίσκεται στον Τομέα Τεχνολογίας των Κατεργασιών του ΕΜΠ καθώς και ο περιβάλλων χώρος του εργαστηρίου, χρησιμοποιώντας το λογισμικό ανάπτυξης τρισδιάστατων εφαρμογών, Unity 3D. Στόχος είναι η διερεύνηση του offline προγραμματισμού του βραχίονα σε περιβάλλον εικονικής πραγματικότητας, καθώς και το ακριβώς αντίθετο, δηλαδή η τηλεπαρακολούθηση της λειτουργίας του.

Στα πλαίσια της μοντελοποίησης έγινε αρχικά εισαγωγή της γεωμετρίας του εργαστηρίου και του μοντέλου του ρομπότ και αναπτύχθηκε η κινηματική του αλυσίδα. Ακολούθησε η ευθεία και η αντίστροφη κινηματική ανάλυση του βραχίονα, η οποία προγραμματίστηκε με τη γλώσσα C# σε μορφή script. Το script της αντίστροφης κινηματικής επισυνάπτεται στο μοντέλο του ρομπότ χρησιμοποιώντας την λειτουργία component που διαθέτει το Unity. Για την υλοποίηση του προγραμματισμού τροχιάς χρησιμοποιήθηκε το χειριστήριο της κονσόλας Xbox 360, ενώ για την τηλεπαρακολούθηση της λειτουργίας του μελετήθηκε το Wii Remote της Nintendo. Η διερεύνηση έγινε μάλιστα με τέτοιο τρόπο, ώστε τα χειριστήρια να λειτουργούν πάνω στην ίδια εφαρμογή και να μην χρειάζεται «χτίσιμο» δύο διαφορετικών προγραμμάτων.

Απώτερος στόχος της παραπάνω ανάλυσης είναι η εισαγωγή νέων, αποδοτικότερων και ασφαλέστερων μεθοδολογιών διαχείρισης βαρέως μηχανολογικού εξοπλισμού. Αν και το αποτέλεσμα είναι ακόμη σε αρχικό στάδιο, η ταχεία ανάπτυξη της εικονικής πραγματικότητας και των διαθέσιμων αισθητήρων υπόσχονται πολλές βελτιώσεις, που μπορούν καταστήσουν τα νέα αυτά συστήματα ανταγωνιστικά στην αγορά.

Ευχαριστίες

Θα ήθελα να εκφράσω τις θερμές που ευχαριστίες στον κ. Γ.-Χ. Βοσνιάκο, καθηγητή του Ε.Μ.Π., για την ανάθεση της παρούσας διπλωματικής εργασίας, καθώς και για την υποστήριξη και τις συμβουλές που μου παρείχε κατά τη διάρκεια της εκπόνησής της.

Ευχαριστώ επίσης τον Ηλία Μάτσα, Υποψήφιο Διδάκτορα της σχολής Μηχανολόγων Μηχανικών Ε.Μ.Π., για όλη την καθοδήγηση που μου παρείχε σχετικά με το θέμα της εικονικής πραγματικότητας, καθώς και τον Δρ. Ξενοφώντα Γωγουβίτη, Μηχανολόγο Μηχανικό Ε.Μ.Π., για τη συμβολή του στην μελέτη της τηλεπαρακολούθησης.

Τέλος, δεν θα μπορούσα να παραλείψω του γονείς μου, τους οποίους ευχαριστώ πολύ για όλη την στήριξη που μου παρείχαν καθ' όλη την διάρκεια των σπουδών μου.

Σεραφείμ Μίχας
Αθήνα, Οκτώβριος 2015

Πίνακας περιεχομένων

Εικόνες

Πίνακες

1 Εισαγωγή

1.1 Η εικονική πραγματικότητα

1.1.1 Ορισμός

Ο όρος Εικονική Πραγματικότητα χρησιμοποιήθηκε για πρώτη φορά από τον Jaron Lanier (Τζάρον Λέινιερ) το 1989. Ο Lanier είναι ένας από τους πρωτοπόρους της Εικονικής Πραγματικότητας και ιδρυτής της εταιρείας VPL Research (από τη φράση Virtual Programming Languages), η οποία ανέπτυξε μερικά από τα πρώτα συστήματα τη δεκαετία του 1980. Σαφής και αυστηρός ορισμός δεν υπάρχει, μιας και ο ίδιος ο όρος είναι αντιφατικός και οδηγεί σε παρεξηγήσεις. Ο ίδιος ο «πατέρας» του όρου, περιέγραψε την Εικονική Πραγματικότητα ως:

"Ένα αλληλεπιδραστικό, τρισδιάστατο περιβάλλον, φτιαγμένο από υπολογιστή, στο οποίο μπορεί κάποιος να εμβυθιστεί."

Έπειτα από αυτόν δόθηκαν ποικίλοι ορισμοί από διάφορους ερευνητές, εκ των οποίων, ένας από τους περιγραφικότερους είναι αυτός που δόθηκε από τους C.Manetta και Blade R. το 1995:

"Ένα υπολογιστικό σύστημα το οποίο χρησιμοποιείται για τη δημιουργία εικονικών κόσμων, στους οποίους ο χρήστης έχει την εντύπωση της ύπαρξης του σε αυτούς και επιπλέον έχει την ικανότητα να πλοηγηθεί και να χειριστεί τα αντικείμενά τους."

Δηλαδή, η Εικονική Πραγματικότητα χρησιμοποιεί ηλεκτρονικούς υπολογιστές, για να δημιουργήσει και να προσομοιώσει υπαρκτά ή μη περιβάλλοντα, από τα οποία ο χρήστης έχει την ψευδαίσθηση ότι περιβάλλεται και στα οποία μπορεί να κινηθεί

ελεύθερα, αλληλεπιδρώντας παράλληλα με τα αντικείμενα που περιλαμβάνουν, όπως θα έκανε και στον πραγματικό κόσμο.

Τελευταία, στην επιστημονική κοινότητα αποφεύγεται η χρήση του όρου Εικονική Πραγματικότητα λόγω της αντιφατικότητάς του και εναλλακτικά χρησιμοποιείται ο όρος Εικονικό Περιβάλλον ή Virtual Environment στα αγγλικά (VE). [1]

1.1.2 Ιστορική Αναδρομή

Οι πρώτη αναφορά στην εικονική πραγματικότητα γίνεται πριν το 1950 και συναντάται στην ταινία επιστημονικής φαντασίας "Pygmalion's Spectacles". Η ιστορία περιγράφει ένα σύστημα εικονικής πραγματικότητας βασισμένο στην χρήση γυαλιών, το οποίο εκτός από εικόνα, δίνει στον χρήστη και την ρεαλιστική αίσθηση της όσφρησης και της αφής.

Στη δεκαετία του 1950 ο Αμερικανός κινηματογραφιστής Morton Heilig προτείνει «το σινεμά του μέλλοντος», το οποίο θα περικυκλώνει το θεατή με αισθήσεις φτιαγμένες από μηχανήματα και θα μεταφέρει τους θεατές σε μια άλλη διάσταση. [2] Το Sensorama που κατασκευάζεται από τον ίδιο το 1956, προσφέρει μια βόλτα με μοτοσυκλέτα στους δρόμους του Μανχάταν. Χρησιμοποιούνται 3Δ γραφικά, στερεοσκοπικός ήχος και συσκευές δόνησης. Ο χρήστης μπορεί επίσης να νοιώσει τον αέρα να τον χτυπάει στο πρόσωπο και να μυρίσει αρώματα της πόλης, όπως γιασεμί και ιβίσκο. Τελικά όμως το Sensorama αποδεικνύεται πολύ επαναστατικό για την εποχή του και αποτυγχάνει. [1]

Το 1966 αρχίζουν να εμφανίζονται οι πρώτοι εξομοιωτές πτήσης που χρησιμοποιούνται από την αεροπορία για την εκπαίδευση των πιλότων. Την ίδια δεκαετία αναπτύσσεται η πρώτη κάσκα εικονικής (Virtual Reality) και επαυξημένης πραγματικότητας (Augmented Reality) η οποία απομονώνει τον χρήστη από το περιβάλλον για ρεαλιστικότερα αποτελέσματα.

Το 1989 χρησιμοποιείται για πρώτη φορά ο όρος «Εικονική Πραγματικότητα» από τον Jaron Lanier, ιδρυτή της εταιρίας VPL Research. Η εταιρία του δημιούργησε αισθητήρες όπως το γάντι δεδομένων (Data Glove) και την σφαίρα ήχου (Audio Sphere) που επέτρεψαν την καλύτερη αλληλεπίδραση χρήστη και εικονικού κόσμου.

Την δεκαετία του 1990 γίνεται η πρώτη προσπάθεια ενσωμάτωσης της εικονικής πραγματικότητας στα ηλεκτρονικά παιχνίδια. Η εταιρία Sega ανακοινώνει το "Sega VR Headset for Arcade Games". Η συσκευή χρησιμοποιεί LCD οθόνες, στερεοσκοπικό ήχο και αισθητήρες αδρανείας, οι οποίοι επέτρεπαν στη συσκευή να παρακολουθεί και να αντιδρά στις κινήσεις του κεφαλιού. [2]

Σήμερα, οι συσκευές εικονικής πραγματικότητας που επικρατούν είναι το Oculus Rift της Oculus VR και τα HoloLens της Microsoft. Το Oculus Rift, είναι μια συσκευή που προσαρμόζεται στο κεφάλι και προσφέρει στερεοσκοπική αντίληψη του εικονικού περιβάλλοντος. Τα HoloLens ενσωματώνονται περισσότερο στην κατηγορία της μεικτής πραγματικότητας και είναι ακόμη σε πειραματικό στάδιο. Είναι διαφανή γυαλιά που επιτρέπουν στον χρήστη να βλέπει το πραγματικό περιβάλλον, με τη ιδιότητα όμως ότι μπορούν να προβάλουν 3D αντικείμενα μέσα στον φυσικό χώρο, με τα οποία μάλιστα η χρήστης μπορεί να αλληλεπιδράσει.

1.1.3 Πεδία Χρήσης

Η χρήση της εικονικής πραγματικότητας στα video games είναι πλέον πολύ διαδεδομένη. Ήδη από τη δεκαετία του 1990 αρκετές γνωστές εταιρίες δημιούργησαν «κάσκες» για χρήση στις παιχνιδομηχανές τους. Μετά το 2000 έκαναν την εμφάνισή τους και οι πρώτες συσκευές ανάδρασης με το παιχνίδι μέσω φυσικής κίνησης, οι δημοφιλέστερες από της οποίες είναι το Wii Remote της Nintendo, το Kinect της Microsoft και το PlayStation Move της Sony. Σήμερα, οι περισσότερες εταιρίες που αναπτύσσουν ηλεκτρονικά παιχνίδια, ασχολούνται περισσότερο με την ρεαλιστικότερη και φυσικότερη αναπαράσταση του εικονικού κόσμου με χρήση στερεοσκοπικών «κασκών» και συστημάτων μεικτής πραγματικότητας.

Πολύ σημαντική εφαρμογή της εικονικής πραγματικότητας είναι και στον τομέα της εκπαίδευσης. Η χρήση της αποσκοπεί στην εξάσκηση των επαγγελματιών σε εικονικό περιβάλλον, στο οποίο μπορούν να ελέγξουν και να βελτιώσουν τις ικανότητές τους, χωρίς να υποστούν τις συνέπειες της αποτυχίας. Μερικά παραδείγματα ευρείας εφαρμογής στον επαγγελματικό χώρο είναι: η εκπαίδευση πιλότων, οι πτώσεις με αλεξίπτωτο, η διεξαγωγή χειρουργείων από γιατρούς και η εκμάθηση λειτουργίας βαρέων μηχανημάτων.

Άλλος τομέας χρήσης είναι αυτός του Design. Αρχιτέκτονες, πολιτικοί μηχανικοί, σχεδιαστές κλπ. χρησιμοποιούν ευρύτατα την εικονική πραγματικότητα για να ελέγξουν την αισθητική των κατασκευών τους. Σε συνδυασμό με την ανάπτυξη των εύχρηστων design software, η επιθεώρηση ενός έργου πριν την κατασκευή του έγινε πολύ ταχύτερη και αξιόπιστη. [2]

Οι παραπάνω τομείς είναι οι επικρατέστεροι στον τομέα της εικονικής πραγματικότητας. Υπάρχουν ακόμη αρκετοί, κυρίως σε αρχικό στάδιο, η αναφορά όμως στον καθένα ξεχωριστά δεν κρίνεται σκόπιμη.

1.2 Αντικείμενο της διπλωματικής

Το αντικείμενο έρευνας της εργασίας είναι ο προγραμματισμός τροχιάς και η τηλεπαρακολούθηση λειτουργίας ενός ρομποτικού βραχίονα, χρησιμοποιώντας περιβάλλον εικονικής πραγματικότητας και αισθητήρες προερχόμενους από ηλεκτρονικά ευρείας χρήσης. Πιο συγκεκριμένα, επιχειρούνται:

- Μοντελοποίηση του ρομποτικού βραχίονα Stäubli RX90L που βρίσκεται στο υπόγειο του τομέα Τεχνολογίας των Κατεργασιών του ΕΜΠ, καθώς και του περιβάλλοντα χώρου, χρησιμοποιώντας το λογισμικό ανάπτυξης παιχνιδιών, Unity 3D.
- Ανάπτυξη κινηματικής αλυσίδας μεταξύ των συνδέσμων του ρομπότ και υλοποίηση αντίστροφης κινηματικής.
- Προγραμματισμός τροχιάς του ρομπότ μέσω του χειριστηρίου της κονσόλας Xbox 360 της Microsoft.
- Τηλεπαρακολούθηση λειτουργίας του βραχίονα μέσω του αισθητήρα/χειριστηρίου Wii Remote.

Στόχοι της παραπάνω έρευνας είναι:

- Απλοποίηση της εργασίας προγραμματισμού τροχιάς του ρομπότ, εφόσον ο χειριστής ελέγχει το τελικό σημείο δράσης και οι τιμές των γωνιών των αρθρώσεων προκύπτουν από την αντίστροφη κινηματική.
- Μείωση των λαθών κατά την παραγωγή, αφού τα λάθη κατά τον προγραμματισμό τροχιάς σε εικονικό περιβάλλον δεν έχουν φυσικές συνέπειες και μπορούν να διορθωθούν.

- Δυνατότητα παρακολούθησης και τροποποίησης της λειτουργίας του ρομποτικού βραχίονα, χωρίς να απαιτείται να βρίσκεται κάποιος στον χώρο του ρομπότ. Το τελευταίο εξαλείφει το ενδεχόμενο τραυματισμού του ανθρώπινου παράγοντα από τα κινούμενα μηχανικά μέρη.

1.3 Ανασκόπηση Βιβλιογραφίας

Τα ρομποτικά κύτταρα αποτελούν ένα ιδιαίτερο είδος συστημάτων παραγωγής που απασχολούν βιομηχανικά ρομπότ κυρίως για την χειρισμό υλικών, π.χ. φόρτωμα/ξεφόρτωμα εργαλειομηχανών, αλλά και για την πρωτογενή κατεξεργασία των υλικών π.χ. συγκόλληση, την δευτερογενή επεξεργασία υλικών π.χ. καθαρισμό, αφαίρεση γρεζιών κλπ. καθώς και την συναρμολόγηση προϊόντων αποτελούμενων από πολλά μέρη. Η παραγωγή σε ρομποτικά κύτταρα πρέπει να λαμβάνει υπόψη ορισμένες ιδιαιτερότητες, οι οποίες είναι: η πιο σύνθετη κινηματική των ρομπότ σε σχέση με αυτή των εργαλειομηχανών και η ανάγκη του offline προγραμματισμού του ρομπότ να είναι όσο το δυνατό διαισθητικός και φιλικός προς τον χρήστη. Αυτές οι ιδιαιτερότητες έχουν αντιμετωπιστεί μέχρι στιγμής από λογισμικά προσομοίωσης που εκτείνονται σε διάφορα πλαίσια, π.χ. μηχανολογικό σχεδιασμό, έλεγχο και ενοποίηση με βιομηχανικό περιβάλλον και αισθητήρες. Τέτοια λογισμικά ουσιαστικά υλοποιούν κινηματική, δυναμική και τεχνητή όραση, με περιορισμένη όμως αλληλεπίδραση με τον χρήστη. [3] [4]

Λόγω της ανάπτυξης των σχετικών τεχνολογιών, μόλις πρόσφατα σοβαρές προσεγγίσεις έδειξαν να εκμεταλλεύονται την εικονική, την επαυξημένη και την μεικτή πραγματικότητα στην βιομηχανική ρομποτική, τόσο όσον αφορά την ρεαλιστική συμπεριφορά του ρομπότ και του περιβάλλοντος παραγωγής, όσο και από την άποψη της βελτίωσης της χρηστοκεντρικής αντίληψης [5]. Αναπτύσσονται νέοι διαισθητικοί τρόποι μετακίνησης και προγραμματισμού των ρομπότ, στηρίζοντας τον χρήστη με βοηθητικούς αλγορίθμους αποφυγής εμποδίων και αυτόματου προγραμματισμού τροχιάς, οι οποίοι συμπεριλαμβάνουν τις έννοιες των συσκευών εισόδου και των στρατηγικών προγραμματισμού τροχιάς. [6] Ένα πλήρες πρόγραμμα εκμάθησης για την ανάπτυξη ολοκληρωμένων περιβαλλόντων εικονικής πραγματικότητας για ρομποτικές εφαρμογές – σε πλαίσιο ευρύτερο του βιομηχανικού – με βάση εμπορικά διαθέσιμα λογισμικά ανάπτυξης δίνεται στο [7]. Επίσης, επινοούνται πολυτροπικές

διεπαφές οι οποίες αντικαθιστούν τις παραδοσιακές, πχ ποντίκια, πληκτρολόγια και δισδιάστατες συσκευές απεικόνισης, όπως π.χ. σύστημα CAVE και απτική επιτραπέζια συσκευή που αλληλεπιδρά με ένα εικονικό ρομποτικό περιβάλλον, με στόχο τον off-line προγραμματισμό δύο ρομπότ σε ένα κύτταρο δύο εργαλειομηχανών [8]. Οι Chen, MacDonald και Wunsche παρουσίασαν ένα διαδραστικό σχέδιο που περιλάμβανε τα απαραίτητα στάδια για την δημιουργία αλληλεπιδράσεων μεταξύ πραγματικών και εικονικών αντικειμένων, για την κατασκευή περιβαλλόντων μεικτής πραγματικότητας προσομοίωσης ρομπότ [9].

Ο προγραμματισμός των ρομπότ, συμπεριλαμβανομένου του προγραμματισμού τροχιάς/διαδρομής, στα κύτταρα ή στις γραμμές παραγωγής, έχει επωφεληθεί πολύ από την ανάπτυξη της εικονικής/επαυξημένης/μεικτής πραγματικότητας, είναι όμως κάτι πολύ πρόσφατο ακόμη, και σε πλήρη ανάπτυξη [10]. Οι διαισθητικές προσεγγίσεις επιδιώκονται με υψηλό βαθμό πολυτροπικής αλληλεπίδρασης του χρήστη και αποφεύγεται όσο το δυνατόν περισσότερο η παρουσία σχετικών μελών του πραγματικού κόσμου [11]. Ένα σύστημα επαυξημένης πραγματικότητας, επικεντρωμένο στην ικανότητα του ανθρώπου να προγραμματίζει γρήγορα τροχιές απαλλαγμένες εμποδίων, μέσω του χειροκίνητου προσδιορισμού του ελεύθερου όγκου παρουσιάζεται στο [12]. Η αντίστροφη και η ορθή κινηματική είναι ενσωματωμένες στο σύστημα και ένα Head Mounted Display (HMD) προβάλει το εικονικό ρομπότ και άλλα εικονικά στοιχεία. Μια μέθοδος εκμάθησης καμπυλών, βασισμένη στα Bayesian νευρωνικά δίκτυα, η οποία χρησιμοποιείται για να καθορίσει τροχιές από δεδομένα σημεία και προσανατολισμούς του τελικού σημείου δράσης κατά μήκος της διδαγμένης καμπύλης, σχεδιάζεται με τη βοήθεια της επαυξημένης πραγματικότητας [13]. Επίσης, λαμβάνονται υπόψη οι δυναμικοί περιορισμοί του ρομπότ και ένα κύβος-δείκτης υιοθετείται ως το κύριο αντικείμενο διεπαφής του χρήστη, το οποίο του επιτρέπει να βλέπει την προεπισκόπηση της προσομοιωμένης κίνησης, να αντιλαμβάνεται πιθανές υπερακοντίσεις και να επιλύει τις διαφορές μεταξύ προγραμματισμένης και προσομοιωμένης τροχιάς [14]. Ένα εξ αποστάσεως προγραμματιστικό σύστημα προσανατολισμένο στις συγκολλήσεις laser, το οποίο λαμβάνει υπόψη τα δεδομένα του πραγματικού τεμαχίου και τα όρια της διαδικασίας και βασίζεται σε ένα διαδραστικό σύστημα επαυξημένης πραγματικότητας αναφέρεται στο [15]. Με βάση αυτό, γρήγορες και αποτελεσματικές χωροταξικές μέθοδοι αλληλεπίδρασης επιτρέπουν ακόμη και σε νέους χρήστες να καθορίζουν γρήγορα λειτουργίες σε επίπεδο εργασιών.

Η ενσωμάτωση ενός ρομπότ σε ένα κύτταρο παραγωγής, γίνεται με έμφαση περισσότερο στην εικονική πραγματικότητα σε αντίθεση με την επαυξημένη πραγματικότητα. Μια δομημένη προσέγγιση οδήγησε στην ανάπτυξη εικονικού περιβάλλοντος, το οποίο υποστηρίζει τον προγραμματισμό έργων του ρομπότ εστιάζοντας σε ενδιαφέροντα συμβάντα, αντικείμενα και χρονικά πλαίσια [16].

1.4 Στόχοι και διάρθωση της εργασίας

Οι στόχοι που έχουν τεθεί προς επίτευξη απαιτούν εργασία, η οποία μπορεί να χωριστεί σε επιμέρους τμήματα και βήματα. Αυτά είναι:

1. Εισαγωγή του μοντέλου του ρομπότ στο Unity και ανάπτυξη κινηματικής αλυσίδας μεταξύ των συνδέσμων του.
2. Ανάπτυξη κώδικα που υλοποιεί την ορθή και την αντίστροφη κινηματική, με σκοπό τον έλεγχο του ρομπότ.
3. Προσθήκη του χώρου του εργαστηρίου και ανάπτυξη σχέσεων αλληλεπίδρασης με τα κινούμενα μέρη του ρομπότ.
4. Σχεδιασμός σεναρίων ελέγχου της λειτουργικότητας της αντίστροφης κινηματικής και των αλληλεπιδράσεων μεταξύ των αντικειμένων.
5. Εξαγωγή τροχιάς από το Unity με σκοπό την αναπαραγωγή της στο πραγματικό ρομπότ και σύγκριση των πειραματικών αποτελεσμάτων με αυτά που παίρνουμε από το εικονικό περιβάλλον.
6. Πρόσδεση του Wii Remote στο ρομπότ και έλεγχος των αποτελεσμάτων της τηλεπαρακολούθησης.

Τα επιμέρους κεφάλαια της εργασίας όπου εξυπηρετούνται οι παραπάνω στόχοι είναι τα εξής:

Στο Κεφ. 2 παρουσιάζεται το λογισμικό ανάπτυξης του εικονικού περιβάλλοντος, Unity 3D. Στα πλαίσια αυτή της παρουσίασης αναλύεται εν συντομία η αρχή λειτουργίας του, οι «οντότητες» που διαθέτει για την δημιουργία των τρισδιάστατων σκηνών, καθώς οι δυνατότητες που μπορούν αυτά να αποκτήσουν με τη λειτουργία των components.

Στο Κεφ. 3 γίνεται μια σύντομη περιγραφή του αρθρωτού βιαμηχανικού βραχίονα Stäubli RX90L. Συγκεκριμένα αναφέρονται τα βασικά τεχνικά και γεωμετρικά χαρακτηριστικά του, τα οποία χρησιμοποιούνται και στην μετέπειτα κινηματική του

ανάλυση. Γίνεται επίσης και μια πολύ σύντομη αναφορά στον τρόπο προγραμματισμού του.

Στο Κεφ. 4 περιγράφονται τα εργαλεία που διαθέτει το Unity για την δημιουργία δυσκολότερων και ρεαλιστικότερων σκηνών. Στα πλαίσια αυτής της περιγραφής, γίνεται λεπτομερής αναφορά της διαδικασίας εισαγωγής των μοντέλων του περιβάλλοντα χώρου του εργαστηρίου και του ρομπότ και της δημιουργίας της κινηματικής του αλυσίδας. Γίνεται επίσης διάκριση μεταξύ του συστήματος συντεταγμένων που χρηιμοποιεί το ρομπότ σε σχέση με αυτό που χρησιμοποιεί το Unity και περιγράφονται όλες οι κινήσεις που έγιναν για να ταυτιστούν.

Στο Κεφ. 5 μελετάται η κινηματική που διέπει τον αρθρωτό ρομποτικό βραχίωνα Stäubli RX90L. Συγκεκριμένα, γίνεται η ευθεία κινηματική του ανάλυση χρησιμοποιώντας την μεθοδολογία των Denavit – Hartenberg και η αντίστροφη κινηματική του ανάλυση χρησιμοποιώντας την μέθοδο απόπλεξης/αποσύζευξης του Pieper. Στη συνέχεια περιγράφεται η μεταφορά των μαθηματικών αποτελεσμάτων σε μορφή κώδικα στο Unity.

Στο Κεφ. 6 παρουσιάζεται ο τρόπος υλοποίησης του προγραμματισμού τροχιάς. Αρχικά, γίνεται σύντομη περιγραφή των δυνατοτήτων του χειριστηρίου της κοσνόλας Xbox 360 που χρησιμοποιήθηκε. Στη συνέχεια, εξηγείται πώς χρησιμοποιήθηκαν τα διάφορα κουμπιά που διαθέτει για τον έλεγχο του ρομπότ, κάνοντας χρήση του κώδικα της αντίστροφης κινηματικής. Τέλος, περιγράφονται τα σενάρια που δημιουργήθηκαν για τον έλεγχο της λειτουργικότητά του και γίνεται σύγκριση των προσομοιωμένων και των πραγματικών αποτελεσμάτων.

Στο Κεφ. 7 αναλύεται η προσπάθεια υλοποίησης της τηλεπαρακολούθησης της λειτουργίας του ρομποτικού βραχίονα του εργαστηρίου χρησιμοποιώντας το Wii Remote της Nintendo. Αφού αναγνωριστούν οι αισθητήρες που διαθέτει το χειριστήριο και γίνει περιγραφή του τρόπου λειτουργίας τους, καταγράφεται η βασική ιδεά υλοποίησης, τα μαθηματικά που την διέπουν, καθώς οι λόγοι που κατέστησαν αδύνατη την υλοποίησή της.

Τέλος, στο Κεφ. 8 συνοψίζεται η δουλειά των προηγούμενων κεφαλαίων, γίνεται αποτίμηση του οφέλους από την έρευνα και προτείνονται ιδέες για μελλοντική επέκτασή της.

2 Το λογισμικό

2.1 Τι είναι το Unity 3D

Το Unity 3D είναι μια πλατφόρμα ανάπτυξης ηλεκτρονικών παιχνιδιών, η οποία δημιουργήθηκε από την εταιρία Unity Technologies, με σκοπό την ανάπτυξη παιχνιδιών για ηλεκτρονικούς υπολογιστές, κονσόλες, κινητές συσκευές καθώς και ιστοσελίδες.

Στην αρχή το λογισμικό ανακοινώθηκε μόνο για Mac OS X στο Apple's Worldwide Developers Conference το 2005. Έκτοτε, η λειτουργικότητά του επεκτάθηκε και πλέον η χρήση του στοχεύει σε παραπάνω από 15 πλατφόρμες. Είναι μάλιστα το βασικό πακέτο ανάπτυξης παιχνιδιών (SDK) για το Wii U της Nintendo.

Η τελευταία έκδοση, Unity 5.0 είναι διαθέσιμη δωρεάν για ιδιώτες και επιχειρήσεις με ετήσιο εισόδημα μικρότερο από $100,000. Είναι σημαντικό να επισημανθεί, ότι το 2006 η Apple έθεσε το Unity υποψήφιο για την κατηγορία «Καλύτερη χρήση γραφικών του Mac OS X». [17]

Εικόνα 1 - Περιβάλλον Unity

Είναι προφανές ότι εφόσον πρόκειται για πλατφόρμα ανάπτυξης παιχνιδιών, το Unity διαθέτει περιβάλλον σύνταξης κώδικα για τον προγραμματισμό των projects που δημιουργούνται. Οι γλώσσες προγραμματισμού που υποστηρίζονται είναι:

- JavaScript
- C#
- Boo (Εμπνευσμένη από την σύνταξη της Python)

Όλες οι παραπάνω γλώσσες ανήκουν στην κατηγορία των αντικειμενοστραφών γλωσσών προγραμματισμού, γεγονός που όχι μόνο κάνει το λογισμικό «μοντέρνο», αλλά προσφέρει και σημαντικές δυνατότητες στα projects.

Αξίζει να σημειωθεί, ότι η εταιρία που παρέχει το λογισμικό, διαθέτει ευρύ και αναλυτικό Documentation και Tutorials, τόσο για τα GameObjects, όσο και για την κάθε γλώσσα προγραμματισμού.

2.2 Game Objects

Η λειτουργία του Unity βασίζεται στη φιλοσοφία της ύπαρξης μιας «σκηνής», μέσα στην οποία υπάρχουν αντικείμενα, τα οποία είτε είναι ανεξάρτητα μεταξύ τους, είτε συνδέονται μεταξύ τους με σχέσεις γονέα – παιδιού, έχουν συγκεκριμένη συμπεριφορά και αλληλεπιδρούν μεταξύ τους με διάφορους τρόπους. Τα αντικείμενα αυτά ονομάζονται GameObjects.

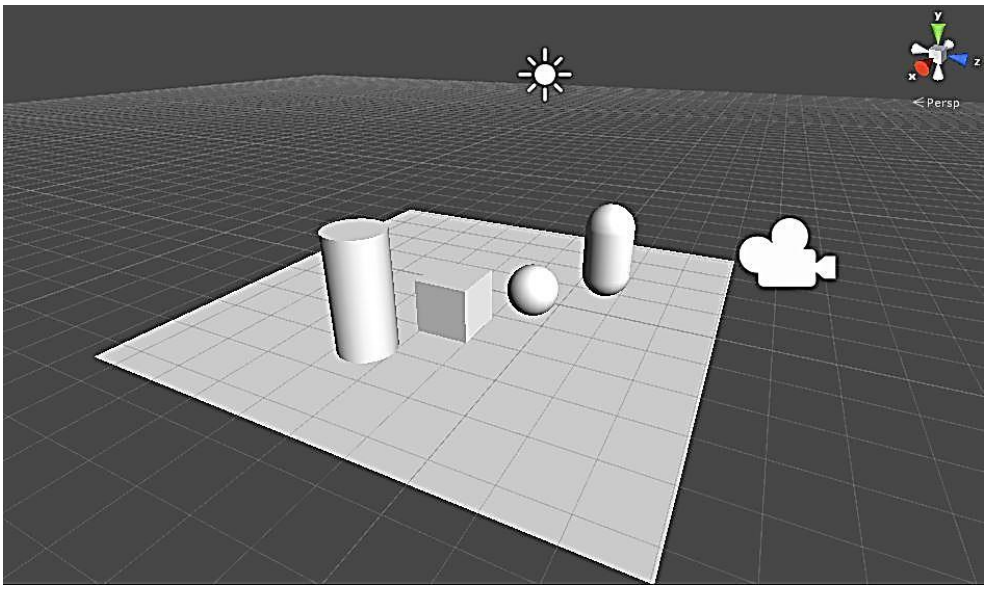
Η συμπεριφορά κάθε GameObject μέσα στη σκηνή εξαρτάται από τα Components που διαθέτει. Τα Components είναι κλάσεις (αρχεία κώδικα) που έχει διαθέσιμες το Unity, οι οποίες επισυνάπτονται στο αντικείμενο και ελέγχουν την συμπεριφορά του. Στα Components συμπεριλαμβάνονται και τα αρχεία κώδικα που δημιουργεί ο χρήστης.

Οι βασικές κατηγορίες αντικειμένων που διαθέτει το Unity είναι:

- 3D Objects
    - Κύβος
    - Σφαίρα
    - Κάψουλα
    - Κύλινδρος
    - Επίπεδο
- Light

    - Directional Light
    - Point Light
    - Spotlight
- Audio
- User Interface (UI)
- Particle System
- Camera

Η κατηγοριοποίηση προκύπτει από τα components που διαθέτει ως προεπιλογή κάθε αντικείμενο. Ανάλυση των components θα γίνει σε επόμενη ενότητα.



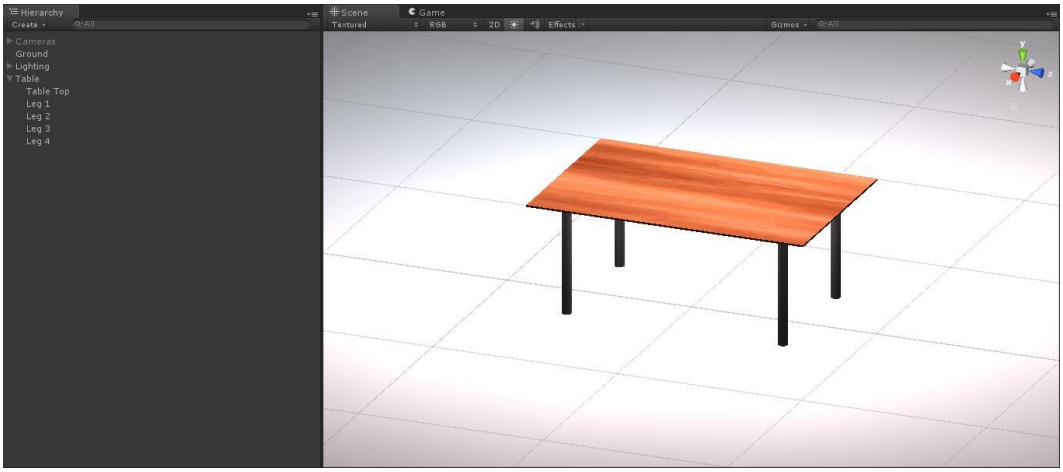
Εικόνα 2 - Σκηνή με κάποια βασικά GameObjects

Για την μοντελοποίηση δυσκολότερων γεωμετριών, τα βασικά 3D Objects που προσφέρει το λογισμικό δεν επαρκούν ή, εάν επαρκούν, η δημιουργία τους είναι δύσκολη και χρονοβόρα.

Αντί αυτού, το Unity παρέχει την δυνατότητα εισαγωγής 3Δ μοντέλων από άλλα σχεδιαστικά προγράμματα. Οι υποστηριζόμενοι τύποι αρχείων είναι .FBX, .dae, .3DS, .dxf και .obj. Στην παρούσα εργασία, ο σχεδιασμός του ρομπότ και του περιβάλλοντος χώρου έγινε στο λογισμικό SolidWorks και το μοντέλο αποθηκεύτηκε σε format IGES. Στη συνέχεια το αρχείο ανοίχθηκε από το λογισμικό 3DS Max και έγινε εξαγωγή του σε format FBX.

Τέλος, μιλήσαμε για σχέσεις γονέα – παιδιού μεταξύ των GameObjects. Αρχικά να πούμε ότι, οποιοδήποτε αντικείμενο βρίσκεται εντός της σκηνής που δημιουργείται,

εμφανίζεται στην αριστερή καρτέλα του project που ονομάζεται Hierarchy. Η ιεραρχία είναι σαν το παράθυρο περιήγησης των Windows, δηλαδή υπάρχουν φάκελοι, που περιέχουν άλλους φακέλους και αυτοί μέσα τους διάφορα αρχεία.



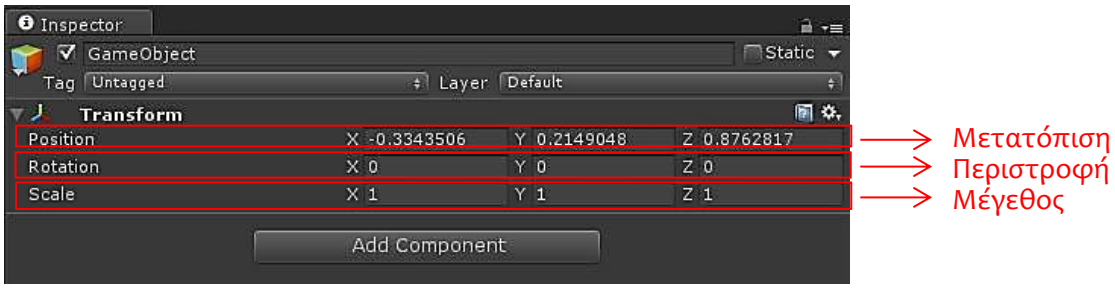
Εικόνα 3 - Hierarchy

Ας δούμε πως λειτουργεί η ιεραρχία στο Unity κοιτώντας το παράδειγμα της εικόνας 3. Το τραπεζάκι της εικόνας αποτελείται από 4 πόδια και 1 επιφάνεια ή αλλιώς 4 κυλίνδρους και 1 παραλληλεπίπεδο. Όπως φαίνεται στην Ιεραρχία, τα 5 κομμάτια είναι παιδιά του γονέα "Table". Το GameObject "Table" στην προκειμένη περίπτωση είναι ένα κενό GameObject, δεν έχει δηλαδή καθόλου components. Το Unity παρέχει τη δυνατότητα δημιουργίας κενών αντικειμένων, μόνο και μόνο για την ομαδοποίηση άλλων. Ο χρήστης έχει τη δυνατότητα να τροποποιήσει κάθε αντικείμενο ξεχωριστά, ακόμη κι αν είναι παιδί άλλου. Αν όμως θελήσει να τροποποιήσει τον γονέα, για παράδειγμα να αλλάξει το μέγεθός του ή να τον μετακινήσει, τα παιδιά του θα μεγεθυνθούν ή θα μετακινηθούν ανάλογα, χωρίς να αλλάξει το σχετικό μέγεθος ή η σχετική θέση γονέα – παιδιού. Υφίσταται δηλαδή η έννοια της κληρονομικότητας που κυριαρχεί στον αντικειμενοστραφή προγραμματισμό. Να τονίσουμε ότι δεν απαιτείται ο γονέας να είναι κενό αντικείμενο για να έχει παιδιά.

2.3 Components

2.3.1 Προεπιλεγμένα Components

Όπως αναφέρθηκε προηγουμένως, το Unity παρέχει κάποιες κατηγορίες έτοιμων GameObjects οι οποίες καθορίζονται από τα components που διαθέτουν. Τα components εμφανίζονται στην δεξιά καρτέλα του project που ονομάζεται Inspector. Όλες οι κατηγορίες, συμπεριλαμβανομένων και των κενών αντικειμένων, έχουν ένα κοινό component το οποίο ονομάζεται transform και δεν μπορεί να αφαιρεθεί.



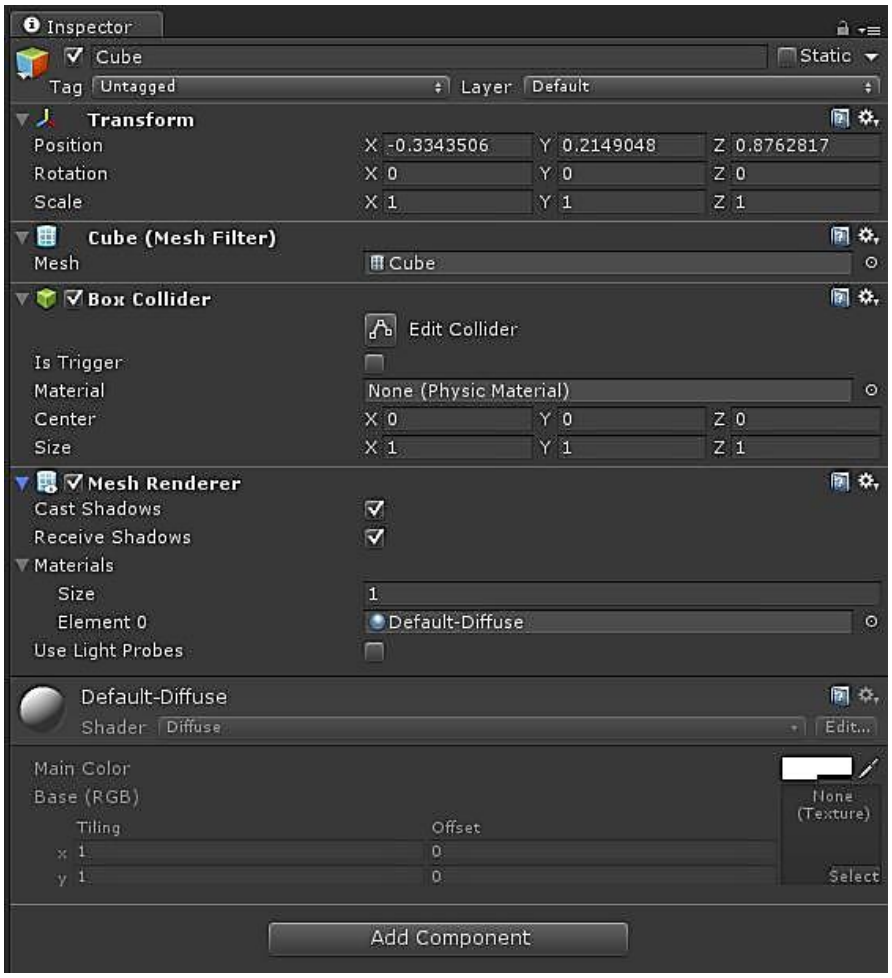
Εικόνα 4 - Transform Component

Το Transform component είναι ουσιαστικά ο ομογενής μετασχηματισμός των GameObjects και περιλαμβάνει: την θέση ως προς την αρχή των αξόνων, την περιστροφή γύρω από τους τρείς άξονες εκφρασμένη σε γωνίες Euler και το μέγεθος του αντικειμένου σε κάθε άξονα. Σχετικά με τους ομογενείς μετασχηματισμούς και τις γωνίες Euler ο αναγνώστης παραπέμπεται στο Παράρτημα Α. Επειδή όπως ήδη είπαμε, τα components είναι κλάσεις που προσφέρονται έτοιμες από το Unity, το transform component μπορεί να προσπελασθεί μέσω κώδικα που φτιάχνει ο χρήστης και να τροποποιηθούν οι τιμές του.

Εκτός του κοινού αυτού χαρακτηριστικού, ανάλογα με την κατηγορία, τα GameObjects περιέχουν διαφορετικά components τα οποία θα αναλυθούν στη συνέχεια ξεχωριστά.

3D Objects

Τα 3D Objects είναι αυτά που έχουν γεωμετρία τριών διαστάσεων όπως ένας κύβος ή ένας κύλινδρος. Τα προεπιλεγμένα components που εισάγει το Unity με τη δημιουργία του 3Δ αντικειμένου εκτός του Transform, είναι το Mesh Renderer και το Collider.



Εικόνα 5 - 3D Object Components

Το Mesh Renderer είναι αυτό που κάνει το αντικείμενο ορατό στη σκηνή. Η γεωμετρία του καθορίζεται από έναν ελάχιστο αριθμό γραμμών οι οποίες τέμνονται μεταξύ τους σε συγκεκριμένα σημεία σχηματίζοντας τρίγωνα και οριοθετούν τον όγκο του αντικειμένου. Ο χώρος που ορίζεται από αυτές τις γραμμές «γεμίζεται» με χρώμα και το αντικείμενο είναι ορατό στη σκηνή μας. Οι γραμμές αυτές ονομάζονται πλέγμα ή στα αγγλικά mesh. Όπως φαίνεται και στην εικόνα 5, το ίδιο το component δίνει στο χρήστη την επιλογή να αλλάξει κάποιες επιλογές του πλέγματος, όπως το αν θα δέχεται/ρίχνει σκιές ή το υλικό από το οποίο είναι φτιαγμένο.

Το Collider είναι το component που δίνει τη δυνατότητα στα αντικείμενα να συγκρούονται. Πρόκειται για ένα πλέγμα το οποίο περιβάλει το τρισδιάστατο αντικείμενο και είναι σαν τοίχος, δηλαδή αν έρθουν σε επαφή δύο αντικείμενα δεν μπαίνουν το ένα μέσα στο άλλο (overlap), αλλά αλληλεπιδρούν (σπρώχνονται, σταματούν κλπ).

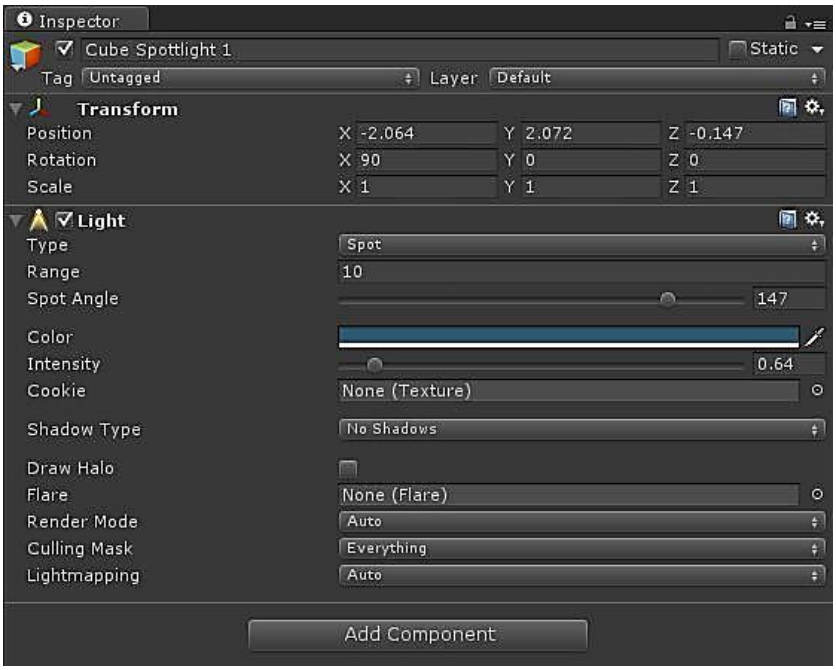
Υπάρχουν οι primitive colliders, οι οποίοι έχουν το σχήμα των βασικών 3Δ αντικειμένων που διαθέτει το Unity και οι mesh colliders οι οποίοι χρησιμοποιούνται σε δυσκολότερες γεωμετρίες και έχουν το σχήμα του πλέγματος του αντικειμένου. Οι τελευταίοι, ενώ δίνουν ακριβέστερα αποτελέσματα, δεν πρέπει να χρησιμοποιούνται περισσότερο από όσο είναι αναγκαίο επειδή καταναλώνουν πολύ υπολογιστική ισχύ.

Οι γλώσσες προγραμματισμού που χρησιμοποιεί το Unity διαθέτουν τις συναρτήσεις OnCollisionEnter(), OnCollisionStay() και OnCollisionExit() οι οποίες είναι φτιαγμένες ειδικά για να καλούνται όταν συμβαίνει αυτό που λέει το όνομά τους, δηλαδή όταν ξεκινάει κάποια σύγκρουση, όσο διαρκεί μια σύγκρουση και όταν τελειώνει μια σύγκρουση αντίστοιχα. Μέσα σε αυτές, ο χρήστης μπορεί να γράψει κώδικα που να τροποποιεί τη συμπεριφορά του παιχνιδιού.

Επιπλέον ο χρήστης μπορεί να αλλάξει τις διαστάσεις και τη θέση του collider ή να τον κάνει Trigger. Όταν η επιλογή Is Trigger είναι επιλεγμένη, ο collider συνεχίζει να εντοπίζει τις συγκρούσεις, επιτρέποντας όμως στα αντικείμενα να μπαίνουν το ένα μέσα στο άλλο. Αυτό είναι πολύ χρήσιμο όταν μας ενδιαφέρει για παράδειγμα να δίνουμε κάποιο ηχητικό σήμα όταν συμβαίνει μια σύγκρουση, χωρίς να μας ενδιαφέρει να σταματάει η κίνηση. Οι αντίστοιχες συναρτήσεις που διαθέτει το Unity είναι οι: OnTriggerEnter(), OnTriggerStay() και OnTriggerExit().

Lights

Τα φώτα είναι εφέ, τα οποία επιδρούν στον χρωματισμό της σκηνής και δημιουργούν την αίσθηση του φωτισμού. Μια σκηνή χωρίς κανένα φως στην ιεραρχία είναι σκοτεινή. Αυτή η κατηγορία περιέχει μόνο ένα component, το Light.



Εικόνα 6 - Light Component

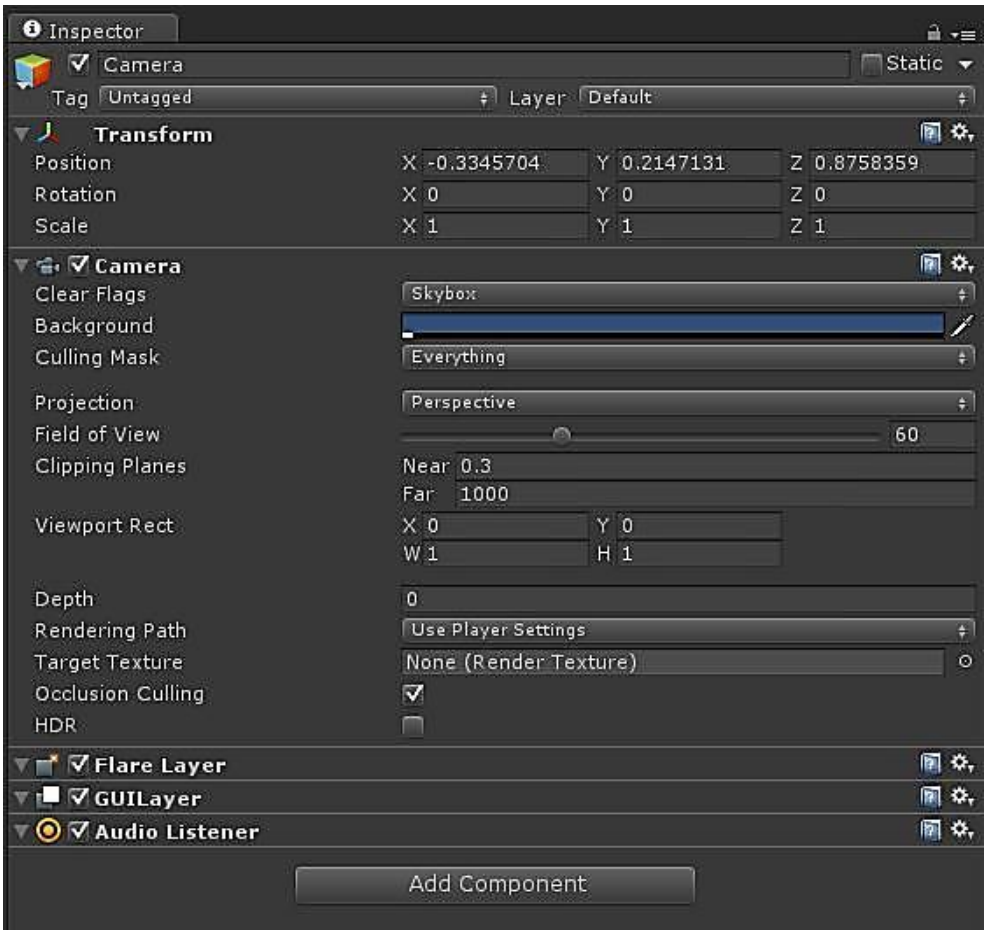
Από εκεί ο χρήστης μπορεί να επιλέξει το είδος του εφέ που θα δημιουργεί ο φωτισμός, συγκεκριμένα:

- Τα directional lights λειτουργούν όπως ο ήλιος. Το πώς θα φωτίζεται ένα αντικείμενο δεν επηρεάζεται από τη θέση του αντικειμένου αλλά από τον προσανατολισμό του και μόνο. Γι' αυτό και στο transform component του directional light η θέση δεν παίζει κανένα ρόλο, παρά μόνο ο προσανατολισμός του.
- Τα spotlights δίνουν την αίσθηση των σποτ.
- Τα point lights είναι σαν τα spotlights με την διαφορά ότι φέγγουν προς όλες τις κατευθύνσεις και όχι μόνο προς τα κάτω.

Από το component αυτό ρυθμίζονται επίσης η απόσταση στην οποία θα φτάνει το φως, το εύρος γύρω από την πηγή, το χρώμα του φωτισμού και η έντασή του.

Camera

Η κάμερα μαζί με τις δύο παραπάνω κατηγορίες είναι τα βασικότερα στοιχεία για την δημιουργία μια σκηνής. Είναι εκείνο το GameObject το οποίο καθορίζει τι θα βλέπει ο χρήστης όταν τρέχει η εφαρμογή.



Εικόνα 7 - Camera Component

Περιέχει δύο σημαντικά components: το Camera και το Audio Listener. Από το camera component μπορούν να ρυθμιστούν το αν η κάμερα θα έχει προοπτική ή αν θα εστιάζει στις δύο διαστάσεις, το βάθος πεδίου, το εύρος του φακού κ.α.

Το audio listener component λειτουργεί όπως το μικρόφωνο μιας κάμερας, δηλαδή λαμβάνει ήχους που προέρχονται από το εικονικό περιβάλλον, εφόσον υπάρχουν, και στη συνέχεια αυτοί αναπαράγονται από τα ηχεία του υπολογιστή.

Οι υπόλοιπες κατηγορίες δεν χρειάζεται να αναλυθούν εξονυχιστικά επειδή δεν έχουν ιδιαίτερη σημασία στα πλαίσια την μηχανολογικής αυτής έρευνας. Σύντομα αναφέρεται ότι:

- Στην κατηγορία Audio εντάσσεται οποιοσδήποτε ήχος αναπαράγεται στη σκηνή.
- Ως User Interface (UI) εννοείται οποιοδήποτε γραφικό στοιχείο, μήνυμα, πλαίσιο κειμένου, κουμπί, μενού, γραμμή κύλισης κ.α. εμφανίζεται στην οθόνη.
- Particle System είναι τα συνεχώς κινούμενα εφέ. Σε αυτά θα μπορούσαν να ενταχθούν ο καπνός από μία καμινάδα, το εφέ του ψεκασμού ή η φωτιά.

2.3.2 Επιπλέον διαθέσιμα Components

Εκτός από τα προκαθορισμένα components που υπάρχουν στα GameObjects ανά κατηγορία, το Unity παρέχει τη δυνατότητα προσθήκης και άλλων προκειμένου κάποιο αντικείμενο να αποκτήσει επιπλέον ιδιότητες. Η προσθήκη γίνεται πατώντας το κουμπί Add Component που φαίνεται στις προηγούμενες εικόνες.

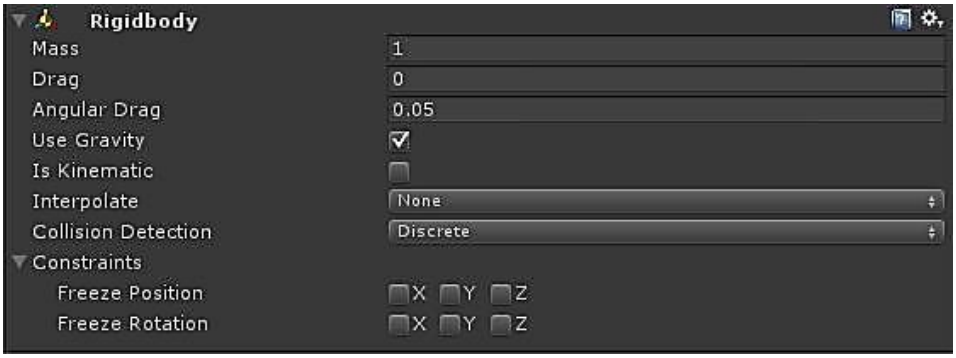
Η λίστα με τα διαθέσιμα components είναι πολύ μεγάλη και γι' αυτό το λόγο θα αναφερθούμε μόνο σε αυτά που χρησιμοποιήθηκαν στην εργασία αυτή.

RigidBody

Το component αυτό θέτει την κίνηση ενός αντικειμένου υπό την επίδραση της φυσικής. Για παράδειγμα, ένα 3D Object που διαθέτει RigidBody, έλκεται από την βαρύτητα χωρίς να έχει κάποιο script που να ελέγχει την κίνηση του.

Επιπλέον, προκειμένου να λειτουργήσουν οι colliders, ένα από τα δύο στερεά πρέπει να έχει οπωσδήποτε RigidBody component.

Στην επόμενη εικόνα φαίνεται το περιεχόμενό του:



Εικόνα 8 - RigidBody Component

Όπως βλέπουμε, τα βασικά χαρακτηριστικά που έχει ένα στερεό σώμα στον πραγματικό κόσμο, όπως η μάζα και η αντίσταση, διατίθενται προς επεξεργασία και στον εικονικό. Επίσης, μέσω συναρτήσεων που περιλαμβάνει η κλάση RIgidBody, μπορούμε να ασκήσουμε δυνάμεις στο στερεό.

Πολύ χρήσιμες είναι οι επιλογές Use Gravity και Is Kinematic. Αν θέλουμε ένα αντικείμενο να μην έλκεται από τη βαρύτητα αλλά να συνεχίζει να επηρεάζεται από τις δυνάμεις, τότε η επιλογή Use Gravity δεν πρέπει να είναι επιλεγμένη. Η επιλογή Is Kinematic αφαιρεί τόσο την επίδραση της βαρύτητας, όσο και οποιασδήποτε άλλης δύναμης.

Θα έμοιαζε λογικό, αν δεν θέλαμε να έχουμε επίδραση φυσικών δυνάμεων να μην βάζαμε καθόλου RigidBody στο αντικείμενο. Όμως, έτσι δεν θα μπορούσε να αλληλεπιδράσει με colliders, αφού όπως είπαμε ένα από τα δύο αντικείμενα πρέπει να έχει RigidBody προκειμένου να δουλέψει η ανίχνευση συγκρούσεων. Άρα βάζοντας RigidBody και επιλέγοντας Is Kinematic, διατηρούμε την ικανότητα ανίχνευσης συγκρούσεων, λέγοντας στο σύστημα να μην ψάχνει για επίδραση άλλων δυνάμεων στο αντικείμενο, εξοικονομώντας έτσι υπολογιστικούς πόρους.

Τέλος, μπορούμε να περιορίσουμε την κίνηση και την περιστροφή του αντικειμένου γύρω από συγκεκριμένους άξονες.

Scripts

Η δυνατότητα ανάπτυξης κώδικα είναι το μεγαλύτερο πλεονέκτημα του Unity. Στην εργασία αυτή η γλώσσα που χρησιμοποιήθηκε είναι η C#. Σε αυτή την παράγραφο θα αναλυθεί εν συντομία ο τρόπος λειτουργίας του Unity με τα scripts.

Αρχικά, πρέπει να πούμε ότι το λογισμικό μπορεί να βρίσκεται σε δύο καταστάσεις: είτε σε Edit Mode, είτε σε Play Mode. Κατά το Play Mode το Unity παράγει στιγμιότυπα της κατάστασης των αντικειμένων που βρίσκονται στη σκηνή, τα οποία ονομάζονται frames. Τα frames λειτουργούν ακριβώς όπως και στις βιντεοκάμερες, δηλαδή αποτυπώνουν περίπου 50-60 frames το δευτερόλεπτο (fps) προκειμένου το αποτέλεσμα στο ανθρώπινο μάτι να μοιάζει με ομαλή κίνηση.

Το προεπιλεγμένο περιβάλλον σύνταξης κώδικα που χρησιμοποιεί το πρόγραμμα είναι το MonoDevelop, αλλά μπορεί να αλλαχθεί με άλλο εφόσον το επιθυμεί ο χρήστης.

Στην επόμενη εικόνα παρατίθεται ένα κομμάτι κώδικα για να εξηγηθεί η λειτουργία του.

```
1 using UnityEngine;
2 using System.Collections;
3
4 public class NewBehaviourScript : MonoBehaviour
5 {
6
7     void Awake()
8     {
9
10    }
11
12    void Start ()
13    {
14
15    }
16
17
18    void Update ()
19    {
20
21    }
22
23    void FixedUpdate()
24    {
25        |
26    }
27 }
28
```

Εικόνα 9 - Βασικές συναρτήσεις Unity

Όπως έχουμε πει και σε προηγούμενο κεφάλαιο, κάθε component είναι μια κλάση, η οποία στο συγκεκριμένο παράδειγμα έχει το όνομα “NewBehaviourScript”. Γενικά το όνομα της κλάσης πρέπει να είναι το ίδιο με το όνομα του αρχείου.

Προκειμένου να μπορέσουμε να εκμεταλλευτούμε τις δυνατότητες του Unity, όλες οι κλάσεις κληρονομούν τις ιδιότητες της μητρικής κλάσης του Unity που ονομάζεται "MonoBehaviour". Η τελευταία περιέχει τις συναρτήσεις Awake(), Start(), Update(), FixedUpdate() και OnGUI().

Η Awake() καλείται μία μόνο φορά πριν το πρώτο frame και χρησιμοποιείται για αρχικοποίηση μεταβλητών κυρίως. Την ίδια χρήση έχει και η Start(), με τη διαφορά ότι καλείται στο πρώτο frame. Το αποτέλεσμα μεταξύ των δύο είναι ορατό όταν για παράδειγμα αλλάξουμε θέση σε ένα αντικείμενο μόλις ξεκινήσει η εφαρμογή χρησιμοποιώντας την Start(). Ναι μεν θα έχουμε αρχικοποιήσει τη θέση, η μετακίνηση όμως σε αυτή θα φανεί.

Η Update() καλείται σε κάθε frame εκτελώντας συνεχώς τους υπολογισμούς που βρίσκονται μέσα της.

Η FixedUpdate() καλείται κάθε φορά που πρόκειται να γίνει ένας υπολογισμός φυσικής. Όταν για παράδειγμα πρόκειται να ασκηθεί μια δύναμη σε ένα αντικείμενο λόγω σύγκρουσης, καλείται η FixedUpdate(). Θα μπορούσαμε λοιπόν να συνδέσουμε την συνάρτηση αυτή με υπολογισμούς που αφορούν τα αντικείμενα που διαθέτουν RigidBody. Το documentation μάλιστα της εταιρίας προτείνει οι υπολογισμοί φυσικής να αποφεύγεται να γράφονται στην Update() για εξοικονόμηση υπολογιστικής ισχύος.

Τέλος, η OnGUI() καλείται κάθε φορά που αποτυπώνουμε κάποιο γραφικό στοιχείο στην οθόνη. Τέτοια είναι αναδυόμενα μηνύματα, πλαίσια κειμένου, κουμπιά, μπάρες κύλισης κλπ.

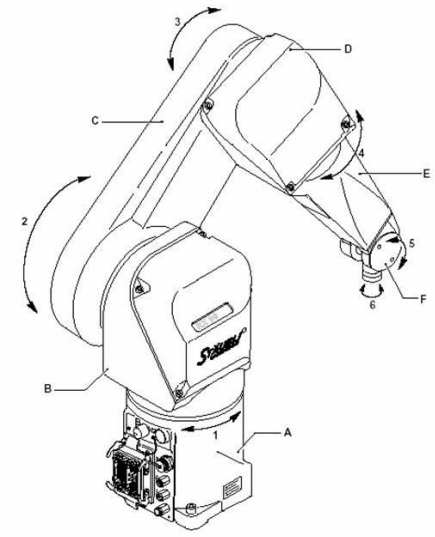
3 Το ρομπότ

3.1 Περιγραφή

Το ρομπότ που χρησιμοποιήθηκε στην εργασία και διατίθεται στο εργαστήριο Τεχνολογίας των Κατεργασιών του ΕΜΠ είναι το Stäubli Rx90L Paint Robot. Ο συγκεκριμένος βραχίονας χρησιμοποιείται σε ποικίλες βιομηχανικές εφαρμογές για το φινίρισμα μεγάλου εύρους προϊόντων. Ενδεικτικές κατηγορίες χρήσης είναι:

- Πλαστικά και μεταλλικά μέρη κινητήρων
- Κατασκευή ποδηλάτων
- Αγροτικός εξοπλισμός
- Αεροναυπηγική
- Οπτικά [18]

Ο βραχίονας αποτελείται από 6 μέλη τα οποία συνδέονται ανά δύο με 6 περιστροφικές αρθρώσεις. Τα μέλη ονομάζονται αλλιώς σύνδεσμοι και στην παρακάτω εικόνα συμβολίζονται με κεφαλαία γράμματα.



Εικόνα 10 - Stäubli RX90L Paint Robot

Επειδή το ρομπότ ανήκει στην κατηγορία των ανθρωπομορφικών βραχιόνων, οι σύνδεσμοι έχουν ονόματα εμπνευσμένα από τα μέρη του ανθρώπινου χεριού:

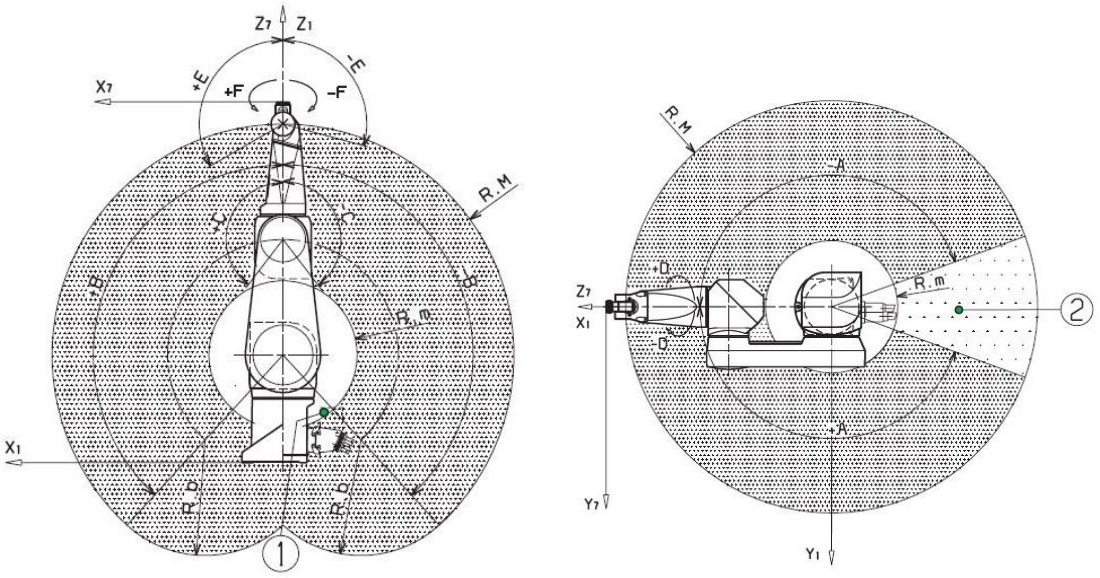
- A – Base
- B – Shoulder
- C – Arm
- D – Elbow
- E – Forearm
- F – Wrist

Κάθε άρθρωση διαθέτει τον δικό της αυτόνομο κινητήρα που της επιτρέπει να περιστρέφεται ανεξάρτητα από τις άλλες και κινεί τον επόμενο σύνδεσμο από τους δύο που συνδέει. Δηλαδή η άρθρωσή 1 για παράδειγμα κινεί τον σύνδεσμο Β. Κάθε άρθρωση περιστρέφεται εντός κάποιων ορίων, τα οποία είναι αυτά που φαίνονται στον επόμενο πίνακα.

Όρια Αρθρώσεων						
Άρθρωση	1	2	3	4	5	6
Εύρος (°)	320	275	285	540	225	540
Όρια(°)	±160	-227.5, 47.5	-52.5, 232.5	±270	-105, 120	±270

Πίνακας 1 - Όρια αρθρώσεων Stäubli RX90L

Τα όρια των αρθρώσεων καθορίζουν τον χώρο εργασίας του βραχίονα. Ως χώρος εργασίας ορίζεται ο χώρος στον οποίο μπορεί να βρεθεί το τελικό σημείο δράσης με οποιονδήποτε προσανατολισμό



Εικόνα 11 - Χώρος εργασίας Stäubli RX90L

Όταν το ρομπότ είναι αδρανές, η εταιρία συνιστά να τοποθετείται στη θέση στην οποία οι τιμές των αρθρώσεων 1 – 6 να είναι αντίστοιχα 0, -90, 90, 0, 0, 0. Σε αυτή τη θέση οι κινητήρες καταπονούνται κατά το ελάχιστο και ονομάζεται θέση "Do Ready".

Στη συνέχεια παρουσιάζονται οι τεχνικές προδιαγραφές του βραχίονα, όπως αυτές κοινοποιούνται από την εταιρία. [19]

Τεχνικά Χαρακτηριστικά	
Εύρος Κίνησης (mm) (R.M)	1185
Ονομαστικό Φορτίο (kg)	3.5
Μέγιστο Φορτίο (kg)	6
Μέγιστη ταχύτητα εργασία (m/s)	11
Βαθμοί ελευθερίας	6
Εγκατάσταση	Πάτωμα/Οροφή

Πίνακας 2 - Τεχνικά χαρακτηριστικά Stäubli RX90L

3.2 Τρόπος προγραμματισμού τροχιάς

Ως προγραμματισμός τροχιάς ορίζεται ο καθορισμός των διαδοχικών τιμών που πρέπει να πάρουν οι αρθρώσεις, προκειμένου το τελικό σημείο δράσης να φτάσει από ένα αρχικό σημείο και προσανατολισμό σε ένα άλλο με διαφορετικό προσανατολισμό. Ο προγραμματισμός τροχιάς του βραχίονα μπορεί να γίνει με δύο τρόπους, είτε online είτε offline.

Ο online προγραμματισμός γίνεται με χρήση του χειριστηρίου διδασκαλίας (teach pendant) όταν το ρομπότ είναι σε λειτουργία. Το teach pendant είναι το χειριστήριο του ρομπότ και επιτρέπει στον χειριστή, είτε να μετακινεί και να προσανατολίζει κατευθείαν το τελικό σημείο δράσης στο χώρο, είτε να προσδιορίζει κατευθείαν τις τιμές των αρθρώσεων. Συνήθως μια τροχιά περνάει από πολλά ενδιάμεσα σημεία τα οποία πρέπει να καθοριστούν ένα – ένα. Κάθε ένα από αυτά τα σημεία αποθηκεύονται στη μνήμη για την αναπαραγωγή τους στον επόμενο κύκλο εργασίας. [20] Αυτός ο τρόπος προγραμματισμού τροχιάς έχει το πλεονέκτημα της άμεσης διεπαφής ρομπότ – ανθρώπου, είναι όμως αργός. Δηλαδή ο χρόνος προγραμματισμού είναι πολύ μεγαλύτερος σε σχέση με τον χρόνο ολοκλήρωσης ενός κύκλου εργασιών.

Εικόνα 12 - Teach Pendant

Ο offline προγραμματισμός γίνεται με χρήση γλωσσών προγραμματισμού. Το συγκεκριμένο ρομπότ χρησιμοποιεί την γλώσσα προγραμματισμού V+ της Adept Technology. Η V+ είναι ένα σύστημα ελέγχου με υπολογιστή που λειτουργεί σε πραγματικό χρόνο, δίνοντας τη δυνατότητα της εκτέλεσης δύσκολων τροχιών σε μικρό χρόνο. Διαθέτει όλες τις λειτουργίες μιας σύγχρονης γλώσσας προγραμματισμού υψηλού επιπέδου όπως υπορουτίνες, δομές ελέγχου, περιβάλλον multitasking και δομές επανάληψης. [21]

Χρησιμοποιώντας την γλώσσα προγραμματισμού αυτή, ο χειριστής μπορεί να ορίσει τα διαδοχικά σημεία από τα οποία πρέπει να περάσει το ρομπότ για την εκτέλεση μιας τροχιάς. Το πρόβλημα αυτής της μεθόδου είναι ότι δεν υπάρχει οπτική επαφή των σημείων, γεγονός που μπορεί να οδηγήσει σε σύγκρουση του ρομπότ σε περίπτωση λάθους.

4 Εισαγωγή 3Δ μοντέλων

4.1 Asset Importing

Το περιβάλλον εργασίας του Unity περιέχει ακόμη μια καρτέλα για την οποία δεν έχουμε μιλήσει ακόμη. Αυτή βρίσκεται στο κάτω μέρος της οθόνης και ονομάζεται Project.

Εικόνα 13 - Καρτέλα Project

Πρέπει να επισημάνουμε ότι στο Unity δημιουργούμε έργα (projects) και όχι σκηνές. Ένα project μπορεί να έχει περισσότερες από μια σκηνές, όπως ακριβώς ένα παιχνίδι μπορεί να έχει περισσότερες από μία πίστες. Κάθε project έχει στοιχεία που το απαρτίζουν τα οποία ονομάζονται Assets. Σε αυτά ανήκουν τα 3Δ μοντέλα που εισάγονται από εξωτερικά σχεδιαστικά προγράμματα, τα αρχεία texture και τα αρχεία ήχου.

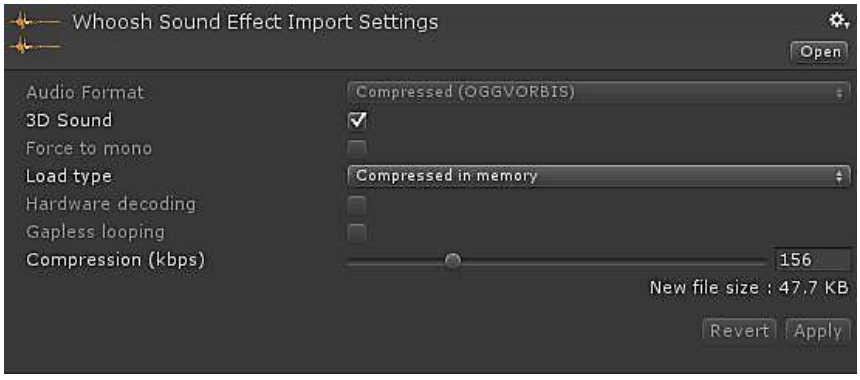
Τα αρχεία texture είναι απλές εικόνες οι οποίες χρησιμοποιούνται για να φτιαχτούν υλικά, τα οποία με τη σειρά τους προστίθενται στα τρισδιάστατα μοντέλα για να γίνουν ρεαλιστικότερα.

Όσον αφορά τα αρχεία ήχου, το Unity υποστηρίζει δύο format: Uncompressed Audio και Ogg Vorbis. Επειδή όμως τα συνηθέστερα αρχεία που χρησιμοποιούνται είναι mp3, wav και aiff, το λογισμικό δίνει την δυνατότητα εισαγωγής τους. Κατά την εισαγωγή τους τα αρχεία αυτά μετατρέπονται αυτόματα σε ένα από τα δύο υποστηριζόμενα format σύμφωνα με τον παρακάτω πίνακα:

.AIFF	Μετατρέπεται σε uncompressed audio κατά την εισαγωγή, κατάλληλο για ηχητικά εφέ μικρής διάρκειας
.WAV	Μετατρέπεται σε uncompressed audio κατά την εισαγωγή, κατάλληλο για ηχητικά εφέ μικρής διάρκειας
.MP3	Μετατρέπεται σε Ogg Vorbis κατά την εισαγωγή, κατάλληλο για ήχους μεγαλύτερης διάρκειας.

Πίνακας 3 - Μετατροπή αρχείων ήχου στα υποστηριζόμενα format από το Unity

Πατώντας πάνω σε κάποιο αρχείο ήχου που έχει εισαχθεί στα assets εμφανίζεται το παράθυρο επιλογών εισαγωγής. Εκεί φαίνεται το format στο οποίο έχει μετατραπεί το αρχείο και μπορούν να ρυθμιστούν παράμετροι όπως το αν θα είναι τρισδιάστατος ήχος ή μονοφωνικός και ο ρυθμός συμπίεσης.

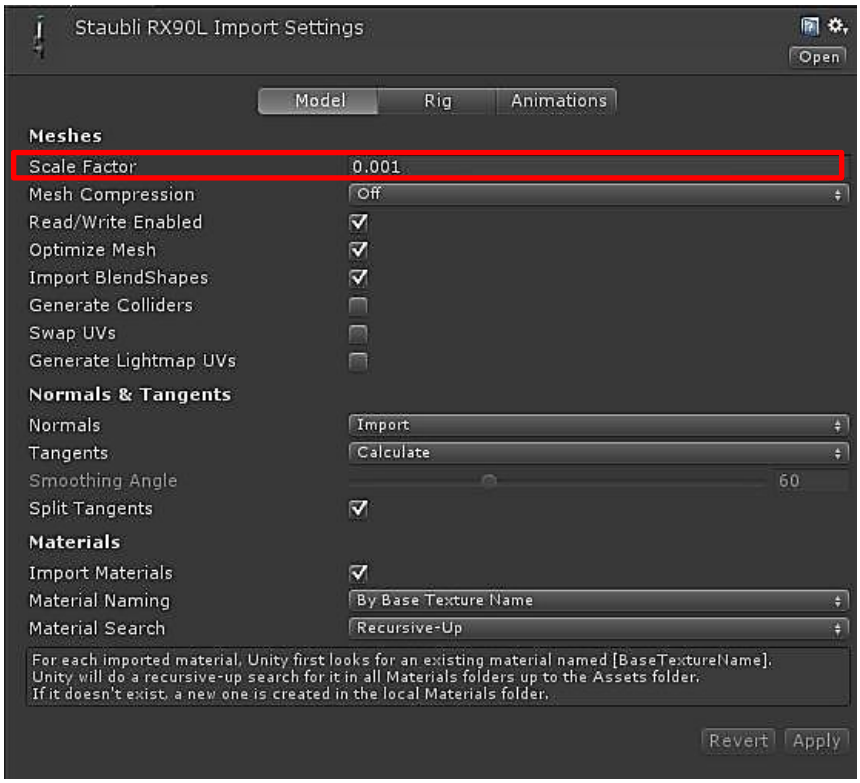


Εικόνα 14 - Παράθυρο επιλογών εισαγωγής αρχείων ήχου

Η εισαγωγή τρισδιάστατων μοντέλων αφέθηκε σκόπιμα τελευταία επειδή παρουσιάζει ιδιαίτερο ενδιαφέρον στην μηχανολογία. Υπάρχει πολύ μεγάλη ποικιλία σχεδιαστικών πακέτων που μπορεί να χρησιμοποιήσει κάποιος για να σχεδιάσει δύσκολες γεωμετρίες, τα αρχεία όμως που αναγνωρίζονται από το Unity είναι τα FBX, dae, 3DS, dxf και obj. Τα μοντέλα αυτά μπορούν να είναι είτε μεμονωμένα parts είτε assemblies. Μόλις εισαχθεί κάποιο μοντέλο, εμφανίζεται, όμοια με τα αρχεία ήχου, το παράθυρο επιλογών εισαγωγής.

Από τις πολλές επιλογές που διατίθενται, μια είναι αυτή που ενδιαφέρει και έχει κόστος στην ακρίβεια του αποτελέσματος της έρευνας, το Scale Factor. Το Unity σε όλα του τα μεγέθη δεν χρησιμοποιεί μονάδες αλλά υποθέτει ότι 1 unity unit = 1 m. Η μετατροπή όμως δεν συμβαίνει αυτόματα και ο χρήστης πρέπει να ρυθμίσει το scale factor προκειμένου να πάρει ρεαλιστικά αποτελέσματα. Γιατί είναι τόσο σημαντικό να έχουμε ρεαλιστικά αποτελέσματα και όχι κάτι περίπου ακριβές;

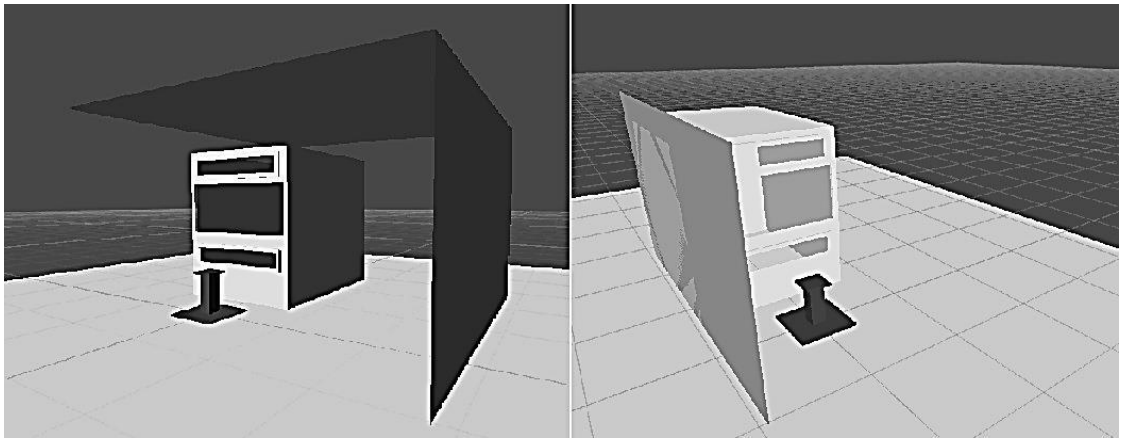
Σκοπός της εργασίας είναι να κινούμε το εικονικό ρομπότ στις 3 διαστάσεις και να διαβάζουμε την θέση και την περιστροφή του τελικού σημείου δράσης (ΤΣΔ) στους 3 άξονες. Από αυτούς του 6 αριθμούς, θα προκύψουν μέσω της αντίστροφης κινηματικής (βλ. κεφάλαιο 5) οι τιμές των γωνιών των αρθρώσεων, οι οποίες στη συνέχεια θα φορτωθούν στο πραγματικό ρομπότ προκειμένου να πραγματοποιήσει την παρεμβολή. Τα μαθηματικά όμως που διέπουν την κινηματική αλυσίδα του ρομπότ εξαρτώνται από τις φυσικές του διαστάσεις. Αν λοιπόν οι διαστάσεις του μοντέλου δεν είναι ακριβώς ίδιες με αυτές του πραγματικού ρομπότ, τα αποτελέσματα όχι μόνο δεν θα είναι ακριβή, αλλά υπάρχει περίπτωση να μην αντιστοιχούν καθόλου στην κίνηση του πραγματικού ρομπότ.



Εικόνα 15 - Παράθυρο επιλογών εισαγωγής 3Δ μοντέλων

Στην εικόνα 15 φαίνεται το scaling factor του μοντέλου του ρομπότ που χρησιμοποιήθηκε στην εργασία. Το σχεδιαστικό πακέτο που χρησιμοποιήθηκε για τη δημιουργία του είναι το SolidWorks, το οποίο χρησιμοποιεί ως μονάδα μέτρησης τα χιλιοστά (mm). Το πραγματικό ρομπότ έχει συνολικό ύψος 1185mm. Γνωρίζουμε ότι το Unity υποθέτει 1 unity unit = 1 m, άρα αν εισάγαμε το μοντέλο με scaling factor = 1 θα είχαμε ένα ρομπότ ύψους 1185 m, ενώ το πραγματικό του ύψος είναι 1185 mm. Θέτοντας την τιμή του scale factor ίση με 0,001 οποιοδήποτε μέγεθος πολλαπλασιάζεται με αυτή την τιμή. Έτσι προέκυψε το εικονικό ισοδύναμο ρομπότ με αυτό που βρίσκεται στο υπόγειο του τομέα Τεχνολογίας των Κατεργασιών του ΕΜΠ.

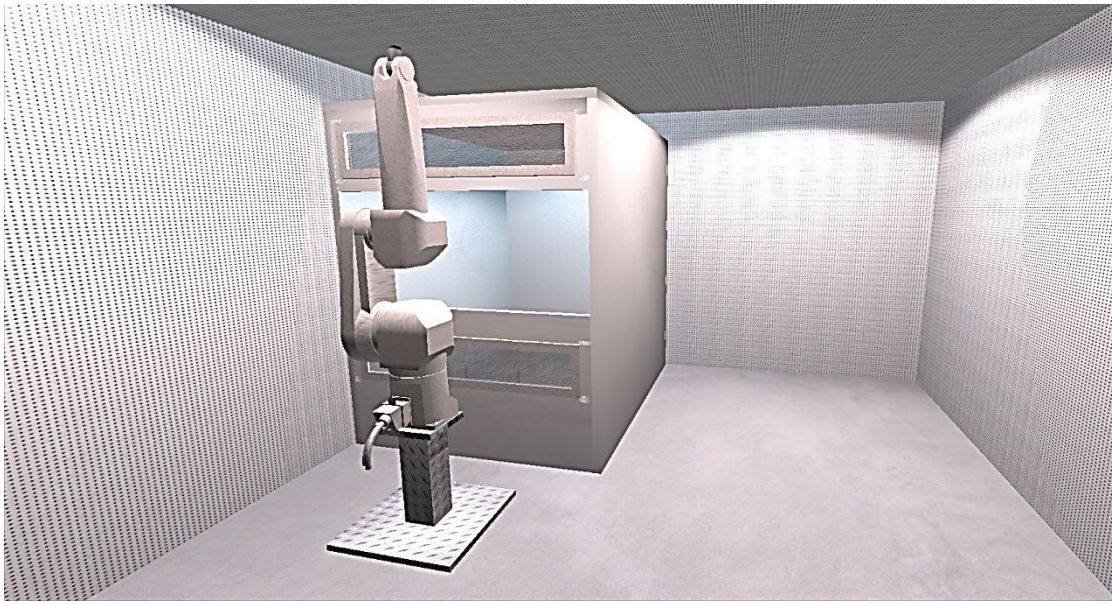
Ο χώρος του εργαστηρίου σχεδιάστηκε στο λογισμικό 3DS Max με μονάδες SI, άρα εισήχθηκε στο Unity με scale factor = 1. Κατά την εισαγωγή του όμως παρουσιάστηκε ένα πρόβλημα: το Unity αναγνωρίζει ως τρισδιάστατες επιφάνειες μόνο αυτές που προκύπτουν από extrude. Αν μια επιφάνεια δεν έχει πάχος, δεν μπορούν να αναπαρασταθούν στη σκηνή και οι δυο πλευρές της αλλά μόνο η μία. Το πρόβλημα αυτό φαίνεται στην εικόνα 13 στην οποία ο χώρος του εργαστηρίου προβάλλεται από διαφορετικές γωνίες, ώστε να είναι ορατή η αδυναμία αναπαράστασης και των δύο πλευρών των τοίχων.



Εικόνα 16 - Πρόβλημα αναγνώρισης 2Δ επιφανειών

Το πρόβλημα θα μπορούσε να λυθεί κάνοντας extrude τους τοίχους του εργαστηρίου στο SolidWorks και να επαναληφθεί η διαδικασία εισαγωγής από την αρχή. Επειδή όμως οι τοίχοι είναι απλά παραλληλεπίπεδα, προτιμήθηκε να δημιουργηθούν με χρήση των primitive GameObjects που προσφέρει το Unity.

Μετά την εισαγωγή του ρομπότ και του χώρου του εργαστηρίου, έγινε εισαγωγή των κατάλληλων textures ώστε να μοιάζουν περισσότερο με το πραγματικό εργαστήριο. Η διαδικασία αυτή γίνεται κάνοντας drag and drop τα textures που έχουν εισαχθεί, πάνω στο αντίστοιχο GameObject στην ιεραρχία. Το αποτέλεσμα φαίνεται στην επόμενη εικόνα.

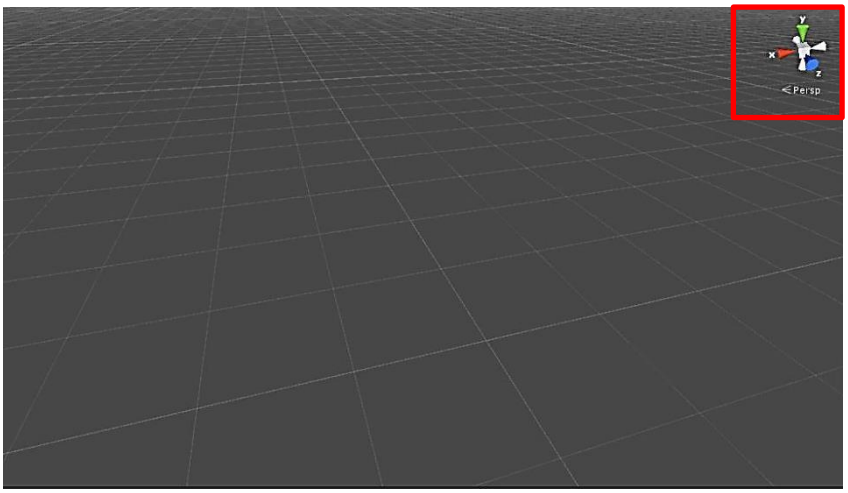


Εικόνα 17 - Εικονικός χώρος εργαστηρίου

4.2 Συστήματα συντεταγμένων

Τα συστήματα συντεταγμένων που χρησιμοποιεί το Unity είναι δύο, το αδρανειακό (Global) και το τοπικό (Local) τα οποία είναι αριστερόστροφα, δηλαδή ακολουθούν τον κανόνα του αριστερού χεριού και όχι του δεξιού όπως έχουμε συνηθίσει.

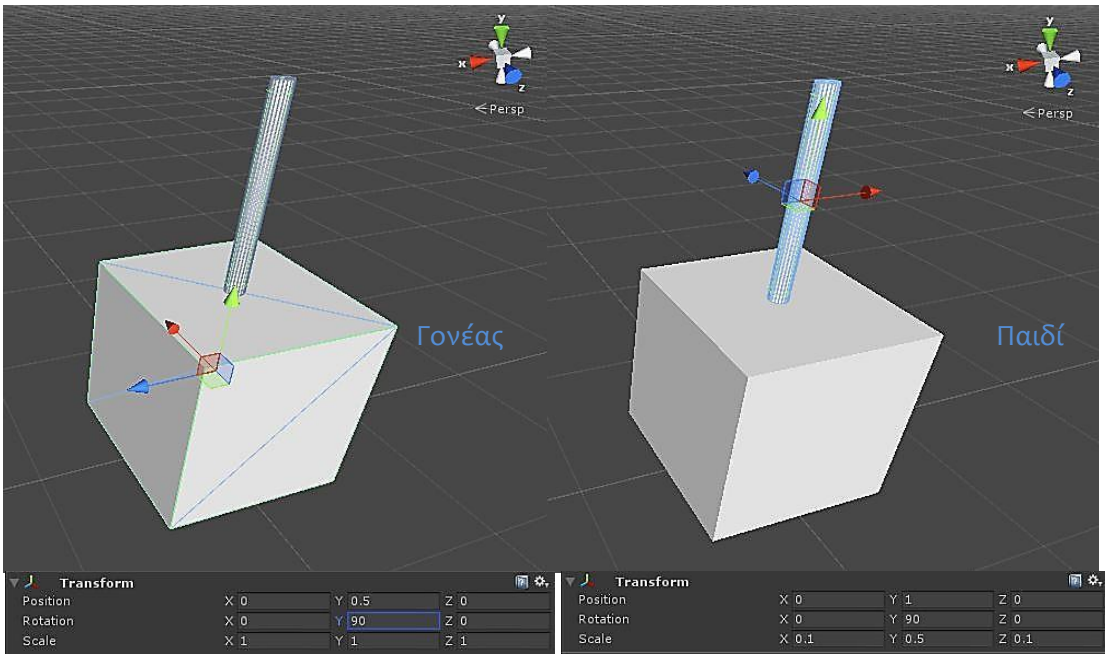
Το αδρανειακό σύστημα συντεταγμένων έχει την αρχή του στο σημείο που το Unity θεωρεί ως σημείο μηδέν. Οποιαδήποτε μετακίνηση αντικειμένου κατά τους τρείς άξονες μετράται ως προς αυτό το σημείο. Για να βρεθεί κάποιο αντικείμενο στο σημείο μηδέν, αρκεί στο transform component του οι τιμές της θέσης να τεθούν όλες ίσες με μηδέν. Προκειμένου να έχουμε εποπτεία των μετακινήσεων και των περιστροφών, εμφανίζονται πάντα στην πάνω δεξιά γωνία της οθόνης, οι άξονες του αδρανειακού συστήματος συντεταγμένων. Για την διευκόλυνση της ανάγνωσής του χρησιμοποιείται χρωματική κωδικοποίηση σύμφωνα με τον κανόνα R(Red), G(Green), B(Blue) → X, Y, Z.



Εικόνα 18 - Εικονίδιο αδρανειακού συστήματος συντεταγμένων

Το τοπικό σύστημα συντεταγμένων δείχνει τη θέση και την περιστροφή ενός αντικειμένου ως προς το σύστημα συντεταγμένων του γονέα του. Πατώντας πάνω σε ένα GameObject εμφανίζεται πάνω του ένα σύστημα συντεταγμένων. Αυτό δείχνει πως έχει μετακινηθεί/περιστραφεί το σύστημα συντεταγμένων του παιδιού σε σχέση με αυτό του γονέα. Ακόμη κι αν ένα αντικείμενο δεν είναι παιδί κάποιου άλλου, είναι παιδί του Unity, άρα το τοπικό σύστημα συντεταγμένων δείχνει την περιστροφή του ως προς το σύστημα που χρησιμοποιεί το Unity.

Τα τοπικά συστήματα συντεταγμένων θα γίνουν πιο κατανοητά μέσω της επόμενης εικόνας:



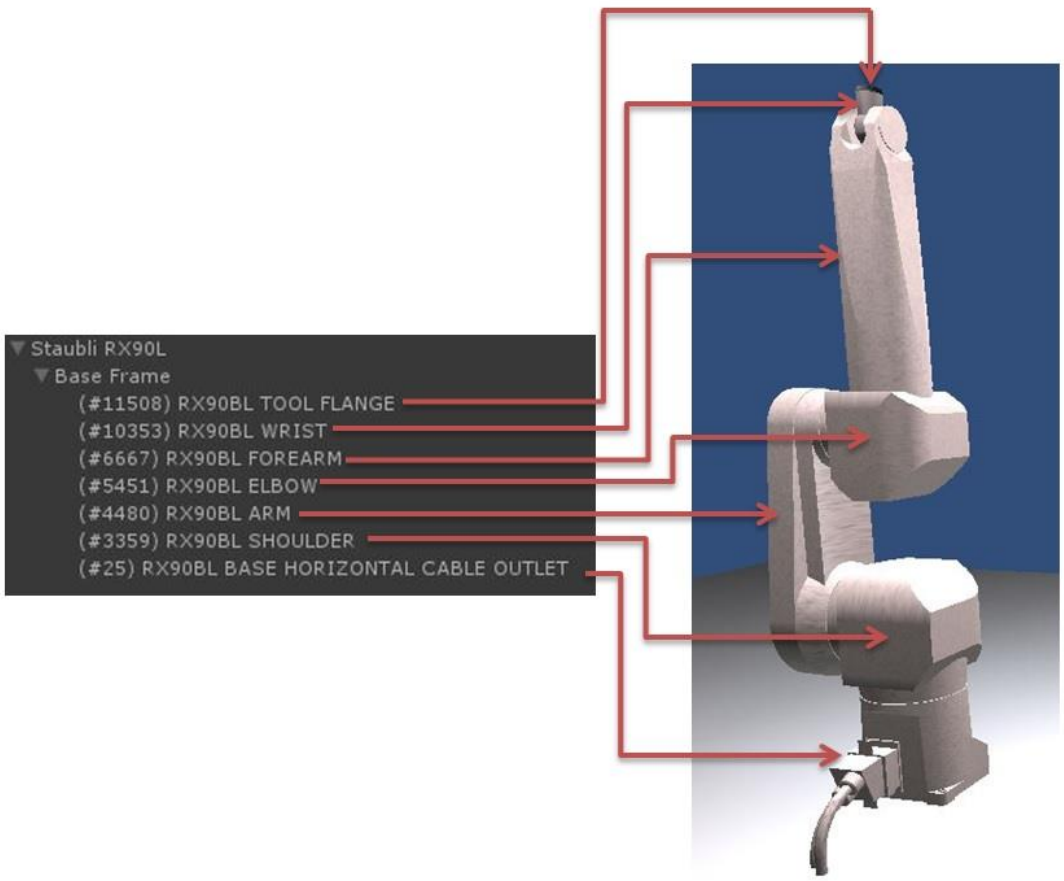
Εικόνα 19 - Τοπικό σύστημα συντεταγμένων

Στην εικόνα 19 ο κύλινδρος είναι παιδί του κύβου. Συγκρίνοντας το σύστημα συντεταγμένων του κύβου με το αδρανειακό του Unity, παρατηρούμε μια περιστροφή 90° γύρω από το άξονα Y, η οποία αναγράφεται και στο transform του κύβου.

Κοιτώντας τώρα το σύστημα συντεταμένων του κυλίνδρου παρατηρούμε μια περιστροφή 180° γύρω από τον άξονα Y, στο transform όμως αναγράφεται περιστροφή 90°. Αυτό συμβαίνει επειδή ο κύλινδρος έχει περιστραφεί 90° ως προς το σύστημα του κύβου, αλλά έχει υιοθετήσει και την περιστροφή του κύβου ως προς το σύστημα του Unity.

4.3 Δημιουργία κινηματικής αλυσίδας

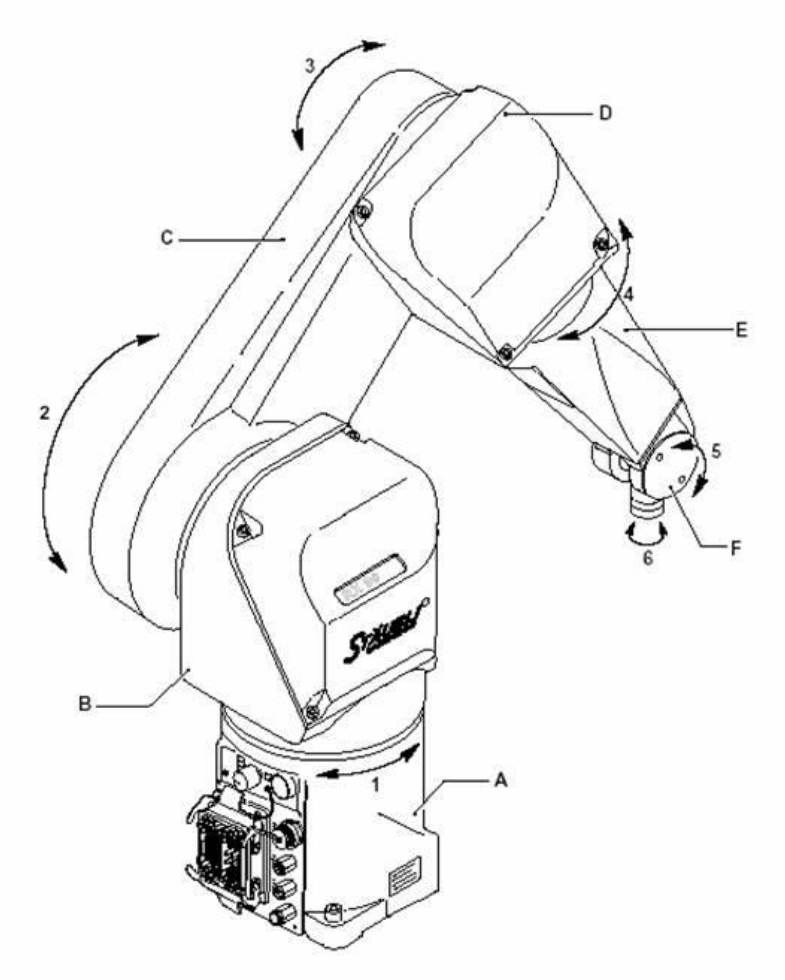
Το μοντέλο του ρομπότ δεν σχεδιάστηκε σαν ένα ενιαίο κομμάτι, αλλά ως πολλά parts τα οποία ενώνονται σε ένα assembly. Κατά την εισαγωγή του όμως, οι σχέσεις που υπήρχαν μεταξύ των parts χάθηκαν επειδή το Unity δεν μπορεί να τις αναγνωρίσει. Η ιεραρχία των κομματιών αμέσως μετά την εισαγωγή του μοντέλου ήταν η επόμενη:



Εικόνα 20 - Κομμάτια μοντέλου κατά την εισαγωγή

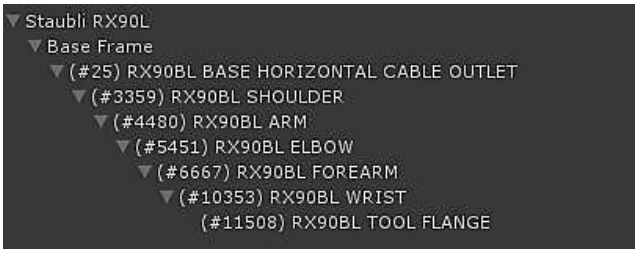
Τα GameObjects “Staubli RX90L” και “Base Frame” είναι κενά και τα υπόλοιπα είναι αυτά που περιέχουν τις γεωμετρίες των επιμέρους κομματιών όπως φαίνεται και στη εικόνα 20. Όπως είναι προφανές, αν προσπαθούσαμε, έτσι όπως είναι η ιεραρχία, να μετακινήσουμε ή να περιστρέψουμε κάποιο κομμάτι, αυτό θα έφευγε από τη θέση του και θα αιωρούταν μακριά από τα υπόλοιπα επειδή δεν υπάρχει κάποια σχέση γονέα - παιδιού που να τα ενώνει.

Για την ανάπτυξη σχέσεων μεταξύ των μελών πρέπει να δούμε πως δουλεύει το ρομπότ. Το Staubli RX90L είναι ένας ανθρωπομορφικός βραχίονας που διαθέτει 6 περιστροφικές αρθρώσεις ή αλλιώς στην γλώσσα της ρομποτικής, είναι ένα 6R Robot.



Εικόνα 21 - Αρθρώσεις Stäubli RX90L

Κάθε άρθρωση περιστρέφεται ανεξάρτητα από τις άλλες και συνδέει ανά 2 τους συνδέσμους του ρομπότ. Συνδυασμός των περιστροφών όλων των αρθρώσεων οδηγεί σε μετακίνηση και προσανατολισμό του τελικού σημείου δράσης. Όταν περιστρέφεται μια άρθρωση, περιστρέφονται μαζί της όλοι οι σύνδεσμοι μετά από αυτή. Για παράδειγμα αν περιστραφεί η άρθρωση 2, θα περιστραφούν και οι σύνδεσμοι C, D, E, F όχι όμως οι A και B. Καταλαβαίνουμε λοιπόν ότι κάθε σύνδεσμος «κληρονομεί» την περιστροφή των προηγούμενων συνδέσμων. Αυτή η παρατήρηση οδήγησε στην δημιουργία της παρακάτω ιεραρχίας:



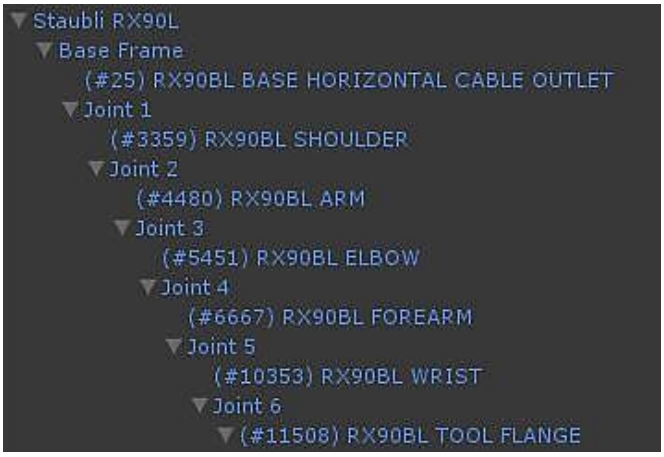
Εικόνα 22 - Ιεραρχία τμημάτων του ρομπότ με σχέσεις γονέα – παιδιού

Η δημιουργία μια τέτοιας ιεραρχίας είχε τα επιθυμητά αποτελέσματα αφού κάθε παιδί στην ιεραρχία κληρονομεί τις ιδιότητες όλων των γονέων του. Δηλαδή, το εσώτερο στην ιεραρχία κληρονομεί τις ιδιότητες των εξώτερων. Άρα, όταν περιστρέφεται ο σύνδεσμος "(#3359) RX90BL SHOULDER" για παράδειγμα περιστρέφονται μαζί του και όλα τα παιδιά του.

Γύρω από ποιόν άξονα όμως; Η απάντηση στη ερώτηση είναι η εξής:

Εφόσον όλοι οι σύνδεσμοι είναι παιδιά κάποιου άλλου, οι τιμές των περιστροφών που θα αναγράφονται στα transform component τους, θα είναι γύρω από τοπικούς άξονες. Οι άξονες όμως αυτοί, είναι διαφορετικοί για κάθε σύνδεσμο. Συγκεκριμένα, οι αρθρώσεις 1, 4 και 6 περιστρέφονται γύρω από τον τοπικό άξονα Z, ενώ οι αρθρώσεις 2, 3 και 5 περιστρέφονται γύρω από τον τοπικό άξονα Y.

Για μαθηματικό λόγο που θα γίνει κατανοητός στο κεφάλαιο της κινηματικής, θα ήταν επιθυμητό όλες οι αρθρώσεις να περιστρέφονται γύρω από τους τοπικούς άξονες Z. Προκειμένου να επιτευχθεί αυτό, προστέθηκαν κενά GameObjects που αναπαριστούν τις αρθρώσεις, ανάμεσα στα GameObjects των συνδέσμων που αυτές ενώνουν. Αυτά μάλιστα περιστράφηκαν κατάλληλα έτσι ώστε ο τοπικός άξονας περιστροφής τους να είναι πάντα ο Z. Έτσι προέκυψε η ιεραρχία της επόμενης εικόνας.



Εικόνα 23 - Τελική ιεραρχία κινηματικής αλυσίδας

Τώρα πλέον, προκειμένου να αλλάξουν τιμή οι γωνίες των αρθρώσεων, περιστρέφουμε γύρω από τον άξονα Z τα GameObjects "Joint 1 – 6" και επειδή οι αντίστοιχοι σύνδεσμοι είναι παιδιά τους, περιστρέφονται κι εκείνοι ανάλογα.

5 Κινηματική ανάλυση βραχίονα

5.1 Εισαγωγή

Η κινηματική είναι η επιστήμη που μελετά την κίνηση των σωμάτων χωρίς να λαμβάνει υπόψη τις δυνάμεις που την προκαλούν. [22] Στην ρομποτική η κινηματική μελετά την κίνηση των επιμέρους αρθρώσεων που απαρτίζουν το ρομπότ και του επιτρέπουν να μετακινεί και να προσανατολίζει το τελικό σημείο δράσης. Για την ακρίβεια, αυτό που ενδιαφέρει είναι η σχέση της θέσης και του προσανατολισμού του τελικού σημείου δράσης, με τις τιμές των γωνιών των επιμέρους αρθρώσεων. Οι αρθρώσεις μπορούν να είναι περιστροφικές, πρισματικές ή σφαιρικές και ενώνουν τα κομμάτια του ρομπότ, που από εδώ και στο εξής θα αναφέρονται ως σύνδεσμοι. Στην προκειμένη περίπτωση οι αρθρώσεις είναι όλες περιστροφικές.

Υπάρχουν δύο προβλήματα που πρέπει να λυθούν: το ευθύ κινηματικό πρόβλημα και το αντίστροφο κινηματικό πρόβλημα.

Το ευθύ κινηματικό πρόβλημα λαμβάνει ως δεδομένες τις τιμές των γωνιών των αρθρώσεων και μέσω των γεωμετρικών χαρακτηριστικών του ρομπότ προσδιορίζει την θέση και τον προσανατολισμό του τελικού σημείου δράσης ως προς το σύστημα συντεταγμένων της βάσης του ρομπότ. Για απλές περιπτώσεις βραχιόνων 2 ή 3 βαθμών ελευθερίας μπορούν να εφαρμοστούν γεωμετρικές λύσεις. Σε πολυπλοκότερα συστήματα όμως, η εφαρμογή των γεωμετρικών λύσεων δεν είναι εύκολα υλοποιήσιμη και γι' αυτό προτιμούνται αναλυτικές λύσεις.

Η αντίστροφη κινηματική μελετά το ακριβώς αντίθετο. Δεδομένης της θέσης και της περιστροφής του τελικού σημείου δράσης, ποιες είναι οι τιμές των γωνιών των αρθρώσεων που το οδηγούν σε αυτή τη θέση; Η λύση σε αυτό το πρόβλημα δεν υπάρχει πάντα και εφόσον υπάρχει, οι λύσεις μπορεί να είναι πολλαπλές.

5.2 Ευθεία κινηματική 6R ρομποτικού βραχίονα

5.2.1 Η μέθοδος Denavit – Hartenberg

Η μέθοδος των Denavit – Hartenberg (D - H) αναπτύχθηκε για την περιγραφή της κινηματικής σύνθετων μηχανισμών από έναν ελάχιστο αριθμό παραμέτρων. [23] Η μέθοδος αυτή έχει στόχο την παραγωγή των ομογενών μετασχηματισμών που συνδέουν τη θέση και τον προσανατολισμό ενός συνδέσμου ως προς τον προηγούμενο. Οι ομογενείς μετασχηματισμοί είναι πίνακες 4x4 και περιέχουν και την θέση και τον προσανατολισμό. [20] Πολλαπλασιασμός των επιμέρους πινάκων διαδοχικά από τη βάση μέχρι το εργαλείο δίνει τον ομογενή μετασχηματισμό του εργαλείου ως προς τη βάση του ρομπότ.

Η μέθοδος χωρίζεται σε δύο βήματα:

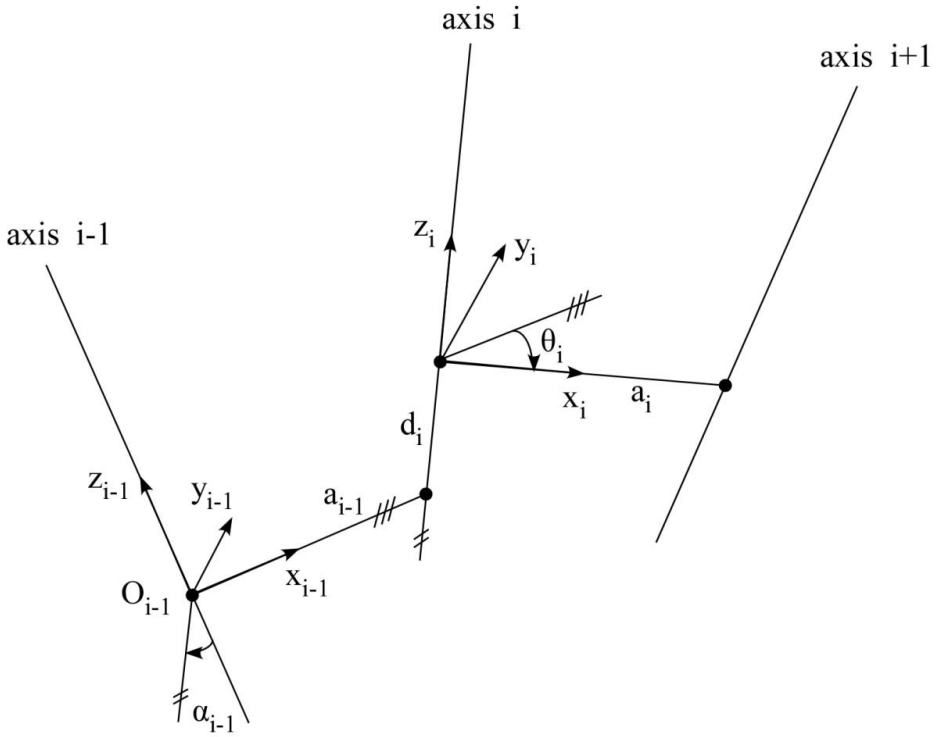
α) Προσάρτηση συστημάτων συντεταγμένων (ΣΣ) στους συνδέσμους

β) Υπολογισμός παραμέτρων Denavit – Hartenberg

Η προσάρτηση των συστημάτων συντεταγμένων γίνεται ως εξής:

- Προσαρτούμε το πρώτο σύστημα συντεταγμένων στη βάση του ρομπότ και το ονομάζουμε ΣΣ {0}. Αυτό το σύστημα συντεταγμένων δεν κινείται και συνήθως είναι το σύστημα αναφοράς. Αυτό το ΣΣ μπορεί να επιλεγεί τυχαία, βοηθάει όμως να επιλεγεί έτσι ώστε το μοναδιαίο $\hat{z}_0$ να έχει τη διεύθυνση του πρώτου άξονα (δηλαδή του $\hat{z}_1$) και το ΣΣ {0} να συμπίπτει με το ΣΣ {1} όταν η πρώτη μεταβλητή άρθρωσης είναι μηδέν.
- Για τους συνδέσμους 1 έως i – 1 (όπου i είναι ο αριθμός των συνδέσμων), το μοναδιαίο $\hat{z}_i$ είναι κατά τον άξονα περιστροφής της άρθρωσης i. Το διάνυσμα $\hat{x}_i$ είναι κατά μήκος της κοινής καθέτου των αξόνων $\hat{z}_i$ και $\hat{z}_{i+1}$. Η αρχή των αξόνων είναι η τομή του $\hat{x}_i$ με τον $\hat{z}_i$ και είναι επιθυμητό να βρίσκεται πάνω στην άρθρωση γύρω από την οποία περιστρέφεται ο σύνδεσμος i. Το $\hat{y}_i$ έχει προσανατολισμό τέτοιο ώστε να προκύπτει δεξιόστροφο σύστημα συντεταγμένων. Αν οι άξονες $\hat{z}_i$ και $\hat{z}_{i+1}$ είναι κάθετοι, τότε ο $\hat{x}_i$ επιλέγεται έτσι ώστε να είναι κάθετος στο επίπεδο που ορίζεται από τα μοναδιαία $\hat{z}_i$ και $\hat{z}_{i+1}$. Σε αυτή την περίπτωση η αρχή του ΣΣ {i} είναι στην τομή των $\hat{z}_i$ και $\hat{z}_{i+1}$.

- Το σύστημα συντεταγμένων του τελευταίου συνδέσμου προσαρτείται έτσι ώστε το μοναδιαίο $\hat{z}_i$ να είναι κατά τον άξονα περιστροφής της άρθρωσης. Η διεύθυνση του $\hat{x}_N$ επιλέγεται έτσι ώστε να είναι συγγραμικό με το $\hat{x}_{N-1}$, όταν η μεταβλητή άρθρωσης $\theta_N = 0$. Το $\hat{y}_N$ έχει προσανατολισμό τέτοιο ώστε να προκύπτει δεξιόστροφο σύστημα συντεταγμένων. [23]



Εικόνα 24 - Προσάρτηση ΣΣ που απαιτούνται κατά την εφαρμογή της μεθόδου D - H **[23]**

Οι παράμετροι D – H που απαιτούνται ορίζονται ως εξής:

- Μήκη συνδέσμων, a_{i-1} : απόσταση $\hat{z}_{i-1}$, $\hat{z}_i$ κατά το $\hat{x}_{i-1}$
- Στρέψεις συνδέσμων, α_{i-1}: γωνία $\hat{z}_{i-1}$, $\hat{z}_i$ γύρω από το $\hat{x}_{i-1}$
- Μετατοπίσεις συνδέσμων, d_i : απόσταση $\hat{x}_{i-1}$, $\hat{x}_i$ κατά το $\hat{z}_i$
- Γωνίες αρθρώσεων θ_i : γωνία $\hat{x}_{i-1}$, $\hat{x}_i$ γύρω από το $\hat{z}_i$

Για κάθε σύνδεσμο λοιπόν υπολογίζονται 4 παράμετροι και συμπληρώνουμε τον παρακάτω πίνακα:

i	α_{i-1}	a_{i-1}	d_i	θ_i
1				
2				
…				
n				

Πίνακας 4 - Πίνακας παραμέτρων D – H

Αφού έχει συμπληρωθεί ο πίνακας D – H μπορούν να βρεθούν στη συνέχεια οι ομογενείς μετασχηματισμοί κάθε συνδέσμου ως προς τον προηγούμενο. Ο συμβολισμός που χρησιμοποιείται για τους ομογενείς μετασχηματισμούς είναι ο ${}^{i-1}T_i$ και σημαίνει: Η θέση και η περιστροφή του συστήματος συντεταγμένων i ως προς το σύστημα συντεταγμένων i – 1.

Αυτός προκύπτει ως διαδοχικές περιστροφές – μετατοπίσεις κατά τις ποσότητες D – H. Δηλαδή:

$$ {}^{i-1}T_i = ROT_x(a_{i-1}) \cdot TRANS(\mathrm{a}_{i-1}) \cdot ROT_z(\theta_i) \cdot TRANS(d_i) $$

όπου:

$$ \mathrm{a}_{i-1} = \begin{bmatrix} \mathrm{a}_{i-1} \\ 0 \\ 0 \end{bmatrix} \text{ και } d_i = \begin{bmatrix} 0 \\ 0 \\ d_i \end{bmatrix}. $$

Αναλυτικά η παραπάνω εξίσωση δίνει:

$$ {}^{i-1}T_i = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & ca_{i-1} & -sa_{i-1} & 0 \\ 0 & sa_{i-1} & ca_{i-1} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & \mathrm{a}_{i-1} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c\theta_i & -s\theta_i & 0 & 0 \\ s\theta_i & c\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} = $$

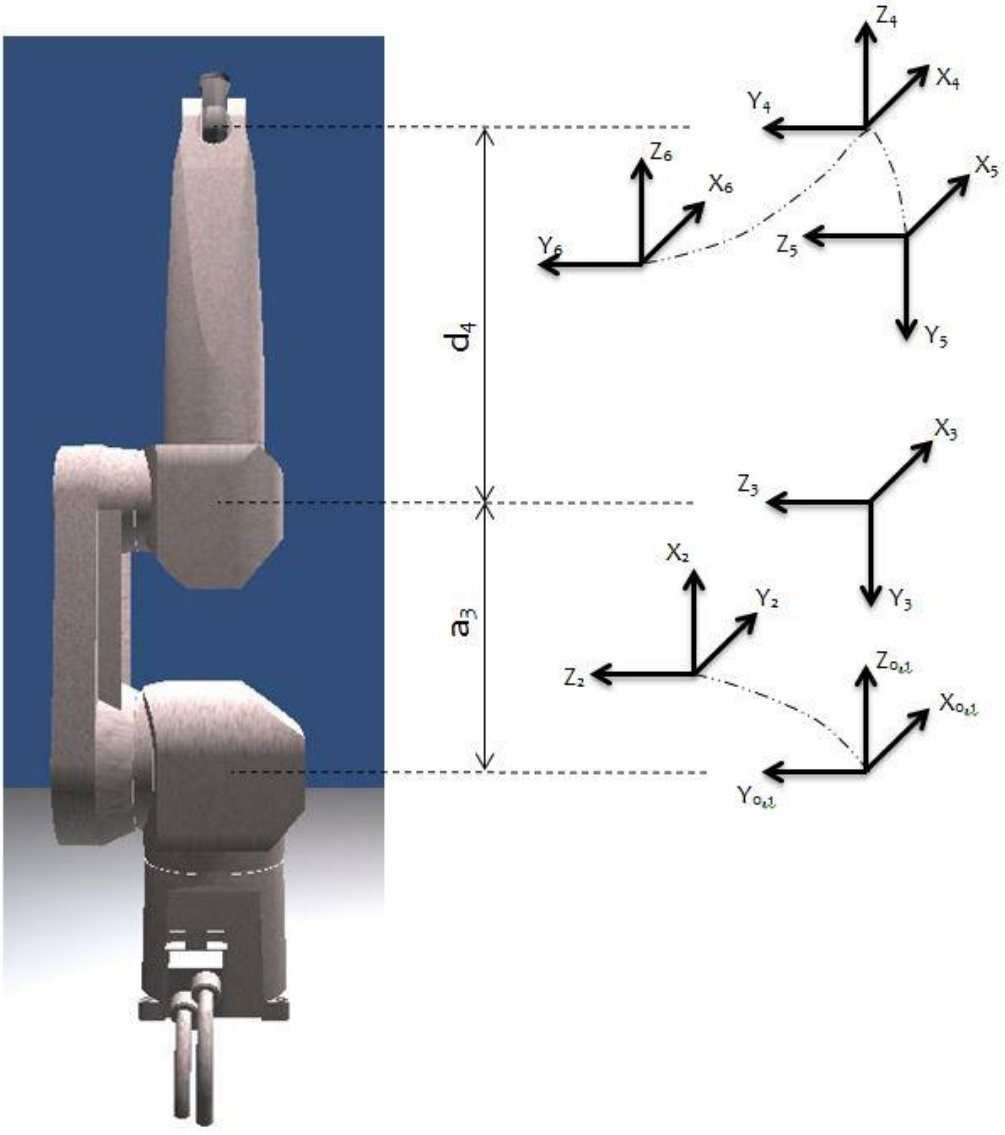
$$ {}^{i-1}T_i = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & \mathrm{a}_{i-1} \\ ca_{i-1} \cdot s\theta_i & ca_{i-1} \cdot c\theta_i & -sa_{i-1} & -sa_{i-1} \cdot d_i \\ sa_{i-1} \cdot s\theta_i & sa_{i-1} \cdot c\theta_i & ca_{i-1} & ca_{i-1} \cdot d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} [23] $$

Ο ομογενής μετασχηματισμός του τελικού σημείου δράσης σε σχέση με το σύστημα συντεταγμένων της αρχής προκύπτει από διαδοχικό πολλαπλασιασμό των επιμέρους ομογενών μετασχηματισμών.

$$^{0}T_{N} = {}^{0}T_{1} \cdot {}^{1}T_{2} \cdot \ldots \cdot {}^{N-1}T_{N}$$

5.2.2 Υλοποίηση για το Stäubli RX90-L

Η μέθοδος που περιγράφηκε παραπάνω εφαρμόστηκε στο ρομπότ που χρησιμοποιήθηκε στην εργασία. Αρχικά έγινε τοποθέτηση των συστημάτων συντεταγμένων όπως φαίνεται στην παρακάτω εικόνα.



Εικόνα 25 – Συστήματα συντεταγμένων της μεθοδολογίας D - H για στο Stäubli RX90L

Το ΣΣ {0} τοποθετήθηκε στην πρώτη άρθρωση του ρομπότ, έτσι ώστε όταν η μεταβλητή της άρθρωσης 1 είναι μηδέν, τα ΣΣ {0} και ΣΣ {1} να ταυτίζονται. Επίσης, το πραγματικό ρομπότ μετράει τη θέση του τελικού σημείου δράσης ως προς αυτό το σύστημα.

Ο πίνακας των παραμέτρων D – H για το ρομπότ που προέκυψε είναι ο επόμενος.

i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	0	θ_1
2	0	-90	0	θ_2
3	a_3	0	0	θ_3
4	0	90	d_4	θ_4
5	0	-90	0	θ_5
6	0	90	0	θ_6

Πίνακας 5 - Πίνακας παραμέτρων D - H για το Staubli RX90L

Στην συνέχεια, χρησιμοποιώντας την εξίσωση που προέκυψε παραπάνω, δημιουργήθηκαν οι πίνακες ομογενούς μετασχηματισμού κάθε συνδέσμου[1].

$$^0T_1 = \begin{bmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad ^1T_2 = \begin{bmatrix} c_2 & -s_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -s_2 & -c_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^2T_3 = \begin{bmatrix} c_3 & -s_3 & 0 & a_3 \\ s_3 & c_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad ^3T_4 = \begin{bmatrix} c_4 & -s_4 & 0 & 0 \\ 0 & 0 & -1 & -d_4 \\ s_4 & c_4 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^4T_5 = \begin{bmatrix} c_5 & -s_5 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -s_5 & -c_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad ^5T_6 = \begin{bmatrix} c_6 & -s_6 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ s_6 & c_6 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

[1] $c_1 = \cos(\theta_1)$, $s_1 = \sin(\theta_1)$, $c_{12} = \cos(\theta_1 + \theta_2)$, $s_{12} = \sin(\theta_1 + \theta_2)$

Ο ομογενής μετασχηματισμός του άκρου ως προς τη βάση προέκυψε:

$$ {}^{0}T_{6} \;=\; {}^{0}T_{1} \;\cdot\; {}^{1}T_{2} \;\cdot\; {}^{2}T_{3} \;\cdot\; {}^{3}T_{4} \;\cdot\; {}^{4}T_{5} \;\cdot\; {}^{5}T_{6} \;= \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

όπου:

$r_{11} = c_1 \cdot [c_{23} \cdot (c_4 \cdot c_5 \cdot c_6 - s_4 \cdot s_6) - s_{23} \cdot s_5 \cdot s_6] + s_1 \cdot (c_4 \cdot s_6 - s_4 \cdot c_5 \cdot c_6)$

$r_{12} = c_1 \cdot [c_{23} \cdot (-c_4 \cdot c_5 \cdot s_6 - s_4 \cdot c_6) + s_{23} \cdot s_5 \cdot s_6] + s_1 \cdot (s_4 \cdot c_5 \cdot s_6 - c_4 \cdot c_6)$

$r_{13} = c_1 \cdot (c_{23} \cdot c_4 \cdot s_5 + s_{23} \cdot c_5) - s_1 \cdot s_4 \cdot s_5$

$r_{21} = s_1 \cdot [-s_{23} \cdot s_5 \cdot c_6 - c_{23} \cdot (s_4 \cdot s_6 - c_4 \cdot c_5 \cdot c_6)] + c_1 \cdot (s_4 \cdot c_5 \cdot c_6 + c_4 \cdot s_6)$

$r_{22} = s_1 \cdot [c_{23} \cdot (-c_4 \cdot c_5 \cdot s_6 - s_4 \cdot c_6) + s_{23} \cdot s_5 \cdot s_6] - c_1 \cdot (s_4 \cdot c_5 \cdot s_6 - c_4 \cdot c_6)$

$r_{23} = -s_{23} \cdot (c_4 \cdot c_5 \cdot c_6 - s_4 \cdot s_6) - c_{23} \cdot s_5 \cdot c_6$

$r_{31} = -s_{23} \cdot (c_4 \cdot c_5 \cdot c_6 - s_4 \cdot s_6) - c_{23} \cdot s_5 \cdot c_6$

$r_{32} = s_{23} \cdot (c_4 \cdot c_5 \cdot s_6 + s_4 \cdot c_6) + c_{23} \cdot s_5 \cdot s_6$

$r_{33} = c_{23} \cdot c_5 - s_{23} \cdot c_4 \cdot s_5$

$p_x = c_1 \cdot s_{23} \cdot d_4 + c_1 \cdot c_2 \cdot \mathrm{a}_3$

$p_y = s_1 \cdot s_{23} \cdot d_4 + s_1 \cdot c_2 \cdot \mathrm{a}_3$

$p_z = c_{23} \cdot d_4 - \mathrm{a}_3 \cdot s_2$

5.3 Αντίστροφη κινηματική 6R ρομποτικού βραχίονα

5.3.1 Η μέθοδος απόπλεξης/αποσύζευξης του Pieper

Ζητούμενο σε αυτή την ενότητα είναι η εύρεση των τιμών των γωνιών των αρθρώσεων, όταν γνωρίζουμε την θέση και την περιστροφή του τελικού σημείου δράσης. Παρόλο που το πρόβλημα της αντίστροφης κινηματικής είναι δύσκολο, στην περίπτωση των βραχιόνων που έχουν 6 αρθρώσεις, από τις οποίες οι 3 τελευταίες είναι περιστροφικές και οι άξονές τους τέμνονται στο ίδιο σημείο, υπάρχει κλειστή λύση. [24]

Πριν προχωρήσουμε στην ανάλυση της αντίστροφης κινηματικής πρέπει να αναφέρουμε το εξής: στην προηγούμενη ενότητα που τοποθετήθηκαν τα συστήματα συντεταγμένων, το ΣΣ {6} τοποθετήθηκε στον καρπό του ρομπότ και όχι στην άκρη της φλάντζας. Με αυτόν τον τρόπο χάνουμε το μήκος μετά τον καρπό από τους υπολογισμούς, δηλαδή ουσιαστικά μετράμε την θέση του καρπού. Η περιστροφή όμως του τελικού σημείου δράσης δεν εξαρτάται από αυτό το μήκος, επομένως περιγράφεται πλήρως.

Με την τεχνική αυτή κερδίσαμε την τομή των 3 τελευταίων συστημάτων συντεταγμένων. Η μέθοδος αυτή ονομάζεται μέθοδος απόπλεξης/αποσύζευξης του Pieper. Σύμφωνα τη μέθοδο αυτή, το πρόβλημα της αντίστροφης κινηματικής αναλύεται σε δύο απλούστερα, τα οποία ονομάζονται αντίστοιχα **αντίστροφη κινηματική θέσης** και **αντίστροφη κινηματική προσανατολισμού.** Η αντίστροφη κινηματική θέσης υπολογίζει τις τιμές των πρώτων 3 αρθρώσεων χρησιμοποιώντας μόνο την θέση του καρπού και έπειτα με την αντίστροφη κινηματική προσανατολισμού υπολογίζονται οι 3 τελευταίες χρησιμοποιώντας τον πίνακα περιστροφών. [24]

5.3.2 Υλοποίηση για το Stäubli RX90-L

4.3.2.1 Υπολογισμός θ_1, θ_2, θ_3

Όπως είναι γνωστό και από προηγούμενη ενότητα ισχύει:

$$ {}^0T_6 = {}^0T_1 \cdot {}^1T_2 \cdot {}^2T_3 \cdot {}^3T_4 \cdot {}^4T_5 \cdot {}^5T_6 = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (1) $$

Για τον υπολογισμό της θ_1 επαναδιατυπώνουμε την παραπάνω εξίσωση αντιστρέφοντας τον πίνακα ${}^0T_1$ και μεταφέροντάς τον στο αριστερό μέλος. Προκύπτει:

$$ \left({}^0T_6\right) \cdot \left({}^0T_1\right)^{-1} = {}^1T_2 \cdot {}^2T_3 \cdot {}^3T_4 \cdot {}^4T_5 \cdot {}^5T_6 \implies $$

$$ \begin{bmatrix} c_1 & s_1 & 0 & 0 \\ -s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \cdot \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} {}^1r_{11} & {}^1r_{12} & {}^1r_{13} & {}^1p_x \\ {}^1r_{21} & {}^1r_{22} & {}^1r_{23} & {}^1p_y \\ {}^1r_{31} & {}^1r_{32} & {}^1r_{33} & {}^1p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Εξισώνοντας τα στοιχεία (2,4) στα δύο μέλη της εξίσωσης παίρνουμε:

$$-s_1 \cdot p_x + c_1 \cdot p_y = 0 \Longrightarrow \tag{2}$$

$$\frac{s_1}{c_1} = \frac{p_y}{p_x} \Longrightarrow \tan(\theta_1) = \frac{p_y}{p_x} \Longrightarrow$$

$$\theta_1 = \begin{cases} Atan2(p_y, p_x) \\ Atan2(-p_y, -p_x) \end{cases}$$

Εξισώνουμε τους όρους (1,4) και (3,4) της παραπάνω εξίσωσης και προκύπτει το σύστημα:

$$c_1 \cdot p_x + s_1 \cdot p_y = \mathrm{a}_3 \cdot c_2 + d_4 \cdot s_{23} \tag{3}$$

$$p_z = d_4 \cdot c_{23} - \mathrm{a}_3 \cdot s_2 \tag{4}$$

Υψώνουμε στο τετράγωνο τις εξισώσεις (2), (3) και (4) και προκύπτει το παρακάτω σύστημα τριών εξισώσεων:

$$c_1^2 \cdot p_y^2 + s_1^2 \cdot p_x^2 - 2 \cdot c_1 \cdot s_1 \cdot p_x \cdot p_y = 0 \tag{5}$$

$$c_1^2 \cdot p_x^2 + s_1^2 \cdot p_y^2 + 2 \cdot c_1 \cdot s_1 \cdot p_x \cdot p_y = \mathrm{a}_3^2 \cdot c_2^2 + d_4 \cdot s_{23}^2 + 2 \cdot \mathrm{a}_3 \cdot d_4 \cdot c_2 \cdot s_{23} \tag{6}$$

$$p_z^2 = d_4^2 \cdot c_{23}^2 + a_3^2 \cdot s_2^2 - 2 \cdot d_4 \cdot \mathrm{a}_3 \cdot s_2 \cdot c_{23} \tag{7}$$

Προσθέτουμε τις εξισώσεις (5), (6) και (7) κατά μέλη και μετά την απαλοιφή των αντίθετων όρων έχουμε:

$$p_x^2 + p_y^2 + p_z^2 = \mathrm{a}_3^2 + d_4^2 + 2 \cdot \mathrm{a}_3 \cdot d_4 \cdot (c_2 \cdot s_{23} - s_2 \cdot c_{23}) \Longrightarrow$$

$$p_x^2 + p_y^2 + p_z^2 = \mathrm{a}_3^2 + d_4^2 + 2 \cdot \mathrm{a}_3 \cdot d_4 \cdot s_3 \Longrightarrow$$

$$s_3 = \frac{p_x^2 + p_y^2 + p_z^2 - \mathrm{a}_3^2 - d_4^2}{2 \cdot \mathrm{a}_3 \cdot d_4} \Longrightarrow$$

$$\theta_3 = Asin\left(\frac{p_x^2 + p_y^2 + p_z^2 - \mathrm{a}_3^2 - d_4^2}{2 \cdot \mathrm{a}_3 \cdot d_4}\right)$$

Η συνάρτηση Asin επιστρέφει τιμές από -90° έως 90°. Επειδή ο αριθμητής στη προκειμένη περίπτωση έχει όλες τις τιμές υψωμένες στο τετράγωνο, οι τιμές της γωνίας που θα προκύψουν για αρνητικές τιμές της θέσης δεν θα αντιστοιχούν στις πραγματικές. Επίσης χρειαζόμαστε μεγαλύτερο εύρος γωνιών για να καλύψουμε τα όρια στα οποία φτάνει η άρθρωση 3. Γι' αυτούς τους λόγους παίρνουμε και τις υπόλοιπες γωνίες που δίνουν το ίδιο ημίτονο, δηλαδή:

$$\theta_3 = \begin{cases} Asin\left(\dfrac{p_x^2 + p_y^2 + p_z^2 - \mathrm{a}_3^2 - d_4^2}{2 \cdot \mathrm{a}_3 \cdot d_4}\right) \\ -Asin\left(\dfrac{p_x^2 + p_y^2 + p_z^2 - \mathrm{a}_3^2 - d_4^2}{2 \cdot \mathrm{a}_3 \cdot d_4}\right) \\ \pi + Asin\left(\dfrac{p_x^2 + p_y^2 + p_z^2 - \mathrm{a}_3^2 - d_4^2}{2 \cdot \mathrm{a}_3 \cdot d_4}\right) \\ \pi - Asin\left(\dfrac{p_x^2 + p_y^2 + p_z^2 - \mathrm{a}_3^2 - d_4^2}{2 \cdot \mathrm{a}_3 \cdot d_4}\right) \end{cases}$$

Με γνωστές τις θ_1 και θ_3, επαναδιατυπώνουμε την εξίσωση (1) ως εξής:

$${}^0T_6 = {}^0T_1 \cdot {}^1T_2 \cdot {}^2T_3 \cdot {}^3T_4 \cdot {}^4T_5 \cdot {}^5T_6 \implies$$

$${}^0T_6 = {}^0T_3 \cdot {}^3T_6 \implies \left({}^0T_3\right)^{-1} \cdot \left({}^0T_6\right) = {}^3T_6 \implies$$

$$\begin{bmatrix} c_1c_{23} & s_1c_{23} & -s_{23} & -\mathrm{a}_3c_3 \\ -c_1s_{23} & -s_1s_{23} & -c_{23} & \mathrm{a}_3s_3 \\ -s_1 & c_1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} =$$

$$\begin{bmatrix} c_4 \cdot c_5 \cdot c_6 - s_4 \cdot s_6 & -c_4 \cdot c_5 \cdot s_6 - s_4 \cdot c_6 & c_4 \cdot s_5 & 0 \\ c_6 \cdot s_5 & -s_5 \cdot s_6 & -c_5 & -d_4 \\ s_4 \cdot c_5 \cdot c_6 + c_4 \cdot s_6 & c_4 \cdot c_6 - s_4 \cdot c_5 \cdot c_6 & s_4s_5 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (8)$$

Εξισώνουμε τα στοιχεία (1,4) και (2,4) και έχουμε:

$$\left.\begin{array}{l} c_1c_{23}\cdot p_x + s_1c_{23}\cdot p_y - s_{23}\cdot p_z - \mathrm{a}_3c_3 = 0 \\ -c_1s_{23}\cdot p_x - s_1s_{23}\cdot p_y - c_{23}\cdot p_z + \mathrm{a}_3s_3 = -d_4 \end{array}\right\} \Rightarrow$$

$$\begin{cases} c_{23} = \dfrac{\mathrm{a}_3c_3 + s_{23}\cdot p_z}{c_1\cdot p_x + s_1\cdot p_y} & (9) \\ s_{23} = \dfrac{d_4 + \mathrm{a}_3s_3 - c_{23}\cdot p_z}{c_1\cdot p_x + s_1\cdot p_y} & (10) \end{cases}$$

Αντικαθιστούμε την (9) στην (10) και την (10) στην (9) αντίστοιχα και το σύστημα γίνεται:

$$c_{23} = \frac{(d_4 + \mathrm{a}_3s_3)p_z + \mathrm{a}_3c_3\left(c_1\cdot p_x + s_1\cdot p_y\right)}{\left(c_1\cdot p_x + s_1\cdot p_y\right)^2 + p_z^2}$$

$$s_{23} = \frac{-\mathrm{a}_3c_3p_z + (d_4 + \mathrm{a}_3s_3)\cdot\left(c_1\cdot p_x + s_1\cdot p_y\right)}{\left(c_1\cdot p_x + s_1\cdot p_y\right)^2 + p_z^2}$$

Διαιρώντας τις τελευταίες κατά μέλη προκύπτει:

$$\tan(\theta_2+\theta_3) = \frac{-\mathrm{a}_3c_3p_z + (d_4 + \mathrm{a}_3s_3)\cdot\left(c_1\cdot p_x + s_1\cdot p_y\right)}{(d_4 + \mathrm{a}_3s_3)p_z + \mathrm{a}_3c_3\left(c_1\cdot p_x + s_1\cdot p_y\right)} \Rightarrow$$

$$\theta_2+\theta_3 = Atan2\Big(-\mathrm{a}_3c_3p_z + (d_4 + \mathrm{a}_3s_3)\cdot\left(c_1\cdot p_x + s_1\cdot p_y\right),$$
$$(d_4 + \mathrm{a}_3s_3)p_z + \mathrm{a}_3c_3\left(c_1\cdot p_x + s_1\cdot p_y\right)\Big)$$

και επειδή η θ_3 είναι γνωστή:

$$\theta_2 = (\theta_2+\theta_3) - \theta_3$$

4.3.2.2 Υπολογισμός θ_4, θ_5, θ_6

Τώρα πλέον το αριστερό μέλος της εξίσωσης (8) είναι γνωστό. Εξισώνουμε τα στοιχεία (2,3) της εξίσωσης και παίρνουμε:

$$-c_1 s_{23} r_{13} - s_1 s_{23} r_{23} - c_{23} r_{33} = -c_5 \Longrightarrow c_5 = c_1 s_{23} r_{13} + s_1 s_{23} r_{23} + c_{23} r_{33} \Longrightarrow$$

$$\theta_5 = \pm\, Acos(c_1 s_{23} r_{13} + s_1 s_{23} r_{23} + c_{23} r_{33})$$

Εξισώνουμε τα στοιχεία (1,3) και (3,3) της ίδιας εξίσωσης:

$$c_1 c_{23} r_{13} + s_1 c_{23} r_{23} - s_{23} r_{33} = c_4 \cdot s_5 \tag{11}$$

$$-s_1 r_{13} + c_1 r_{23} = s_4 s_5 \tag{12}$$

Αν $\sin(\theta_5) \neq 0$ διαιρούμε τις (11) και (12) κατά μέλη και προκύπτει:

$$\tan(\theta_4) = \frac{-s_1 r_{13} + c_1 r_{23}}{c_1 c_{23} r_{13} + s_1 c_{23} r_{23} - s_{23} r_{33}} \Longrightarrow$$

$$\theta_4 = Atan2(-s_1 r_{13} + c_1 r_{23},\ \ c_1 c_{23} r_{13} + s_1 c_{23} r_{23} - s_{23} r_{33})$$

Για να καλύψουμε τα όρια στα οποία φτάνει η άρθρωση 4 παίρνουμε ακόμη μία τιμή της θ_4 σύμφωνα με τις παρακάτω περιπτώσεις:

- Αν θ_4 = 90 ÷ 180 τότε:

$$\theta_{42} = -360 + Atan2(-s_1 r_{13} + c_1 r_{23},\ \ c_1 c_{23} r_{13} + s_1 c_{23} r_{23} - s_{23} r_{33})$$

- Αν θ_4 = -90 ÷ -180 τότε:

$$\theta_{42} = 360 + Atan2(-s_1 r_{13} + c_1 r_{23},\ \ c_1 c_{23} r_{13} + s_1 c_{23} r_{23} - s_{23} r_{33})$$

Αν $\sin(\theta_5)$ = 0 οι γωνίες θ_4 και θ_6 δεν υπολογίζονται ξεχωριστά, αλλά και οι δύο μαζί. Σε αυτή την περίπτωση υποθέτουμε ότι θ_4 = 0 και όλη την περιστροφή την παίρνει η θ_6.

Μένει να υπολογιστεί η γωνία θ_6. Για τον υπολογισμό της γράφουμε την εξίσωση (1) στη μορφή:

$$\left({}^0T_5 \right)^{-1} \cdot \left({}^0T_6 \right) = {}^5T_6$$

Εξισώνουμε τα στοιχεία (3,1) της παραπάνω εξίσωσης και παίρνουμε:

$$-(s_1c_4 + c_1s_4c_{23})r_{11} + (c_1c_4 - s_1s_4c_{23})r_{21} + s_4s_{23}r_{31} = s_6 \Longrightarrow$$

$$\theta_6 = Asin(-(s_1c_4 + c_1s_4c_{23})r_{11} + (c_1c_4 - s_1s_4c_{23})r_{21} + s_4s_{23}r_{31})$$

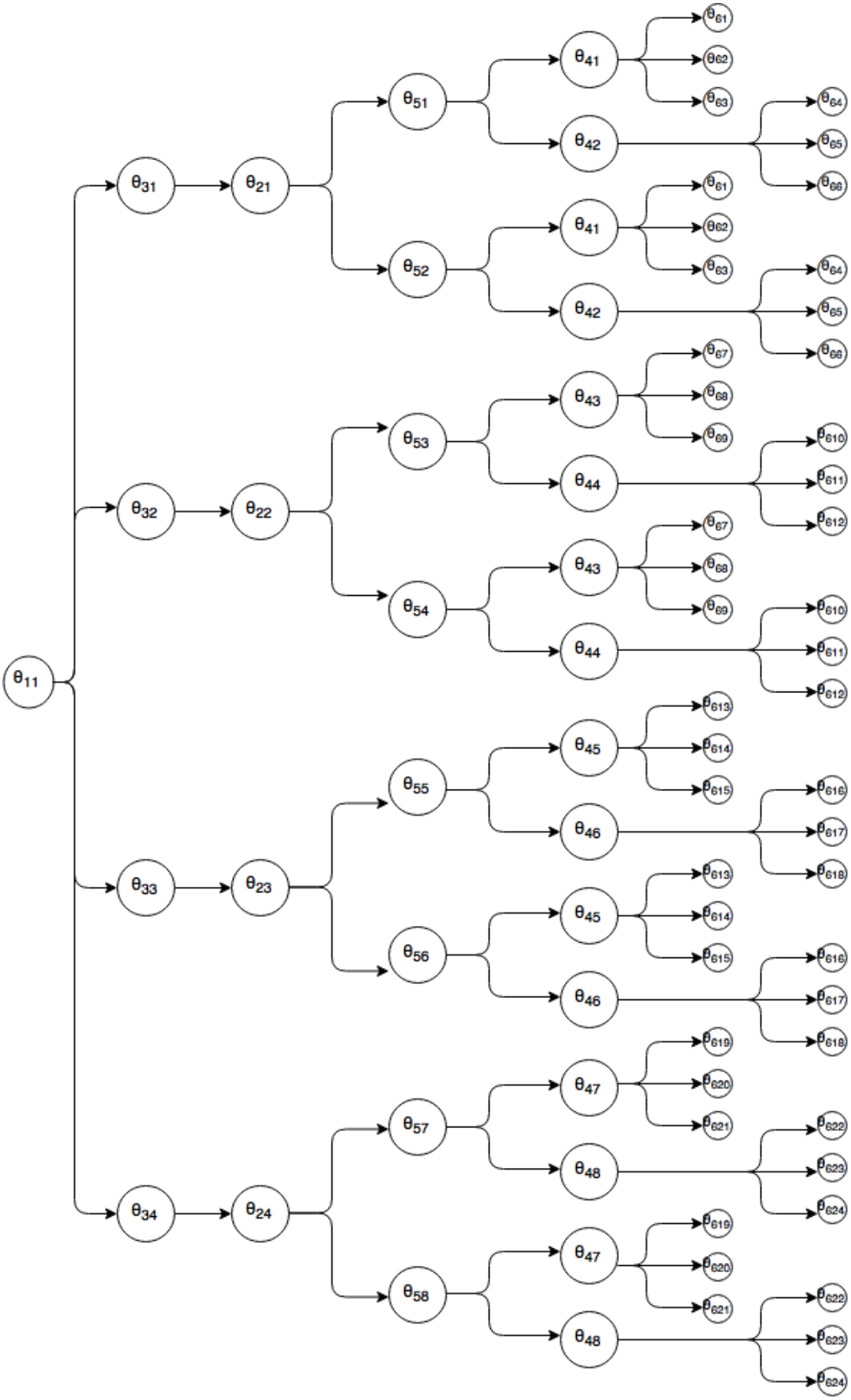
Κι εδώ προκειμένου να καλύψουμε τα όρια στα οποία φτάνει η άρθρωση 6 παίρνουμε επιπλέον τις περιπτώσεις:

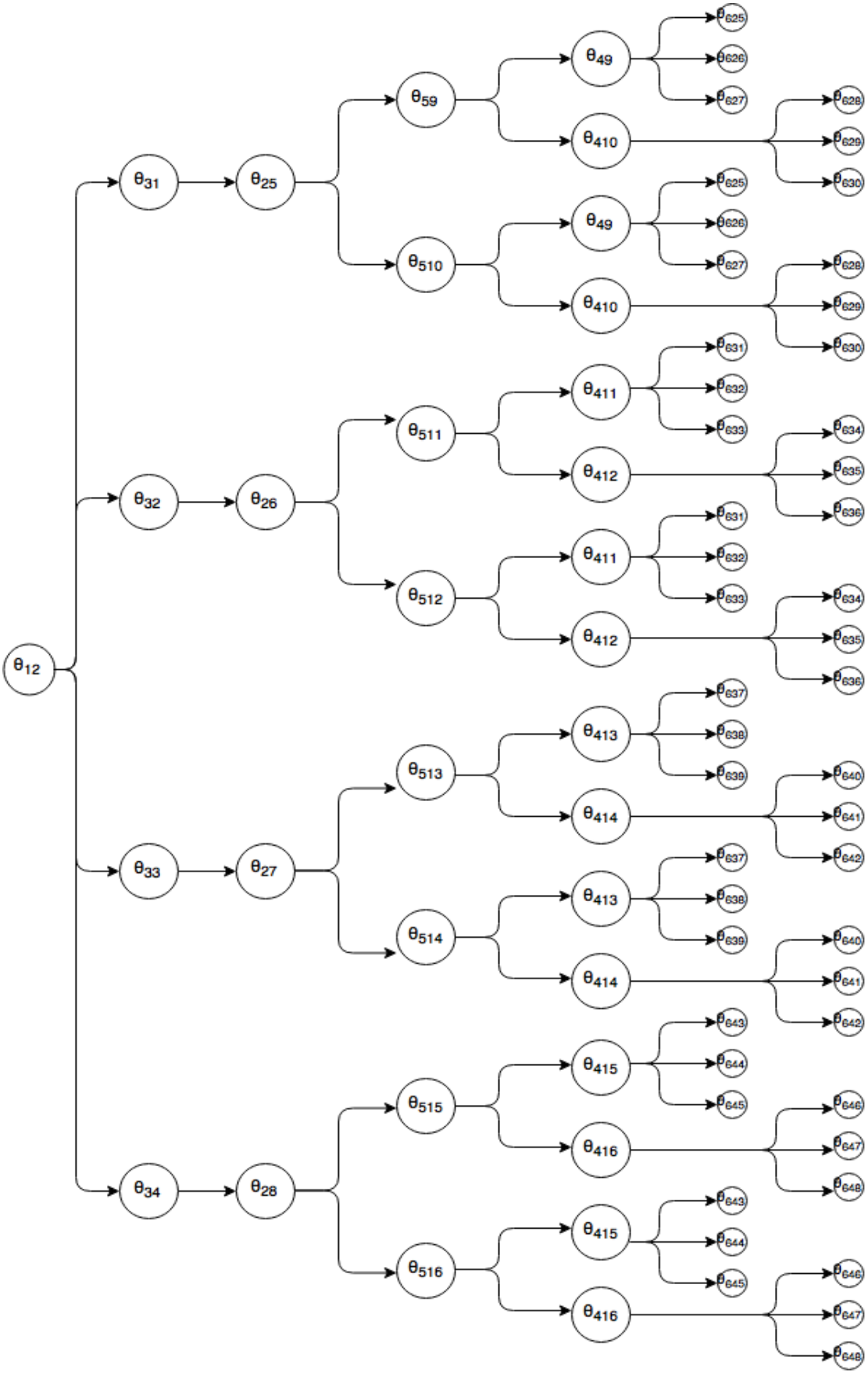
$$\theta_{62} = 180 - Asin(-(s_1c_4 + c_1s_4c_{23})r_{11} + (c_1c_4 - s_1s_4c_{23})r_{21} + s_4s_{23}r_{31})$$

$$\theta_{63} = -180 - Asin(-(s_1c_4 + c_1s_4c_{23})r_{11} + (c_1c_4 - s_1s_4c_{23})r_{21} + s_4s_{23}r_{31})$$

4.3.2.3 Επιλογή μοναδικής λύσης

Η πολλαπλότητα των λύσεων της αντίστροφης κινηματικής είναι εμφανής μετά την ανάλυση της προηγούμενης ενότητας. Το σύνολο των λύσεων που προέκυψαν παρουσιάζεται σχηματικά σε μορφή δέντρου.


θ11
θ31
θ32
θ33
θ34
θ21
θ22
θ23
θ24
θ51
θ52
θ53
θ54
θ55
θ56
θ57
θ58
θ41
θ42
θ43
θ44
θ45
θ46
θ47
θ48
θ61
θ62
θ63
θ64
θ65
θ66
θ67
θ68
θ69
θ610
θ611
θ612
θ613
θ614
θ615
θ616
θ617
θ618
θ619
θ620
θ621
θ622
θ623
θ624



Εικόνα 26 - Δέντρο λύσεων αντίστροφης κινηματικής

Από αυτούς τους 96 συνδυασμούς που προκύπτουν, ορισμένοι πρέπει να απορριφτούν επειδή οι τιμές των γωνιών που προέκυψαν υπερβαίνουν τα όρια των αρθρώσεων του πραγματικού ρομπότ. Από τους υπόλοιπους που μένουν πρέπει να επιλεγεί ένας. Εφόσον δεν υπάρχει άλλος περιορισμός, μοναδικό κριτήριο είναι η ελαχιστοποίηση της συνολικής κίνησης, δηλαδή επιθυμούμε την ελάχιστη συνολική μεταβολή των αρθρώσεων. [20]

$$\sum_{i=1}^{6} |\delta\theta_i| = min$$

5.4 Προγραμματισμός κινηματικής στο Unity

Όλη η προηγούμενη διαδικασία πρέπει να προγραμματιστεί στο Unity προκειμένου το εικονικό ρομπότ να κινείται σύμφωνα με τις εντολές του χρήστη. Αυτό που ενδιαφέρει κυρίως είναι η αντίστροφη κινηματική, αφού ο σκοπός είναι ο χρήστης να δίνει ως είσοδο στο σύστημα την θέση του καρπού και τον προσανατολισμό του τελικού σημείου δράσης και να προκύπτουν οι γωνίες των αρθρώσεων μέσω των υπολογισμών της αντίστροφης κινηματικής.

Η ευθεία κινηματική ανάλυση δεν χρειάζεται να προγραμματιστεί σε κώδικα στην προκειμένη περίπτωση για δύο λόγους:

- Στο περιβάλλον του Unity, το μοντέλο του ρομπότ τοποθετήθηκε στην θέση μηδέν του αδρανειακού συστήματος συντεταγμένων. Το σύστημα γνωρίζει κάθε στιγμή τη θέση κάθε οντότητας που βρίσκεται στη σκηνή και μπορούμε να την διαβάσουμε μέσω του transform component. Εφόσον η θέση μηδέν του συστήματος συμπίπτει με το σύστημα βάσης του ρομπότ, η θέση του καρπού που διαβάζουμε στο Unity αντιστοιχεί πλήρως στην πραγματική.
- Την ευθεία κινηματική την χρησιμοποιούμε μόνο για την εκτύπωση των μεταβλητών θέσης στην οθόνη.

Ήταν απαραίτητη όμως για την λύση του αντίστροφου κινηματικού προβλήματος.

Στη συνέχεια θα γίνει περιγραφή του κώδικα που γράφτηκε για τον έλεγχο του ρομπότ, κάνοντας αναφορά σε ορισμένες κλήσεις κλάσεων, συναρτήσεων και μεταβλητών που διαθέτει το σύστημα. Επειδή εκτενής αναφορά σε κάθε μία από

αυτές δεν κρίνεται σκόπιμη, ο αναγνώστης καλείται να ανατρέξει στο documentation του Unity για την καλύτερη κατανόησή τους.

Όπως είχαμε αναφέρει, έγινε τροποποίηση της ιεραρχίας των GameObjects έτσι ώστε όλες οι αρθρώσεις να περιστρέφονται γύρω από τον τοπικό άξονα Z. Αυτό προτιμήθηκε γιατί και στην μεθοδολογία Denavit – Hartenberg ο άξονας περιστροφής της κάθε άρθρωση είναι ο τοπικός άξονας Z. Συνεπώς, όποιες τιμές γωνιών προκύψουν από την αντίστροφη κινηματική, θα χρησιμοποιηθούν για την αλλαγή του rotation z στο transform component κάθε άρθρωσης, αφήνοντας τις περιστροφές ως προς τους άλλους άξονες ως έχουν. Με αυτό τον τρόπο διατηρείται η κινηματική αλυσίδα.

Σαν input στο σύστημα δίνουμε 6 τιμές, τις p_{wx}, p_{wy}, p_{wz}, a_x, a_y και a_z. Οι πρώτες τρείς είναι η μετατόπιση του καρπού κατά τους τρείς άξονες και οι υπόλοιπες είναι οι περιστροφές του τελικού σημείου δράσης γύρω από τους τρείς άξονες. Οι γωνίες όπως δίνονται στο σύστημα είναι σε περιγραφή Euler (βλ. Παράρτημα Α).

Για τον υπολογισμό των γωνιών των 3 πρώτων αρθρώσεων, ο τρόπος που δίνεται η μετατόπιση αρκεί, αφού είπαμε ότι οι 3 πρώτες αρθρώσεις είναι ανεξάρτητες του προσανατολισμού του τελικού σημείου δράσης. Για τον υπολογισμό όμως των 3 τελευταίων χρειαζόμαστε τον πίνακα περιστροφής. Η περιστροφή γύρω από κάθε άξονα μπορεί να γραφτεί σε μορφή πίνακα ως εξής:

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(a_x) & -\sin(a_x) \\ 0 & \sin(a_x) & \cos(a_x) \end{bmatrix}$$

$$R_y = \begin{bmatrix} \cos(a_y) & 0 & \sin(a_y) \\ 0 & 1 & 0 \\ -\sin(a_y) & 0 & \cos(a_y) \end{bmatrix}$$

$$R_z = \begin{bmatrix} \cos(a_z) & -\sin(a_z) & 0 \\ \sin(a_z) & \cos(a_z) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Και ο πίνακας περιστροφής προκύπτει κάνοντας τον πολλαπλασιασμό:

$$R = R_z \cdot R_y \cdot R_x$$

Το script της αντίστροφης κινηματικής έχει ονομαστεί IKMover.cs και έχει επισυναφθεί ως component στο GameObject "Staubli RX90L". Όλα τα scripts που θα αναφερθούν μέσα στην εργασία βρίσκονται πλήρη στο Παράρτημα Β. Ο κώδικας υπολογισμού της αντίστροφης κινηματικής έχει γραφτεί μέσα στην συνάρτηση Update() γιατί θέλουμε να τρέχει συνεχώς σε κάθε frame και να ανανεώνει τις τιμές των γωνιών.

Αρχικά γίνεται η εισαγωγή της μετατόπισης και της περιστροφής από τον χρήστη. Ο τρόπος εισαγωγής θα αναλυθεί σε επόμενο κεφάλαιο. Στη συνέχεια σχηματίζεται ο πίνακας περιστροφής, ακολουθώντας προαναφερθείσα διαδικασία. Το Unity διαθέτει την κλάση Mathf(), η οποία περιέχει όλες τις γνωστές μαθηματικές συναρτήσεις, συμπεριλαμβανομένων και των τριγωνομετρικών.

Προσοχή πρέπει να δοθεί στο γεγονός ότι οι τριγωνομετρικές συναρτήσεις απαιτούν την γωνία σε rad. Η μετατροπή των γωνιών από μοίρες σε rad γίνεται πολλαπλασιάζοντας την γωνία με την μεταβλητή Mathf.Deg2Rad.

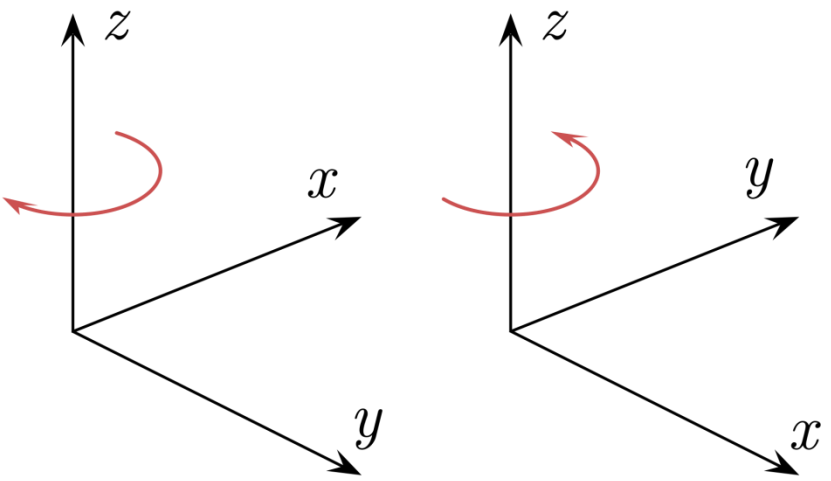
Ακολουθεί ο υπολογισμός όλων των πιθανών λύσεων. Μετά την απόρριψη όσων δεν είναι μέσα στα όρια των αρθρώσεων του πραγματικού ρομπότ, υπολογίζονται όλα τα αθροίσματα της συνολικής μετακίνησης και επιλέγεται το μικρότερο και συνεπώς οι γωνίες στις οποίες αυτό αντιστοιχεί.

Με τις τιμές των αρθρώσεων γνωστές, πρέπει να γίνει η μετακίνηση του μοντέλου στη σκηνή. Στο transform component οι τιμές των γωνιών εμφανίζονται σε περιγραφή Euler προκειμένου να είναι ευκολότερα αντιληπτές, αλλά το Unity λειτουργεί με Quaternions για την μείωση των ιδιομορφιών (βλ. Παράρτημα Α). Προκειμένου να δημιουργήσουμε το Quaternion κάθε γωνίας χρησιμοποιούμε την εξής κλάση:

```
jRot = Quaternion.Euler(rotX, rotY, initRotZ-(rotZ-initRotZ));
```

Η κλάση Quaternion περιέχει την συνάρτηση Euler η οποία μετασχηματίζει τις γωνίες Euler σε περιγραφή Quaternion. Αυτή δέχεται σαν ορίσματα τις περιστροφές γύρω από τους 3 άξονες. Οι περιστροφές γύρω από του άξονες x και y δεν αλλάζουν άρα

παραμένουν ως έχουν. Η περιστροφή γύρω από τον άξονα z έχει γραφτεί με τέτοιο τρόπο ώστε το σύστημα από δεξιόστροφο να μετασχηματιστεί σε αριστερόστροφο και να συμφωνεί με αυτό του Unity.



Εικόνα 27 - Αριστερά αριστερόστροφη θετική γωνία, δεξιά δεξιόστροφη θετική γωνία

Ο μετασχηματισμός γίνεται με την εξής λογική: η νέα γωνία θα ισούται με την παλιά επαυξημένη κατά την αντίθετη διαφορά της νέα από την παλιά, ή μαθηματικά:

$$Rot_{new,counterclockwise} = Rot_{old} - \left(Rot_{new,clockwise} - Rot_{old}\right) [25]$$

Το σύνολο των εντολών που υπολογίζουν το Quaternion κάθε άρθρωσης έχουν γραφτεί σε μια συνάρτηση η οποία ονομάζεται CalculateTarget(), η οποία καλείται σε κάθε frame μετά τον υπολογισμό των καινούριων γωνιών.

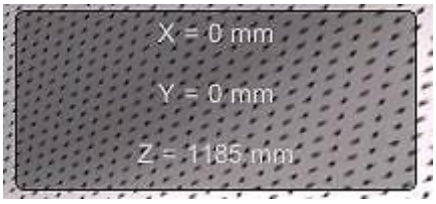
Αφού δημιουργήθηκε το Quaternion της κάθε γωνίας, χρησιμοποιούμε την συνάρτηση Lerp της κλάσης Quaternion για την πραγματοποίηση της μετακίνησης. Το όνομα Lerp προκύπτει από το Linear Interpolation ή στα ελληνικά γραμμική παρεμβολή. Η συνάρτηση αυτή έχει την σύνταξη:

joint.localRotation = Quaternion.Lerp (Quaternion.Euler (joint.localEulerAngles), jRot, rotationSpeed*Time.deltaTime);

και λειτουργεί ως εξής: μετακινεί σταδιακά την άρθρωση από την αρχική της περιστροφή, στην περιστροφή – στόχο με ταχύτητα rotationSpeed. Η μεταβλητή Time.deltaTime μετασχηματίζει την ταχύτητα από speed/frame σε speed/second για να

είναι η κίνηση ομαλή. Το σύνολο των εντολών αυτών για όλες τις αρθρώσεις έχει γραφτεί μέσα στη συνάρτηση MoveToTarget() η οποία καλείται σε κάθε frame μετά την εκτέλεση της CalculateTarget().

Τέλος, χρησιμοποιούμε ένα άλλο script το οποίο ονομάζεται DisplayEndEffectorCoordinates.cs για την εκτύπωση των συντεταγμένων του ΤΔΣ ως προς τη βάση του ρομπότ. Το αρχείο αυτό περιέχει την συνάρτηση OnGUI() η οποία τρέχει σε κάθε frame και εκτυπώνει γραφικά στοιχεία στην οθόνη. Στην προκειμένη περίπτωση δημιουργήθηκε ένα ορθογώνιο παραλληλόγραμμο το οποίο τοποθετήθηκε στην πάνω δεξιά πλευρά της οθόνης, μέσα στο οποίο αναγράφονται οι συντεταγμένες ως προς τους 3 άξονες.



Εικόνα 28 - Παράθυρο εμφάνισης συντεταγμένων ΤΣΔ

Η γενική σύνταξη της εντολής που δημιουργεί το παράθυρο είναι η ακόλουθη:

GUI.Box (new Rect (x, y, width, height), 'Printed Text')

Οι αριθμοί x και y αναφέρονται στις συντεταγμένες της πάνω αριστερής γωνίας του τετραγώνου (σε pixels), με το (0,0) να βρίσκεται στην πάνω αριστερή γωνία της οθόνης. Οι μεταβλητές width και height συμβολίζουν προφανώς το πλάτος και το ύψος του τετραγώνου. Στο πεδίο του κειμένου που εκτυπώνεται τοποθετούνται οι συντεταγμένες του τελικού σημείου δράσης. Αυτές όμως βρίσκονται στο script IKMover.cs. Για να μπορέσει ένα script να αποκτήσει πρόσβαση και να χρησιμοποιήσει μεταβλητές που βρίσκονται σε κάποιο άλλο, δημιουργούμε ένα **αντικείμενο** της κλάσης της οποίας οι μεταβλητές και συναρτήσεις μας ενδιαφέρουν. Αυτό γίνεται με την εντολή:

private IKMover robotMoverInstance;

λέμε δηλαδή στον κώδικα, να φτιάξει μια μεταβλητή τύπου IKMover, η οποία ονομάζεται robotMoverInstance. Στη συνέχεια, προκειμένου η μεταβλητή αυτή να αποκτήσει τις ιδιότητες του αρχείου IKMover γράφουμε:

robotMoverInstance = GameObject.Find("Staubli RX90L").GetComponent<IKMover>();

δηλαδή ψάχνει το GameObject με όνομα "Staubli RX90L" και παίρνει τις ιδιότητες του script, συμπεριλαμβανομένων των συναρτήσεων και των μεταβλητών που περιέχει.

Αφού πλέον ο τρέχων κώδικας έχει πρόσβαση στις μεταβλητές που επιθυμούμε, μένει να γίνει σωστή εκτύπωση των συντεταγμένων. Για να ταιριάζουν οι συντεταγμένες που διαβάζουμε από το transform component με τις πραγματικές του ρομπότ, εφαρμόζουμε τον μετασχηματισμό: Real(x) = - Unity(x), Real(y) = - Unity(z), Real(z) = Unity(y). Με αυτόν τον τρόπο μετασχηματίζουμε το αριστερόστροφο σύστημα συντεταγμένων σε δεξιόστροφο.

6 Προγραμματισμός Τροχιάς

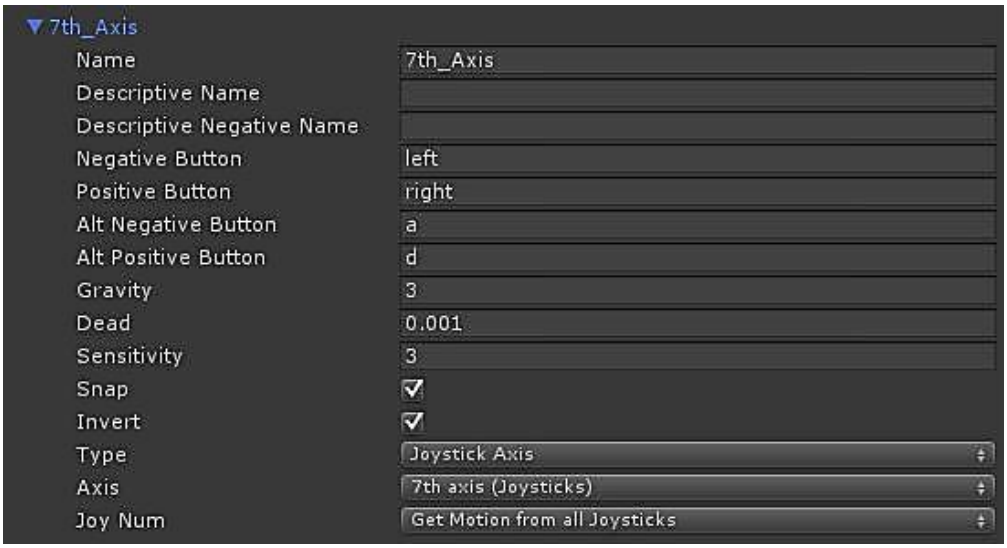
6.1 Το χειριστήριο

Όπως αναφέρθηκε και στην προηγούμενη ενότητα, το σύστημα δέχεται σαν είσοδο 6 μεταβλητές, τρείς για την περιγραφή της θέσης του καρπού και τρείς για την περιγραφή του προσανατολισμού του ΤΣΔ. Οι τιμές των μεταβλητών αυτών δεν δίνονται σε μορφή απόλυτων αριθμών, αλλά το ρομπότ βρίσκεται σε κάποια αρχική θέση και οι μεταβλητές αλλάζουν κατά κάποια τιμή ανά frame.

Το χειριστήριο που επιλέχθηκε για τον έλεγχο του εικονικού ρομπότ, είναι αυτό της παιχνιδομηχανής Xbox 360 της Microsoft.

Εικόνα 29 - Xbox Controller

Το χειριστήριο αποτελείται από έναν αριθμό κουμπιών, τα οποία είναι δύο ειδών. Το ένα είδος είναι το καθαρά ψηφιακό, το οποίο επιστρέφει 0 όταν δεν είναι πατημένο και 1 όταν είναι πατημένο και στο Unity αναγνωρίζεται ως JoystickButton i, όπου i = 1:n. Το δεύτερο είδος είναι το αναλογικό, το οποίο επιστρέφει συνεχείς τιμές από 0 έως 1 ανάλογα με το πόσο πατιέται και αναγνωρίζεται από το Unity ως άξονας (Axis). Ο κωδικός αριθμός κάθε κουμπιού και άξονα, όπως αυτά αναγνωρίζονται από το Unity, φαίνεται στην εικόνα 29. Για τα κουμπιά ο προγραμματιστής δεν χρειάζεται να κάνει κάτι, επειδή οι κωδικοί τους είναι προκαθορισμένοι στο Unity. Για να αναγνωριστούν οι άξονες όμως, πρέπει να τους δοθεί κάποιο όνομα. Αυτό γίνεται από τον Input Manager του λογισμικού, ο οποίος είναι διαθέσιμος από το μενού Edit → Project Setting → Input.



Εικόνα 30 - Input Manager

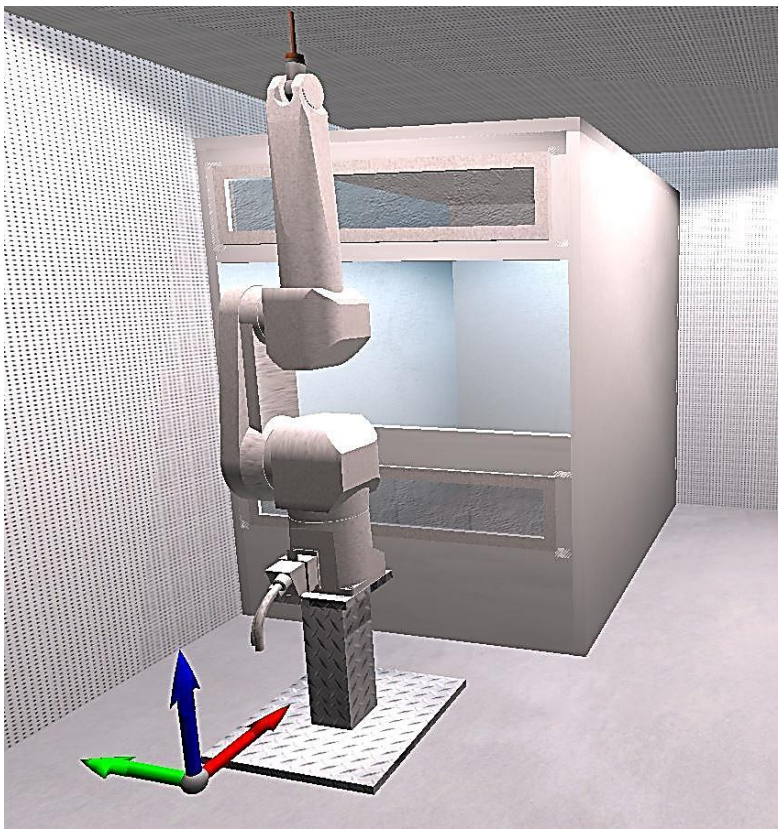
Ο Input Manager εμφανίζεται στην δεξιά πλευρά της οθόνης, στη θέση του Inspector και έχει περιεχόμενο της μορφής της εικόνας 30. Ένα project στην αρχή περιέχει μόνο 2 άξονες, τον Vertical και τον Horizontal οι οποίοι αντιστοιχούν στα βελάκια του πληκτρολογίου ή στους αντίστοιχους άξονες οποιοδήποτε χειριστηρίου. Στην περίπτωση όμως που διαθέτουμε περισσότερους άξονες και θέλουμε να τους αξιοποιήσουμε, πρέπει να τους προσθέσουμε στον Input Manager, να τους δώσουμε ένα όνομα για να τους προσπελαύνουμε από τον κώδικα και να ορίσουμε ποιοί θα είναι αυτοί χρησιμοποιώντας το dropdown menu Axis.

Στην παρούσα εργασία χρησιμοποιήθηκαν όλα τα κουμπιά και οι περισσότεροι από τους διαθέσιμους άξονες. Η δουλειά την οποία επιτελεί το καθένα καθώς και ο τρόπος με τον οποίο γίνεται αυτή θα αναλυθούν στην επόμενη ενότητα.

6.2 Εισαγωγή των μεταβλητών κίνησης

Η εφαρμογή ξεκινάει με το ρομπότ να βρίσκεται στη θέση Do Ready. Στην θέση αυτή οι τιμές των 6 μεταβλητών είναι:

- $p_{wx} = 0$
- $p_{wy} = 0$
- $p_{wz} = 1.1$
- $a_x = 0$
- $a_y = 0$
- $a_z = 0$



Εικόνα 31 - Αρχική θέση του ρομπότ κατά την έναρξη της εφαρμογής

Για την μετακίνηση του καρπού χρησιμοποιήθηκαν 3 άξονες. Οι δύο από αυτούς είναι ο Horizontal και ο Vertical, που υπάρχουν από την δημιουργία του project, ενώ για την κάλυψη της μετατόπισης κατά την τρίτη κατεύθυνση δημιουργήθηκε ο "7^{th}_Axis" στον

Input Manager. Η τιμή που επιστρέφει κάθε άξονας αποθηκεύεται σε μια μεταβλητή, χρησιμοποιώντας την κλάση Input του Unity με την παρακάτω σύνταξη π.χ. για τον άξονα z:

zAxis = Input.GetAxis ("7th_Axis") * ikSpeed;

Η τιμή αυτής της μεταβλητής θα χρησιμοποιηθεί για να αυξομειώνει την τιμή της παραμέτρου p_{wz}. Επειδή αυτή η εντολή βρίσκεται εντός της συνάρτησης Update() και τρέχει σε κάθε frame, χρησιμοποιήθηκε η μεταβλητή ikSpeed ως παράγοντας μείωσης της τιμής εισόδου, προκειμένου να μειωθεί η ταχύτητα μετακίνησης του καρπού σε διάστημα ενός δευτερολέπτου και να είναι το ρομπότ ελέγξιμο από τον χρήστη.

Όπως προαναφέρθηκε και στο κεφάλαιο 4, το ρομπότ έχει έναν συγκεκριμένο χώρο επιδεξιότητας στον οποίο μπορεί να φτάσει με οποιονδήποτε προσανατολισμό. Πρέπει να υπάρχει λοιπόν μια συνθήκη που να μην επιτρέπει την περαιτέρω αύξηση ή μείωση κάθε συντεταγμένης, όταν το ρομπότ έχει βγει από τον χώρο επιδεξιότητας. Η μαθηματική εύρεση του χώρου εργασίας είναι επίπονη διαδικασία οπότε εφαρμόστηκε μια εναλλακτική μέθοδος.

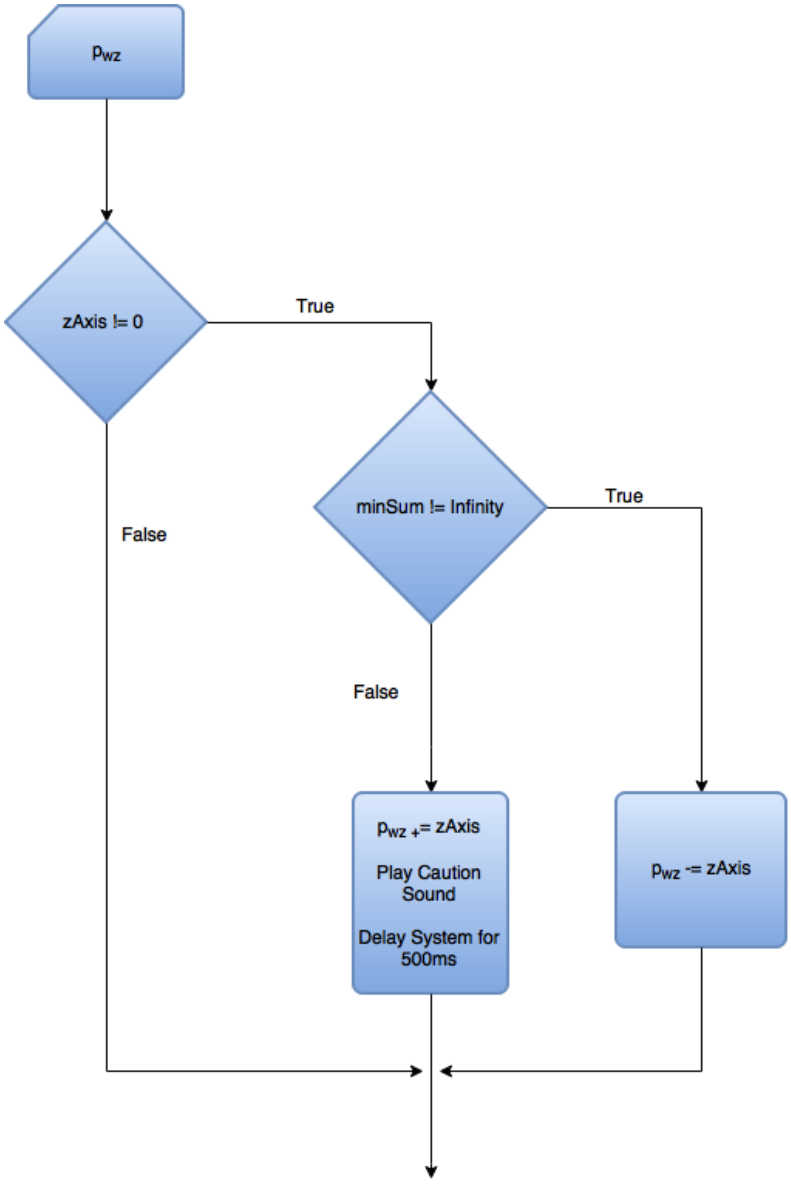
Μοναδικό κριτήριο για την επιλογή μιας μοναδικής λύσης εκ των 96 που προκύπτουν από την αντίστροφη κινηματική, είναι η ελαχιστοποίηση της συνολικής μεταβολής των αρθρώσεων. Στην αρχή της συνάρτησης Update() αρχικοποιείται η μεταβλητή:

minSum = ∞

Από τους 96 συνδυασμούς που προκύπτουν, το άθροισμα κάθε συνδυασμού υπολογίζεται μόνο στην περίπτωση που και οι 6 αρθρώσεις είναι εντός ορίων, ειδάλλως τίθεται αυθαίρετα άπειρο. Αν κατά την αύξηση ή μείωση κάποια συντεταγμένης ο καρπός φτάσει σε σημείο εκτός του χώρου επιδεξιότητας, τότε κάποια ή κάποιες τιμές αρθρώσεων που θα προκύψουν από την αντίστροφη κινηματική θα είναι εκτός ορίων. Συνεπώς όλα τα αθροίσματα θα έχουν την τιμή άπειρο, άρα και το minSum θα ισούται με άπειρο. Σε αυτή την περίπτωση λοιπόν, σταματάμε την μεταβολή της εκάστοτε συντεταγμένης και μετακινούμε τον καρπό προς την αντίθετη κατεύθυνση κατά το εκάστοτε input για να βρισκόμαστε πάντα εντός του χώρου επιδεξιότητας.

Επιπλέον για την προειδοποίηση του χρήστη, σε αυτή την περίπτωση αναπαράγεται ένας ήχος και η εκτέλεση του κώδικα σταματά για 500ms για να προλάβει ο χρήστης

να αφήσει το πλήκτρο που πατά. Ο ήχος εισάγεται στο project με τη διαδικασία που έχει αναφερθεί στο κεφάλαιο 4.1, επισυνάπτεται σε κάποιο GameObject και έτσι το τελευταίο αποκτά το Audio Component. Στον κώδικα υπολογισμού της αντίστροφης κινηματικής φτιάχνουμε ένα αντικείμενο του Audio Component το οποίο είναι τύπου AudioSource με όνομα outOfDexterousSpace. Η κλάση AudioSource περιέχει την συνάρτηση Play, όποτε γράφοντας outOfDexterousSpace.Play() αναπαράγεται ο ήχος. Όσον αφορά την παύση εκτέλεσης του κώδικα, υλοποιείται με την εντολή Thread.Sleep(500). Η προηγούμενη διαδικασία για την κατεύθυνση z φαίνεται συνοπτικά στο επόμενο διάγραμμα ροής. Για τις υπόλοιπες κατευθύνσεις υπάρχει αντίστοιχος κώδικας.

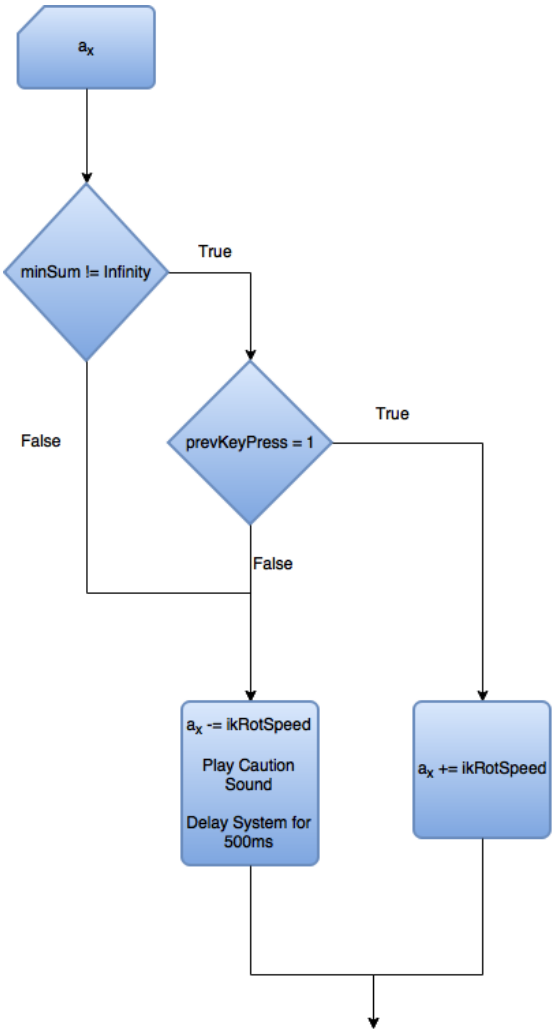


Εικόνα 32 - Διάγραμμα ροής εισαγωγής μετατοπίσεων

Οι παράγοντες περιστροφής του καρπού δίνονται με τη χρήση 3 κουμπιών. Επιλέχθηκε σκόπιμα η περιστροφή γύρω από τον x άξονα να γίνεται με το κουμπί B που έχει κόκκινο χρώμα, η περιστροφή γύρω από τον άξονα y να γίνεται με το κουμπί A που έχει πράσινο χρώμα και η περιστροφή γύρω από τον άξονα z να γίνεται με το κουμπί X που έχει μπλε χρώμα. Με αυτό τον τρόπο ο χρωματισμός συμφωνεί με αυτόν που χρησιμοποιεί το Unity. Αν θέλουμε να περιστρέψουμε προς την αρνητική φορά των γωνιών, τότε πατάμε το Left Bumper και το αντίστοιχο κουμπί περιστροφής. Η εντολή που διαβάζει το πάτημα ενός κουμπιού είναι η:

Input.GetKey (KeyCode.JoystickButtoni)

όπου i = 1,2,3,…,n ο κωδικός αριθμός κάθε κουμπιού. Επειδή τα κουμπιά είναι ψηφιακά και επιστρέφουν είτε 1 είτε 0, χρησιμοποιήθηκε η μεταβλητή ikRotSpeed σαν ταχύτητα περιστροφής.



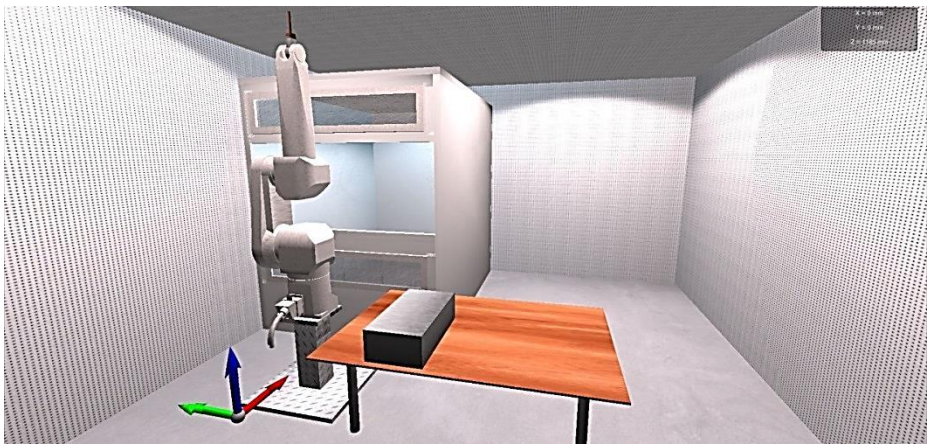
Εικόνα 33 - Διάγραμμα ροής εισαγωγής γωνιών

Με την ίδια λογική όπως και στις μετατοπίσεις, όταν το ΤΣΔ βγει από τον χώρο επιδεξιότητας σταματάμε την αύξηση ή μείωση της γωνίας και αναπαράγεται ο προειδοποιητικός ήχος. Στην περίπτωση των γωνιών όμως έγινε μια μικρή αλλαγή. Κάθε περιστροφή κωδικοποιήθηκε με έναν ακέραιο αριθμό, ο οποίος αποθηκεύτηκε στην μεταβλητή prevKeyPress. Έτσι προέκυψαν 6 κωδικοί, δύο για κάθε άξονα (θετική και αρνητική περιστροφή). Η συνθήκη που σταματάει τη μεταβολή μιας γωνίας είναι όταν το minSum ισούται με άπειρο και ταυτόχρονα το τρέχον πατημένο πλήκτρο είναι αυτό που πατιόταν και στο προηγούμενο frame. Αν ο έλεγχος γινόταν μόνο στο minSum, ακόμη και να αλλάζαμε το κουμπί που πατάγαμε για να κινηθούμε προς άλλη κατεύθυνση, όταν το ρομπότ είχε βγει από τον χώρο επιδεξιότητας, ο κώδικας δεν θα επέτρεπε την μεταβολή οποιασδήποτε γωνίας, με αποτέλεσμα το πρόγραμμα να «παγώσει». Το αντίστοιχο διάγραμμα ροής για την περίπτωση της θετικής περιστροφής γύρω από τον άξονα x φαίνεται στην εικόνα 33.

6.3 Εφαρμογές προγραμματισμού τροχιάς

6.3.1 Ευθεία συγκόλληση 3 διευθύνσεων

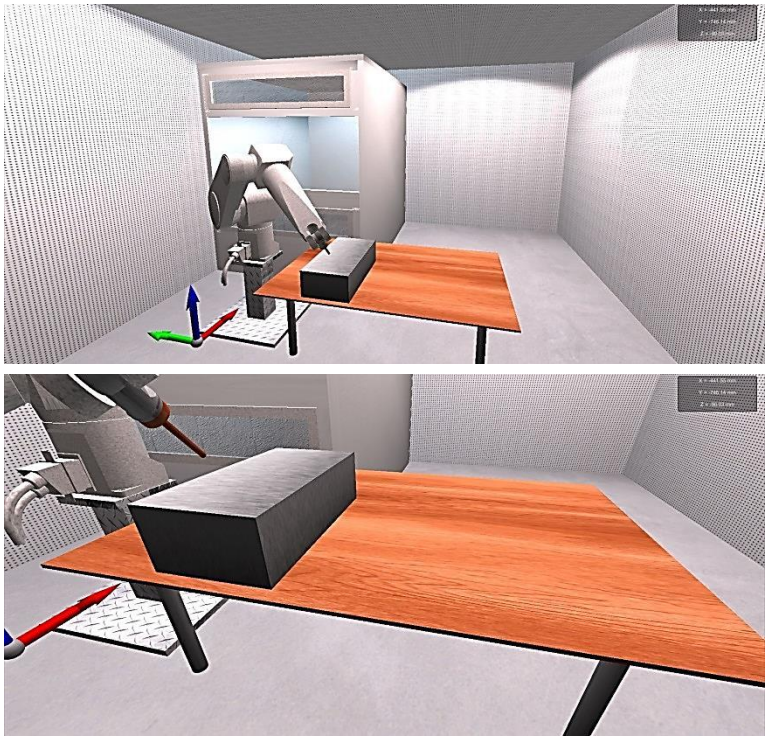
Η συγκόλληση είναι πολύ συχνή εφαρμογή των βιομηχανικών ρομπότ και συναντάται έντονα στην αυτοκινητοβιομηχανία και την αεροναυπηγική. Κύριος παράγοντας της επιτυχούς συγκόλλησης είναι η γωνία υπό την οποία γίνεται. Συνήθως επιλέγεται η γωνία του ηλεκτροδίου ως προς την ακμή συγκόλλησης να είναι 45°. Το σενάριο που υλοποιήθηκε στην παρούσα εργασία είναι η συγκόλληση των 4 ακμών ενός μεταλλικού κουτιού.



Εικόνα 34 - Εφαρμογή συγκόλλησης ακμών μεταλλικού κουτιού

Στην σκηνή της εικόνας 34, το ύψος του τραπεζιού και η θέση του κουτιού σε σχέση με το ρομπότ είναι γνωστές. Χρησιμοποιώντας το χειριστήριο του Xbox πρέπει να οδηγήσουμε το ΤΣΔ δράσης υπό γωνία 45° ως προς κάθε πλευρά, για να προσομοιώσει συγκόλληση.

Η εικόνα που βλέπουμε είναι αυτή που δείχνει η κύρια κάμερα (Main Camera) της σκηνής, η οποία χρησιμοποιείται για να έχουμε εποπτεία όλου του χώρου, αλλά δεν μας επιτρέπει να παρακολουθούμε την λεπτομέρεια της κίνησης στις 3 διαστάσεις. Για τον λόγο αυτό προστέθηκε στην ιεραρχία ακόμη μία κάμερα που ονομάστηκε Close Camera. Το transform component της κάμερας αυτής μεταβάλλεται σε κάθε frame, έτσι ώστε να βρίσκεται πάντα σε συγκεκριμένη απόσταση κατά τους 3 άξονες από το τελικό σημείο δράσης ίση με: Δx = 0.5m, Δy = 0.5m, Δz = 0.1m. Ο κώδικας που ελέγχει την κίνηση της δεύτερης κάμερας βρίσκεται στο αρχείο SecondCameraFollow.cs το οποίο έχει επισυναφθεί στην Close Camera.

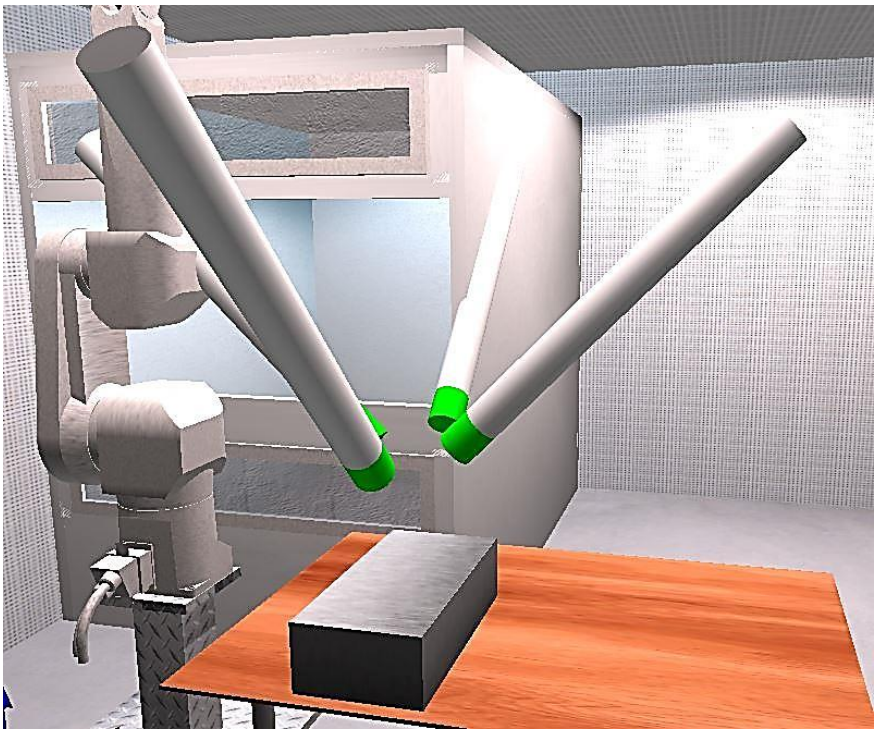


Εικόνα 35 - Σύγχριση Main Camera - Close Camera View

Κάθε φορά όμως στην οθόνη μας μπορούμε να βλέπουμε την εικόνα μόνο μίας εκ των δύο καμερών. Η κάμερα που είναι ενεργή κάθε φορά είναι αυτή που έχει το μεγαλύτερο βάθος (depth) στο Camera Component. Χρειάστηκε λοιπόν η δημιουργία του ChangeView.cs του οποίου η μοναδική δουλειά είναι να αλλάζει το βάθος της δεύτερης κάμερας. Η αλλαγή γίνεται πατώντας το κουμπί No 9 στο χειριστήριο. Η Close Camera έχει την τιμή -2 όταν είναι ανενεργή και την τιμή 1 όταν είναι ενεργή.

Η προσθήκη της δεύτερης κάμερας βελτίωσε μεν την ακρίβεια με την οποία μπορούμε να παρακολουθήσουμε την κίνηση, μοναδικός όμως παράγοντας για την επίτευξη της γωνίας 45° παραμένει η όραση του χρήστη. Αντί για αυτό θα θέλαμε να έχουμε μαθηματική ακρίβεια. Για να επιτευχθεί αυτό, τοποθετήθηκαν μέσα στη σκηνή

«κρυφά» GameObjects με προσανατολισμό 45° ανά πλευρά του κουτιού. Με τον όρο «κρυφά» εννοείται ότι υπάρχουν στη σκηνή, αλλά έχουμε επιλέξει να μην μπορεί να τα δει η κάμερα. Αυτό γίνεται απενεργοποιώντας τον Mesh Renderer από τον Inspector. Τα GameObjects αυτά είναι κύλινδροι και έχουν όνομα Direction1, Direction2, Direction3 και Direction4 και αντιστοιχούν στις 4 πλευρές του κουτιού.



Εικόνα 36 - Οι 4 επιθυμητοί προσανατολισμοί συγκόλλησης ορατοί στη σκηνή

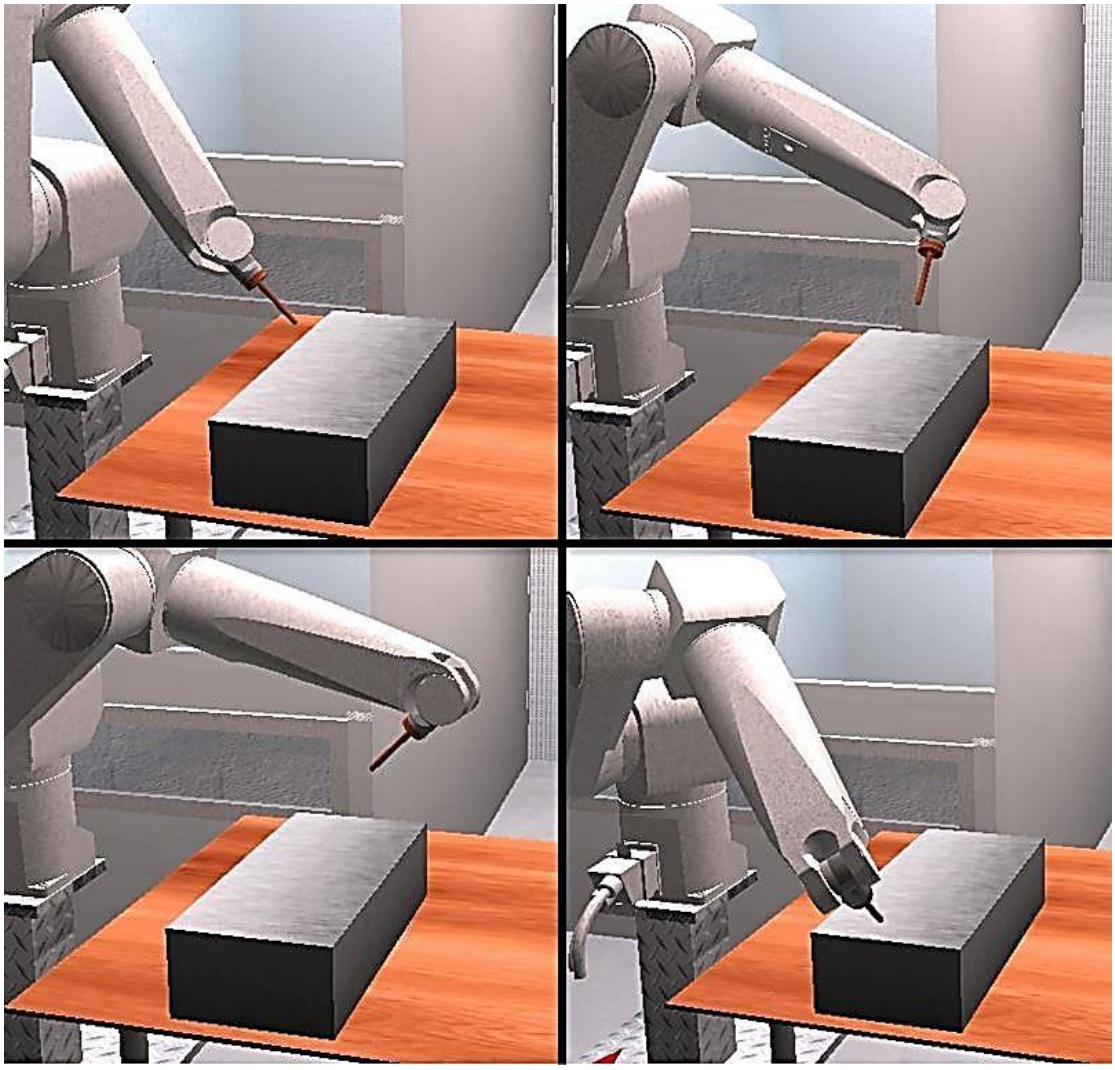
Όπως έχει αναφερθεί και σε προηγούμενο κεφάλαιο, το Unity περιγράφει τον προσανατολισμό ενός αντικειμένου με γωνίες Euler, σε μορφή δηλαδή κατάλληλη για την δημιουργία του πίνακα περιστροφής.

Ο κώδικας IKMover.cs τροποποιήθηκε έτσι ώστε με το πάτημα του κουμπιού Y στο χειριστήριο να εναλλάσσονται κυκλικά οι 4 επιθυμητοί προσανατολισμοί. Ένας μετρητής με όνομα directionCoding που παίρνει τιμές από 1 έως 4 αυξάνεται κατά 1 όταν πιέζεται το πλήκτρο Y. Ανάλογα με την τιμή του μετρητή οι μεταβλητές a_x, a_y και a_z παίρνουν τις τιμές των περιστροφών ανά άξονα του αντίστοιχου επιθυμητού προσανατολισμού. Για παράδειγμα αν ο μετρητής έχει την τιμή 1 εκτελείται ο κώδικάς:

```
ax = desirableRot1.rotation.eulerAngles.x;
ay = desirableRot1.rotation.eulerAngles.z;
                    az = - desirableRot1.rotation.eulerAngles.y;
```

Προφανώς η μεταβλητή desirableRot1 είναι το αντικείμενο του GameObject Direction1. Επίσης, οι εναλλαγή των περιστροφών y και z καθώς και το αρνητικό πρόσημο έγιναν για να μετατραπεί το σύστημα συντεταγμένων από αριστερόστροφο σε δεξιόστροφο και να μπορεί να το διαχειριστεί η αντίστροφη κινηματική.

Στη συνέχεια, ο ήδη υπάρχων κώδικας υπολογισμού της αντίστροφης κινηματικής υπολογίζει τις τιμές των γωνιών των αρθρώσεων. Με το πλήκτρο Back η κυκλική εναλλαγή των προσανατολισμών πραγματοποιείται αντίστροφα.



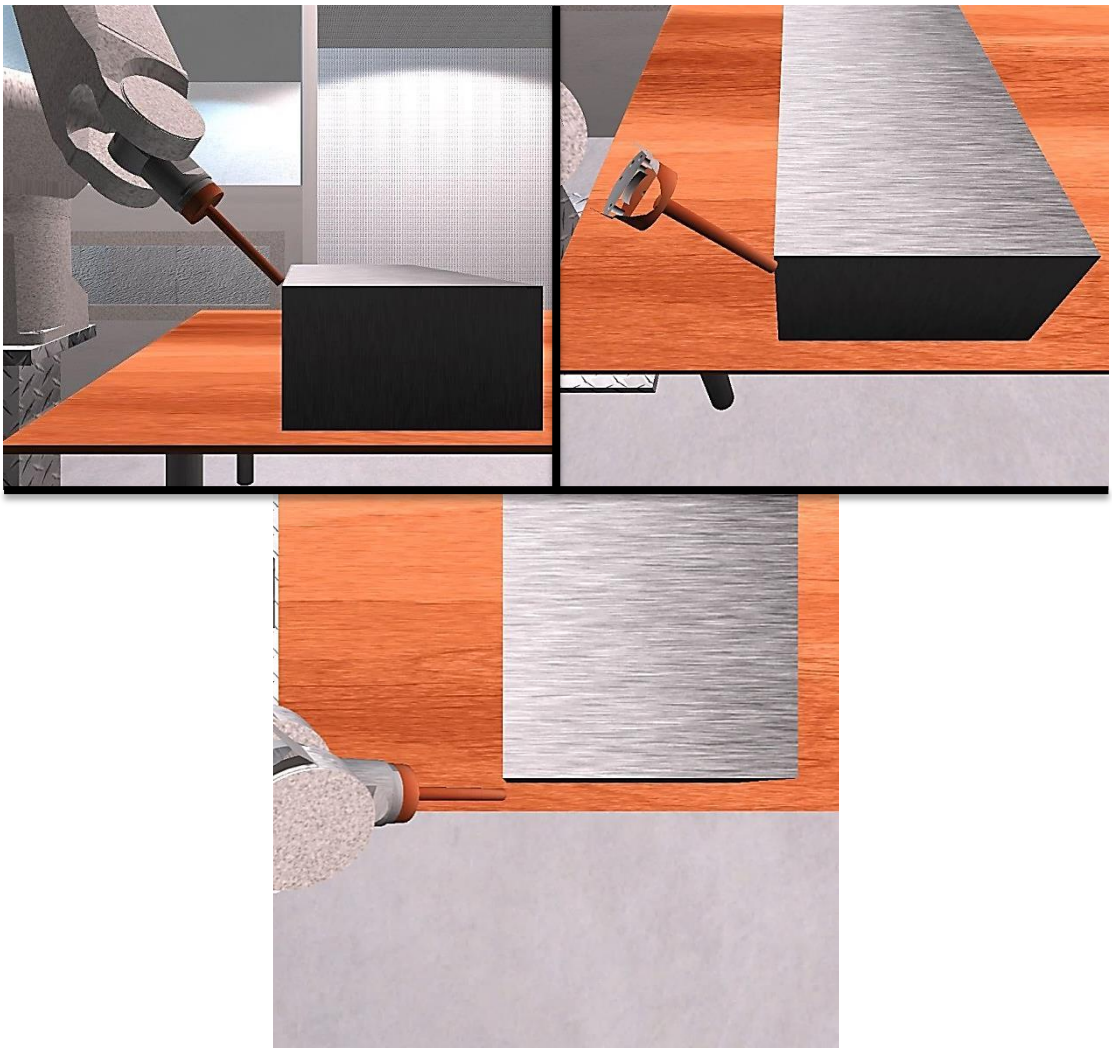
Εικόνα 37 - Οι 4 επιθυμητοί προσανατολισμοί στο ρομπότ

Η τελική τροποποίηση που έγινε αφορά τον κώδικα SecondCameraFollow.cs. Όταν ενεργοποιείται η δεύτερη κάμερα, είναι οριζόντια στο ύψος του ΤΣΔ και ο χρήστης έχει εποπτεία του ύψους στο οποίο αυτό βρίσκεται. Στη συνέχεια για να μπορεί να βλέπει τη θέση του στο επίπεδο x-y χρησιμοποιήθηκε και ο άξονας 6 του χειριστηρίου. Όπως είπαμε ένας άξονας επιστρέφει τιμές από 0 έως 1, ενώ εμείς θέλουμε η κάμερα να

περιστρέφεται γύρω από τον τοπικό άξονα x κάθε προσανατολισμού από 0° έως 90°. Η αντιστοίχιση των τιμών γίνεται με το λεγόμενο mapping. Mapping σημαίνει ότι οι τιμές ενός διαστήματος με γνωστά άκρα [x1, x2], αντιστοιχίζονται στις τιμές ενός άλλου διαστήματος με άκρα [y1, y2]. Η ερώτηση στην οποία απαντάει είναι: «Αν μετρήσουμε μια τιμή x, ποια είναι η αντίστοιχη y;» Και η απάντηση δίνεται από την σχέση:

$$y = \frac{x - x_1}{x_2 - x_1}(y_2 - y_1) + y_1$$

Άρα, ο κώδικας διαβάζει την τιμή που επιστρέφει ο άξονας 6 και την τροποποιεί ώστε να αντιστοιχεί σε μοίρες περιστροφής της δεύτερης κάμερας ως προς τον τοπικό άξονα x κάθε προσανατολισμού. Το αποτέλεσμα φαίνεται στην επόμενη εικόνα, στην οποία βλέπουμε την ίδια θέση του ΤΣΔ από 3 διαφορετικές οπτικές γωνίες.



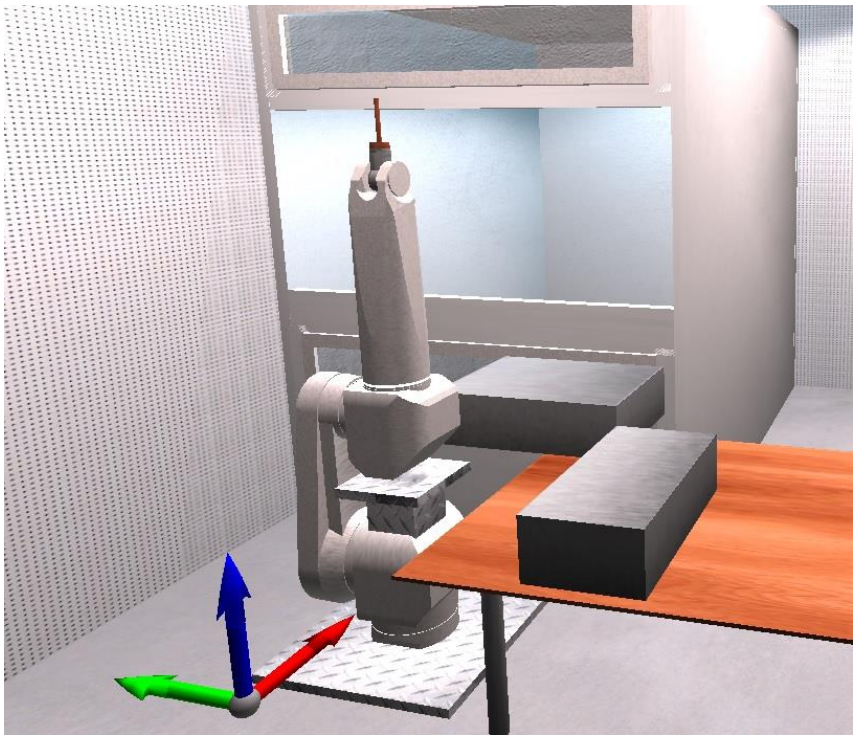
Εικόνα 38 - Διάφορες όψεις της Close Camera χρησιμοποιώντας τον άξονα 6

6.3.2 Ανίχνευση συγκρούσεων

Οι συγκρούσεις του ρομποτικού βραχίονα με τον περιβάλλοντα χώρο, μπορούν να προκαλέσουν σημαντικές ζημιές τόσο στο ίδιο το ρομπότ, όσο και στην εργασία η οποία πραγματοποιείται. Στο εικονικό περιβάλλον, αντίθετα με το πραγματικό, οι συγκρούσεις δεν έχουν καμία συνέπεια και σκοπός τους είναι η προειδοποίηση του χρήστη. Έτσι ο χειριστής που προγραμματίζει την τροχιά του βραχίονα, μπορεί να τις αποτρέψει πριν αυτές συμβούν.

Όπως έχει αναφερθεί και σε προηγούμενο κεφάλαιο, για να αναγνωριστεί μια σύγκρουση μεταξύ δύο αντικειμένων, τουλάχιστον ένα από τα δύο πρέπει να διαθέτει RigidBody Component, ενώ και τα δύο πρέπει να διαθέτουν Collider Component. Στην περίπτωσή μας, το αντικείμενο που προκαλεί την σύγκρουση είναι το ρομπότ και αυτό που δέχεται τη σύγκρουση είναι οποιοδήποτε εντάσσεται στον περιβάλλοντα χώρο του ρομπότ, συμπεριλαμβανομένης της βάσης του.

Επιλέχθηκε το RigidBody Component να το έχουν οι σύνδεσμοι του ρομπότ. Αυτό όμως είχε ως αποτέλεσμα, οι τελευταίοι να επηρεάζονται πλέον από τη βαρύτητα και να «πέφτουν» στο κενό.



Εικόνα 39 - Το ρομπότ έλκεται από τη βαρύτητα λόγω του RigidBody Component

Για την επίλυση του προβλήματος, στο RigidBody Component κάθε συνδέσμου επιλέχθηκε Is Kinematic. Η επιλογή αυτή επιτρέπει την ανίχνευση συγκρούσεων, αγνοεί όμως την επίδραση όλων των δυνάμεων.

Colliders προστέθηκαν τόσο στο ρομπότ όσο και σε όλα τα αντικείμενα του περιβάλλοντα χώρου. Ιδιαίτερα στα Collider Components των αντικειμένων του περιβάλλοντα χώρου, ενεργοποιήθηκε η επιλογή Is Trigger. Η επιλογή αυτή αναγνωρίζει την σύγκρουση μεταξύ δύο αντικειμένων, επιτρέποντας παράλληλα την αλληλοεπικάλυψη τους (overlap).

Αφού έγινε η απαιτούμενη προεργασία, τώρα πρέπει να καθορίσουμε τις ενέργειες που θα γίνονται όταν συμβαίνει μια σύγκρουση. Ένα νέο script δημιουργήθηκε με όνομα CollisionDetection.cs το οποίο περιέχει τις συναρτήσεις OnTriggerEnter(Collider other), OnTriggerStay(Collider other) και OnTriggerExit(Collider other). Ο κώδικας αυτός προκειμένου να δουλέψει πρέπει να είναι επισυνημμένος σε όλα τα αντικείμενα που έχουν Collider ο οποίος δεν είναι Trigger. Όλες οι παραπάνω συναρτήσεις δέχονται ως όρισμα την μεταβλητή τύπου **Collider** με όνομα **other**. Σε αυτή αποθηκεύεται ο μη-Trigger Collider κατά την κλήση της συνάρτησης για λόγο που θα φανεί στη συνέχεια.

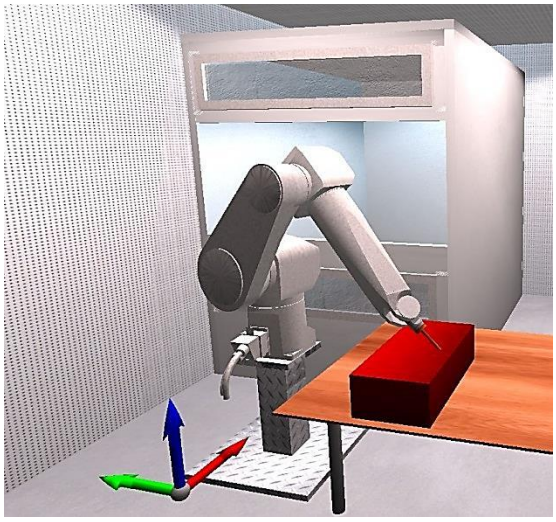
Η συνάρτηση OnTriggerEnter καλείται κάθε φορά που ένας TriggerCollider ξεκινάει να εισέρχεται σε κάποιο αντικείμενο με επισυνημμένο το CollisionDetection.cs. Εντός αυτής της συνάρτησης έχει γραφτεί κώδικας που αναπαράγει έναν ήχο για να προειδοποιεί τον χρήστη ότι συνέβη κάποια σύγκρουση.

Η συνάρτηση OnTriggerStay καλείται για όσο χρόνο συμβαίνει το overlap τον δυο αντικειμένων και μέσα της έχει γραφτεί κώδικας που «χρωματίζει» κόκκινο το αντικείμενο με το οποίο συγκρούεται το ρομπότ. Η πρόσβαση στο χρώμα του αντικειμένου αυτού γίνεται μέσω του **Collider other** που ορίζεται κατά την κλήση της συνάρτησης.

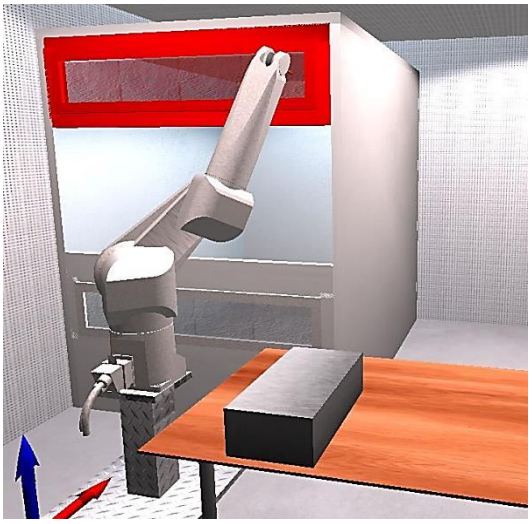
other.gameObject.renderer.material.SetColor("_Color", Color.red);

Τέλος, η συνάρτηση OnTriggerExit καλείται όταν σταματήσει να υφίσταται επαφή των δύο αντικειμένων και χρησιμοποιήθηκε για την επαναφορά του χρώματος τους.

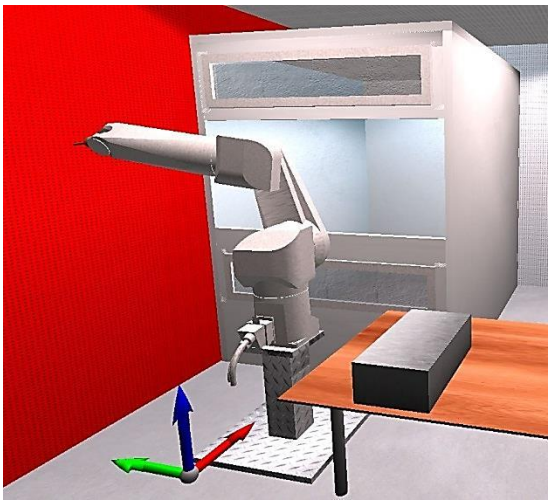
Οι επόμενες εικόνες δείχνουν διάφορες περιπτώσεις συγκρούσεων του ρομπότ με άλλα αντικείμενα.



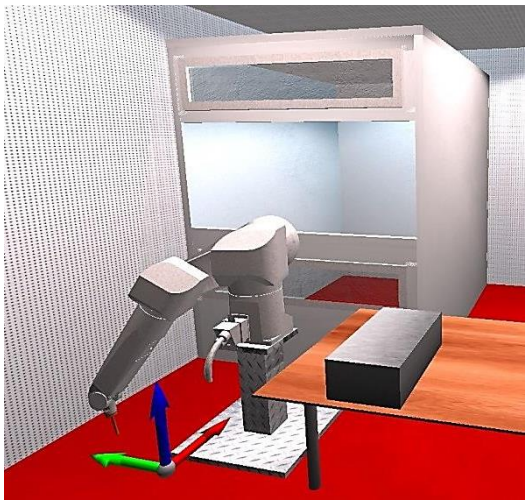
Εικόνα 41 - Σύγκρουση με το αντικείμενο συγκόλλησης



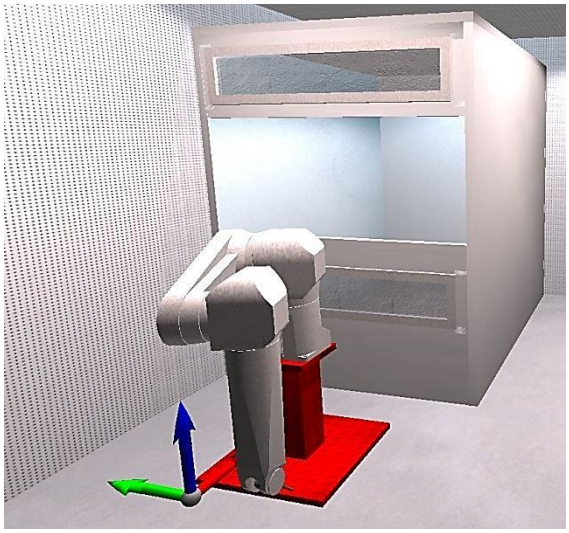
Εικόνα 40 - Σύγκρουση με το δωμάτιο ψεκασμού



Εικόνα 43 - Σύγκρουση με τον τοίχο του εργαστηρίου



Εικόνα 42 - Σύγκρουση με το δάπεδο



Εικόνα 44 - Σύγκρουση με τη βάση του ρομπότ

6.4 Εξαγωγή τροχιάς και πειραματικά αποτελέσματα

Η τροχιά που κάνει το ρομπότ στο εικονικό περιβάλλον πρέπει να μεταφερθεί και στο πραγματικό. Η τροχιά είναι ουσιαστικά ένα σύνολο εξάδων αριθμών που περιγράφουν τις τιμές των αρθρώσεων 1-6. Κάθε εξάδα αντιστοιχεί σε ένα μοναδικό σημείο και προσανατολισμό του τελικού σημείου δράσης στον χώρο.

Το ρομπότ προγραμματίζεται χρησιμοποιώντας τη γλώσσα V+. Μέσα στο script IKMover.cs, μετά την μετακίνηση του ρομπότ στην επόμενη θέση, έχει γραφτεί γραμμή κώδικα που καταγράφει τις τιμές των αρθρώσεων της θέσης αυτής, σε μορφή «κατανοητή» από το ρομπότ, όταν ο χρήστης πατήσει το κουμπί 5 στο χειριστήριο. Η μορφή αυτή είναι:

DO DECOMPOSE J_V[i]=#PPOINT(j1_Val, j2_Val, j3_Val, j4_Val, j5_Val, j6_Val) ;

όπου i = 0, 6, 12,… ,n.

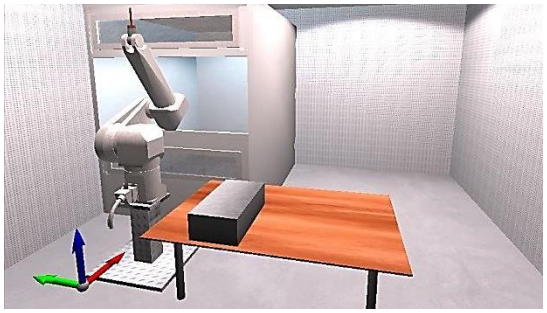
Το σύνολο των εντολών αυτών που απαρτίζουν την τροχιά, αποθηκεύεται στο αρχείο output.txt στον τρέχοντα κατάλογο της εφαρμογής. Το αρχείο αυτό ανοίγεται στη συνέχεια στο λογισμικό ελέγχου του ρομπότ με απλό OPEN. Για την εκτέλεσή των εντολών που περιέχει έχει γραφτεί ο επόμενος κώδικας που τις εκτελεί γραμμή – γραμμή:

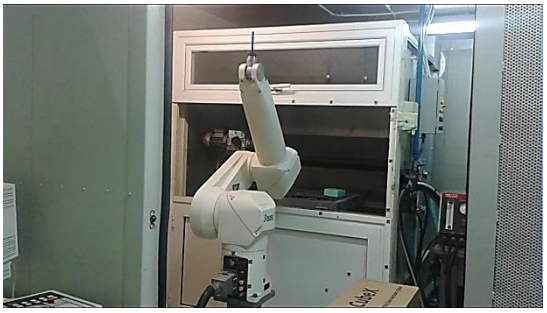
```
AUTO #j_v[6]
AUTO REAL j
        j = 1
FOR i = 0 TO LAST(j_v[]) STEP 6
      MOVE #PPOINT(j_v[i],j_v[i+1],j_v[i+2],j_v[i+3],j_v[i+4],j_v[i+5])
      BREAK
      DELAY 0
      TYPE "COMPLETED SET OF JOINTS NUMBER", $ENCODE(j)
      j = j+1
END
```

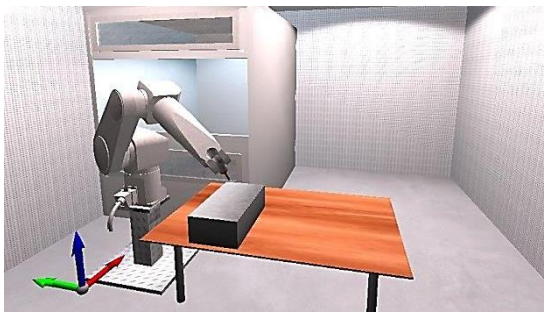
Ο κώδικας αυτός έχει αποθηκευτεί με όνομα FOB_06.PG και φορτώνεται χρησιμοποιώντας την εντολή LOAD FOB_06.PG.

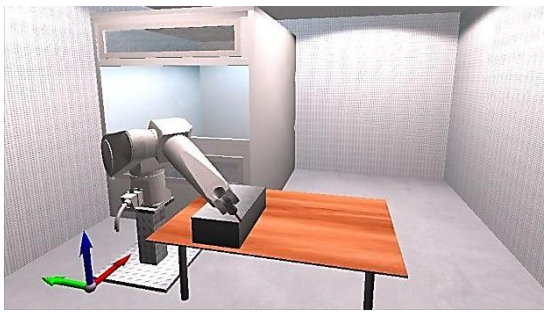
Εφόσον έχουμε φορτωμένα και το αρχείο με την τροχιά και τον κώδικα, το ρομπότ μπορεί να αναπαράγει την τροχιά πληκτρολογώντας την εντολή EXECUTE FOB_06().

Στο πείραμα που έγινε προσομοιώθηκε συγκόλληση των 3 εκ των 4 πλευρών του κουτιού. Το ρομπότ ξεκίνησε από τη θέση Do Ready και οδηγήθηκε με γωνία 45° ως προς κάθε πλευρά. Κατά την πορεία αποθηκεύτηκαν στο αρχείο εξόδου σημεία της τροχιάς και φορτώθηκαν στο πραγματικό ρομπότ για αναπαραγωγή. Οι επόμενες εικόνες δείχνουν σε σύγκριση την τροχιά του πραγματικού και του εικονικού ρομπότ.



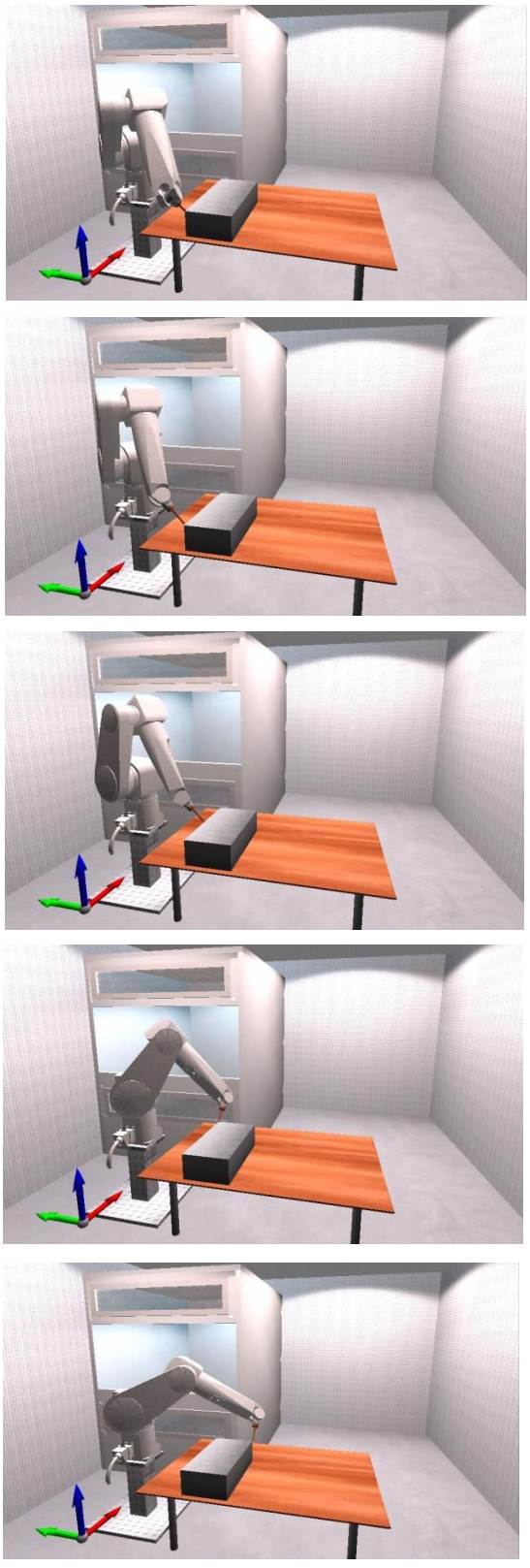






Εικόνα 46 - Τροχιά συγκόλλησης εικονικού ρομπότ

Εικόνα 45 - Τροχιά συγκόλλησης πραγματικού ρομπότ



Εικόνα 47 - Τροχιά συγκόλλησης εικονικού ρομπότ (συνέχεια)

Εικόνα 48 - Τροχιά συγκόλλησης πραγματικού ρομπότ (συνέχεια)

Παράλληλα, σε κάθε ενδιάμεση θέση της τροχιάς καταγραφόταν η θέση του ΤΣΔ σε mm που εκτυπώνεται από το Unity και αυτή που εκτυπώνεται από τον controller του ρομπότ. Τα αποτελέσματα φαίνονται στον επόμενο πίνακα:

X – Unity	Y – Unity	Z - Unity	X – Real	Y – Real	Z - Real
0	0	809.70	0.01	0	809.95
-325.98	-428.74	810.08	-325.36	-429.35	809.92
225.02	-612.53	430.47	-225.02	-612.53	430.04
-355.74	-781.32	-0.10	-355.77	-781.37	-0.11
-506.89	-781.25	-104.86	-506.90	-781.25	-105.1
-517.19	-907.96	-105.10	-517.19	-907.96	-105.11
-517.18	-907.95	-175.10	-517.18	-907.96	-175.11
-517.22	-858.01	-175.10	-517.29	-858.53	-175.13
-517.27	-803.92	-175.10	-517.29	-804.35	-175.13
-517.22	-757.97	-175.10	-517.27	-757.48	-175.11
-517.22	-713.52	-175.10	-517.15	-713.87	-175.09
-517.24	-667.75	-175.10	-517.29	-667.21	-175.14
-517.20	-620.32	-175.09	-517.29	-619.88	-175.12
-517.24	-590.92	-175.11	-517.30	-590.43	-175.13
-577.22	-398.86	-175.07	-577.22	-398.87	-175.08
-577.33	-494.06	-175.10	-577.34	-494.07	-175.10
-437.22	-494.02	-175.11	-434.07	-493.92	-175.09
-381.42	-494.03	-175.10	-381.36	-494.19	-175.12
-136.85	-493.96	-175.08	-136.61	-494.04	-175.09
73.86	-493.93	-175.14	73.98	-493.83	-174.94
121.70	-494.06	-175.07	121.82	-493.73	-175.04
166.01	-494.07	-175.12	166.21	-493.78	-175.14
306.74	-492.05	-175.02	306.74	-492.05	-175.02
264.64	-432.15	-190.13	264.65	-432.15	-190.13
264.63	-583.74	-190.08	264.64	-584.07	-190.10
264.67	-649.29	-190.11	264.64	-648.67	-190.12

Πίνακας 6 – Σύγκριση των συντεταγμένων του ΤΣΔ που επιστρέφει το Unity και ο Controller του ρομπότ

Όπως είναι φανερό, το εικονικό περιβάλλον δίνει αποτελέσματα ακρίβειας χιλιοστού σε σχέση με το πραγματικό. Παρατηρήθηκε ωστόσο κάτι που μπορεί να επηρεάσει έντονα το αποτέλεσμα μια τροχιάς. Το ρομπότ παίρνει εντολή να μετακινήσει τις αρθρώσεις από τη θέση που βρίσκονται σε μια επόμενη. Ο Controller είναι έτσι προγραμματισμένος έτσι ώστε όλες οι αρθρώσεις να ξεκινούν και να σταματούν ταυτόχρονα την κίνησή τους, με σκοπό την ελάττωση των ταλαντώσεων λόγω αδρανειακών δυνάμεων και την μεγιστοποίηση της ακρίβειας.

Στην περίπτωση της συγκόλλησης, κατά την κίνηση κατά μήκος μιας πλευράς, το ηλεκτρόδιο πρέπει να παραμένει όσο το δυνατόν γίνεται σε σταθερή απόσταση από την ακμή συγκόλλησης. Αν ο χειριστής δώσει σαν είσοδο στο ρομπότ δύο σημεία τα οποία απέχουν αρκετά μεταξύ τους, τότε το ρομπότ θα ακολουθήσει την δική του παρεμβολή γωνιών και παρόλο που θα πάει ακριβώς στο σημείο που ζητήθηκε, τα ενδιάμεσα σημεία δεν θα αντιστοιχούν στην τροχιά που προγραμματίστηκε.

Για την επίλυση αυτού του προβλήματος, ο χρήστης πρέπει να παίρνει όσες περισσότερες μετρήσεις είναι δυνατόν κατά τον προγραμματισμό της τροχιάς στο εικονικό περιβάλλον, για να μην έχει περιθώριο απόκλισης ο controller. Η βέλτιστη περίπτωση είναι οι μετρήσεις να παίρνονται σε κάθε frame, έτσι ώστε η τροχιά να είναι ακριβώς η ίδια με αυτή που προβάλλεται στη Unity.

7 Τηλεπαρακολούθηση Λειτουργίας

7.1 Το χειριστήριο

Στην περίπτωση της τηλεπαρακολούθησης της λειτουργία του ρομποτικού βραχίονα, θέλουμε να πετύχουμε ακριβώς το αντίθετο από αυτό που υλοποιήθηκε στον προγραμματισμό τροχιάς. Καθώς το ρομπότ λειτουργεί, θέλουμε να βλέπουμε στη σκηνή του Unity ακριβώς την τροχιά που κάνει. Για να γίνει αυτό, πρέπει με κάποιο τρόπο να δίνουμε σαν είσοδο στο Unity, την θέση του καρπού και τον προσανατολισμό του ΤΣΔ, χωρίς να παίρνουμε δεδομένα κατευθείαν από τον controller του ρομπότ.

Η μελέτη έγινε χρησιμοποιώντας το χειριστήριο Wii Remote (Wiimote για συντομία) της Nintendo. Το συγκεκριμένο χειριστήριο χρησιμοποιήθηκε στην κονσόλα Wii και ήταν επαναστατικό στον τομέα των ηλεκτρονικών παιχνιδιών, λόγω της ικανότητας ανίχνευσης κίνησης, η οποία επιτρέπει στο χρήστη να αλληλεπιδρά και να διαχειρίζεται αντικείμενα στην οθόνη, μέσω της αναγνώρισης χειρονομιών.

Εικόνα 49 - Το Wii Remote

Το Wiimote έχει τη δυνατότητα να ανιχνεύει επιτάχυνση κατά μήκος των τριών αξόνων με τη χρήση του επιταχυνσιόμετρου ADXL330. Επιπλέον, χρησιμοποιεί ένα αισθητήρα υπερύθρων που του επιτρέπει να λειτουργεί σαν δείκτης ποντικιού στην οθόνη. Η επικοινωνία του επιταχυνσιόμετρου με την κονσόλα γίνεται με Bluetooth, ενώ η επικοινωνία μέσω υπερύθρων γίνεται με το Sensor Bar το οποίο εκπέμπει υπέρυθρο φως από 10 Led.

Εικόνα 50 - Wii Sensor Bar

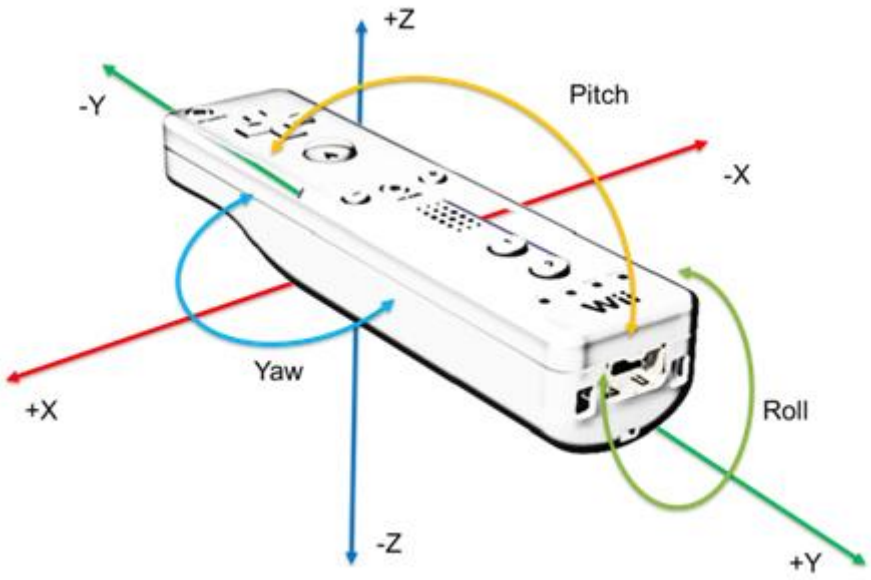
Αργότερα, προστέθηκε και γυροσκόπιο στο χειριστήριο το οποίο κούμπωνε στο κάτω μέρος του, δίνοντάς του τη δυνατότητα να μετράει γωνίες γύρω από του 3 άξονες (roll, pitch, yaw). Η τελευταία έκδοση είχε το γυροσκόπιο ενσωματωμένο στην κύρια συσκευή. [26]

Εικόνα 51 - Το Wii Remote με το γυροσκόπιο συνδεμένο

Το επιταχυνσιόμετρο μετράει επιταχύνσεις σε εύρος ±3g. Μπορεί να μετρήσει την στατική επιτάχυνση που δέχεται κάθε άξονας εξαιτίας της βαρύτητας, καθώς και την δυναμική επιτάχυνση που προκαλείται από την κίνησή του. Προφανώς η μέτρηση της στατικής επιτάχυνσης εξαρτάται από το ποιος άξονας κοιτάει το έδαφος. Τα δύο ήδη

μετρήσεων είναι πεπλεγμένα, δηλαδή δεν μπορούμε να διαβάσουμε μόνο την δυναμική επιτάχυνση αμελώντας την επιτάχυνση της βαρύτητας. [27]

Το γυροσκόπιο επιστρέφει τιμές απόλυτων γωνιών με εύρος ±90°. Για την μέτρησή τους χρησιμοποιεί την επίδραση της βαρύτητας. Για παράδειγμα, αν θέλει να μετρήσει την γωνία roll, μετράει με τι επιταχύνσεις έλκει η βαρύτητα τους άξονες ±X και υπολογίζει την γωνία που έχει στραφεί το χειριστήριο γύρω από τον άξονα X. Οι φορές των αξόνων και των περιστροφών φαίνονται στην εικόνα 52.



Εικόνα 52 - Οι άξονες και οι περιστροφές του Wii Remote

7.2 Η βασική ιδέα

Για την επίτευξη της τηλεπαρακολούθησης χρειάζονται δύο Wiimote, ένα προσαρτημένο στον καρπό του ρομπότ με σκοπό να μετράει επιταχύνσεις και ένα στο τελικό σημείο δράσης για να μετράει τις περιστροφές.

Το ρομπότ πρέπει να ξεκινάει πάντα την κίνηση του από ένα γνωστό σημείο, έστω την θέση Do Ready. Καθώς κινείται το ρομπότ, σε κάθε frame διαβάζουμε την επιτάχυνσή του σε κάθε άξονα και τις περιστροφές γύρω από αυτούς. Οι περιστροφές επιστρέφονται από το χειριστήριο σε μορφή απολύτων αριθμών και μάλιστα οι γωνίες

roll, pitch και yaw είναι στην πράξη σε περιγραφή Euler. Αυτό σημαίνει ότι με τη διαδικασία που περιγράφτηκε στο κεφάλαιο 5.4 μπορούμε να δημιουργήσουμε τον πίνακα περιστροφής.

Για τον υπολογισμό της θέσης βασιζόμαστε στην κλασσική φυσική. Ένα σώμα επιταχυνόμενο με σταθερή επιτάχυνση a_x, το οποίο ξεκινάει από γνωστή θέση x(t) με αρχική ταχύτητα $v_x(t)$, σε χρόνο dt θα έχει ταχύτητα και θέση αντίστοιχα:

$$v_x(t+dt) = v_x(t) + a_x \cdot dt$$

$$x(t+dt) = x(t) + v_x(t) \cdot dt + \frac{1}{2} \cdot a_x \cdot (dt)^2$$

Ομοίως, στις άλλες κατευθύνσεις:

$$v_y(t+dt) = v_y(t) + a_y \cdot dt$$

$$y(t+dt) = y(t) + v_y(t) \cdot dt + \frac{1}{2} \cdot a_y \cdot (dt)^2$$

$$v_z(t+dt) = v_z(t) + a_z \cdot dt$$

$$z(t+dt) = z(t) + v_z(t) \cdot dt + \frac{1}{2} \cdot a_z \cdot (dt)^2$$

Εφόσον το ρομπότ ξεκινάει από την ακινησία από γνωστή θέση, γνωρίζουμε την αρχική ταχύτητα και θέση στο πρώτο frame. Άρα καθώς τρέχει το πρόγραμμα, υπολογίζονται σε κάθε frame η καινούρια ταχύτητα και θέση, έχοντας ως δεδομένο την επιτάχυνση που επιστρέφει το Wiimote ανά άξονα. Οι επιταχύνσεις όμως υπολογίζονται με βάση το τοπικό σύστημα συντεταγμένων του χειριστηρίου. Για να μετατρέψουμε την έκφραση των θέσεων που προκύπτουν από το τοπικό σύστημα συνταγμένων στο αδρανειακό, πολλαπλασιάζουμε τον πίνακα στήλη των μετατοπίσεων με τον πίνακα των περιστροφών, δηλαδή:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = R \cdot \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix}$$

Άρα το μόνο που χρειαζόμαστε για να επιτύχουμε τον στόχο της τηλεπαρακολούθησης, είναι η πρόσβαση στις 3 περιστροφές και 3 επιταχύνσεις του wiimote.

7.3 Προσπάθεια υλοποίησης

Για να μπορέσουμε να πάρουμε τις μετρήσεις που χρειαζόμαστε, πρέπει να αναπτυχθεί καταρχήν επικοινωνία μεταξύ του χειριστηρίου και του Unity. Για την επίτευξή της, χρησιμοποιήθηκε η βιβλιοθήκη UniWii.dll, η οποία δημιουργήθηκε από τον Rob Terrell της εταιρίας Stinkbot LLC. Η βιβλιοθήκη περιέχει διάφορες συναρτήσεις, οι οποίες επιστρέφουν δεδομένα του χειριστηρίου, όπως η στάθμη της μπαταρίας του, το πάτημα των κουμπιών του, οι μετρήσεις των αισθητήρων κ.α.

Το αρχείο τοποθετήθηκε στα Assets του project στον κατάλογο Plugins. Για να μπορέσουμε να χρησιμοποιήσουμε τις συναρτήσεις που μας ενδιαφέρουν, πρέπει να προστεθούν κάποιες γραμμές κώδικα στην αρχή του IKMover.cs. Αυτές είναι οι επόμενες:

```
[DllImport ("UniWii")]
private static extern void wiimote_start();
[DllImport ("UniWii")]
private static extern void wiimote_stop();
[DllImport ("UniWii")]
private static extern byte wiimote_getAccX(int which);
[DllImport ("UniWii")]
private static extern byte wiimote_getAccY(int which);
[DllImport ("UniWii")]
private static extern byte wiimote_getAccZ(int which);
[DllImport ("UniWii")]
private static extern float wiimote_getRoll(int which);
[DllImport ("UniWii")]
private static extern float wiimote_getPitch(int which);
[DllImport ("UniWii")]
private static extern float wiimote_getYaw(int which);
```

Τα ονόματα των συναρτήσεων υποδεικνύουν τι κάνουν, γι' αυτό περαιτέρω ανάλυση δεν θα γίνει. Η βιβλιοθήκη επιτρέπει την σύνδεση έως και 16 wiimote ταυτόχρονα. Το όρισμα int which που δέχονται ορισμένες συναρτήσεις, είναι ο αύξων αριθμός κάθε χειριστηρίου.

Στην ιδανική περίπτωση, αν διαβάζαμε τις επιταχύνσεις και τις περιστροφές από τις αντίστοιχες συναρτήσεις, το εγχείρημα της τηλεπαρακολούθησης θα είχε επιτευχθεί, αφού ο κώδικας της αντίστροφης κινηματικής υπάρχει ήδη από τον προγραμματισμό τροχιάς. Κατά την διάρκεια της υλοποίησης όμως παρουσιάστηκαν ορισμένα προβλήματα.

Αρχικά έγινε προσπάθεια λήψης των γωνιών για την υλοποίηση του προσανατολισμού. Επειδή οι γωνίες επιστρέφονται ως απόλυτες τιμές και είναι γωνίες Euler, οι τιμές των γωνιών θα εκχωρηθούν στις μεταβλητές a_x, a_y και a_z χρησιμοποιώντας τις εντολές:

a_x = wiimote_getPitch(0);

a_y = wiimote_getRoll(0);

a_z = wiimote_getYaw(0);

Οι περιστροφές pitch και roll επιστρέφονταν κανονικά, όμως η εντολή getYaw επέστρεφε πάντα μηδέν, οποιονδήποτε προσανατολισμό και να είχε το χειριστήριο. Πιθανότατα αυτό να οφείλεται στο γεγονός ότι το wiimote μετρά την γωνία yaw με χρήση των υπερύθρων και του Sensor Bar. Το συγκεκριμένο εξάρτημα δεν μπόρεσε να συνδεθεί στον υπολογιστή επειδή έχει ειδικό ακροδέκτη, επομένως δεν μπόρεσε και να ελεγχθεί. Μια εναλλακτική που δοκιμάστηκε για να λυθεί το πρόβλημα, ήταν η χρήση δύο wiimote σε διάταξη σταυρού.

Εικόνα 53 - Δύο Wii Remotes σε διάταξη σταυρού

Σε αυτή την περίπτωση, αντί να μετρήσουμε την γωνία Yaw του χειριστηρίου 1, επιχειρούμε να μετρήσουμε την γωνία roll του χειριστηρίου 2 με την εντολή:

a_z = wiimote_getRoll(1);

Όπως είπαμε προηγουμένως όμως, τα γυροσκόπια μετρούν τις γωνίες βάσει της βαρύτητας. Στην συγκεκριμένη περίπτωση, η γωνία roll στο χειριστήριο 2 δεν επηρεάζεται από την βαρύτητα επειδή ο άξονας X είναι οριζόντιος. Επομένως όταν πραγματοποιούταν η στροφή, η τιμή που επιστρεφόταν ήταν πάντα μηδέν. Άρα όχι μόνο δεν λύθηκε το πρόβλημα, αλλά εμφανίστηκε ένα ακόμη. Αυτό ήταν ότι καθώς το χειριστήριο πήγαινε σταδιακά σε κάποια θέση που ένας από του άξονες δεν επηρεάζεται από την βαρύτητα, χανόταν σταδιακά ένα μέρος της πραγματικής μέτρησης.

Όσον αφορά τις επιταχύνσεις, το πρόβλημα ήταν ότι μέσα στις μετρήσεις συμπεριλαμβανόταν και η τιμή της βαρύτητας. Για παράδειγμα, όταν το χειριστήριο ήταν έτσι στραμμένο ώστε ο άξονας +Y να κοιτάει το έδαφος, η επιτάχυνση a_y αντί να επιστρέφει μηδέν, επέστρεφε +g. Επειδή το σύστημα συντεταγμένων είναι ορθογώνιο, αν μπορούσαμε να πάρουμε τις μετρήσεις όλων των γωνιών σωστά, θα μπορούσαμε να απαλείψουμε την συμβολή της βαρύτητας σύμφωνα με τις παρακάτω πράξεις:

$$acc_x = acc_x - g \cdot \cos(roll)$$

$$acc_y = acc_y - g \cdot \cos(pitch)$$

$$acc_z = acc_z - g \cdot \cos(yaw)$$

Προφανώς όμως το τελευταίο δεν ήταν εφαρμόσιμο επειδή δεν έχουμε τις γωνίες.

Ως συμπέρασμα μετά από τις προσπάθειες επίλυσης των προβλημάτων αυτών, ήταν ότι το συγκεκριμένο χειριστήριο κατασκευάστηκε να λειτουργεί με αυτόν τον τρόπο για να εξυπηρετήσει τις λειτουργίες των παιχνιδιών του Wii, δεν είναι όμως κατάλληλο για την συγκεκριμένη μηχανολογική εφαρμογή. Συνεπώς η εφαρμογή της τηλεπαρακολούθησης δεν υλοποιήθηκε.

8 Συμπεράσματα

8.1 Συνεισφορά

Στην παρούσα εργασία περιγράφτηκε μια μεθοδολογία ανάπτυξης του μοντέλου ενός βιομηχανικού χώρου σε περιβάλλον εικονικής πραγματικότητας. Η διαδικασία περιλάμβανε τόσο την εισαγωγή των 3Δ μοντέλων για την αναπαράσταση του φυσικού χώρου, όσο και τον προγραμματισμό τους ώστε τα κινούμενα μέρη να αποκτήσουν «ζωή» και να αλληλεπιδρούν με τα υπόλοιπα. Βασικός πυλώνας του εικονικού συστήματος είναι το λογισμικό Unity, ένα πρόγραμμα που αρχικά δημιουργήθηκε για την ανάπτυξη ηλεκτρονικών παιχνιδιών, λόγω όμως των δυνατοτήτων που παρέχει, το πεδίο χρήσης επεκτείνεται σταδιακά.

Η δημιουργία των διαφόρων οντοτήτων και η ανάπτυξη της μεθοδολογίας αλληλεπίδρασης συστήματος – χρήστη είναι η βασική συνεισφορά της εργασίας. Μπορούν να αποτελέσουν μάλιστα υπόβαθρο για την επέκτασή της ή για την ανάπτυξη άλλων σχετικών με το αντικείμενο. Συγκεκριμένα προσφέρονται:

1. Μοντέλο του ρομποτικού βραχίονα Stäubli RX90L με πλήρως ανεπτυγμένη κινηματική αλυσίδα μεταξύ των συνδέσμων.

2. Μοντέλο πραγματικών διαστάσεων του χώρου του εργαστηρίου εκτόξευσης πλάσματος.

3. Ευθεία και αντίστροφη κινηματική ανάλυση 6R αρθρωτών ρομποτικών βραχιόνων.

4. Πειραματικά αποτελέσματα από την διαχείριση μηχανολογικού εξοπλισμού μέσω συστήματος εικονικής πραγματικότητας.

8.2 Πλεονεκτήματα του εικονικού περιβάλλοντος

Ο συμβατικός προγραμματισμός εργασιών ενός βιομηχανικού ρομπότ, στηριζόταν ανέκαθεν στην εμπειρία του χειριστή, ο οποίος προγραμμάτιζε την τροχιά του ενώ βρισκόταν σε λειτουργία. Ο τρόπος αυτός εκτός από το ότι ήταν χρονοβόρος, εγκυμονούσε τον κίνδυνο του λάθους, θέτοντας σε κίνδυνο την ακεραιότητα του χειριστή, του βιομηχανικού εξοπλισμού καθώς και της εργασίας.

Στη σημερινή εποχή, νέοι μηχανικοί και τεχνίτες καλούνται να επιτελέσουν την εργασία του προγραμματισμού. Λόγω της εξοικείωσής τους με τις τεχνολογίες εικονικής πραγματικότητας, η χρήση της τελευταίας στην εργασία αυτή δίνει γρήγορα, αξιόπιστα και ασφαλή αποτελέσματα.

Όφελος υπάρχει προφανώς και από οικονομικής άποψης. Χρηματικοί πόροι εξοικονομούνται λόγω της μείωσης των απαιτούμενων εργατοωρών, των αποτυχημένων προσπαθειών καθώς και των ζημιών που προκαλούν οι συγκρούσεις. Επίσης, ένα τέτοιο σύστημα μπορεί να χρησιμοποιηθεί για την σωστή μελέτη τοποθέτησης πρόσθετου μηχανολογικού εξοπλισμού σε ένα ήδη υπάρχον περιβάλλον.

Όσον αφορά την τηλεπαρακολούθηση λειτουργίας του ρομποτικού βραχίονα, το όφελος έγκειται στην απομάκρυνση του προσωπικού από τα βαρέα μηχανήματα. Στις περισσότερες βιομηχανίες πάντα υπάρχει ένας μηχανικός στον χώρο της εργασίας για να επιβλέπει τυχόν προβλήματα με τον εξοπλισμό και τη ροή της παραγωγής. Αν η εποπτεία γίνεται εξ αποστάσεως, εξαλείφεται η πιθανότητα τραυματισμού του εργαζόμενου.

8.3 Προτάσεις για μελλοντική έρευνα

Η ανάπτυξη του συστήματος προγραμματισμού τροχιάς ήταν επιτυχής, υπάρχουν όμως πολλά περιθώρια βελτίωσης.

- Το σενάριο της συγκόλλησης περιλάμβανε τον προσανατολισμό του ΤΣΔ με γωνία 45° ως προς το οριζόντιο επίπεδο και για τις 4 πλευρές. Μια πιθανή επέκταση είναι ο συνεχής προσανατολισμός σε καμπύλες γραμμές χωρίς να υπάρχει παρέμβαση του χρήστη. Ένα παράδειγμα είναι η προσομοίωση

συγκόλλησης ενός κύκλου, παραμένοντας πάντα με γωνία 45^{o} ως προς το επίπεδο του σε όλο το μήκος της ακμής του.

- Στην συγκεκριμένη εργασία, η έρευνα επικεντρώθηκε στην κινηματική ανάλυση του βραχίονα, αγνοώντας εντελώς την δυναμική που τον διέπει. Θα μπορούσε να γίνει προσπάθεια δυναμικής προσομοίωσής του, με σκοπό την ανάλυση των ταλαντώσεων κατά τη λειτουργία καθώς και την επίδραση που αυτές έχουν στην ακρίβεια της τροχιάς.

- Η εισαγωγή του χρόνου στο project έχει σημασία. Παίρνοντας δεδομένα της παρεμβολής που κάνει το πραγματικό ρομπότ και του χρόνου που χρειάζεται για να την ολοκληρώσει, μπορεί να προσομοιωθεί ο χρόνος περαίωσης μιας εργασίας. Η εφαρμογή τότε, ξεφεύγει από τα όρια του απλού προγραμματισμού τροχιάς και μπορεί να χρησιμοποιηθεί στον προγραμματισμό εργασιών μιας ολόκληρης παραγωγικής μονάδας.

- Τελευταίο και δυσκολότερο αποτελεί ο online εξ αποστάσεως χειρισμός του ρομπότ. Αυτό σημαίνει ότι αντί ο χρήστης να παίρνει αρχείο εξόδου από το Unity και να το περνά στον controller του ρομπότ, οι εντολές που δίνει μέσω του χειριστηρίου να εκτελούνται σε πραγματικό χρόνο από το ρομπότ. Η μεγαλύτερη δυσκολία ενός τέτοιου εγχειρήματος, είναι οι ιδιομορφίες της αντίστροφης κινηματικής που μπορούν να οδηγήσουν σε πολύ μεγάλες δυνάμεις στους επενεργητές των αρθρώσεων και συνεπώς στην καταστροφή τους.

Η τηλεπαρακολούθηση της λειτουργίας του βραχίονα, αν και δεν επιτεύχθηκε, πιθανόν να είναι εφικτή με χρήση άλλου εξοπλισμού. Μετά από έρευνα των αισθητήρων που υπάρχουν στο εμπόριο, οι αισθητήρες που προτείνονται για μελλοντική έρευνα είναι οι εξής:

- **Γυροσκόπιο** το οποίο να μετρά τις γωνίες γύρω από όλους τους άξονες, είτε σε μορφή απολύτων γωνιών, είτε σε μορφή γωνιακών επιταχύνσεων.
- **Μαγνητόμετρο** για την μέτρηση του μαγνητικού βορρά σε περίπτωση που το γυροσκόπιο δεν μπορεί να μετρήσει την γωνία yaw. Αν γνωρίζουμε την θέση

του μαγνητικού βορρά, μπορούμε να βαθμονομήσουμε την μέτρηση για να αντιστοιχεί στο μηδέν της yaw.

- **Γραμμικό επιταχυνσιόμετρο** το οποίο μετρά επιταχύνσεις στους τρεις άξονες, απαλλαγμένες από την επίδραση της βαρύτητας.

Πρόσφατα αναπτύχθηκαν εφαρμογές για smartphone, οι οποίες χρησιμοποιούν τους αισθητήρες του κινητού και κάνουν stream τις μετρήσεις στον υπολογιστή. Ιδέα μελέτης θα ήταν η ανάπτυξη επικοινωνίας μεταξύ του Unity και μιας τέτοιας εφαρμογής και η απόπειρα χρήσης των μετρήσεων για την υλοποίηση της τηλεπαρακολούθησης.

9 Βιβλιογραφία

[1] Wikipedia, «Εικονική Πραγματικότητα,» [Ηλεκτρονικό]. Available: https://el.wikipedia.org/wiki/%CE%95%CE%B9%CE%BA%CE%BF%CE%BD%CE%B9%CE%BA%CE%AE_%CF%80%CF%81%CE%B1%CE%B3%CE%BC%CE%B1%CF%84%CE%B9%CE%BA%CF%8C%CF%84%CE%B7%CF%84%CE%B1. [Πρόσβαση 14 Σεπτεμβρίου 2015].

[2] Wikipedia, «Virtual Reality,» [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/Virtual_reality. [Πρόσβαση 14 Σεπτεμβρίου 2015].

[3] L. Zlajpah, «Simulation in Robotics,» *Mathematics and Computers in Simulation, Vol.79,* pp. 879-897, 2008.

[4] C. A. Jara, F. A. Candelas, P. Gil, F. Torres, F. Esquembre και S. Dormido, «An interactive tool for industrial robots simulation, Computer Vision and remote operation,» *Robotics and Autonomous Systems, Vol. 59 No6,* pp. 389-401, 2011.

[5] J. Rossman, «Advanced virtual testbeds: Robotics know how for virtual worlds,» σε *Proceedings of the 11th International Conference on Control Automation Robotics & Vision (ICARCV)*, Singapore, 7-10 December 2010, pp. 133-138.

[6] B. Hein και H. Worn, «Intuitive and model-based on-line programming of industrial robots: New input devices,» σε *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2009)*, 10-15 October 2009, pp. 3064-3069.

[7] O. K. Gupta και R. A. Jarvis, «Using a Virtual World to Design a Simulation Platform for Vision and Robotic Systems,» *Advances in Visual Computing, Lecture Notes in*

Computer Science, Vol. 5875, pp. 233-242, 2009.

[8] G. Mogran, D. Talaba, F. Girbacia, T. Butnaru, S. Sisca και C. Aron, «A generic multimodal interface for design and manufacturing applications,» σε *Proceedings of the 2nd International Workshop Virtual Manufacturing (VirMan08) - part of the 5th INTUITION International Conference: Virtual Reality in Industry and Society: From Research to Application*, Torino, Italy, 6 - 8 October 2008, p. 10.

[9] Y.-H. Chen, B. A. MacDonald και B. C. Wunsche, «Designing a Mixed Reality Framework of Enriching Interactions in Robot Simulation,» σε *Proceedings of the International Conference on Computer Graphics Theory an Applications (GRAPP2010)*, France, 17-21 May 2010, pp. 331-338.

[10] Z. Pan, J. Polden, N. Larkin, S. v. Duin και J. Norrish, «Recent progress on programming methods for industrial robots,» *Robotics and Computer-Integrated Manufacturing, Vol. 28 No 2,* pp. 87-94, 2012.

[11] B. Akan, A. Ameri, B. Curuklu και L. Asplund, «Intuitive industrial robot programming through incremental multimodal language and augmented reality,» σε *Proceedings of the 2011 IEEE International Conference on Robotics and Automation (ICRA2011)*, Shanghai, China, 9-13 May 2011, pp. 3934-3939.

[12] J. W. S. Chong, S. K. Ong, A. Y. C. Nee και K. Yousef-Youmi, «Robot programming using augmented reality: An interactive method for plannin collision-free paths,» *Robotics and Computer-Integrated Manufacturin, Vol. 25 No 3,* pp. 689-701, 2009.

[13] S. K. Ong, J. W. S. Chong και A. Y. C. Nee, «A novel AR-based robot programming and path planning methodology,» *Robotics and Computer-Integrated Manufacturing, Vol. 28 No 2,* pp. 87-94, 2010.

[14] H. C. Fang, S. K. Ong και A. Y. C. Nee, «Interactive robot trajectory planning and simulation using Augmented Reality,» *Robotics and Computer Integrated Manufacturing, Vol. 28 No 2,* pp. 227-238, 2012.

[15] G. Reinhart, U. Munzert και W. Vogl, «A programming system for robot-based remote-laser-welding with conventional optics,» *CIRP Annals - Manufacturing*

Technology, Vol. 57, pp. 37-40, 2008.

[16] R. d. Amicis, G. d. Gironimo και A. Marzano, «Design of a Virtual Reality Architecture for Robotic Workcells Simulation,» σε *Proceedings of the Virtual Concept 2006 Conference*, Playa Del Carmen, Mexico, 26 November-1 December 2006, p. 6.

[17] Wikipedia, «Unity (Game Engine),» [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/Unity_(game_engine). [Πρόσβαση 16 Σεπτεμβρίου 2015].

[18] Stäubli, Stäubli Robotics: Painting Brochure.

[19] Stäubli, Stäubli Robotics: Arm - RX Series 90B family Characteristics, 2004.

[20] Ε. Λεβεδιανός, «Διερεύνηση Μεθοδολογίας και Εργαλείων Κατασκευής Μοντελων Εικονικής Πραγματικότητας για Ρομποτικά Κύτταρα Κατεργασιών,» Αθήνα, 2013.

[21] Adept Technology Inc., V+ Language User's Guide, 1997.

[22] J. Craig, Introduction to Robotics: Mechanics and Control, Third Edition, 1989.

[23] Ε. Παπαδόπουλος, «Κινηματική,» σε *Σημειώσεις Ρομποτικής*.

[24] M. Spong, S. Hutcinson και M. Vidyasagar, Robot Modeling and Control, 2005.

[25] D. Eberly, «Conversion of Left-Handed Coordinates to Right-Handed Coordinates,» 2003.

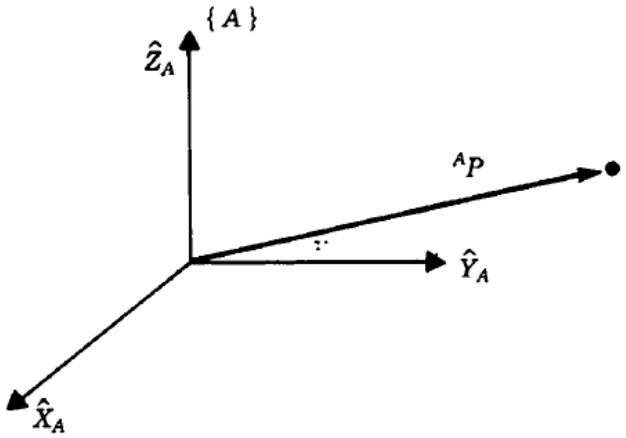
[26] Wikipedia, «Wii Remote,» [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/Wii_Remote. [Πρόσβαση 30 Σεπτέμβριος 2015].

[27] Analog Devices, iMEMS ADXL 3-Axis Accelerometer, 2007.

ΠΑΡΑΡΤΗΜΑ Α: Θέση και προσανατολισμός [23]

ΠΕΡΙΓΡΑΦΗ ΘΕΣΗΣ

Η θέση ενός σώματος σε σχέση με ένα άλλο καθορίζεται από το διάνυσμα θέσης. Το διάνυσμα θέσης είναι ένα μοναδικό διάνυσμα που δείχνει τη θέση ενός σημείου P ως προς κάποια αρχή Α.



Εικόνα 54 - Διάνυσμα θέσης σημείου P ως προς αρχή Α

Ένα διάνυσμα θέσης μπορεί να γραφεί ως:

$$\boldsymbol{p} = {}^{A}p_x\,\hat{\boldsymbol{x}}_{\boldsymbol{A}} + {}^{A}p_y\,\hat{\boldsymbol{y}}_{\boldsymbol{A}} + {}^{A}p_z\,\hat{\boldsymbol{z}}_{\boldsymbol{A}}$$

όπου:

$${}^{A}p_x = \boldsymbol{p}\cdot\hat{\boldsymbol{x}}_{\boldsymbol{A}}$$

$${}^{A}p_y = \boldsymbol{p}\cdot\hat{\boldsymbol{y}}_{\boldsymbol{A}}$$

$${}^{A}p_z = \boldsymbol{p}\cdot\hat{\boldsymbol{z}}_{\boldsymbol{A}}$$

οι προβολές του p στα μοναδιαία $\hat{\boldsymbol{x}}_{\boldsymbol{A}}, \hat{\boldsymbol{y}}_{\boldsymbol{A}}, \hat{\boldsymbol{z}}_{\boldsymbol{A}}$.

Σε μορφή πίνακα η θέση περιγράφεται από το διάνυσμα στήλης, το οποίο είναι ένα σύνολο τριών αριθμών που περιγράφουν ένα διάνυσμα P ως προς ένα σύστημα συντεταγμένων {A}.

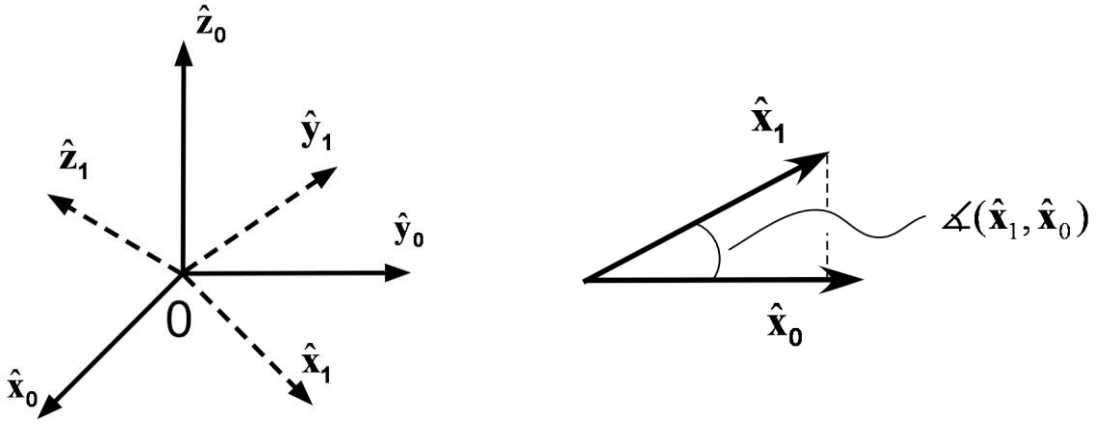
$$ {}^{A}\boldsymbol{p} = \begin{bmatrix} {}^{A}p_x \\ {}^{A}p_y \\ {}^{A}p_z \end{bmatrix} $$

ΠΡΟΣΑΝΑΤΟΛΙΣΜΟΣ ΣΩΜΑΤΟΣ

Συνημίτονα κατεύθυνσης

Για να περιγράψουμε τον προσανατολισμό ενός σώματος:

1. Προσαρτούμε ένα σωματόδετο ΣΣ σ΄ αυτό.
2. Δίνουμε μια περιγραφή του σωματόδετου ΣΣ ως προς το σύστημα συντεταγμένων που μας ενδιαφέρει.



Εικόνα 55 - Σχετικός προσανατολισμός δύο συστημάτων συντεταγμένων

Για να περιγράψουμε τον προσανατολισμό του μοναδιαίου $\hat{\boldsymbol{x}}_1$ ως προς το ΣΣ {0}, το προβάλλουμε στα μοναδιαία ($\hat{\boldsymbol{x}}_0$, $\hat{\boldsymbol{y}}_0$, $\hat{\boldsymbol{z}}_0$) του ΣΣ {0}. Με αυτό τον τρόπο, παίρνουμε τις τρεις συντεταγμένες του μοναδιαίου διανύσματος $\hat{\boldsymbol{x}}_1$ ως προς το ΣΣ {0} και σχηματίζουμε το διάνυσμα στήλης ${}^{0}\hat{\boldsymbol{x}}_1$ **.** Επαναλαμβάνουμε τη διαδικασία με τα υπόλοιπα δύο μοναδιαία διανύσματα του ΣΣ {1}. Το αποτέλεσμα είναι τρεις στήλες, ως εξής:

$$^{\mathbf{0}}\hat{\boldsymbol{x}}_{\mathbf{1}} = \begin{bmatrix} \cos(\hat{\boldsymbol{x}}_{\mathbf{1}}, \hat{\boldsymbol{x}}_{\mathbf{0}}) \\ \cos(\hat{\boldsymbol{x}}_{\mathbf{1}}, \hat{\boldsymbol{y}}_{\mathbf{0}}) \\ \cos(\hat{\boldsymbol{x}}_{\mathbf{1}}, \boldsymbol{z}_{\mathbf{0}}) \end{bmatrix} = \begin{bmatrix} r_{11} \\ r_{21} \\ r_{31} \end{bmatrix}$$

$$^{\mathbf{0}}\hat{\boldsymbol{y}}_{\mathbf{1}} = \begin{bmatrix} \cos(\boldsymbol{y}_{\mathbf{1}}, \hat{\boldsymbol{x}}_{\mathbf{0}}) \\ \cos(\hat{\boldsymbol{y}}_{\mathbf{1}}, \hat{\boldsymbol{y}}_{\mathbf{0}}) \\ \cos(\boldsymbol{y}_{\mathbf{1}}, \boldsymbol{z}_{\mathbf{0}}) \end{bmatrix} = \begin{bmatrix} r_{12} \\ r_{22} \\ r_{32} \end{bmatrix}$$

$$^{\mathbf{0}}\hat{\boldsymbol{z}}_{\mathbf{1}} = \begin{bmatrix} \cos(\hat{\boldsymbol{z}}_{\mathbf{1}}, \hat{\boldsymbol{x}}_{\mathbf{0}}) \\ \cos(\hat{\boldsymbol{z}}_{\mathbf{1}}, \hat{\boldsymbol{y}}_{\mathbf{0}}) \\ \cos(\hat{\boldsymbol{z}}_{\mathbf{1}}, \boldsymbol{z}_{\mathbf{0}}) \end{bmatrix} = \begin{bmatrix} r_{13} \\ r_{23} \\ r_{33} \end{bmatrix}$$

Τα τρία αυτά μοναδιαία διανύσματα σχηματίζουν έναν (3x3) πίνακα περιστροφής:

$$^{0}R_{1} = \left[\, ^{\mathbf{0}}\hat{\boldsymbol{x}}_{\mathbf{1}} \quad ^{\mathbf{0}}\hat{\boldsymbol{y}}_{\mathbf{1}} \quad ^{\mathbf{0}}\hat{\boldsymbol{z}}_{\mathbf{1}} \, \right] = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

και η περιγραφή αυτή του προσανατολισμού ονομάζεται συνημίτονα κατεύθυνσης.

Οι περιστροφές γύρω από του άξονες x, y, z, αν εφαρμόσουμε την παραπάνω διαδικασία προκύπτουν:

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta_x) & -\sin(\theta_x) \\ 0 & \sin(\theta_x) & \cos(\theta_x) \end{bmatrix} \qquad R_y = \begin{bmatrix} \cos(\theta_y) & 0 & \sin(\theta_y) \\ 0 & 1 & 0 \\ -\sin(\theta_y) & 0 & \cos(\theta_y) \end{bmatrix}$$

$$R_z = \begin{bmatrix} \cos(\theta_z) & -\sin(\theta_z) & 0 \\ \sin(\theta_z) & \cos(\theta_z) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

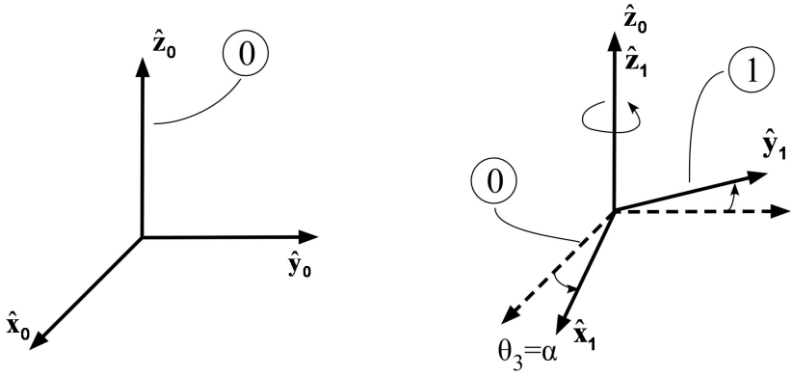
Ο πίνακας περιστροφής εκτός της περιγραφής της περιστροφής του ΣΣ{1} ως προς το ΣΣ{0}, μετασχηματίζει τις συνιστώσες ενός διανύσματος ως προς ΣΣ{1} στις συνιστώσες του ως προς το ΣΣ{0} με τον πολλαπλασιασμό:

$$\begin{bmatrix} ^{0}p_x \\ ^{0}p_y \\ ^{0}p_z \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} ^{1}p_x \\ ^{1}p_y \\ ^{1}p_z \end{bmatrix}$$

Γωνίες Euler

Οι γωνίες Euler ορίζονται ως οι γωνίες περιστροφής ενός σώματος γύρω από διαδοχικούς χωρόδετους άξονές του. Ανάλογα με τη διαδοχή περιστροφών και επιλογή αξόνων περιστροφής, μπορούμε να έχουμε δώδεκα τριάδες γωνιών Euler. Από αυτές, μελετάμε εδώ τη διαδοχή περιστροφών z-y-x, που είναι και γνωστές ως roll-pitch-yaw.

Όπως φαίνεται στο επόμενο σχήμα, η πρώτη περιστροφή γίνεται γύρω από τον άξονα $\hat{z}_0 = \hat{z}_1$. Η δεύτερη περιστροφή γίνεται γύρω από τον άξονα y του σωματόδετου ΣΣ, δηλαδή γύρω από το $\hat{y}_1 = \hat{y}_2$. Η τρίτη περιστροφή γίνεται γύρω από τον άξονα x του σωματόδετου ΣΣ, δηλαδή τον $\hat{x}_2$.



Εικόνα 56 - Z-Y-X Γωνίες Euler

Επομένως, η τελική θέση του σωματόδετου ΣΣ προκύπτει από:

- Περιστροφή του ΣΣ {0} κατά $R_{z_0}(\theta_3)$ έτσι ώστε να πάρουμε τελικά το ΣΣ {1}.
- Περιστροφή του ΣΣ {1} κατά $R_{y_1}(\theta_2)$ έτσι ώστε να πάρουμε τελικά το ΣΣ {2}.
- Περιστροφή του ΣΣ {2} κατά $R_{x_2}(\theta_1)$ έτσι ώστε να πάρουμε τελικά το ΣΣ {3}.

Ένα διάνυσμα στο τελικό ΣΣ {3} φαίνεται από το ΣΣ {0} ως:

$$^0\boldsymbol{p} = {}^0\boldsymbol{R_1} \cdot {}^1\boldsymbol{R_2} \cdot {}^2\boldsymbol{R_3} \cdot {}^3\boldsymbol{p} = \boldsymbol{R_{z_0}(\theta_3)} \cdot \boldsymbol{R_{y_1}(\theta_2)} \cdot \boldsymbol{R_{x_2}(\theta_1)} \cdot {}^3\boldsymbol{p} = {}^0\boldsymbol{R_3} \cdot {}^3\boldsymbol{p}$$

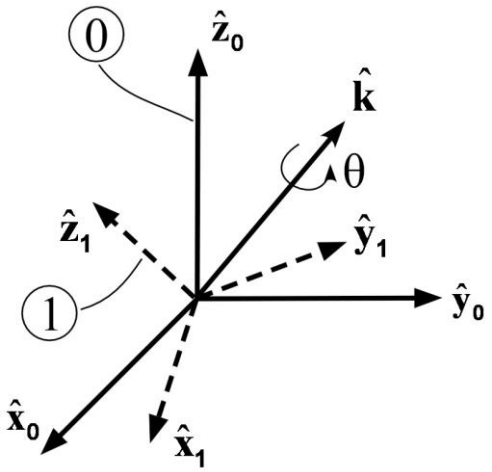
$$^{0}\boldsymbol{R_3}\ (\boldsymbol{\theta_1}, \boldsymbol{\theta_2}, \boldsymbol{\theta_3}) =$$

$$= \begin{bmatrix} \cos(\theta_3) & -\sin(\theta_3) & 0 \\ \sin(\theta_3) & \cos(\theta_3) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\theta_2) & 0 & \sin(\theta_2) \\ 0 & 1 & 0 \\ -\sin(\theta_2) & 0 & \cos(\theta_2) \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta_1) & -\sin(\theta_1) \\ 0 & \sin(\theta_1) & \cos(\theta_1) \end{bmatrix} =$$

$$= \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

Ζεύγος Ισοδύναμης Γωνίας – Άξονα Περιστροφής

Στην περίπτωση αυτή, η νέα θέση {1} του σωματόδετου ΣΣ προκύπτει από μία περιστροφή του ΣΣ {0} κατά γωνία θ γύρω από κάποιο άξονα $\widehat{\boldsymbol{k}} = \left[k_x, k_y, k_z\right]^T$



Εικόνα 57 - Ζεύγος Ισοδύναμης Γωνίας - Άξονα Περιστροφής

Εάν γνωρίζουμε τη γωνία και τον άξονα περιστροφής, τότε ο πίνακας περιστροφής δίνεται από την εξίσωση:

$$^{0}\boldsymbol{R_1} = \boldsymbol{R_k}(\theta) = \boldsymbol{I_3} cos\theta + \widehat{\boldsymbol{k}}\widehat{\boldsymbol{k}}^{\boldsymbol{T}}(1 - cos\theta) + \widehat{\boldsymbol{k}}^{x} sin\theta$$

όπου I_3 ο μοναδιαίος πίνακας 3x3 και:

$$\widehat{\boldsymbol{k}}^{x} = \begin{bmatrix} 0 & -k_z & k_y \\ k_z & 0 & -k_x \\ -k_y & k_x & 0 \end{bmatrix}$$

Αν τα συνημίτονα κατεύθυνσης του $^{\mathbf{0}}\boldsymbol{R_1}$ είναι γνωστά, οι παράμετροι που μας ενδιαφέρουν δίνονται από τις ακόλουθες εξισώσεις:

$$\theta = cos^{-1}\left(\frac{r_{11} + r_{22} + r_{33} - 1}{2}\right)$$

$$\widehat{\boldsymbol{k}} = \frac{1}{2sin\theta}\begin{bmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{bmatrix}$$

Παράμετροι Euler (Quaternion)

Οι παράμετροι Euler ορίζονται ως ένα διάνυσμα **ε** και μια παράμετρο $\boldsymbol{\varepsilon_4}$:

$$\boldsymbol{\varepsilon} = \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \end{bmatrix} = \widehat{\boldsymbol{k}} sin\left(\frac{\theta}{2}\right), \qquad \varepsilon_4 = cos\left(\frac{\theta}{2}\right)$$

Εάν είναι γνωστές οι παράμετροι Euler, τότε ο πίνακας περιστροφής δίνεται από την εξίσωση:

$$\boldsymbol{R}_{\varepsilon} = (2\varepsilon_4^2 - 1)\boldsymbol{I_3} + 2\varepsilon_4\boldsymbol{\varepsilon^x} + 2\boldsymbol{\varepsilon\varepsilon^T}$$

Το διάνυσμα στήλης $(\boldsymbol{\varepsilon}^T, \varepsilon_4)^T = [\varepsilon_1, \varepsilon_2, \varepsilon_3, \varepsilon_4]^T$ λέγεται μοναδιαίο quaternion (unit quaternion). Τα $(\boldsymbol{\varepsilon}^T, \varepsilon_4)$ λέγονται και τετραγωνικές αναλλοίωτες (quadratic invariants).

Αν δίνεται ο πίνακας περιστροφής **R**, τότε οι παράμετροι υπολογίζονται ως εξής:

$$\varepsilon_4 = \frac{1}{2}\sqrt{(r_{11} + r_{22} + r_{33} + 1)}\,, \qquad 0 \le \theta \le \pi$$

$$\varepsilon = \frac{1}{4\varepsilon_4}\begin{bmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{bmatrix}$$

Η περιγραφή αυτή χρησιμοποιείται στη ρομποτική διότι παρουσιάζει τις λιγότερες ιδιομορφίες.

Ομογενής Μετασχηματισμός

Ο ομογενής μετασχηματισμός $^{0}\boldsymbol{T_1}$ από το ΣΣ {1} στο ΣΣ {0} ορίζεται ως ο πίνακας 4×4 που αποτελείται από τον πίνακα περιστροφής $^{0}\boldsymbol{R_1}$ και το διάνυσμα θέσης $^{0}\boldsymbol{b}$ και μία σειρά με τρία μηδέν και μία μονάδα**:**

$$^{0}\boldsymbol{T_1} = \begin{bmatrix} & ^{0}\boldsymbol{R_1} & & ^{0}\boldsymbol{b} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

όπου:

- $^{0}\boldsymbol{R_1}$, ο προσανατολισμός του ΣΣ {1} ως προς το {0}
- $^{0}\boldsymbol{b}$, η θέση της αρχής του ΣΣ {1} ως προς το {0}
- 1, συντελεστής κλίμακας

Ο ομογενής μετασχηματισμός έχει τις εξής χρήσεις:

a) Περιγράφει τη θέση και τον προσανατολισμό του ΣΣ {1} ως προς το {0}. (Περιγράφτηκε παραπάνω)

b) Μετασχηματίζει τις συνιστώσες ενός διανύσματος θέσης στο ΣΣ {1}, στις συνιστώσες του ως προς το ΣΣ {0}

$$^{0}\boldsymbol{p} = {}^{0}\boldsymbol{b} + {}^{0}\boldsymbol{R_1}\ {}^{1}\boldsymbol{p}$$

όπου:

$^{0}\boldsymbol{p}$**:** Διάνυσμα θέσης του P ως προς το {0}, εκφρασμένο στο {0}

$^{1}\boldsymbol{p}$**:** Διάνυσμα θέσης του P ως προς το {1}, εκφρασμένο στο {1}

$^{0}\boldsymbol{b}$**:** Διάνυσμα θέσης του {1} ως προς το {0}, εκφρασμένο στο {0}

Η παραπάνω εξίσωση μπορεί να γραφτεί συμπαγώς ως:

$$^{0}\boldsymbol{p} = {}^{0}\boldsymbol{T_1}\ {}^{1}\boldsymbol{p}$$

c) Περιστρέφει ένα διάνυσμα θέσης και μετά το μετατοπίζει, δίνοντας έτσι τις συνιστώσες του τελικού διανύσματος θέσης:

$$\boldsymbol{p'} = \boldsymbol{T} \cdot \boldsymbol{p}$$

όπου **p**' και **p** πίνακες 4x4.

ΠΑΡΑΡΤΗΜΑ Β: Κώδικες

ChangeView.cs

```
using UnityEngine;
using System.Collections;

public class ChangeView : MonoBehaviour {

	private Camera secondCamera;

	void Start ()
	{
		secondCamera = GameObject.Find ("Close
Camera").GetComponent<Camera> ();
		secondCamera.camera.depth = -2;
	}

	void Update ()
	{
		if
(Input.GetKeyDown(KeyCode.JoystickButton9))
		{

	if(secondCamera.camera.depth != 1)
				{

	secondCamera.camera.depth = 1;
				}
				else
				{

	secondCamera.camera.depth = -2;
				}
		}
	}
}
```

CollisionDetection.cs

```
using UnityEngine;
using System.Collections;

public class CollisionDetection : MonoBehaviour
{
	private AudioSource collisionSound;

	private GameObject upSideU;
	private GameObject downSideU;
	private GameObject leftSideU;
	private GameObject rightSideU;
	private GameObject upFrame;

	private GameObject upSideD;
	private GameObject downSideD;
	private GameObject leftSideD;
	private GameObject rightSideD;
	private GameObject downFrame;



	void Awake()
	{
		collisionSound = GameObject.Find
("Metal Crash").GetComponent<AudioSource> ();

		upSideU = GameObject.Find ("Up Side
(U)");
		downSideU = GameObject.Find ("Down Side
(U)");
		leftSideU = GameObject.Find ("Left Side
(U)");
		rightSideU = GameObject.Find ("Right
Side (U)");
		upFrame = GameObject.Find ("Upper
Frame");

		upSideD = GameObject.Find ("Up Side
(D)");
		downSideD = GameObject.Find ("Down Side
(D)");
		leftSideD = GameObject.Find ("Left Side
(D)");
		rightSideD = GameObject.Find ("Right
Side (D)");
		downFrame = GameObject.Find ("Lower
Frame");
	}

	void OnTriggerEnter(Collider other)
	{
		if(other.gameObject.tag ==
"RobotCrashSpace")
		{
			collisionSound.Play ();
		}

	}

	void OnTriggerStay(Collider other)
	{
		if(other.gameObject.tag ==
"RobotCrashSpace")
		{
			if(other.gameObject.name ==
"Upper Window")
			{

	other.gameObject.renderer.material.SetColor("_Color
", Color.red);

	upSideU.renderer.material.SetColor("_Color",
Color.red);

	downSideU.renderer.material.SetColor("_Color",
Color.red);

	leftSideU.renderer.material.SetColor("_Color",
Color.red);

	rightSideU.renderer.material.SetColor("_Color",
Color.red);

	upFrame.renderer.material.SetColor("_Color",
Color.red);
			}
			else
if(other.gameObject.name == "Lower Window")
			{

	other.gameObject.renderer.material.SetColor("_Color
", Color.red);

	upSideD.renderer.material.SetColor("_Color",
Color.red);

	downSideD.renderer.material.SetColor("_Color",
Color.red);

	leftSideD.renderer.material.SetColor("_Color",
Color.red);

	rightSideD.renderer.material.SetColor("_Color",
Color.red);

	downFrame.renderer.material.SetColor("_Color",
Color.red);
			}
			else
			{

	other.gameObject.renderer.material.SetColor("_Color
", Color.red);
			}
		}
	}

	void OnTriggerExit(Collider other)
	{
		if(other.gameObject.tag ==
"RobotCrashSpace")
		{
			if(other.gameObject.name ==
"Upper Window")
			{

	other.gameObject.renderer.material.SetColor("_Color
", Color.white);

	upSideU.renderer.material.SetColor("_Color",
Color.white);

	downSideU.renderer.material.SetColor("_Color",
Color.white);

	leftSideU.renderer.material.SetColor("_Color",
Color.white);

	rightSideU.renderer.material.SetColor("_Color",
Color.white);

	upFrame.renderer.material.SetColor("_Color",
Color.white);
			}
			else
if(other.gameObject.name == "Lower Window")
			{

	other.gameObject.renderer.material.SetColor("_Color
", Color.white);

	upSideD.renderer.material.SetColor("_Color",
Color.white);

	downSideD.renderer.material.SetColor("_Color",
Color.white);

	leftSideD.renderer.material.SetColor("_Color",
Color.white);

	rightSideD.renderer.material.SetColor("_Color",
Color.white);

	downFrame.renderer.material.SetColor("_Color",
Color.white);
			}
			else
			{

	other.gameObject.renderer.material.SetColor("_Color
", Color.white);
			}
		}
	}

}
```

DisplayEndEffectorCoordinates.cs

```
using UnityEngine;
using System.Collections;

public class DisplayEndEffectorCoordinates : MonoBehaviour
{
            private IKMover robotMoverInstance;

            void Start()
            {
                        robotMoverInstance                    =
GameObject.Find("Staubli RX90L").GetComponent<IKMover>();
            }

            // Displays the End-Effector coordinates as printed
by the real robot
            void OnGUI()
            {
                        GUI.Box (new Rect (Screen.width - 220,
5, 200, 90),
                                    "X          =          "        +
(Mathf.Round((robotMoverInstance.endEffectorLabPosition.x)     *
100000f)/100f).ToString()+ " mm" + "\n" + "\n" +
                                    "Y          =          "        +
(Mathf.Round((robotMoverInstance.endEffectorLabPosition.y)     *
100000f)/100f).ToString()+ " mm" + "\n" + "\n" +
                                    "Z          =          "        +
(Mathf.Round((robotMoverInstance.endEffectorLabPosition.z)     *
100000f)/100f).ToString()+ " mm");
            }
}
```

IKMover.cs

```
using UnityEngine;
using System;
using System.IO;
using System.Collections;
using System.Threading;

public class IKMover : MonoBehaviour
{

            private Camera secondCamera;

            public float rotationSpeed = 15f; //Ταχυτητα Lerp
            private   float   ikSpeed   =   0.005f;   //Ταχυτητα
μετακινησης ΤΣΔ
            private   float   ikRotSpeed   =   0.5f;   //Ταχυτητα
περιστροφης ΤΣΔ

            // Οι γωνιες των αρθρωσεων συμφωνες με το Σ.Σ. του
ρομποτ. Στη συνεχεια του προγραμματος πρεπει να τις δωσω με
τροπο που να αντιστοιχουν στο
            // Σ.Σ. του unity.
            [Range (-160f,160f)] public float rotZ1 = 0;
            [Range (-227.5f,47.5f)] public float rotZ2 = -90;
            [Range (-52.5f,232.5f)] public float rotZ3 = 90;
            [Range (-270f,270f)] public float rotZ4 = 0;
            [Range (-105f,120f)] public float rotZ5 = 0;
            [Range (-270f,270f)] public float rotZ6 = 0;

            // ΟΙ θεσεις που χρησιμοποιω για να υπολογισω τη
θεση του Τ.Σ.Δ. στο Σ.Σ. του ρομποτ.
            public Vector3 endEffectorLabPosition;
            public Vector3 wristLabPosition;
            private Transform endEffectorUnityPosision;
            private Transform wristUnityPosition;
            private Transform referenceFrameUnityPosition;

            //Επιθυμητος προσανατολισμος
            private Transform desirableRot1;
            private Transform desirableRot2;
            private Transform desirableRot3;
            private Transform desirableRot4;
            public int directionCoding = 0;

            // Προηγουμενο πληκτρο που πατηθηκε
            private int prevKeyPress = 100;

            // Ηχοι Συστηματος
            private AudioSource outOfDexterousSpace;

            // Αρχειο εξοδου
            private TextWriter ppPoints;

            //Iterator για το γραψιμο του αρχειου
            private int decomposeIter = 0;

            // Οι γωνιες DO READY. Απο αυτες αλλαζω μονο τις Ζ.
Οι αλλες απαιτουνται μονο απο το Quaternion του Unity.
            private float rotX1 = -90f;
            private float rotX2 = -90f;
            private float rotX3 = 0f;
            private float rotX4 = 90f;
            private float rotX5 = -90f;
            private float rotX6 = 90f;
            private float rotYAll = 0f;
            private float initRotZ1 = 0f;
            private float initRotZ2 = -90f;
            private float initRotZ3 = 90f;
            private float initRotZ4 = 0f;
            private float initRotZ5 = 0f;
            private float initRotZ6 = 0f;

            private Transform joint1;
            private Transform joint2;
            private Transform joint3;
            private Transform joint4;
            private Transform joint5;
            private Transform joint6;

            // DO READY Rotations of joints
            private Quaternion j1Def;
            private Quaternion j2Def;
            private Quaternion j3Def;
            private Quaternion j4Def;
            private Quaternion j5Def;
            private Quaternion j6Def;

            // Target Rotations of joints
            private Quaternion j1Rot;
            private Quaternion j2Rot;
            private Quaternion j3Rot;
            private Quaternion j4Rot;
            private Quaternion j5Rot;
            private Quaternion j6Rot;

            //Πινακας περιστροφης Rx
            private float rx1;
            private float rx2;
            private float rx3;
            private float rx4;
            private float rx5;
            private float rx6;
            private float rx7;
            private float rx8;
            private float rx9;

            //Πινακας περιστροφης Ry
            private float ry1;
            private float ry3;
            private float ry4;
            private float ry6;
            private float ry7;
            private float ry9;

            //Πινακας περιστροφης Rz
            private float rz1;
            private float rz2;
            private float rz3;
            private float rz4;
            private float rz5;
            private float rz6;
            private float rz7;
            private float rz8;
            private float rz9;

            //Μεταβλητες που χρησιμοποιω απο τον πικανα T
            private float pwx;
            private float pwy;
            private float pwz;
            private float r13;
            private float r23;
            private float r33;
            private float r11;
            private float r21;
            private float r31;

            // Ολες οι πιθανες λυσεις γωνιων
            private float q11;
            private float q12;
            private float q31;
            private float q32;
            private float q33;
            private float q34;
            private float q21;
            private float q22;
            private float q23;
            private float q24;
            private float q25;
            private float q26;
            private float q27;
            private float q28;
            private float q51;
            private float q52;
            private float q53;
            private float q54;
            private float q55;
            private float q56;
            private float q57;
            private float q58;
            private float q59;
            private float q510;
            private float q511;
            private float q512;
            private float q513;
            private float q514;
            private float q515;
            private float q516;
            private float q41;
            private float q42;
            private float q43;
            private float q44;
            private float q45;
            private float q46;
            private float q47;
            private float q48;
            private float q49;
            private float q410;
            private float q411;
            private float q412;
            private float q413;
            private float q414;
            private float q415;
            private float q416;
            private float q61;
            private float q62;
            private float q63;
            private float q64;
            private float q65;
            private float q66;
            private float q67;
            private float q68;
            private float q69;
            private float q610;
```

```
private float q611;
private float q612;
private float q613;
private float q614;
private float q615;
private float q616;
private float q617;
private float q618;
private float q619;
private float q620;
private float q621;
private float q622;
private float q623;
private float q624;
private float q625;
private float q626;
private float q627;
private float q628;
private float q629;
private float q630;
private float q631;
private float q632;
private float q633;
private float q634;
private float q635;
private float q636;
private float q637;
private float q638;
private float q639;
private float q640;
private float q641;
private float q642;
private float q643;
private float q644;
private float q645;
private float q646;
private float q647;
private float q648;

//Γωνιες πριν την επομενη μετακινηση
private float q1prev;
private float q2prev;
private float q3prev;
private float q4prev;
private float q5prev;
private float q6prev;

//Ολα τα πιθανα αθροισματα
private float sum1;
private float sum2;
private float sum3;
private float sum4;
private float sum5;
private float sum6;
private float sum7;
private float sum8;
private float sum9;
private float sum10;
private float sum11;
private float sum12;
private float sum13;
private float sum14;
private float sum15;
private float sum16;
private float sum17;
private float sum18;
private float sum19;
private float sum20;
private float sum21;
private float sum22;
private float sum23;
private float sum24;
private float sum25;
private float sum26;
private float sum27;
private float sum28;
private float sum29;
private float sum30;
private float sum31;
private float sum32;
private float sum33;
private float sum34;
private float sum35;
private float sum36;
private float sum37;
private float sum38;
private float sum39;
private float sum40;
private float sum41;
private float sum42;
private float sum43;
private float sum44;
private float sum45;
private float sum46;
private float sum47;
private float sum48;
private float sum49;
private float sum50;
private float sum51;
private float sum52;
private float sum53;
private float sum54;
private float sum55;
private float sum56;
private float sum57;
private float sum58;
private float sum59;
private float sum60;
private float sum61;
private float sum62;
private float sum63;
private float sum64;
private float sum65;
private float sum66;
private float sum67;
private float sum68;
private float sum69;
private float sum70;
private float sum71;
private float sum72;
private float sum73;
private float sum74;
private float sum75;
private float sum76;
private float sum77;
private float sum78;
private float sum79;
private float sum80;
private float sum81;
private float sum82;
private float sum83;
private float sum84;
private float sum85;
private float sum86;
private float sum87;
private float sum88;
private float sum89;
private float sum90;
private float sum91;
private float sum92;
private float sum93;
private float sum94;
private float sum95;
private float sum96;
private float minSum;

//User Input
private float xAxis; //Μετατοπισεις
private float yAxis;
private float zAxis;
private float ax; //Γωνιες Euler
private float ay;
private float az;

void Awake ()
{
                joint1   =   GameObject.Find   ("Joint
1").GetComponent<Transform> ();
                joint2   =   GameObject.Find   ("Joint
2").GetComponent<Transform> ();
                joint3   =   GameObject.Find   ("Joint
3").GetComponent<Transform> ();
                joint4   =   GameObject.Find   ("Joint
4").GetComponent<Transform> ();
                joint5   =   GameObject.Find   ("Joint
5").GetComponent<Transform> ();
                joint6   =   GameObject.Find   ("Joint
6").GetComponent<Transform> ();

                endEffectorUnityPosision             =
GameObject.Find         ("(#11508)         RX90BL         TOOL
FLANGE").GetComponent<Transform> ();
                wristUnityPosition
=GameObject.Find("(#10353)                              RX90BL
WRIST").GetComponent<Transform>();
                referenceFrameUnityPosition          =
GameObject.Find ("Joint 2").GetComponent<Transform> ();

                secondCamera = GameObject.Find ("Close
Camera").GetComponent<Camera> ();

                desirableRot1     =     GameObject.Find
("Direction1").GetComponent<Transform> ();
                desirableRot2     =     GameObject.Find
("Direction2").GetComponent<Transform> ();
                desirableRot3     =     GameObject.Find
("Direction3").GetComponent<Transform> ();
                desirableRot4     =     GameObject.Find
("Direction4").GetComponent<Transform> ();

                outOfDexterousSpace                  =
GameObject.Find("Censor               Beep               Sound
Effect").GetComponent<AudioSource> ();

                ppPoints     =     new     StreamWriter
("output.txt");
                ppPoints.WriteLine("DO SPEED 20");

                j1Def   =   Quaternion.Euler   (rotX1,
rotYAll, initRotZ1);
                j2Def   =   Quaternion.Euler   (rotX2,
rotYAll, initRotZ2);
                j3Def   =   Quaternion.Euler   (rotX3,
rotYAll, initRotZ3);
                j4Def   =   Quaternion.Euler   (rotX4,
rotYAll, initRotZ4);
                j5Def   =   Quaternion.Euler   (rotX5,
rotYAll, initRotZ5);
                j6Def   =   Quaternion.Euler   (rotX6,
rotYAll, initRotZ6);
        }

        void Start()
        {
                //  Initializes  robot  to  "Do  Ready"
position
                joint1.localRotation = j1Def;
                joint2.localRotation = j2Def;
                joint3.localRotation = j3Def;
                joint4.localRotation = j4Def;
                joint5.localRotation = j5Def;
                joint6.localRotation = j6Def;

                q1prev = initRotZ1;
                q2prev = initRotZ2;
                q3prev = initRotZ3;
                q4prev = initRotZ4;
                q5prev = initRotZ5;
                q6prev = initRotZ6;

                pwx = 0f;
                pwy = 0f;
                pwz = 1.1f;
                ax = 0f;
                ay = 0f;
                az = 0f;

                minSum = 0f;

                endEffectorLabPosition               =
CoordinateSystemChange     (endEffectorUnityPosision.position)-
CoordinateSystemChange (referenceFrameUnityPosition.position);
```

```
                        wristLabPosition                        =
CoordinateSystemChange            (wristUnityPosition.position)-
CoordinateSystemChange (referenceFrameUnityPosition.position);
            }


            void Update()
            {

                        //Δινω  τα  περιεχομενα  του  πινακα
περιστροφης
                        //***Παραγοντες μετακινησης του καρπου
κατα X,Y,Z***//
                        zAxis  =  Input.GetAxis  ("7th_Axis")  *
ikSpeed;
                        xAxis = Input.GetAxis ("Horizontal") *
ikSpeed;
                        yAxis  =  Input.GetAxis  ("Vertical")  *
ikSpeed;

                        if  (secondCamera.camera.depth  ==  -2)
//Αλλαζω τα κουμπια για να ταιριαζουν στο καθε view
                        {
                                    if (zAxis != 0f)
                                    {
                                                if(!(minSum  ==
Mathf.Infinity))
                                                {
                                                            pwz
-= zAxis;
                                                }
                                                else
                                                {
                                                            pwz
+= zAxis;
            outOfDexterousSpace.Play();

            Thread.Sleep (500);
                                                }
                                    }

                                    if (yAxis != 0f)
                                    {
                                                if(!(minSum  ==
Mathf.Infinity))
                                                {
                                                            pwx
+= yAxis;
                                                }
                                                else
                                                {
                                                            pwx
-= yAxis;
            outOfDexterousSpace.Play();

            Thread.Sleep (500);
                                                }
                                    }

                                    if (xAxis != 0f)
                                    {
                                                if(!(minSum  ==
Mathf.Infinity))
                                                {
                                                            pwy
-= xAxis;
                                                }
                                                else
                                                {
                                                            pwy
+= xAxis;
            outOfDexterousSpace.Play();

            Thread.Sleep (500);
                                                }
                                    }
                        }
                        else  if  (secondCamera.camera.depth  ==
1)
                        {
                                    switch (directionCoding)
                                    {
                                    case 1:
                                                if   (zAxis   !=
0f)
                                                {

            if(!(minSum == Mathf.Infinity))
                                                            {

            pwz -= zAxis;
                                                            }

            else
                                                            {

            pwz += zAxis;

            outOfDexterousSpace.Play();

            Thread.Sleep (500);
                                                            }
                                                }

                                                if   (yAxis   !=
0f)
                                                {

            if(!(minSum == Mathf.Infinity))
                                                            {

            pwx += yAxis;
                                                            }

            else
                                                            {

            pwx -= yAxis;

            outOfDexterousSpace.Play();

            Thread.Sleep (500);
                                                            }
                                                }

                                                if   (xAxis   !=
0f)
                                                {

            if(!(minSum == Mathf.Infinity))
                                                            {

            pwy -= xAxis;
                                                            }

            else
                                                            {

            pwy += xAxis;

            outOfDexterousSpace.Play();

            Thread.Sleep (500);
                                                            }
                                                }
                                                break;

                                    case 2:
                                                if   (zAxis   !=
0f)
                                                {

            if(!(minSum == Mathf.Infinity))
                                                            {

            pwz -= zAxis;
                                                            }

            else
                                                            {

            pwz += zAxis;

            outOfDexterousSpace.Play();

            Thread.Sleep (500);
                                                            }
                                                }

                                                if   (xAxis   !=
0f)
                                                {

            if(!(minSum == Mathf.Infinity))
                                                            {

            pwx += xAxis;
                                                            }

            else
                                                            {

            pwx -= xAxis;

            outOfDexterousSpace.Play();

            Thread.Sleep (500);
                                                            }
                                                }

                                                if   (yAxis   !=
0f)
                                                {

            if(!(minSum == Mathf.Infinity))
                                                            {

            pwy += yAxis;
                                                            }

            else
                                                            {

            pwy -= yAxis;

            outOfDexterousSpace.Play();

            Thread.Sleep (500);
                                                            }
                                                }
                                                break;

                                    case 3:
                                                if   (zAxis   !=
0f)
                                                {

            if(!(minSum == Mathf.Infinity))
                                                            {

            pwz -= zAxis;
                                                            }

            else
                                                            {

            pwz += zAxis;

            outOfDexterousSpace.Play();

            Thread.Sleep (500);
                                                            }
                                                }

                                                if   (yAxis   !=
0f)
                                                {

            if(!(minSum == Mathf.Infinity))
                                                            {
```

```
            pwx += yAxis;
                                                }
            else
                                                {
            pwx -= yAxis;
            outOfDexterousSpace.Play();
            Thread.Sleep (500);
                                                }
                                            }
                                            if   (xAxis   !=
0f)
                                            {
            if(!(minSum == Mathf.Infinity))
                                                {
            pwy -= xAxis;
                                                }
            else
                                                {
            pwy += xAxis;
            outOfDexterousSpace.Play();
            Thread.Sleep (500);
                                                }
                                            }
                                            break;
                                case 4:
                                            if   (zAxis   !=
0f)
                                            {
            if(!(minSum == Mathf.Infinity))
                                                {
            pwz -= zAxis;
                                                }
            else
                                                {
            pwz += zAxis;
            outOfDexterousSpace.Play();
            Thread.Sleep (500);
                                                }
                                            }
                                            if   (xAxis   !=
0f)
                                            {
            if(!(minSum == Mathf.Infinity))
                                                {
            pwx += xAxis;
                                                }
            else
                                                {
            pwx -= xAxis;
            outOfDexterousSpace.Play();
            Thread.Sleep (500);
                                                }
                                            }
                                            if   (yAxis   !=
0f)
                                            {
            if(!(minSum == Mathf.Infinity))
                                                {
            pwy += yAxis;
                                                }
            else
                                                {
            pwy -= yAxis;
            outOfDexterousSpace.Play();
            Thread.Sleep (500);
                                                }
                                            }
                                            break;
                                default:
                                            if   (zAxis   !=
0f)
                                            {
            if(!(minSum == Mathf.Infinity))
                                                {
            pwz -= zAxis;
                                                }
            else
                                                {
            pwz += zAxis;
            outOfDexterousSpace.Play();
            Thread.Sleep (500);
                                                }
                                            }
                                            if   (yAxis   !=
0f)
                                            {
            if(!(minSum == Mathf.Infinity))
                                                {
            pwx += yAxis;
                                                }
            else
                                                {
            pwx -= yAxis;
            outOfDexterousSpace.Play();
            Thread.Sleep (500);
                                                }
                                            }
                                            if   (xAxis   !=
0f)
                                            {
            if(!(minSum == Mathf.Infinity))
                                                {
            pwy -= xAxis;
                                                }
            else
                                                {
            pwy += xAxis;
            outOfDexterousSpace.Play();
            Thread.Sleep (500);
                                                }
                                            }
                                            break;
                                }
                                            if   (yAxis   !=
0f)
                                            {
            if(!(minSum == Mathf.Infinity))
                                                {
            pwx += yAxis;
                                                }
            else
                                                {
            pwx -= yAxis;
            outOfDexterousSpace.Play();
            Thread.Sleep (500);
                                                }
                                            }
                                            if   (xAxis   !=
0f)
                                            {
            if(!(minSum == Mathf.Infinity))
                                                {
            pwy -= xAxis;
                                                }
            else
                                                {
            pwy += xAxis;
            outOfDexterousSpace.Play();
            Thread.Sleep (500);
                                                }
                                            }
                                            break;
                                case 4:
                                            if   (zAxis   !=
0f)
                                            {
            if(!(minSum == Mathf.Infinity))
                                                {
            pwz -= zAxis;
                                                }
            else
                                                {
            pwz += zAxis;
            outOfDexterousSpace.Play();
            Thread.Sleep (500);
                                                }
                                            }
                                            if   (xAxis   !=
0f)
                                            {
            if(!(minSum == Mathf.Infinity))
                                                {
            pwx += xAxis;
                                                }
            else
                                                {
            pwx -= xAxis;
            outOfDexterousSpace.Play();
            Thread.Sleep (500);
                                                }
                                            }
                                            if   (yAxis   !=
0f)
                                            {
            if(!(minSum == Mathf.Infinity))
                                                {
            pwy += yAxis;
                                                }
            else
                                                {
            pwy -= yAxis;
            outOfDexterousSpace.Play();
            Thread.Sleep (500);
                                                }
                                            }
                                            break;
                                default:
                                            if   (zAxis   !=
0f)
                                            {
            if(!(minSum == Mathf.Infinity))
                                                {
            pwz -= zAxis;
                                                }
            else
                                                {
            pwz += zAxis;
            outOfDexterousSpace.Play();
            Thread.Sleep (500);
                                                }
                                            }
                                            if   (yAxis   !=
0f)
                                            {
            if(!(minSum == Mathf.Infinity))
                                                {
            pwx += yAxis;
                                                }
            else
                                                {
            pwx -= yAxis;
            outOfDexterousSpace.Play();
            Thread.Sleep (500);
                                                }
                                            }
                                            if   (xAxis   !=
0f)
                                            {
            if(!(minSum == Mathf.Infinity))
                                                {
            pwy -= xAxis;
                                                }
            else
                                                {
            pwy += xAxis;
            outOfDexterousSpace.Play();
            Thread.Sleep (500);
                                                }
                                            }
                                            break;
                                }
                        }
                        //***Παραγοντες     περιστροφης     του
ακρου***//
                        if                      (!Input.GetKey
(KeyCode.JoystickButton4))
                        {
                                if              (Input.GetKey
(KeyCode.JoystickButton1))
                                {
                                            directionCoding
= 0;
                                            if (!(minSum ==
Mathf.Infinity && prevKeyPress == 1))
                                            {
                                                        ax
= ax + ikRotSpeed;
                                                prevKeyPress = 1;
                                            }
                                            else
                                            {
                                                        ax
= ax - ikRotSpeed;
                                                outOfDexterousSpace.Play();
                                                Thread.Sleep (500);
                                            }
                                }
                                if              (Input.GetKey
(KeyCode.JoystickButton0))
                                {
                                            directionCoding
= 0;
                                            if (!(minSum ==
Mathf.Infinity && prevKeyPress == 0))
                                            {
                                                        ay
= ay + ikRotSpeed;
                                                prevKeyPress = 0;
                                            }
                                            else
                                            {
                                                        ay
= ay - ikRotSpeed;
                                                outOfDexterousSpace.Play();
                                                Thread.Sleep (500);
                                            }
                                }
                                if              (Input.GetKey
(KeyCode.JoystickButton2))
                                {
                                            directionCoding
= 0;
                                            if (!(minSum ==
Mathf.Infinity && prevKeyPress == 2))
                                            {
                                                        az
= az + ikRotSpeed;
                                                prevKeyPress = 2;
                                            }
                                            else
                                            {
                                                        az
= az - ikRotSpeed;
                                                outOfDexterousSpace.Play();
```

```
            Thread.Sleep (500);
                                    }
                                }
                        }

                        else          if          (Input.GetKey
(KeyCode.JoystickButton4))
                        {
                                if            (Input.GetKey
(KeyCode.JoystickButton1))
                                {
                                        directionCoding
= 0;

                                        if (!(minSum ==
Mathf.Infinity && prevKeyPress == 10))
                                        {
                                                ax
= ax - ikRotSpeed;

            prevKeyPress = 10;
                                        }
                                        else
                                        {
                                                ax
= ax + ikRotSpeed;

            outOfDexterousSpace.Play();

            Thread.Sleep (500);
                                        }
                                }

                                if            (Input.GetKey
(KeyCode.JoystickButton0))
                                {
                                        directionCoding
= 0;

                                        if (!(minSum ==
Mathf.Infinity && prevKeyPress == 11))
                                        {
                                                ay
= ay - ikRotSpeed;

            prevKeyPress = 11;
                                        }
                                        else
                                        {
                                                ay
= ay + ikRotSpeed;

            outOfDexterousSpace.Play();

            Thread.Sleep (500);
                                        }
                                }

                                if            (Input.GetKey
(KeyCode.JoystickButton2))
                                {
                                        directionCoding
= 0;

                                        if (!(minSum ==
Mathf.Infinity && prevKeyPress == 12))
                                        {
                                                az
= az - ikRotSpeed;

            prevKeyPress = 12;
                                        }
                                        else
                                        {
                                                az
= az + ikRotSpeed;

            outOfDexterousSpace.Play();

            Thread.Sleep (500);
                                        }
                                }

                        if                      (Input.GetKeyDown
(KeyCode.JoystickButton3))    //    Αλλαζει    τον    επιθυμητο
προσανατολισμο κυκλικα
                        {
                                if(directionCoding < 4)
                                {
                                        directionCoding
+=1;
                                }
                                else
                                {
                                        directionCoding
= 1;
                                }

                        }
                        else        if        (Input.GetKeyDown
(KeyCode.JoystickButton6))
                        {
                                if(directionCoding > 1)
                                {
                                        directionCoding
--;
                                }
                                else
                                {
                                        directionCoding
= 4;
                                }
                        }


                        switch  (directionCoding)  //  Παιρνει
τους 4 επιθυμητους προσανατολισμους
                        {
                                case 1:
                                        ax            =
desirableRot1.rotation.eulerAngles.x;
                                        ay            =
desirableRot1.rotation.eulerAngles.z;
                                        az     =     -
desirableRot1.rotation.eulerAngles.y;
                                        break;
                                case 2:
                                        ax            =
desirableRot2.rotation.eulerAngles.x;
                                        ay            =
desirableRot2.rotation.eulerAngles.z;
                                        az     =     -
desirableRot2.rotation.eulerAngles.y;
                                        break;
                                case 3:
                                        ax            =
desirableRot3.rotation.eulerAngles.x;
                                        ay            =
desirableRot3.rotation.eulerAngles.z;
                                        az     =     -
desirableRot3.rotation.eulerAngles.y;
                                        break;
                                case 4:
                                        ax            =
desirableRot4.rotation.eulerAngles.x;
                                        ay            =
desirableRot4.rotation.eulerAngles.z;
                                        az     =     -
desirableRot4.rotation.eulerAngles.y;
                                        break;
                        }

                        //Σχηματισμος Πινακα Rx
                        rx1 = 1f;
                        rx2 = 0f;
                        rx3 = 0f;
                        rx4 = 0f;
                        rx5 = Mathf.Cos (ax * Mathf.Deg2Rad);
                        rx6 = -Mathf.Sin (ax * Mathf.Deg2Rad);
                        rx7 = 0f;
                        rx8 = Mathf.Sin (ax * Mathf.Deg2Rad);
                        rx9 = Mathf.Cos (ax * Mathf.Deg2Rad);

                        //Σχηματισμος Πινακα Ry
                        ry1 = Mathf.Cos (ay * Mathf.Deg2Rad);
                        ry3 = Mathf.Sin (ay * Mathf.Deg2Rad);
                        ry4 = 0f;
                        ry6 = 0f;
                        ry7 = -Mathf.Sin (ay * Mathf.Deg2Rad);
                        ry9 = Mathf.Cos (ay * Mathf.Deg2Rad);

                        //Σχηματισμος Πινακα Rz
                        rz1 = Mathf.Cos (az * Mathf.Deg2Rad);
                        rz2 = -Mathf.Sin (az * Mathf.Deg2Rad);
                        rz3 = 0f;
                        rz4 = Mathf.Sin (az * Mathf.Deg2Rad);
                        rz5 = Mathf.Cos (az * Mathf.Deg2Rad);
                        rz6 = 0f;
                        rz7 = 0f;
                        rz8 = 0f;
                        rz9 = 1f;

                        //Στοιχεια τελικου πινακα περιστροφης
                        r13 = ry3 * (rx1 * rz1 + rx4 * rz2 +
rx7 * rz3) + ry6 * (rx2 * rz1 + rx5 * rz2 + rx8 * rz3) + ry9 *
(rx3 * rz1 + rx6 * rz2 + rx9 * rz3);
                        r23 = ry3 * (rx1 * rz4 + rx4 * rz5 +
rx7 * rz6) + ry6 * (rx2 * rz4 + rx5 * rz5 + rx8 * rz6) + ry9 *
(rx3 * rz4 + rx6 * rz5 + rx9 * rz6);
                        r33 = ry3 * (rx1 * rz7 + rx4 * rz8 +
rx7 * rz9) + ry6 * (rx2 * rz7 + rx5 * rz8 + rx8 * rz9) + ry9 *
(rx3 * rz7 + rx6 * rz8 + rx9 * rz9);
                        r11 = ry1 * (rx1 * rz1 + rx4 * rz2 +
rx7 * rz3) + ry4 * (rx2 * rz1 + rx5 * rz2 + rx8 * rz3) + ry7 *
(rx3 * rz1 + rx6 * rz2 + rx9 * rz3);
                        r21 = ry1 * (rx1 * rz4 + rx4 * rz5 +
rx7 * rz6) + ry4 * (rx2 * rz4 + rx5 * rz5 + rx8 * rz6) + ry7 *
(rx3 * rz4 + rx6 * rz5 + rx9 * rz6);
                        r31 = ry1 * (rx1 * rz7 + rx4 * rz8 +
rx7 * rz9) + ry4 * (rx2 * rz7 + rx5 * rz8 + rx8 * rz9) + ry7 *
(rx3 * rz7 + rx6 * rz8 + rx9 * rz9);


                        //***  Υπολογισμος  ολων  των  πιθανων
γωνιων***//
                        // Υπολογισμος q1
                        q11 = Mathf.Rad2Deg * Mathf.Atan2 (pwy,
pwx);
                        q12  =  Mathf.Rad2Deg  *  Mathf.Atan2  (-
pwy, -pwx);

                        // Υπολογισμος q3
                        q31   =   Mathf.Rad2Deg   *   Mathf.Asin
(Mathf.Round  (((Mathf.Pow(pwx,  2f)  +  Mathf.Pow(pwy,  2f)  +
Mathf.Pow(pwz, 2f) - Mathf.Pow(0.45f, 2f)

- Mathf.Pow(0.65f, 2f)) / (2f * 0.45f * 0.65f))*1000f)/1000f);
                        q32 =  q31;
                        q33 = 180f - Mathf.Rad2Deg * Mathf.Asin
(Mathf.Round  (((Mathf.Pow(pwx,  2f)  +  Mathf.Pow(pwy,  2f)  +
Mathf.Pow(pwz, 2f)

- Mathf.Pow(0.45f, 2f) - Mathf.Pow(0.65f, 2f)) / (2f * 0.45f *
0.65f))*1000f)/1000f);
                        q34 = 180f + Mathf.Rad2Deg * Mathf.Asin
(Mathf.Round  (((Mathf.Pow(pwx,  2f)  +  Mathf.Pow(pwy,  2f)  +
Mathf.Pow(pwz, 2f)

- Mathf.Pow(0.45f, 2f) - Mathf.Pow(0.65f, 2f)) / (2f * 0.45f *
0.65f))*1000f)/1000f);

                        // Υπολογισμος q2
                        q21 = (Mathf.Rad2Deg * Mathf.Atan2 (-
0.45f * Mathf.Cos (q31 * Mathf.Deg2Rad) * pwz + (0.65f + 0.45f
* Mathf.Sin (q31 * Mathf.Deg2Rad)) *

                (Mathf.Cos (q11 * Mathf.Deg2Rad) * pwx + Mathf.Sin
(q11 * Mathf.Deg2Rad) * pwy),

(0.65f + 0.45f * Mathf.Sin (q31 * Mathf.Deg2Rad)) * pwz +
```

```
            0.45f  *  Mathf.Cos  (q31  *  Mathf.Deg2Rad)  *
(Mathf.Cos  (q11  *  Mathf.Deg2Rad)  *  pwx  +  Mathf.Sin  (q11  *
Mathf.Deg2Rad) * pwy))) - q31;

                        q22 = (Mathf.Rad2Deg * Mathf.Atan2 (-
0.45f * Mathf.Cos (q32 * Mathf.Deg2Rad) * pwz + (0.65f + 0.45f
* Mathf.Sin (q32 * Mathf.Deg2Rad)) *

(Mathf.Cos  (q11  *  Mathf.Deg2Rad)  *  pwx  +  Mathf.Sin  (q11  *
Mathf.Deg2Rad) * pwy),

(0.65f + 0.45f * Mathf.Sin (q32 * Mathf.Deg2Rad)) * pwz +

0.45f * Mathf.Cos (q32 * Mathf.Deg2Rad) * (Mathf.Cos (q11 *
Mathf.Deg2Rad) * pwx + Mathf.Sin (q11 * Mathf.Deg2Rad) * pwy)))
- q32;

                        q23 = (Mathf.Rad2Deg * Mathf.Atan2 (-
0.45f * Mathf.Cos (q33 * Mathf.Deg2Rad) * pwz + (0.65f + 0.45f
* Mathf.Sin (q33 * Mathf.Deg2Rad)) *

(Mathf.Cos  (q11  *  Mathf.Deg2Rad)  *  pwx  +  Mathf.Sin  (q11  *
Mathf.Deg2Rad) * pwy),

(0.65f + 0.45f * Mathf.Sin (q33 * Mathf.Deg2Rad)) * pwz +

0.45f * Mathf.Cos (q33 * Mathf.Deg2Rad) * (Mathf.Cos (q11 *
Mathf.Deg2Rad) * pwx + Mathf.Sin (q11 * Mathf.Deg2Rad) * pwy)))
- q33;

                        q24 = (Mathf.Rad2Deg * Mathf.Atan2 (-
0.45f * Mathf.Cos (q34 * Mathf.Deg2Rad) * pwz + (0.65f + 0.45f
* Mathf.Sin (q34 * Mathf.Deg2Rad)) *

(Mathf.Cos  (q11  *  Mathf.Deg2Rad)  *  pwx  +  Mathf.Sin  (q11  *
Mathf.Deg2Rad) * pwy),

(0.65f + 0.45f * Mathf.Sin (q34 * Mathf.Deg2Rad)) * pwz +

0.45f * Mathf.Cos (q34 * Mathf.Deg2Rad) * (Mathf.Cos (q11 *
Mathf.Deg2Rad) * pwx + Mathf.Sin (q11 * Mathf.Deg2Rad) * pwy)))
- q34;

                        q25 = (Mathf.Rad2Deg * Mathf.Atan2 (-
0.45f * Mathf.Cos (q31 * Mathf.Deg2Rad) * pwz + (0.65f + 0.45f
* Mathf.Sin (q31 * Mathf.Deg2Rad)) *

(Mathf.Cos  (q12  *  Mathf.Deg2Rad)  *  pwx  +  Mathf.Sin  (q12  *
Mathf.Deg2Rad) * pwy),

(0.65f + 0.45f * Mathf.Sin (q31 * Mathf.Deg2Rad)) * pwz +

0.45f * Mathf.Cos (q31 * Mathf.Deg2Rad) * (Mathf.Cos (q12 *
Mathf.Deg2Rad) * pwx + Mathf.Sin (q12 * Mathf.Deg2Rad) * pwy)))
- q31;

                        q26 = (Mathf.Rad2Deg * Mathf.Atan2 (-
0.45f * Mathf.Cos (q32 * Mathf.Deg2Rad) * pwz + (0.65f + 0.45f
* Mathf.Sin (q32 * Mathf.Deg2Rad)) *

(Mathf.Cos  (q12  *  Mathf.Deg2Rad)  *  pwx  +  Mathf.Sin  (q12  *
Mathf.Deg2Rad) * pwy),

(0.65f + 0.45f * Mathf.Sin (q32 * Mathf.Deg2Rad)) * pwz +

0.45f * Mathf.Cos (q32 * Mathf.Deg2Rad) * (Mathf.Cos (q12 *
Mathf.Deg2Rad) * pwx + Mathf.Sin (q12 * Mathf.Deg2Rad) * pwy)))
- q32;

                        q27 = (Mathf.Rad2Deg * Mathf.Atan2 (-
0.45f * Mathf.Cos (q33 * Mathf.Deg2Rad) * pwz + (0.65f + 0.45f
* Mathf.Sin (q33 * Mathf.Deg2Rad)) *

(Mathf.Cos  (q12  *  Mathf.Deg2Rad)  *  pwx  +  Mathf.Sin  (q12  *
Mathf.Deg2Rad) * pwy),

(0.65f + 0.45f * Mathf.Sin (q33 * Mathf.Deg2Rad)) * pwz +

0.45f * Mathf.Cos (q33 * Mathf.Deg2Rad) * (Mathf.Cos (q12 *
Mathf.Deg2Rad) * pwx + Mathf.Sin (q12 * Mathf.Deg2Rad) * pwy)))
- q33;

                        q28 = (Mathf.Rad2Deg * Mathf.Atan2 (-
0.45f * Mathf.Cos (q34 * Mathf.Deg2Rad) * pwz + (0.65f + 0.45f
* Mathf.Sin (q34 * Mathf.Deg2Rad)) *

(Mathf.Cos  (q12  *  Mathf.Deg2Rad)  *  pwx  +  Mathf.Sin  (q12  *
Mathf.Deg2Rad) * pwy),

(0.65f + 0.45f * Mathf.Sin (q34 * Mathf.Deg2Rad)) * pwz +

0.45f * Mathf.Cos (q34 * Mathf.Deg2Rad) * (Mathf.Cos (q12 *
Mathf.Deg2Rad) * pwx + Mathf.Sin (q12 * Mathf.Deg2Rad) * pwy)))
- q34;

                        //Υπολογισμος q5
                        q51  =  Mathf.Rad2Deg  *  Mathf.Acos
(Mathf.Round     ((Mathf.Cos     (q11*Mathf.Deg2Rad)*Mathf.Sin
(q21*Mathf.Deg2Rad + q31*Mathf.Deg2Rad)*r13 +

Mathf.Sin  (q11*Mathf.Deg2Rad)*Mathf.Sin  (q21*Mathf.Deg2Rad  +
q31*Mathf.Deg2Rad)*r23 +

Mathf.Cos                (q21*Mathf.Deg2Rad                +
q31*Mathf.Deg2Rad)*r33)*1000f)/1000f);
                        q52 = -q51;

                        q53  =  Mathf.Rad2Deg  *  Mathf.Acos
(Mathf.Round     ((Mathf.Cos     (q11*Mathf.Deg2Rad)*Mathf.Sin
(q22*Mathf.Deg2Rad + q32*Mathf.Deg2Rad)*r13 +

Mathf.Sin  (q11*Mathf.Deg2Rad)*Mathf.Sin  (q22*Mathf.Deg2Rad  +
q32*Mathf.Deg2Rad)*r23 +

Mathf.Cos                (q22*Mathf.Deg2Rad                +
q32*Mathf.Deg2Rad)*r33)*1000f)/1000f);
                        q54 = -q53;

                        q55  =  Mathf.Rad2Deg  *  Mathf.Acos
(Mathf.Round     ((Mathf.Cos     (q11*Mathf.Deg2Rad)*Mathf.Sin
(q23*Mathf.Deg2Rad + q33*Mathf.Deg2Rad)*r13 +

Mathf.Sin  (q11*Mathf.Deg2Rad)*Mathf.Sin  (q23*Mathf.Deg2Rad  +
q33*Mathf.Deg2Rad)*r23 +

Mathf.Cos                (q23*Mathf.Deg2Rad                +
q33*Mathf.Deg2Rad)*r33)*1000f)/1000f);

                        q56 = -q55;

                        q57  =  Mathf.Rad2Deg  *  Mathf.Acos
(Mathf.Round     ((Mathf.Cos     (q11*Mathf.Deg2Rad)*Mathf.Sin
(q24*Mathf.Deg2Rad + q34*Mathf.Deg2Rad)*r13 +

Mathf.Sin  (q11*Mathf.Deg2Rad)*Mathf.Sin  (q24*Mathf.Deg2Rad  +
q34*Mathf.Deg2Rad)*r23 +

Mathf.Cos                (q24*Mathf.Deg2Rad                +
q34*Mathf.Deg2Rad)*r33)*1000f)/1000f);

                        q58 = -q57;

                        q59  =  Mathf.Rad2Deg  *  Mathf.Acos
(Mathf.Round     ((Mathf.Cos     (q12*Mathf.Deg2Rad)*Mathf.Sin
(q25*Mathf.Deg2Rad + q31*Mathf.Deg2Rad)*r13 +

Mathf.Sin  (q12*Mathf.Deg2Rad)*Mathf.Sin  (q25*Mathf.Deg2Rad  +
q31*Mathf.Deg2Rad)*r23 +

Mathf.Cos                (q25*Mathf.Deg2Rad                +
q31*Mathf.Deg2Rad)*r33)*1000f)/1000f);
                        q510 = -q59;

                        q511  =  Mathf.Rad2Deg  *  Mathf.Acos
(Mathf.Round     ((Mathf.Cos     (q12*Mathf.Deg2Rad)*Mathf.Sin
(q26*Mathf.Deg2Rad + q32*Mathf.Deg2Rad)*r13 +

Mathf.Sin  (q12*Mathf.Deg2Rad)*Mathf.Sin  (q26*Mathf.Deg2Rad  +
q32*Mathf.Deg2Rad)*r23 +

Mathf.Cos                (q26*Mathf.Deg2Rad                +
q32*Mathf.Deg2Rad)*r33)*1000f)/1000f);

                        q512 = -q511;

                        q513  =  Mathf.Rad2Deg  *  Mathf.Acos
(Mathf.Round     ((Mathf.Cos     (q12*Mathf.Deg2Rad)*Mathf.Sin
(q27*Mathf.Deg2Rad + q33*Mathf.Deg2Rad)*r13 +

Mathf.Sin  (q12*Mathf.Deg2Rad)*Mathf.Sin  (q27*Mathf.Deg2Rad  +
q33*Mathf.Deg2Rad)*r23 +

Mathf.Cos                (q27*Mathf.Deg2Rad                +
q33*Mathf.Deg2Rad)*r33)*1000f)/1000f);

                        q514 = -q513;

                        q515  =  Mathf.Rad2Deg  *  Mathf.Acos
(Mathf.Round     ((Mathf.Cos     (q12*Mathf.Deg2Rad)*Mathf.Sin
(q28*Mathf.Deg2Rad + q34*Mathf.Deg2Rad)*r13 +

Mathf.Sin  (q12*Mathf.Deg2Rad)*Mathf.Sin  (q28*Mathf.Deg2Rad  +
q34*Mathf.Deg2Rad)*r23 +

Mathf.Cos                (q28*Mathf.Deg2Rad                +
q34*Mathf.Deg2Rad)*r33)*1000f)/1000f);

                        q516 = -q515;


                        // Υπολογισμος q4
                        if (Mathf.Sin (q51) == 0f)
                        {
                                    q41 = q4prev;
                                    q42 = q4prev;
                        }
                        else
                        {
                                    q41   =   Mathf.Rad2Deg   *
(Mathf.Atan2(-Mathf.Sin   (q11*Mathf.Deg2Rad)*r13   +   Mathf.Cos
(q11 * Mathf.Deg2Rad)*r23,

Mathf.Cos  (q11*Mathf.Deg2Rad)*Mathf.Cos  (q21*Mathf.Deg2Rad  +
q31*Mathf.Deg2Rad)*r13 +

Mathf.Sin  (q11*Mathf.Deg2Rad)*Mathf.Cos  (q21*Mathf.Deg2Rad  +
q31*Mathf.Deg2Rad)*r23 -

Mathf.Sin (q21*Mathf.Deg2Rad + q31*Mathf.Deg2Rad)*r33));
                                    if  (q41  <=  90f  &&  q41  >=
0f)
                                    {
                                                q42   =   q41   +
180f;
                                    }
                                    else if (q41 <= 0f && q41
>= -90f)
                                    {
                                                q42   =   q41   -
180f;
                                    }
                                    else
                                    {
                                                q42 = q41;
                                    }
                        }

                        if (Mathf.Sin (q53) == 0f)
                        {
                                    q43 = q4prev;
                                    q44 = q4prev;
                        }
                        else
                        {
                                    q43   =   Mathf.Rad2Deg   *
(Mathf.Atan2(-Mathf.Sin   (q11*Mathf.Deg2Rad)*r13   +   Mathf.Cos
(q11 * Mathf.Deg2Rad)*r23,

Mathf.Cos  (q11*Mathf.Deg2Rad)*Mathf.Cos  (q22*Mathf.Deg2Rad  +
q32*Mathf.Deg2Rad)*r13 +

Mathf.Sin  (q11*Mathf.Deg2Rad)*Mathf.Cos  (q22*Mathf.Deg2Rad  +
q32*Mathf.Deg2Rad)*r23 -
```

```
Mathf.Sin (q22*Mathf.Deg2Rad + q32*Mathf.Deg2Rad)*r33));
                                    if (q43 <= 90f && q43 >=
0f)
                                    {
                                                q44  =  q43  +
180f;
                                    }
                                    else if (q43 <= 0f && q43
>= -90f)
                                    {
                                                q44  =  q43  -
180f;
                                    }
                                    else
                                    {
                                                q44 = q43;
                                    }
                        }

                        if (Mathf.Sin (q55) == 0f)
                        {
                                    q45 = q4prev;
                                    q46 = q4prev;
                        }
                        else
                        {
                                    q45  =  Mathf.Rad2Deg  *
(Mathf.Atan2(-Mathf.Sin  (q11*Mathf.Deg2Rad)*r13  +  Mathf.Cos
(q11 * Mathf.Deg2Rad)*r23,

Mathf.Cos  (q11*Mathf.Deg2Rad)*Mathf.Cos  (q23*Mathf.Deg2Rad  +
q33*Mathf.Deg2Rad)*r13 +

Mathf.Sin  (q11*Mathf.Deg2Rad)*Mathf.Cos  (q23*Mathf.Deg2Rad  +
q33*Mathf.Deg2Rad)*r23 -

Mathf.Sin (q23*Mathf.Deg2Rad + q33*Mathf.Deg2Rad)*r33));
                                    if (q45 <= 90f && q45 >=
0f)
                                    {
                                                q46  =  q45  +
180f;
                                    }
                                    else if (q45 <= 0f && q45
>= -90f)
                                    {
                                                q46  =  q45  -
180f;
                                    }
                                    else
                                    {
                                                q46 = q45;
                                    }
                        }

                        if (Mathf.Sin (q57) == 0f)
                        {
                                    q47 = q4prev;
                                    q48 = q4prev;
                        }
                        else
                        {
                                    q47  =  Mathf.Rad2Deg  *
(Mathf.Atan2(-Mathf.Sin  (q11*Mathf.Deg2Rad)*r13  +  Mathf.Cos
(q11 * Mathf.Deg2Rad)*r23,

Mathf.Cos  (q11*Mathf.Deg2Rad)*Mathf.Cos  (q24*Mathf.Deg2Rad  +
q34*Mathf.Deg2Rad)*r13 +

Mathf.Sin  (q11*Mathf.Deg2Rad)*Mathf.Cos  (q24*Mathf.Deg2Rad  +
q34*Mathf.Deg2Rad)*r23 -

Mathf.Sin (q24*Mathf.Deg2Rad + q34*Mathf.Deg2Rad)*r33));
                                    if (q47 <= 90f && q47 >=
0f)
                                    {
                                                q48  =  q47  +
180f;
                                    }
                                    else if (q47 <= 0f && q47
>= -90f)
                                    {
                                                q48  =  q47  -
180f;
                                    }
                                    else
                                    {
                                                q48 = q47;
                                    }
                        }

                        if (Mathf.Sin (q59) == 0f)
                        {
                                    q49 = q4prev;
                                    q410 = q4prev;
                        }
                        else
                        {
                                    q49  =  Mathf.Rad2Deg  *
(Mathf.Atan2(-Mathf.Sin  (q12*Mathf.Deg2Rad)*r13  +  Mathf.Cos
(q12 * Mathf.Deg2Rad)*r23,

Mathf.Cos  (q12*Mathf.Deg2Rad)*Mathf.Cos  (q25*Mathf.Deg2Rad  +
q31*Mathf.Deg2Rad)*r13 +

Mathf.Sin  (q12*Mathf.Deg2Rad)*Mathf.Cos  (q25*Mathf.Deg2Rad  +
q31*Mathf.Deg2Rad)*r23 -

Mathf.Sin (q25*Mathf.Deg2Rad + q31*Mathf.Deg2Rad)*r33));
                                    if (q49 <= 90f && q49 >=
0f)
                                    {
                                                q410  =  q49  +
180f;
                                    }
                                    else if (q49 <= 0f && q49
>= -90f)
                                    {
                                                q410  =  q49  -
180f;
                                    }
                                    else
                                    {
                                                q410 = q49;
                                    }
                        }

                        if (Mathf.Sin (q511) == 0f)
                        {
                                    q411 = q4prev;
                                    q412 = q4prev;
                        }
                        else
                        {
                                    q411  =  Mathf.Rad2Deg  *
(Mathf.Atan2(-Mathf.Sin  (q12*Mathf.Deg2Rad)*r13  +  Mathf.Cos
(q12 * Mathf.Deg2Rad)*r23,

Mathf.Cos  (q12*Mathf.Deg2Rad)*Mathf.Cos  (q26*Mathf.Deg2Rad  +
q32*Mathf.Deg2Rad)*r13 +

Mathf.Sin  (q12*Mathf.Deg2Rad)*Mathf.Cos  (q26*Mathf.Deg2Rad  +
q32*Mathf.Deg2Rad)*r23 -

Mathf.Sin (q26*Mathf.Deg2Rad + q32*Mathf.Deg2Rad)*r33));
                                    if (q411 <= 90f && q411 >=
0f)
                                    {
                                                q412  =  q411  +
180f;
                                    }
                                    else if (q411 <= 0f && q411
>= -90f)
                                    {
                                                q412  =  q411  -
180f;
                                    }
                                    else
                                    {
                                                q412 = q411;
                                    }
                        }

                        if (Mathf.Sin (q513) == 0f)
                        {
                                    q413 = q4prev;
                                    q414 = q4prev;
                        }
                        else
                        {
                                    q413  =  Mathf.Rad2Deg  *
(Mathf.Atan2(-Mathf.Sin  (q12*Mathf.Deg2Rad)*r13  +  Mathf.Cos
(q12 * Mathf.Deg2Rad)*r23,

Mathf.Cos  (q12*Mathf.Deg2Rad)*Mathf.Cos  (q27*Mathf.Deg2Rad  +
q33*Mathf.Deg2Rad)*r13 +

Mathf.Sin  (q12*Mathf.Deg2Rad)*Mathf.Cos  (q27*Mathf.Deg2Rad  +
q33*Mathf.Deg2Rad)*r23 -

Mathf.Sin (q27*Mathf.Deg2Rad + q33*Mathf.Deg2Rad)*r33));
                                    if (q413 <= 90f && q413 >=
0f)
                                    {
                                                q414  =  q413  +
180f;
                                    }
                                    else if (q413 <= 0f && q413
>= -90f)
                                    {
                                                q414  =  q413  -
180f;
                                    }
                                    else
                                    {
                                                q414 = q413;
                                    }
                        }

                        if (Mathf.Sin (q515) == 0f)
                        {
                                    q415 = q4prev;
                                    q416 = q4prev;
                        }
                        else
                        {
                                    q415  =  Mathf.Rad2Deg  *
(Mathf.Atan2(-Mathf.Sin  (q12*Mathf.Deg2Rad)*r13  +  Mathf.Cos
(q12 * Mathf.Deg2Rad)*r23,

Mathf.Cos  (q12*Mathf.Deg2Rad)*Mathf.Cos  (q28*Mathf.Deg2Rad  +
q34*Mathf.Deg2Rad)*r13 +

Mathf.Sin  (q12*Mathf.Deg2Rad)*Mathf.Cos  (q28*Mathf.Deg2Rad  +
q34*Mathf.Deg2Rad)*r23 -

Mathf.Sin (q28*Mathf.Deg2Rad + q34*Mathf.Deg2Rad)*r33));
                                    if (q415 <= 90f && q415 >=
0f)
                                    {
                                                q416  =  q415  +
180f;
                                    }
                                    else if (q415 <= 0f && q415
>= -90f)
                                    {
                                                q416  =  q415  -
180f;
                                    }
                                    else
                                    {
                                                q416 = q415;
                                    }
                        }


                        // Υπολογισμος q6
                        q61  =  Mathf.Rad2Deg  *  Mathf.Asin
(Mathf.Round         ((-(Mathf.Sin(q11*Mathf.Deg2Rad)*Mathf.Cos
(q41*Mathf.Deg2Rad) +

Mathf.Cos                        (q11*Mathf.Deg2Rad)*Mathf.Sin
```

```
(q41*Mathf.Deg2Rad)*Mathf.Cos        (q21*Mathf.Deg2Rad       +
q31*Mathf.Deg2Rad))*r11 +

(Mathf.Cos(q11*Mathf.Deg2Rad)*Mathf.Cos (q41*Mathf.Deg2Rad) -

                                                  Mathf.Sin
(q11*Mathf.Deg2Rad)*Mathf.Sin      (q41*Mathf.Deg2Rad)*Mathf.Cos
(q21*Mathf.Deg2Rad + q31*Mathf.Deg2Rad))*r21 +

Mathf.Sin  (q41*Mathf.Deg2Rad)*Mathf.Sin  (q21*Mathf.Deg2Rad  +
q31*Mathf.Deg2Rad)*r31)*1000f)/1000f);
                        if (q61 >= 0f)
                        {
                                    q62 = 180f - q61;
                                    q63 = -180f - q61;
                        }
                        else
                        {
                                    q62 = -180f - q61;
                                    q63 = 180f - q61;
                        }

                        q64   =   Mathf.Rad2Deg   *   Mathf.Asin
(Mathf.Round          ((-(Mathf.Sin(q11*Mathf.Deg2Rad)*Mathf.Cos
(q42*Mathf.Deg2Rad) +

Mathf.Cos                          (q11*Mathf.Deg2Rad)*Mathf.Sin
(q42*Mathf.Deg2Rad)*Mathf.Cos        (q21*Mathf.Deg2Rad       +
q31*Mathf.Deg2Rad))*r11 +

(Mathf.Cos(q11*Mathf.Deg2Rad)*Mathf.Cos (q42*Mathf.Deg2Rad) -

                                                  Mathf.Sin
(q11*Mathf.Deg2Rad)*Mathf.Sin      (q42*Mathf.Deg2Rad)*Mathf.Cos
(q21*Mathf.Deg2Rad + q31*Mathf.Deg2Rad))*r21 +

Mathf.Sin  (q42*Mathf.Deg2Rad)*Mathf.Sin  (q21*Mathf.Deg2Rad  +
q31*Mathf.Deg2Rad)*r31)*1000f)/1000f);
                        if (q64 >= 0f)
                        {
                                    q65 = 180f - q64;
                                    q66 = -180f - q64;
                        }
                        else
                        {
                                    q65 = -180f - q64;
                                    q66 = 180f - q64;
                        }


                        q67   =   Mathf.Rad2Deg   *   Mathf.Asin
(Mathf.Round          ((-(Mathf.Sin(q11*Mathf.Deg2Rad)*Mathf.Cos
(q43*Mathf.Deg2Rad) +

Mathf.Cos                          (q11*Mathf.Deg2Rad)*Mathf.Sin
(q43*Mathf.Deg2Rad)*Mathf.Cos        (q22*Mathf.Deg2Rad       +
q32*Mathf.Deg2Rad))*r11 +

(Mathf.Cos(q11*Mathf.Deg2Rad)*Mathf.Cos (q43*Mathf.Deg2Rad) -

                                                  Mathf.Sin
(q11*Mathf.Deg2Rad)*Mathf.Sin      (q43*Mathf.Deg2Rad)*Mathf.Cos
(q22*Mathf.Deg2Rad + q32*Mathf.Deg2Rad))*r21 +

Mathf.Sin  (q43*Mathf.Deg2Rad)*Mathf.Sin  (q22*Mathf.Deg2Rad  +
q32*Mathf.Deg2Rad)*r31)*1000f)/1000f);
                        if (q67 >= 0f)
                        {
                                    q68 = 180f - q67;
                                    q69 = -180f - q67;
                        }
                        else
                        {
                                    q68 = -180f - q67;
                                    q69 = 180f - q67;
                        }

                        q610   =   Mathf.Rad2Deg   *   Mathf.Asin
(Mathf.Round          ((-(Mathf.Sin(q11*Mathf.Deg2Rad)*Mathf.Cos
(q44*Mathf.Deg2Rad) +

Mathf.Cos                          (q11*Mathf.Deg2Rad)*Mathf.Sin
(q44*Mathf.Deg2Rad)*Mathf.Cos        (q22*Mathf.Deg2Rad       +
q32*Mathf.Deg2Rad))*r11 +

(Mathf.Cos(q11*Mathf.Deg2Rad)*Mathf.Cos (q44*Mathf.Deg2Rad) -

                                                  Mathf.Sin
(q11*Mathf.Deg2Rad)*Mathf.Sin      (q44*Mathf.Deg2Rad)*Mathf.Cos
(q22*Mathf.Deg2Rad + q32*Mathf.Deg2Rad))*r21 +

Mathf.Sin  (q44*Mathf.Deg2Rad)*Mathf.Sin  (q22*Mathf.Deg2Rad  +
q32*Mathf.Deg2Rad)*r31)*1000f)/1000f);
                        if (q610 >= 0f)
                        {
                                    q611 = 180f - q610;
                                    q612 = -180f - q610;
                        }
                        else
                        {
                                    q611 = -180f - q610;
                                    q612 = 180f - q610;
                        }

                        q613   =   Mathf.Rad2Deg   *   Mathf.Asin
(Mathf.Round          ((-(Mathf.Sin(q11*Mathf.Deg2Rad)*Mathf.Cos
(q45*Mathf.Deg2Rad) +

Mathf.Cos                          (q11*Mathf.Deg2Rad)*Mathf.Sin
(q45*Mathf.Deg2Rad)*Mathf.Cos        (q23*Mathf.Deg2Rad       +
q33*Mathf.Deg2Rad))*r11 +

(Mathf.Cos(q11*Mathf.Deg2Rad)*Mathf.Cos (q45*Mathf.Deg2Rad) -

                                                  Mathf.Sin
(q11*Mathf.Deg2Rad)*Mathf.Sin      (q45*Mathf.Deg2Rad)*Mathf.Cos
(q23*Mathf.Deg2Rad + q33*Mathf.Deg2Rad))*r21 +

Mathf.Sin  (q45*Mathf.Deg2Rad)*Mathf.Sin  (q23*Mathf.Deg2Rad  +
q33*Mathf.Deg2Rad)*r31)*1000f)/1000f);
                        if (q613 >= 0f)
                        {
                                    q614 = 180f - q613;
                                    q615 = -180f - q613;
                        }
                        else
                        {
                                    q614 = -180f - q613;
                                    q615 = 180f - q613;
                        }

                        q616   =   Mathf.Rad2Deg   *   Mathf.Asin
(Mathf.Round          ((-(Mathf.Sin(q11*Mathf.Deg2Rad)*Mathf.Cos
(q46*Mathf.Deg2Rad) +

Mathf.Cos                          (q11*Mathf.Deg2Rad)*Mathf.Sin
(q46*Mathf.Deg2Rad)*Mathf.Cos        (q23*Mathf.Deg2Rad       +
q33*Mathf.Deg2Rad))*r11 +

(Mathf.Cos(q11*Mathf.Deg2Rad)*Mathf.Cos (q46*Mathf.Deg2Rad) -

                                                  Mathf.Sin
(q11*Mathf.Deg2Rad)*Mathf.Sin      (q46*Mathf.Deg2Rad)*Mathf.Cos
(q23*Mathf.Deg2Rad + q33*Mathf.Deg2Rad))*r21 +

Mathf.Sin  (q46*Mathf.Deg2Rad)*Mathf.Sin  (q23*Mathf.Deg2Rad  +
q33*Mathf.Deg2Rad)*r31)*1000f)/1000f);
                        if (q616 >= 0f)
                        {
                                    q617 = 180f - q616;
                                    q618 = -180f - q616;
                        }
                        else
                        {
                                    q617 = -180f - q616;
                                    q618 = 180f - q616;
                        }

                        q619   =   Mathf.Rad2Deg   *   Mathf.Asin
(Mathf.Round          ((-(Mathf.Sin(q11*Mathf.Deg2Rad)*Mathf.Cos
(q47*Mathf.Deg2Rad) +

Mathf.Cos                          (q11*Mathf.Deg2Rad)*Mathf.Sin
(q47*Mathf.Deg2Rad)*Mathf.Cos        (q24*Mathf.Deg2Rad       +
q34*Mathf.Deg2Rad))*r11 +

(Mathf.Cos(q11*Mathf.Deg2Rad)*Mathf.Cos (q47*Mathf.Deg2Rad) -

                                                  Mathf.Sin
(q11*Mathf.Deg2Rad)*Mathf.Sin      (q47*Mathf.Deg2Rad)*Mathf.Cos
(q24*Mathf.Deg2Rad + q34*Mathf.Deg2Rad))*r21 +

Mathf.Sin  (q47*Mathf.Deg2Rad)*Mathf.Sin  (q24*Mathf.Deg2Rad  +
q34*Mathf.Deg2Rad)*r31)*1000f)/1000f);
                        if (q619 >= 0f)
                        {
                                    q620 = 180f - q619;
                                    q621 = -180f - q619;
                        }
                        else
                        {
                                    q620 = -180f - q619;
                                    q621 = 180f - q619;
                        }

                        q622   =   Mathf.Rad2Deg   *   Mathf.Asin
(Mathf.Round          ((-(Mathf.Sin(q11*Mathf.Deg2Rad)*Mathf.Cos
(q48*Mathf.Deg2Rad) +

Mathf.Cos                          (q11*Mathf.Deg2Rad)*Mathf.Sin
(q48*Mathf.Deg2Rad)*Mathf.Cos        (q24*Mathf.Deg2Rad       +
q34*Mathf.Deg2Rad))*r11 +

(Mathf.Cos(q11*Mathf.Deg2Rad)*Mathf.Cos (q48*Mathf.Deg2Rad) -

                                                  Mathf.Sin
(q11*Mathf.Deg2Rad)*Mathf.Sin      (q48*Mathf.Deg2Rad)*Mathf.Cos
(q24*Mathf.Deg2Rad + q34*Mathf.Deg2Rad))*r21 +

Mathf.Sin  (q48*Mathf.Deg2Rad)*Mathf.Sin  (q24*Mathf.Deg2Rad  +
q34*Mathf.Deg2Rad)*r31)*1000f)/1000f);
                        if (q622 >= 0f)
                        {
                                    q623 = 180f - q622;
                                    q624 = -180f - q622;
                        }
                        else
                        {
                                    q623 = -180f - q622;
                                    q624 = 180f - q622;
                        }

                        q625   =   Mathf.Rad2Deg   *   Mathf.Asin
(Mathf.Round          ((-(Mathf.Sin(q12*Mathf.Deg2Rad)*Mathf.Cos
(q49*Mathf.Deg2Rad) +

Mathf.Cos                          (q12*Mathf.Deg2Rad)*Mathf.Sin
(q49*Mathf.Deg2Rad)*Mathf.Cos        (q25*Mathf.Deg2Rad       +
q31*Mathf.Deg2Rad))*r11 +

(Mathf.Cos(q12*Mathf.Deg2Rad)*Mathf.Cos (q49*Mathf.Deg2Rad) -

                                                  Mathf.Sin
(q12*Mathf.Deg2Rad)*Mathf.Sin      (q49*Mathf.Deg2Rad)*Mathf.Cos
(q25*Mathf.Deg2Rad + q31*Mathf.Deg2Rad))*r21 +

Mathf.Sin  (q49*Mathf.Deg2Rad)*Mathf.Sin  (q25*Mathf.Deg2Rad  +
q31*Mathf.Deg2Rad)*r31)*1000f)/1000f);
                        if (q625 >= 0f)
                        {
                                    q626 = 180f - q625;
                                    q627 = -180f - q625;
                        }
                        else
                        {
```

```
                                    q626 = -180f - q625;
                                    q627 = 180f - q625;
                          }

                          q628  =  Mathf.Rad2Deg  *  Mathf.Asin
(Mathf.Round          ((-(Mathf.Sin(q12*Mathf.Deg2Rad)*Mathf.Cos
(q410*Mathf.Deg2Rad) +

Mathf.Cos                          (q12*Mathf.Deg2Rad)*Mathf.Sin
(q410*Mathf.Deg2Rad)*Mathf.Cos       (q25*Mathf.Deg2Rad       +
q31*Mathf.Deg2Rad))*r11 +

(Mathf.Cos(q12*Mathf.Deg2Rad)*Mathf.Cos (q410*Mathf.Deg2Rad) -


                                                  Mathf.Sin
(q12*Mathf.Deg2Rad)*Mathf.Sin     (q410*Mathf.Deg2Rad)*Mathf.Cos
(q25*Mathf.Deg2Rad + q31*Mathf.Deg2Rad))*r21 +

Mathf.Sin  (q410*Mathf.Deg2Rad)*Mathf.Sin  (q25*Mathf.Deg2Rad  +
q31*Mathf.Deg2Rad)*r31)*1000f)/1000f);
                          if (q628 >= 0f)
                          {
                                    q629 = 180f - q628;
                                    q630 = -180f - q628;
                          }
                          else
                          {
                                    q629 = -180f - q628;
                                    q630 = 180f - q628;
                          }

                          q631  =  Mathf.Rad2Deg  *  Mathf.Asin
(Mathf.Round          ((-(Mathf.Sin(q12*Mathf.Deg2Rad)*Mathf.Cos
(q411*Mathf.Deg2Rad) +

Mathf.Cos                          (q12*Mathf.Deg2Rad)*Mathf.Sin
(q411*Mathf.Deg2Rad)*Mathf.Cos       (q26*Mathf.Deg2Rad       +
q32*Mathf.Deg2Rad))*r11 +

(Mathf.Cos(q12*Mathf.Deg2Rad)*Mathf.Cos (q411*Mathf.Deg2Rad) -


                                                  Mathf.Sin
(q12*Mathf.Deg2Rad)*Mathf.Sin     (q411*Mathf.Deg2Rad)*Mathf.Cos
(q26*Mathf.Deg2Rad + q32*Mathf.Deg2Rad))*r21 +

Mathf.Sin  (q411*Mathf.Deg2Rad)*Mathf.Sin  (q26*Mathf.Deg2Rad  +
q32*Mathf.Deg2Rad)*r31)*1000f)/1000f);
                          if (q631 >= 0f)
                          {
                                    q632 = 180f - q631;
                                    q633 = -180f - q631;
                          }
                          else
                          {
                                    q632 = -180f - q631;
                                    q633 = 180f - q631;
                          }

                          q634  =  Mathf.Rad2Deg  *  Mathf.Asin
(Mathf.Round          ((-(Mathf.Sin(q12*Mathf.Deg2Rad)*Mathf.Cos
(q412*Mathf.Deg2Rad) +

Mathf.Cos                          (q12*Mathf.Deg2Rad)*Mathf.Sin
(q412*Mathf.Deg2Rad)*Mathf.Cos       (q26*Mathf.Deg2Rad       +
q32*Mathf.Deg2Rad))*r11 +

(Mathf.Cos(q12*Mathf.Deg2Rad)*Mathf.Cos (q412*Mathf.Deg2Rad) -


                                                  Mathf.Sin
(q12*Mathf.Deg2Rad)*Mathf.Sin     (q412*Mathf.Deg2Rad)*Mathf.Cos
(q26*Mathf.Deg2Rad + q32*Mathf.Deg2Rad))*r21 +

Mathf.Sin  (q412*Mathf.Deg2Rad)*Mathf.Sin  (q26*Mathf.Deg2Rad  +
q32*Mathf.Deg2Rad)*r31)*1000f)/1000f);
                          if (q634 >= 0f)
                          {
                                    q635 = 180f - q634;
                                    q636 = -180f - q634;
                          }
                          else
                          {
                                    q635 = -180f - q634;
                                    q636 = 180f - q634;
                          }


                          q637  =  Mathf.Rad2Deg  *  Mathf.Asin
(Mathf.Round          ((-(Mathf.Sin(q12*Mathf.Deg2Rad)*Mathf.Cos
(q413*Mathf.Deg2Rad) +

Mathf.Cos                          (q12*Mathf.Deg2Rad)*Mathf.Sin
(q413*Mathf.Deg2Rad)*Mathf.Cos       (q27*Mathf.Deg2Rad       +
q33*Mathf.Deg2Rad))*r11 +

(Mathf.Cos(q12*Mathf.Deg2Rad)*Mathf.Cos (q413*Mathf.Deg2Rad) -


                                                  Mathf.Sin
(q12*Mathf.Deg2Rad)*Mathf.Sin     (q413*Mathf.Deg2Rad)*Mathf.Cos
(q27*Mathf.Deg2Rad + q33*Mathf.Deg2Rad))*r21 +

Mathf.Sin  (q413*Mathf.Deg2Rad)*Mathf.Sin  (q27*Mathf.Deg2Rad  +
q33*Mathf.Deg2Rad)*r31)*1000f)/1000f);
                          if (q637 >= 0f)
                          {
                                    q638 = 180f - q637;
                                    q639 = -180f - q637;
                          }
                          else
                          {
                                    q638 = -180f - q637;
                                    q639 = 180f - q637;
                          }


                          q640  =  Mathf.Rad2Deg  *  Mathf.Asin
(Mathf.Round          ((-(Mathf.Sin(q12*Mathf.Deg2Rad)*Mathf.Cos
(q414*Mathf.Deg2Rad) +

Mathf.Cos                          (q12*Mathf.Deg2Rad)*Mathf.Sin
(q414*Mathf.Deg2Rad)*Mathf.Cos       (q27*Mathf.Deg2Rad       +
q33*Mathf.Deg2Rad))*r11 +

(Mathf.Cos(q12*Mathf.Deg2Rad)*Mathf.Cos (q414*Mathf.Deg2Rad) -


                                                  Mathf.Sin
(q12*Mathf.Deg2Rad)*Mathf.Sin     (q414*Mathf.Deg2Rad)*Mathf.Cos
(q27*Mathf.Deg2Rad + q33*Mathf.Deg2Rad))*r21 +

Mathf.Sin  (q414*Mathf.Deg2Rad)*Mathf.Sin  (q27*Mathf.Deg2Rad  +
q33*Mathf.Deg2Rad)*r31)*1000f)/1000f);
                          if (q640 >= 0f)
                          {
                                    q641 = 180f - q640;
                                    q642 = -180f - q640;
                          }
                          else
                          {
                                    q641 = -180f - q640;
                                    q642 = 180f - q640;
                          }

                          q643  =  Mathf.Rad2Deg  *  Mathf.Asin
(Mathf.Round          ((-(Mathf.Sin(q12*Mathf.Deg2Rad)*Mathf.Cos
(q415*Mathf.Deg2Rad) +

Mathf.Cos                          (q12*Mathf.Deg2Rad)*Mathf.Sin
(q415*Mathf.Deg2Rad)*Mathf.Cos       (q28*Mathf.Deg2Rad       +
q34*Mathf.Deg2Rad))*r11 +

(Mathf.Cos(q12*Mathf.Deg2Rad)*Mathf.Cos (q415*Mathf.Deg2Rad) -


                                                  Mathf.Sin
(q12*Mathf.Deg2Rad)*Mathf.Sin     (q415*Mathf.Deg2Rad)*Mathf.Cos
(q28*Mathf.Deg2Rad + q34*Mathf.Deg2Rad))*r21 +

Mathf.Sin  (q415*Mathf.Deg2Rad)*Mathf.Sin  (q28*Mathf.Deg2Rad  +
q34*Mathf.Deg2Rad)*r31)*1000f)/1000f);
                          if (q643 >= 0f)
                          {
                                    q644 = 180f - q643;
                                    q645 = -180f - q643;
                          }
                          else
                          {
                                    q644 = -180f - q643;
                                    q645 = 180f - q643;
                          }


                          q646  =  Mathf.Rad2Deg  *  Mathf.Asin
(Mathf.Round          ((-(Mathf.Sin(q12*Mathf.Deg2Rad)*Mathf.Cos
(q416*Mathf.Deg2Rad) +

Mathf.Cos                          (q12*Mathf.Deg2Rad)*Mathf.Sin
(q416*Mathf.Deg2Rad)*Mathf.Cos       (q28*Mathf.Deg2Rad       +
q34*Mathf.Deg2Rad))*r11 +

(Mathf.Cos(q12*Mathf.Deg2Rad)*Mathf.Cos (q416*Mathf.Deg2Rad) -


                                                  Mathf.Sin
(q12*Mathf.Deg2Rad)*Mathf.Sin     (q416*Mathf.Deg2Rad)*Mathf.Cos
(q28*Mathf.Deg2Rad + q34*Mathf.Deg2Rad))*r21 +

Mathf.Sin  (q416*Mathf.Deg2Rad)*Mathf.Sin  (q28*Mathf.Deg2Rad  +
q34*Mathf.Deg2Rad)*r31)*1000f)/1000f);
                          if (q646 >= 0f)
                          {
                                    q647 = 180f - q646;
                                    q648 = -180f - q646;
                          }
                          else
                          {
                                    q647 = -180f - q646;
                                    q648 = 180f - q646;
                          }


                          //***Υπολογισμων     αθροισματων     και
επιλογη γωνιων μεσω του μικροτερου αθροισματος***//
                          sum1 = Mathf.Infinity;
                          minSum = sum1;

                          if ((q11 >= -160f && q11 <= 160f) &&
(q31 >= -52.5f && q31 <= 232.5f) && (q21 >= -227.5f && q21 <=
47.5f) && (q51 >= -105f && q51 <= 120f) && (q41 >= -270f && q41
<= 270f) && (q61 >= -270f && q61 <= 270f))
                          {
                                    sum1 = Mathf.Abs (q1prev -
q11) + Mathf.Abs (q2prev - q21) + Mathf.Abs (q3prev - q31) +
Mathf.Abs  (q4prev  -  q41)  +  Mathf.Abs  (q5prev  -  q51)  +
Mathf.Abs(q6prev - q61);

                                    if (sum1 < minSum)
                                    {
                                              minSum = sum1;
                                              rotZ1 = q11;
                                              rotZ2 = q21;
                                              rotZ3 = q31;
                                              rotZ4 = q41;
                                              rotZ5 = q51;
                                              rotZ6 = q61;
                                    }
                          }

                          if ((q11 >= -160f && q11 <= 160f) &&
(q31 >= -52.5f && q31 <= 232.5f) && (q21 >= -227.5f && q21 <=
47.5f) && (q51 >= -105f && q51 <= 120f) && (q41 >= -270f && q41
<= 270f) && (q62 >= -270f && q62 <= 270f))
                          {
                                    sum2 = Mathf.Abs (q1prev -
q11) + Mathf.Abs (q2prev - q21) + Mathf.Abs (q3prev - q31) +
Mathf.Abs  (q4prev  -  q41)  +  Mathf.Abs  (q5prev  -  q51)  +
Mathf.Abs(q6prev - q62);

                                    if (sum2 < minSum)
                                    {
```

```
                                        minSum = sum2;
                                        rotZ1 = q11;
                                        rotZ2 = q21;
                                        rotZ3 = q31;
                                        rotZ4 = q41;
                                        rotZ5 = q51;
                                        rotZ6 = q62;
                                }
                        }

                        if ((q11 >= -160f && q11 <= 160f) &&
(q31 >= -52.5f && q31 <= 232.5f) && (q21 >= -227.5f && q21 <=
47.5f) && (q51 >= -105f && q51 <= 120f) && (q41 >= -270f && q41
<= 270f) && (q63 >= -270f && q63 <= 270f))
                        {
                                sum3 = Mathf.Abs (q1prev -
q11) + Mathf.Abs (q2prev - q21) + Mathf.Abs (q3prev - q31) +
Mathf.Abs  (q4prev  -  q41)  +  Mathf.Abs  (q5prev  -  q51)  +
Mathf.Abs(q6prev - q63);

                                if (sum3 < minSum)
                                {
                                        minSum = sum3;
                                        rotZ1 = q11;
                                        rotZ2 = q21;
                                        rotZ3 = q31;
                                        rotZ4 = q41;
                                        rotZ5 = q51;
                                        rotZ6 = q63;
                                }
                        }

                        if ((q11 >= -160f && q11 <= 160f) &&
(q31 >= -52.5f && q31 <= 232.5f) && (q21 >= -227.5f && q21 <=
47.5f) && (q51 >= -105f && q51 <= 120f) && (q42 >= -270f && q42
<= 270f) && (q64 >= -270f && q64 <= 270f))
                        {
                                sum4 = Mathf.Abs (q1prev -
q11) + Mathf.Abs (q2prev - q21) + Mathf.Abs (q3prev - q31) +
Mathf.Abs  (q4prev  -  q42)  +  Mathf.Abs  (q5prev  -  q51)  +
Mathf.Abs(q6prev - q64);

                                if (sum4 < minSum)
                                {
                                        minSum = sum4;
                                        rotZ1 = q11;
                                        rotZ2 = q21;
                                        rotZ3 = q31;
                                        rotZ4 = q42;
                                        rotZ5 = q51;
                                        rotZ6 = q64;
                                }
                        }

                        if ((q11 >= -160f && q11 <= 160f) &&
(q31 >= -52.5f && q31 <= 232.5f) && (q21 >= -227.5f && q21 <=
47.5f) && (q51 >= -105f && q51 <= 120f) && (q42 >= -270f && q42
<= 270f) && (q65 >= -270f && q65 <= 270f))
                        {
                                sum5 = Mathf.Abs (q1prev -
q11) + Mathf.Abs (q2prev - q21) + Mathf.Abs (q3prev - q31) +
Mathf.Abs  (q4prev  -  q42)  +  Mathf.Abs  (q5prev  -  q51)  +
Mathf.Abs(q6prev - q65);

                                if (sum5 < minSum)
                                {
                                        minSum = sum5;
                                        rotZ1 = q11;
                                        rotZ2 = q21;
                                        rotZ3 = q31;
                                        rotZ4 = q42;
                                        rotZ5 = q51;
                                        rotZ6 = q65;
                                }
                        }

                        if ((q11 >= -160f && q11 <= 160f) &&
(q31 >= -52.5f && q31 <= 232.5f) && (q21 >= -227.5f && q21 <=
47.5f) && (q51 >= -105f && q51 <= 120f) && (q42 >= -270f && q42
<= 270f) && (q66 >= -270f && q66 <= 270f))
                        {
                                sum6 = Mathf.Abs (q1prev -
q11) + Mathf.Abs (q2prev - q21) + Mathf.Abs (q3prev - q31) +
Mathf.Abs  (q4prev  -  q42)  +  Mathf.Abs  (q5prev  -  q51)  +
Mathf.Abs(q6prev - q66);

                                if (sum6 < minSum)
                                {
                                        minSum = sum6;
                                        rotZ1 = q11;
                                        rotZ2 = q21;
                                        rotZ3 = q31;
                                        rotZ4 = q42;
                                        rotZ5 = q51;
                                        rotZ6 = q66;
                                }
                        }

                        if ((q11 >= -160f && q11 <= 160f) &&
(q31 >= -52.5f && q31 <= 232.5f) && (q21 >= -227.5f && q21 <=
47.5f) && (q52 >= -105f && q52 <= 120f) && (q41 >= -270f && q41
<= 270f) && (q61 >= -270f && q61 <= 270f))
                        {
                                sum7 = Mathf.Abs (q1prev -
q11) + Mathf.Abs (q2prev - q21) + Mathf.Abs (q3prev - q31) +
Mathf.Abs  (q4prev  -  q41)  +  Mathf.Abs  (q5prev  -  q52)  +
Mathf.Abs(q6prev - q61);

                                if (sum7 < minSum)
                                {
                                        minSum = sum7;
                                        rotZ1 = q11;
                                        rotZ2 = q21;
                                        rotZ3 = q31;
                                        rotZ4 = q41;
                                        rotZ5 = q52;
                                        rotZ6 = q61;
                                }
                        }

                        if ((q11 >= -160f && q11 <= 160f) &&
(q31 >= -52.5f && q31 <= 232.5f) && (q21 >= -227.5f && q21 <=
47.5f) && (q52 >= -105f && q52 <= 120f) && (q41 >= -270f && q41
<= 270f) && (q62 >= -270f && q62 <= 270f))
                        {
                                sum8 = Mathf.Abs (q1prev -
q11) + Mathf.Abs (q2prev - q21) + Mathf.Abs (q3prev - q31) +
Mathf.Abs  (q4prev  -  q41)  +  Mathf.Abs  (q5prev  -  q52)  +
Mathf.Abs(q6prev - q62);

                                if (sum8 < minSum)
                                {
                                        minSum = sum8;
                                        rotZ1 = q11;
                                        rotZ2 = q21;
                                        rotZ3 = q31;
                                        rotZ4 = q41;
                                        rotZ5 = q52;
                                        rotZ6 = q62;
                                }
                        }

                        if ((q11 >= -160f && q11 <= 160f) &&
(q31 >= -52.5f && q31 <= 232.5f) && (q21 >= -227.5f && q21 <=
47.5f) && (q52 >= -105f && q52 <= 120f) && (q41 >= -270f && q41
<= 270f) && (q63 >= -270f && q63 <= 270f))
                        {
                                sum9 = Mathf.Abs (q1prev -
q11) + Mathf.Abs (q2prev - q21) + Mathf.Abs (q3prev - q31) +
Mathf.Abs  (q4prev  -  q41)  +  Mathf.Abs  (q5prev  -  q52)  +
Mathf.Abs(q6prev - q63);

                                if (sum9 < minSum)
                                {
                                        minSum = sum9;
                                        rotZ1 = q11;
                                        rotZ2 = q21;
                                        rotZ3 = q31;
                                        rotZ4 = q41;
                                        rotZ5 = q52;
                                        rotZ6 = q63;
                                }
                        }


                        if ((q11 >= -160f && q11 <= 160f) &&
(q31 >= -52.5f && q31 <= 232.5f) && (q21 >= -227.5f && q21 <=
47.5f) && (q52 >= -105f && q52 <= 120f) && (q42 >= -270f && q42
<= 270f) && (q64 >= -270f && q64 <= 270f))
                        {
                                sum10 = Mathf.Abs (q1prev -
q11) + Mathf.Abs (q2prev - q21) + Mathf.Abs (q3prev - q31) +
Mathf.Abs  (q4prev  -  q42)  +  Mathf.Abs  (q5prev  -  q52)  +
Mathf.Abs(q6prev - q64);

                                if (sum10 < minSum)
                                {
                                        minSum = sum10;
                                        rotZ1 = q11;
                                        rotZ2 = q21;
                                        rotZ3 = q31;
                                        rotZ4 = q42;
                                        rotZ5 = q52;
                                        rotZ6 = q64;
                                }
                        }

                        if ((q11 >= -160f && q11 <= 160f) &&
(q31 >= -52.5f && q31 <= 232.5f) && (q21 >= -227.5f && q21 <=
47.5f) && (q52 >= -105f && q52 <= 120f) && (q42 >= -270f && q42
<= 270f) && (q65 >= -270f && q65 <= 270f))
                        {
                                sum11 = Mathf.Abs (q1prev -
q11) + Mathf.Abs (q2prev - q21) + Mathf.Abs (q3prev - q31) +
Mathf.Abs  (q4prev  -  q42)  +  Mathf.Abs  (q5prev  -  q52)  +
Mathf.Abs(q6prev - q65);

                                if (sum11 < minSum)
                                {
                                        minSum = sum11;
                                        rotZ1 = q11;
                                        rotZ2 = q21;
                                        rotZ3 = q31;
                                        rotZ4 = q42;
                                        rotZ5 = q52;
                                        rotZ6 = q65;
                                }
                        }

                        if ((q11 >= -160f && q11 <= 160f) &&
(q31 >= -52.5f && q31 <= 232.5f) && (q21 >= -227.5f && q21 <=
47.5f) && (q52 >= -105f && q52 <= 120f) && (q42 >= -270f && q42
<= 270f) && (q66 >= -270f && q66 <= 270f))
                        {
                                sum12 = Mathf.Abs (q1prev -
q11) + Mathf.Abs (q2prev - q21) + Mathf.Abs (q3prev - q31) +
Mathf.Abs  (q4prev  -  q42)  +  Mathf.Abs  (q5prev  -  q52)  +
Mathf.Abs(q6prev - q66);

                                if (sum12 < minSum)
                                {
                                        minSum = sum12;
                                        rotZ1 = q11;
                                        rotZ2 = q21;
                                        rotZ3 = q31;
                                        rotZ4 = q42;
                                        rotZ5 = q52;
                                        rotZ6 = q66;
                                }
                        }

                        if ((q11 >= -160f && q11 <= 160f) &&
(q32 >= -52.5f && q32 <= 232.5f) && (q22 >= -227.5f && q22 <=
47.5f) && (q53 >= -105f && q53 <= 120f) && (q43 >= -270f && q43
<= 270f) && (q67 >= -270f && q67 <= 270f))
                        {
                                sum13 = Mathf.Abs (q1prev -
q11) + Mathf.Abs (q2prev - q22) + Mathf.Abs (q3prev - q32) +
Mathf.Abs  (q4prev  -  q43)  +  Mathf.Abs  (q5prev  -  q53)  +
Mathf.Abs(q6prev - q67);

                                if (sum13 < minSum)
                                {
                                        minSum = sum13;
```

```
                                        rotZ1 = q11;
                                        rotZ2 = q22;
                                        rotZ3 = q32;
                                        rotZ4 = q43;
                                        rotZ5 = q53;
                                        rotZ6 = q67;
                                }
                        }

                        if ((q11 >= -160f && q11 <= 160f) &&
(q32 >= -52.5f && q32 <= 232.5f) && (q22 >= -227.5f && q22 <=
47.5f) && (q53 >= -105f && q53 <= 120f) && (q43 >= -270f && q43
<= 270f) && (q68 >= -270f && q68 <= 270f))
                        {
                                sum14 = Mathf.Abs (q1prev -
q11) + Mathf.Abs (q2prev - q22) + Mathf.Abs (q3prev - q32) +
Mathf.Abs  (q4prev  -  q43)  +  Mathf.Abs  (q5prev  -  q53)  +
Mathf.Abs(q6prev - q68);

                                if (sum14 < minSum)
                                {
                                        minSum = sum14;
                                        rotZ1 = q11;
                                        rotZ2 = q22;
                                        rotZ3 = q32;
                                        rotZ4 = q43;
                                        rotZ5 = q53;
                                        rotZ6 = q68;
                                }
                        }

                        if ((q11 >= -160f && q11 <= 160f) &&
(q32 >= -52.5f && q32 <= 232.5f) && (q22 >= -227.5f && q22 <=
47.5f) && (q53 >= -105f && q53 <= 120f) && (q43 >= -270f && q43
<= 270f) && (q69 >= -270f && q69 <= 270f))
                        {
                                sum15 = Mathf.Abs (q1prev -
q11) + Mathf.Abs (q2prev - q22) + Mathf.Abs (q3prev - q32) +
Mathf.Abs  (q4prev  -  q43)  +  Mathf.Abs  (q5prev  -  q53)  +
Mathf.Abs(q6prev - q69);

                                if (sum15 < minSum)
                                {
                                        minSum = sum15;
                                        rotZ1 = q11;
                                        rotZ2 = q22;
                                        rotZ3 = q32;
                                        rotZ4 = q43;
                                        rotZ5 = q53;
                                        rotZ6 = q69;
                                }
                        }

                        if ((q11 >= -160f && q11 <= 160f) &&
(q32 >= -52.5f && q32 <= 232.5f) && (q22 >= -227.5f && q22 <=
47.5f) && (q53 >= -105f && q53 <= 120f) && (q44 >= -270f && q44
<= 270f) && (q610 >= -270f && q610 <= 270f))
                        {
                                sum16 = Mathf.Abs (q1prev -
q11) + Mathf.Abs (q2prev - q22) + Mathf.Abs (q3prev - q32) +
Mathf.Abs  (q4prev  -  q44)  +  Mathf.Abs  (q5prev  -  q53)  +
Mathf.Abs(q6prev - q610);

                                if (sum16 < minSum)
                                {
                                        minSum = sum16;
                                        rotZ1 = q11;
                                        rotZ2 = q22;
                                        rotZ3 = q32;
                                        rotZ4 = q44;
                                        rotZ5 = q53;
                                        rotZ6 = q610;
                                }
                        }
                        if ((q11 >= -160f && q11 <= 160f) &&
(q32 >= -52.5f && q32 <= 232.5f) && (q22 >= -227.5f && q22 <=
47.5f) && (q53 >= -105f && q53 <= 120f) && (q44 >= -270f && q44
<= 270f) && (q611 >= -270f && q611 <= 270f))
                        {
                                sum17 = Mathf.Abs (q1prev -
q11) + Mathf.Abs (q2prev - q22) + Mathf.Abs (q3prev - q32) +
Mathf.Abs  (q4prev  -  q44)  +  Mathf.Abs  (q5prev  -  q53)  +
Mathf.Abs(q6prev - q611);

                                if (sum17 < minSum)
                                {
                                        minSum = sum17;
                                        rotZ1 = q11;
                                        rotZ2 = q22;
                                        rotZ3 = q32;
                                        rotZ4 = q44;
                                        rotZ5 = q53;
                                        rotZ6 = q611;
                                }
                        }

                        if ((q11 >= -160f && q11 <= 160f) &&
(q32 >= -52.5f && q32 <= 232.5f) && (q22 >= -227.5f && q22 <=
47.5f) && (q53 >= -105f && q53 <= 120f) && (q44 >= -270f && q44
<= 270f) && (q612 >= -270f && q612 <= 270f))
                        {
                                sum18 = Mathf.Abs (q1prev -
q11) + Mathf.Abs (q2prev - q22) + Mathf.Abs (q3prev - q32) +
Mathf.Abs  (q4prev  -  q44)  +  Mathf.Abs  (q5prev  -  q53)  +
Mathf.Abs(q6prev - q612);

                                if (sum18 < minSum)
                                {
                                        minSum = sum18;
                                        rotZ1 = q11;
                                        rotZ2 = q22;
                                        rotZ3 = q32;
                                        rotZ4 = q44;
                                        rotZ5 = q53;
                                        rotZ6 = q612;
                                }
                        }


                        if ((q11 >= -160f && q11 <= 160f) &&
(q32 >= -52.5f && q32 <= 232.5f) && (q22 >= -227.5f && q22 <=
47.5f) && (q54 >= -105f && q54 <= 120f) && (q43 >= -270f && q43
<= 270f) && (q67 >= -270f && q67 <= 270f))
                        {
                                sum19 = Mathf.Abs (q1prev -
q11) + Mathf.Abs (q2prev - q22) + Mathf.Abs (q3prev - q32) +
Mathf.Abs  (q4prev  -  q43)  +  Mathf.Abs  (q5prev  -  q54)  +
Mathf.Abs(q6prev - q67);

                                if (sum19 < minSum)
                                {
                                        minSum = sum19;
                                        rotZ1 = q11;
                                        rotZ2 = q22;
                                        rotZ3 = q32;
                                        rotZ4 = q43;
                                        rotZ5 = q54;
                                        rotZ6 = q67;
                                }
                        }

                        if ((q11 >= -160f && q11 <= 160f) &&
(q32 >= -52.5f && q32 <= 232.5f) && (q22 >= -227.5f && q22 <=
47.5f) && (q54 >= -105f && q54 <= 120f) && (q43 >= -270f && q43
<= 270f) && (q68 >= -270f && q68 <= 270f))
                        {
                                sum20 = Mathf.Abs (q1prev -
q11) + Mathf.Abs (q2prev - q22) + Mathf.Abs (q3prev - q32) +
Mathf.Abs  (q4prev  -  q43)  +  Mathf.Abs  (q5prev  -  q54)  +
Mathf.Abs(q6prev - q68);

                                if (sum20 < minSum)
                                {
                                        minSum = sum20;
                                        rotZ1 = q11;
                                        rotZ2 = q22;
                                        rotZ3 = q32;
                                        rotZ4 = q43;
                                        rotZ5 = q54;
                                        rotZ6 = q68;
                                }
                        }

                        if ((q11 >= -160f && q11 <= 160f) &&
(q32 >= -52.5f && q32 <= 232.5f) && (q22 >= -227.5f && q22 <=
47.5f) && (q54 >= -105f && q54 <= 120f) && (q43 >= -270f && q43
<= 270f) && (q69 >= -270f && q69 <= 270f))
                        {
                                sum21 = Mathf.Abs (q1prev -
q11) + Mathf.Abs (q2prev - q22) + Mathf.Abs (q3prev - q32) +
Mathf.Abs  (q4prev  -  q43)  +  Mathf.Abs  (q5prev  -  q54)  +
Mathf.Abs(q6prev - q69);

                                if (sum21 < minSum)
                                {
                                        minSum = sum21;
                                        rotZ1 = q11;
                                        rotZ2 = q22;
                                        rotZ3 = q32;
                                        rotZ4 = q43;
                                        rotZ5 = q54;
                                        rotZ6 = q69;
                                }
                        }


                        if ((q11 >= -160f && q11 <= 160f) &&
(q32 >= -52.5f && q32 <= 232.5f) && (q22 >= -227.5f && q22 <=
47.5f) && (q54 >= -105f && q54 <= 120f) && (q44 >= -270f && q44
<= 270f) && (q610 >= -270f && q610 <= 270f))
                        {
                                sum22 = Mathf.Abs (q1prev -
q11) + Mathf.Abs (q2prev - q22) + Mathf.Abs (q3prev - q32) +
Mathf.Abs  (q4prev  -  q44)  +  Mathf.Abs  (q5prev  -  q54)  +
Mathf.Abs(q6prev - q610);

                                if (sum22 < minSum)
                                {
                                        minSum = sum22;
                                        rotZ1 = q11;
                                        rotZ2 = q22;
                                        rotZ3 = q32;
                                        rotZ4 = q44;
                                        rotZ5 = q54;
                                        rotZ6 = q610;
                                }
                        }

                        if ((q11 >= -160f && q11 <= 160f) &&
(q32 >= -52.5f && q32 <= 232.5f) && (q22 >= -227.5f && q22 <=
47.5f) && (q54 >= -105f && q54 <= 120f) && (q44 >= -270f && q44
<= 270f) && (q611 >= -270f && q611 <= 270f))
                        {
                                sum23 = Mathf.Abs (q1prev -
q11) + Mathf.Abs (q2prev - q22) + Mathf.Abs (q3prev - q32) +
Mathf.Abs  (q4prev  -  q44)  +  Mathf.Abs  (q5prev  -  q54)  +
Mathf.Abs(q6prev - q611);

                                if (sum23 < minSum)
                                {
                                        minSum = sum23;
                                        rotZ1 = q11;
                                        rotZ2 = q22;
                                        rotZ3 = q32;
                                        rotZ4 = q44;
                                        rotZ5 = q54;
                                        rotZ6 = q611;
                                }
                        }

                        if ((q11 >= -160f && q11 <= 160f) &&
(q32 >= -52.5f && q32 <= 232.5f) && (q22 >= -227.5f && q22 <=
47.5f) && (q54 >= -105f && q54 <= 120f) && (q44 >= -270f && q44
<= 270f) && (q612 >= -270f && q612 <= 270f))
                        {
                                sum24 = Mathf.Abs (q1prev -
q11) + Mathf.Abs (q2prev - q22) + Mathf.Abs (q3prev - q32) +
Mathf.Abs  (q4prev  -  q44)  +  Mathf.Abs  (q5prev  -  q54)  +
Mathf.Abs(q6prev - q612);

                                if (sum24 < minSum)
                                {
                                        minSum = sum24;
```

```
                                        rotZ1 = q11;
                                        rotZ2 = q22;
                                        rotZ3 = q32;
                                        rotZ4 = q44;
                                        rotZ5 = q54;
                                        rotZ6 = q612;
                                }
                        }

                        if ((q11 >= -160f && q11 <= 160f) &&
(q33 >= -52.5f && q33 <= 232.5f) && (q23 >= -227.5f && q23 <=
47.5f) && (q55 >= -105f && q55 <= 120f) && (q45 >= -270f && q45
<= 270f) && (q613 >= -270f && q613 <= 270f))
                        {
                                sum25 = Mathf.Abs (q1prev -
q11) + Mathf.Abs (q2prev - q23) + Mathf.Abs (q3prev - q33) +
Mathf.Abs  (q4prev  -  q45)  +  Mathf.Abs  (q5prev  -  q55)  +
Mathf.Abs(q6prev - q613);

                                if (sum25 < minSum)
                                {
                                        minSum = sum25;
                                        rotZ1 = q11;
                                        rotZ2 = q23;
                                        rotZ3 = q33;
                                        rotZ4 = q45;
                                        rotZ5 = q55;
                                        rotZ6 = q613;
                                }
                        }

                        if ((q11 >= -160f && q11 <= 160f) &&
(q33 >= -52.5f && q33 <= 232.5f) && (q23 >= -227.5f && q23 <=
47.5f) && (q55 >= -105f && q55 <= 120f) && (q45 >= -270f && q45
<= 270f) && (q614 >= -270f && q614 <= 270f))
                        {
                                sum26 = Mathf.Abs (q1prev -
q11) + Mathf.Abs (q2prev - q23) + Mathf.Abs (q3prev - q33) +
Mathf.Abs  (q4prev  -  q45)  +  Mathf.Abs  (q5prev  -  q55)  +
Mathf.Abs(q6prev - q614);

                                if (sum26 < minSum)
                                {
                                        minSum = sum26;
                                        rotZ1 = q11;
                                        rotZ2 = q23;
                                        rotZ3 = q33;
                                        rotZ4 = q45;
                                        rotZ5 = q55;
                                        rotZ6 = q614;
                                }
                        }

                        if ((q11 >= -160f && q11 <= 160f) &&
(q33 >= -52.5f && q33 <= 232.5f) && (q23 >= -227.5f && q23 <=
47.5f) && (q55 >= -105f && q55 <= 120f) && (q45 >= -270f && q45
<= 270f) && (q615 >= -270f && q615 <= 270f))
                        {
                                sum27 = Mathf.Abs (q1prev -
q11) + Mathf.Abs (q2prev - q23) + Mathf.Abs (q3prev - q33) +
Mathf.Abs  (q4prev  -  q45)  +  Mathf.Abs  (q5prev  -  q55)  +
Mathf.Abs(q6prev - q615);

                                if (sum27 < minSum)
                                {
                                        minSum = sum27;
                                        rotZ1 = q11;
                                        rotZ2 = q23;
                                        rotZ3 = q33;
                                        rotZ4 = q45;
                                        rotZ5 = q55;
                                        rotZ6 = q615;
                                }
                        }

                        if ((q11 >= -160f && q11 <= 160f) &&
(q33 >= -52.5f && q33 <= 232.5f) && (q23 >= -227.5f && q23 <=
47.5f) && (q55 >= -105f && q55 <= 120f) && (q46 >= -270f && q46
<= 270f) && (q616 >= -270f && q616 <= 270f))
                        {
                                sum28 = Mathf.Abs (q1prev -
q11) + Mathf.Abs (q2prev - q23) + Mathf.Abs (q3prev - q33) +
Mathf.Abs  (q4prev  -  q46)  +  Mathf.Abs  (q5prev  -  q55)  +
Mathf.Abs(q6prev - q616);

                                if (sum28 < minSum)
                                {
                                        minSum = sum28;
                                        rotZ1 = q11;
                                        rotZ2 = q23;
                                        rotZ3 = q33;
                                        rotZ4 = q46;
                                        rotZ5 = q55;
                                        rotZ6 = q616;
                                }
                        }

                        if ((q11 >= -160f && q11 <= 160f) &&
(q33 >= -52.5f && q33 <= 232.5f) && (q23 >= -227.5f && q23 <=
47.5f) && (q55 >= -105f && q55 <= 120f) && (q46 >= -270f && q46
<= 270f) && (q617 >= -270f && q617 <= 270f))
                        {
                                sum29 = Mathf.Abs (q1prev -
q11) + Mathf.Abs (q2prev - q23) + Mathf.Abs (q3prev - q33) +
Mathf.Abs  (q4prev  -  q46)  +  Mathf.Abs  (q5prev  -  q55)  +
Mathf.Abs(q6prev - q617);

                                if (sum29 < minSum)
                                {
                                        minSum = sum29;
                                        rotZ1 = q11;
                                        rotZ2 = q23;
                                        rotZ3 = q33;
                                        rotZ4 = q46;
                                        rotZ5 = q55;
                                        rotZ6 = q617;
                                }
                        }

                        if ((q11 >= -160f && q11 <= 160f) &&
(q33 >= -52.5f && q33 <= 232.5f) && (q23 >= -227.5f && q23 <=
47.5f) && (q55 >= -105f && q55 <= 120f) && (q46 >= -270f && q46
<= 270f) && (q618 >= -270f && q618 <= 270f))
                        {
                                sum30 = Mathf.Abs (q1prev -
q11) + Mathf.Abs (q2prev - q23) + Mathf.Abs (q3prev - q33) +
Mathf.Abs  (q4prev  -  q46)  +  Mathf.Abs  (q5prev  -  q55)  +
Mathf.Abs(q6prev - q618);

                                if (sum30 < minSum)
                                {
                                        minSum = sum30;
                                        rotZ1 = q11;
                                        rotZ2 = q23;
                                        rotZ3 = q33;
                                        rotZ4 = q46;
                                        rotZ5 = q55;
                                        rotZ6 = q618;
                                }
                        }

                        if ((q11 >= -160f && q11 <= 160f) &&
(q33 >= -52.5f && q33 <= 232.5f) && (q23 >= -227.5f && q23 <=
47.5f) && (q56 >= -105f && q56 <= 120f) && (q45 >= -270f && q45
<= 270f) && (q613 >= -270f && q613 <= 270f))
                        {
                                sum31 = Mathf.Abs (q1prev -
q11) + Mathf.Abs (q2prev - q23) + Mathf.Abs (q3prev - q33) +
Mathf.Abs  (q4prev  -  q45)  +  Mathf.Abs  (q5prev  -  q56)  +
Mathf.Abs(q6prev - q613);

                                if (sum31 < minSum)
                                {
                                        minSum = sum31;
                                        rotZ1 = q11;
                                        rotZ2 = q23;
                                        rotZ3 = q33;
                                        rotZ4 = q45;
                                        rotZ5 = q56;
                                        rotZ6 = q613;
                                }
                        }

                        if ((q11 >= -160f && q11 <= 160f) &&
(q33 >= -52.5f && q33 <= 232.5f) && (q23 >= -227.5f && q23 <=
47.5f) && (q56 >= -105f && q56 <= 120f) && (q45 >= -270f && q45
<= 270f) && (q614 >= -270f && q614 <= 270f))
                        {
                                sum32 = Mathf.Abs (q1prev -
q11) + Mathf.Abs (q2prev - q23) + Mathf.Abs (q3prev - q33) +
Mathf.Abs  (q4prev  -  q45)  +  Mathf.Abs  (q5prev  -  q56)  +
Mathf.Abs(q6prev - q614);

                                if (sum32 < minSum)
                                {
                                        minSum = sum32;
                                        rotZ1 = q11;
                                        rotZ2 = q23;
                                        rotZ3 = q33;
                                        rotZ4 = q45;
                                        rotZ5 = q56;
                                        rotZ6 = q614;
                                }
                        }

                        if ((q11 >= -160f && q11 <= 160f) &&
(q33 >= -52.5f && q33 <= 232.5f) && (q23 >= -227.5f && q23 <=
47.5f) && (q56 >= -105f && q56 <= 120f) && (q45 >= -270f && q45
<= 270f) && (q615 >= -270f && q615 <= 270f))
                        {
                                sum33 = Mathf.Abs (q1prev -
q11) + Mathf.Abs (q2prev - q23) + Mathf.Abs (q3prev - q33) +
Mathf.Abs  (q4prev  -  q45)  +  Mathf.Abs  (q5prev  -  q56)  +
Mathf.Abs(q6prev - q615);

                                if (sum33 < minSum)
                                {
                                        minSum = sum33;
                                        rotZ1 = q11;
                                        rotZ2 = q23;
                                        rotZ3 = q33;
                                        rotZ4 = q45;
                                        rotZ5 = q56;
                                        rotZ6 = q615;
                                }
                        }

                        if ((q11 >= -160f && q11 <= 160f) &&
(q33 >= -52.5f && q33 <= 232.5f) && (q23 >= -227.5f && q23 <=
47.5f) && (q56 >= -105f && q56 <= 120f) && (q46 >= -270f && q46
<= 270f) && (q616 >= -270f && q616 <= 270f))
                        {
                                sum34 = Mathf.Abs (q1prev -
q11) + Mathf.Abs (q2prev - q23) + Mathf.Abs (q3prev - q33) +
Mathf.Abs  (q4prev  -  q46)  +  Mathf.Abs  (q5prev  -  q56)  +
Mathf.Abs(q6prev - q616);

                                if (sum34 < minSum)
                                {
                                        minSum = sum34;
                                        rotZ1 = q11;
                                        rotZ2 = q23;
                                        rotZ3 = q33;
                                        rotZ4 = q46;
                                        rotZ5 = q56;
                                        rotZ6 = q616;
                                }
                        }

                        if ((q11 >= -160f && q11 <= 160f) &&
(q33 >= -52.5f && q33 <= 232.5f) && (q23 >= -227.5f && q23 <=
47.5f) && (q56 >= -105f && q56 <= 120f) && (q46 >= -270f && q46
<= 270f) && (q617 >= -270f && q617 <= 270f))
                        {
                                sum35 = Mathf.Abs (q1prev -
q11) + Mathf.Abs (q2prev - q23) + Mathf.Abs (q3prev - q33) +
Mathf.Abs  (q4prev  -  q46)  +  Mathf.Abs  (q5prev  -  q56)  +
Mathf.Abs(q6prev - q617);

                                if (sum35 < minSum)
                                {
                                        minSum = sum35;
                                        rotZ1 = q11;
```

```
                                        rotZ2 = q23;
                                        rotZ3 = q33;
                                        rotZ4 = q46;
                                        rotZ5 = q56;
                                        rotZ6 = q617;
                                }
                        }

                        if ((q11 >= -160f && q11 <= 160f) &&
(q33 >= -52.5f && q33 <= 232.5f) && (q23 >= -227.5f && q23 <=
47.5f) && (q56 >= -105f && q56 <= 120f) && (q46 >= -270f && q46
<= 270f) && (q618 >= -270f && q618 <= 270f))
                        {
                                sum36 = Mathf.Abs (q1prev -
q11) + Mathf.Abs (q2prev - q23) + Mathf.Abs (q3prev - q33) +
Mathf.Abs  (q4prev  -  q46)  +  Mathf.Abs  (q5prev  -  q56)  +
Mathf.Abs(q6prev - q618);

                                if (sum36 < minSum)
                                {
                                        minSum = sum36;
                                        rotZ1 = q11;
                                        rotZ2 = q23;
                                        rotZ3 = q33;
                                        rotZ4 = q46;
                                        rotZ5 = q56;
                                        rotZ6 = q618;
                                }
                        }


                        if ((q11 >= -160f && q11 <= 160f) &&
(q34 >= -52.5f && q34 <= 232.5f) && (q24 >= -227.5f && q24 <=
47.5f) && (q57 >= -105f && q57 <= 120f) && (q47 >= -270f && q47
<= 270f) && (q619 >= -270f && q619 <= 270f))
                        {
                                sum37 = Mathf.Abs (q1prev -
q11) + Mathf.Abs (q2prev - q24) + Mathf.Abs (q3prev - q34) +
Mathf.Abs  (q4prev  -  q47)  +  Mathf.Abs  (q5prev  -  q57)  +
Mathf.Abs(q6prev - q619);

                                if (sum37 < minSum)
                                {
                                        minSum = sum37;
                                        rotZ1 = q11;
                                        rotZ2 = q24;
                                        rotZ3 = q34;
                                        rotZ4 = q47;
                                        rotZ5 = q57;
                                        rotZ6 = q619;
                                }
                        }

                        if ((q11 >= -160f && q11 <= 160f) &&
(q34 >= -52.5f && q34 <= 232.5f) && (q24 >= -227.5f && q24 <=
47.5f) && (q57 >= -105f && q57 <= 120f) && (q47 >= -270f && q47
<= 270f) && (q620 >= -270f && q620 <= 270f))
                        {
                                sum38 = Mathf.Abs (q1prev -
q11) + Mathf.Abs (q2prev - q24) + Mathf.Abs (q3prev - q34) +
Mathf.Abs  (q4prev  -  q47)  +  Mathf.Abs  (q5prev  -  q57)  +
Mathf.Abs(q6prev - q620);

                                if (sum38 < minSum)
                                {
                                        minSum = sum38;
                                        rotZ1 = q11;
                                        rotZ2 = q24;
                                        rotZ3 = q34;
                                        rotZ4 = q47;
                                        rotZ5 = q57;
                                        rotZ6 = q620;
                                }
                        }

                        if ((q11 >= -160f && q11 <= 160f) &&
(q34 >= -52.5f && q34 <= 232.5f) && (q24 >= -227.5f && q24 <=
47.5f) && (q57 >= -105f && q57 <= 120f) && (q47 >= -270f && q47
<= 270f) && (q621 >= -270f && q621 <= 270f))
                        {
                                sum39 = Mathf.Abs (q1prev -
q11) + Mathf.Abs (q2prev - q24) + Mathf.Abs (q3prev - q34) +
Mathf.Abs  (q4prev  -  q47)  +  Mathf.Abs  (q5prev  -  q57)  +
Mathf.Abs(q6prev - q621);

                                if (sum39 < minSum)
                                {
                                        minSum = sum39;
                                        rotZ1 = q11;
                                        rotZ2 = q24;
                                        rotZ3 = q34;
                                        rotZ4 = q47;
                                        rotZ5 = q57;
                                        rotZ6 = q621;
                                }
                        }


                        if ((q11 >= -160f && q11 <= 160f) &&
(q34 >= -52.5f && q34 <= 232.5f) && (q24 >= -227.5f && q24 <=
47.5f) && (q57 >= -105f && q57 <= 120f) && (q48 >= -270f && q48
<= 270f) && (q622 >= -270f && q622 <= 270f))
                        {
                                sum40 = Mathf.Abs (q1prev -
q11) + Mathf.Abs (q2prev - q24) + Mathf.Abs (q3prev - q34) +
Mathf.Abs  (q4prev  -  q48)  +  Mathf.Abs  (q5prev  -  q57)  +
Mathf.Abs(q6prev - q622);

                                if (sum40 < minSum)
                                {
                                        minSum = sum40;
                                        rotZ1 = q11;
                                        rotZ2 = q24;
                                        rotZ3 = q34;
                                        rotZ4 = q48;
                                        rotZ5 = q57;
                                        rotZ6 = q622;
                                }
                        }

                        if ((q11 >= -160f && q11 <= 160f) &&
(q34 >= -52.5f && q34 <= 232.5f) && (q24 >= -227.5f && q24 <=
47.5f) && (q57 >= -105f && q57 <= 120f) && (q48 >= -270f && q48
<= 270f) && (q623 >= -270f && q623 <= 270f))
                        {
                                sum41 = Mathf.Abs (q1prev -
q11) + Mathf.Abs (q2prev - q24) + Mathf.Abs (q3prev - q34) +
Mathf.Abs  (q4prev  -  q48)  +  Mathf.Abs  (q5prev  -  q57)  +
Mathf.Abs(q6prev - q623);

                                if (sum41 < minSum)
                                {
                                        minSum = sum41;
                                        rotZ1 = q11;
                                        rotZ2 = q24;
                                        rotZ3 = q34;
                                        rotZ4 = q48;
                                        rotZ5 = q57;
                                        rotZ6 = q623;
                                }
                        }

                        if ((q11 >= -160f && q11 <= 160f) &&
(q34 >= -52.5f && q34 <= 232.5f) && (q24 >= -227.5f && q24 <=
47.5f) && (q57 >= -105f && q57 <= 120f) && (q48 >= -270f && q48
<= 270f) && (q624 >= -270f && q624 <= 270f))
                        {
                                sum42 = Mathf.Abs (q1prev -
q11) + Mathf.Abs (q2prev - q24) + Mathf.Abs (q3prev - q34) +
Mathf.Abs  (q4prev  -  q48)  +  Mathf.Abs  (q5prev  -  q57)  +
Mathf.Abs(q6prev - q624);

                                if (sum42 < minSum)
                                {
                                        minSum = sum42;
                                        rotZ1 = q11;
                                        rotZ2 = q24;
                                        rotZ3 = q34;
                                        rotZ4 = q48;
                                        rotZ5 = q57;
                                        rotZ6 = q624;
                                }
                        }

                        if ((q11 >= -160f && q11 <= 160f) &&
(q34 >= -52.5f && q34 <= 232.5f) && (q24 >= -227.5f && q24 <=
47.5f) && (q58 >= -105f && q58 <= 120f) && (q47 >= -270f && q47
<= 270f) && (q619 >= -270f && q619 <= 270f))
                        {
                                sum43 = Mathf.Abs (q1prev -
q11) + Mathf.Abs (q2prev - q24) + Mathf.Abs (q3prev - q34) +
Mathf.Abs  (q4prev  -  q47)  +  Mathf.Abs  (q5prev  -  q58)  +
Mathf.Abs(q6prev - q619);

                                if (sum43 < minSum)
                                {
                                        minSum = sum43;
                                        rotZ1 = q11;
                                        rotZ2 = q24;
                                        rotZ3 = q34;
                                        rotZ4 = q47;
                                        rotZ5 = q58;
                                        rotZ6 = q619;
                                }
                        }

                        if ((q11 >= -160f && q11 <= 160f) &&
(q34 >= -52.5f && q34 <= 232.5f) && (q24 >= -227.5f && q24 <=
47.5f) && (q58 >= -105f && q58 <= 120f) && (q47 >= -270f && q47
<= 270f) && (q620 >= -270f && q620 <= 270f))
                        {
                                sum44 = Mathf.Abs (q1prev -
q11) + Mathf.Abs (q2prev - q24) + Mathf.Abs (q3prev - q34) +
Mathf.Abs  (q4prev  -  q47)  +  Mathf.Abs  (q5prev  -  q58)  +
Mathf.Abs(q6prev - q620);

                                if (sum44 < minSum)
                                {
                                        minSum = sum44;
                                        rotZ1 = q11;
                                        rotZ2 = q24;
                                        rotZ3 = q34;
                                        rotZ4 = q47;
                                        rotZ5 = q58;
                                        rotZ6 = q620;
                                }
                        }

                        if ((q11 >= -160f && q11 <= 160f) &&
(q34 >= -52.5f && q34 <= 232.5f) && (q24 >= -227.5f && q24 <=
47.5f) && (q58 >= -105f && q58 <= 120f) && (q47 >= -270f && q47
<= 270f) && (q621 >= -270f && q621 <= 270f))
                        {
                                sum45 = Mathf.Abs (q1prev -
q11) + Mathf.Abs (q2prev - q24) + Mathf.Abs (q3prev - q34) +
Mathf.Abs  (q4prev  -  q47)  +  Mathf.Abs  (q5prev  -  q58)  +
Mathf.Abs(q6prev - q621);

                                if (sum45 < minSum)
                                {
                                        minSum = sum45;
                                        rotZ1 = q11;
                                        rotZ2 = q24;
                                        rotZ3 = q34;
                                        rotZ4 = q47;
                                        rotZ5 = q58;
                                        rotZ6 = q621;
                                }
                        }

                        if ((q11 >= -160f && q11 <= 160f) &&
(q34 >= -52.5f && q34 <= 232.5f) && (q24 >= -227.5f && q24 <=
47.5f) && (q58 >= -105f && q58 <= 120f) && (q48 >= -270f && q48
<= 270f) && (q622 >= -270f && q622 <= 270f))
                        {
                                sum46 = Mathf.Abs (q1prev -
q11) + Mathf.Abs (q2prev - q24) + Mathf.Abs (q3prev - q34) +
Mathf.Abs  (q4prev  -  q48)  +  Mathf.Abs  (q5prev  -  q58)  +
Mathf.Abs(q6prev - q622);

                                if (sum46 < minSum)
                                {
                                        minSum = sum46;
                                        rotZ1 = q11;
```

```
                                        rotZ2 = q24;
                                        rotZ3 = q34;
                                        rotZ4 = q48;
                                        rotZ5 = q58;
                                        rotZ6 = q622;
                                }
                        }

                        if ((q11 >= -160f && q11 <= 160f) &&
(q34 >= -52.5f && q34 <= 232.5f) && (q24 >= -227.5f && q24 <=
47.5f) && (q58 >= -105f && q58 <= 120f) && (q48 >= -270f && q48
<= 270f) && (q623 >= -270f && q623 <= 270f))
                        {
                                sum47 = Mathf.Abs (q1prev -
q11) + Mathf.Abs (q2prev - q24) + Mathf.Abs (q3prev - q34) +
Mathf.Abs  (q4prev  -  q48)  +  Mathf.Abs  (q5prev  -  q58)  +
Mathf.Abs(q6prev - q623);

                                if (sum47 < minSum)
                                {
                                        minSum = sum47;
                                        rotZ1 = q11;
                                        rotZ2 = q24;
                                        rotZ3 = q34;
                                        rotZ4 = q48;
                                        rotZ5 = q58;
                                        rotZ6 = q623;
                                }
                        }

                        if ((q11 >= -160f && q11 <= 160f) &&
(q34 >= -52.5f && q34 <= 232.5f) && (q24 >= -227.5f && q24 <=
47.5f) && (q58 >= -105f && q58 <= 120f) && (q48 >= -270f && q48
<= 270f) && (q624 >= -270f && q624 <= 270f))
                        {
                                sum48 = Mathf.Abs (q1prev -
q11) + Mathf.Abs (q2prev - q24) + Mathf.Abs (q3prev - q34) +
Mathf.Abs  (q4prev  -  q48)  +  Mathf.Abs  (q5prev  -  q58)  +
Mathf.Abs(q6prev - q624);

                                if (sum48 < minSum)
                                {
                                        minSum = sum48;
                                        rotZ1 = q11;
                                        rotZ2 = q24;
                                        rotZ3 = q34;
                                        rotZ4 = q48;
                                        rotZ5 = q58;
                                        rotZ6 = q624;
                                }
                        }

                        if ((q12 >= -160f && q12 <= 160f) &&
(q31 >= -52.5f && q31 <= 232.5f) && (q25 >= -227.5f && q25 <=
47.5f) && (q59 >= -105f && q59 <= 120f) && (q49 >= -270f && q49
<= 270f) && (q625 >= -270f && q625 <= 270f))
                        {
                                sum49 = Mathf.Abs (q1prev -
q12) + Mathf.Abs (q2prev - q25) + Mathf.Abs (q3prev - q31) +
Mathf.Abs  (q4prev  -  q49)  +  Mathf.Abs  (q5prev  -  q59)  +
Mathf.Abs(q6prev - q625);

                                if (sum49 < minSum)
                                {
                                        minSum = sum49;
                                        rotZ1 = q12;
                                        rotZ2 = q25;
                                        rotZ3 = q31;
                                        rotZ4 = q49;
                                        rotZ5 = q59;
                                        rotZ6 = q625;
                                }
                        }

                        if ((q12 >= -160f && q12 <= 160f) &&
(q31 >= -52.5f && q31 <= 232.5f) && (q25 >= -227.5f && q25 <=
47.5f) && (q59 >= -105f && q59 <= 120f) && (q49 >= -270f && q49
<= 270f) && (q626 >= -270f && q626 <= 270f))
                        {
                                sum50 = Mathf.Abs (q1prev -
q12) + Mathf.Abs (q2prev - q25) + Mathf.Abs (q3prev - q31) +
Mathf.Abs  (q4prev  -  q49)  +  Mathf.Abs  (q5prev  -  q59)  +
Mathf.Abs(q6prev - q626);

                                if (sum50 < minSum)
                                {
                                        minSum = sum50;
                                        rotZ1 = q12;
                                        rotZ2 = q25;
                                        rotZ3 = q31;
                                        rotZ4 = q49;
                                        rotZ5 = q59;
                                        rotZ6 = q626;
                                }
                        }

                        if ((q12 >= -160f && q12 <= 160f) &&
(q31 >= -52.5f && q31 <= 232.5f) && (q25 >= -227.5f && q25 <=
47.5f) && (q59 >= -105f && q59 <= 120f) && (q49 >= -270f && q49
<= 270f) && (q627 >= -270f && q627 <= 270f))
                        {
                                sum51 = Mathf.Abs (q1prev -
q12) + Mathf.Abs (q2prev - q25) + Mathf.Abs (q3prev - q31) +
Mathf.Abs  (q4prev  -  q49)  +  Mathf.Abs  (q5prev  -  q59)  +
Mathf.Abs(q6prev - q627);

                                if (sum51 < minSum)
                                {
                                        minSum = sum51;
                                        rotZ1 = q12;
                                        rotZ2 = q25;
                                        rotZ3 = q31;
                                        rotZ4 = q49;
                                        rotZ5 = q59;
                                        rotZ6 = q627;
                                }
                        }

                        if ((q12 >= -160f && q12 <= 160f) &&
(q31 >= -52.5f && q31 <= 232.5f) && (q25 >= -227.5f && q25 <=
47.5f) && (q59 >= -105f && q59 <= 120f) && (q410 >= -270f &&
q410 <= 270f) && (q628 >= -270f && q628 <= 270f))
                        {
                                sum52 = Mathf.Abs (q1prev -
q12) + Mathf.Abs (q2prev - q25) + Mathf.Abs (q3prev - q31) +
Mathf.Abs  (q4prev  -  q410)  +  Mathf.Abs  (q5prev  -  q59)  +
Mathf.Abs(q6prev - q628);

                                if (sum52 < minSum)
                                {
                                        minSum = sum52;
                                        rotZ1 = q12;
                                        rotZ2 = q25;
                                        rotZ3 = q31;
                                        rotZ4 = q410;
                                        rotZ5 = q59;
                                        rotZ6 = q628;
                                }
                        }

                        if ((q12 >= -160f && q12 <= 160f) &&
(q31 >= -52.5f && q31 <= 232.5f) && (q25 >= -227.5f && q25 <=
47.5f) && (q59 >= -105f && q59 <= 120f) && (q410 >= -270f &&
q410 <= 270f) && (q629 >= -270f && q629 <= 270f))
                        {
                                sum53 = Mathf.Abs (q1prev -
q12) + Mathf.Abs (q2prev - q25) + Mathf.Abs (q3prev - q31) +
Mathf.Abs  (q4prev  -  q410)  +  Mathf.Abs  (q5prev  -  q59)  +
Mathf.Abs(q6prev - q629);

                                if (sum53 < minSum)
                                {
                                        minSum = sum53;
                                        rotZ1 = q12;
                                        rotZ2 = q25;
                                        rotZ3 = q31;
                                        rotZ4 = q410;
                                        rotZ5 = q59;
                                        rotZ6 = q629;
                                }
                        }

                        if ((q12 >= -160f && q12 <= 160f) &&
(q31 >= -52.5f && q31 <= 232.5f) && (q25 >= -227.5f && q25 <=
47.5f) && (q59 >= -105f && q59 <= 120f) && (q410 >= -270f &&
q410 <= 270f) && (q630 >= -270f && q630 <= 270f))
                        {
                                sum54 = Mathf.Abs (q1prev -
q12) + Mathf.Abs (q2prev - q25) + Mathf.Abs (q3prev - q31) +
Mathf.Abs  (q4prev  -  q410)  +  Mathf.Abs  (q5prev  -  q59)  +
Mathf.Abs(q6prev - q630);

                                if (sum54 < minSum)
                                {
                                        minSum = sum54;
                                        rotZ1 = q12;
                                        rotZ2 = q25;
                                        rotZ3 = q31;
                                        rotZ4 = q410;
                                        rotZ5 = q59;
                                        rotZ6 = q630;
                                }
                        }

                        if ((q12 >= -160f && q12 <= 160f) &&
(q31 >= -52.5f && q31 <= 232.5f) && (q25 >= -227.5f && q25 <=
47.5f) && (q510 >= -105f && q510 <= 120f) && (q49 >= -270f &&
q49 <= 270f) && (q625 >= -270f && q625 <= 270f))
                        {
                                sum55 = Mathf.Abs (q1prev -
q12) + Mathf.Abs (q2prev - q25) + Mathf.Abs (q3prev - q31) +
Mathf.Abs  (q4prev  -  q49)  +  Mathf.Abs  (q5prev  -  q510)  +
Mathf.Abs(q6prev - q625);

                                if (sum55 < minSum)
                                {
                                        minSum = sum55;
                                        rotZ1 = q12;
                                        rotZ2 = q25;
                                        rotZ3 = q31;
                                        rotZ4 = q49;
                                        rotZ5 = q510;
                                        rotZ6 = q625;
                                }
                        }

                        if ((q12 >= -160f && q12 <= 160f) &&
(q31 >= -52.5f && q31 <= 232.5f) && (q25 >= -227.5f && q25 <=
47.5f) && (q510 >= -105f && q510 <= 120f) && (q49 >= -270f &&
q49 <= 270f) && (q626 >= -270f && q626 <= 270f))
                        {
                                sum56 = Mathf.Abs (q1prev -
q12) + Mathf.Abs (q2prev - q25) + Mathf.Abs (q3prev - q31) +
Mathf.Abs  (q4prev  -  q49)  +  Mathf.Abs  (q5prev  -  q510)  +
Mathf.Abs(q6prev - q626);

                                if (sum56 < minSum)
                                {
                                        minSum = sum56;
                                        rotZ1 = q12;
                                        rotZ2 = q25;
                                        rotZ3 = q31;
                                        rotZ4 = q49;
                                        rotZ5 = q510;
                                        rotZ6 = q626;
                                }
                        }

                        if ((q12 >= -160f && q12 <= 160f) &&
(q31 >= -52.5f && q31 <= 232.5f) && (q25 >= -227.5f && q25 <=
47.5f) && (q510 >= -105f && q510 <= 120f) && (q49 >= -270f &&
q49 <= 270f) && (q627 >= -270f && q627 <= 270f))
                        {
                                sum57 = Mathf.Abs (q1prev -
q12) + Mathf.Abs (q2prev - q25) + Mathf.Abs (q3prev - q31) +
Mathf.Abs  (q4prev  -  q49)  +  Mathf.Abs  (q5prev  -  q510)  +
Mathf.Abs(q6prev - q627);

                                if (sum57 < minSum)
                                {
                                        minSum = sum57;
                                        rotZ1 = q12;
                                        rotZ2 = q25;
                                        rotZ3 = q31;
```

```
                                        rotZ4 = q49;
                                        rotZ5 = q510;
                                        rotZ6 = q627;
                                }
                        }

                        if ((q12 >= -160f && q12 <= 160f) &&
(q31 >= -52.5f && q31 <= 232.5f) && (q25 >= -227.5f && q25 <=
47.5f) && (q510 >= -105f && q510 <= 120f) && (q410 >= -270f &&
q410 <= 270f) && (q628 >= -270f && q628 <= 270f))
                        {
                                sum58 = Mathf.Abs (q1prev -
q12) + Mathf.Abs (q2prev - q25) + Mathf.Abs (q3prev - q31) +
Mathf.Abs  (q4prev  -  q410)  +  Mathf.Abs  (q5prev  -  q510)  +
Mathf.Abs(q6prev - q628);

                                if (sum58 < minSum)
                                {
                                        minSum = sum58;
                                        rotZ1 = q12;
                                        rotZ2 = q25;
                                        rotZ3 = q31;
                                        rotZ4 = q410;
                                        rotZ5 = q510;
                                        rotZ6 = q628;
                                }
                        }

                        if ((q12 >= -160f && q12 <= 160f) &&
(q31 >= -52.5f && q31 <= 232.5f) && (q25 >= -227.5f && q25 <=
47.5f) && (q510 >= -105f && q510 <= 120f) && (q410 >= -270f &&
q410 <= 270f) && (q629 >= -270f && q629 <= 270f))
                        {
                                sum59 = Mathf.Abs (q1prev -
q12) + Mathf.Abs (q2prev - q25) + Mathf.Abs (q3prev - q31) +
Mathf.Abs  (q4prev  -  q410)  +  Mathf.Abs  (q5prev  -  q510)  +
Mathf.Abs(q6prev - q629);

                                if (sum59 < minSum)
                                {
                                        minSum = sum59;
                                        rotZ1 = q12;
                                        rotZ2 = q25;
                                        rotZ3 = q31;
                                        rotZ4 = q410;
                                        rotZ5 = q510;
                                        rotZ6 = q629;
                                }
                        }

                        if ((q12 >= -160f && q12 <= 160f) &&
(q31 >= -52.5f && q31 <= 232.5f) && (q25 >= -227.5f && q25 <=
47.5f) && (q510 >= -105f && q510 <= 120f) && (q410 >= -270f &&
q410 <= 270f) && (q630 >= -270f && q630 <= 270f))
                        {
                                sum60 = Mathf.Abs (q1prev -
q12) + Mathf.Abs (q2prev - q25) + Mathf.Abs (q3prev - q31) +
Mathf.Abs  (q4prev  -  q410)  +  Mathf.Abs  (q5prev  -  q510)  +
Mathf.Abs(q6prev - q630);

                                if (sum60 < minSum)
                                {
                                        minSum = sum60;
                                        rotZ1 = q12;
                                        rotZ2 = q25;
                                        rotZ3 = q31;
                                        rotZ4 = q410;
                                        rotZ5 = q510;
                                        rotZ6 = q630;
                                }
                        }

                        if ((q12 >= -160f && q12 <= 160f) &&
(q32 >= -52.5f && q32 <= 232.5f) && (q26 >= -227.5f && q26 <=
47.5f) && (q511 >= -105f && q511 <= 120f) && (q411 >= -270f &&
q411 <= 270f) && (q631 >= -270f && q631 <= 270f))
                        {
                                sum61 = Mathf.Abs (q1prev -
q12) + Mathf.Abs (q2prev - q26) + Mathf.Abs (q3prev - q32) +
Mathf.Abs  (q4prev  -  q411)  +  Mathf.Abs  (q5prev  -  q511)  +
Mathf.Abs(q6prev - q631);

                                if (sum61 < minSum)
                                {
                                        minSum = sum61;
                                        rotZ1 = q12;
                                        rotZ2 = q26;
                                        rotZ3 = q32;
                                        rotZ4 = q411;
                                        rotZ5 = q511;
                                        rotZ6 = q631;
                                }
                        }

                        if ((q12 >= -160f && q12 <= 160f) &&
(q32 >= -52.5f && q32 <= 232.5f) && (q26 >= -227.5f && q26 <=
47.5f) && (q511 >= -105f && q511 <= 120f) && (q411 >= -270f &&
q411 <= 270f) && (q632 >= -270f && q632 <= 270f))
                        {
                                sum62 = Mathf.Abs (q1prev -
q12) + Mathf.Abs (q2prev - q26) + Mathf.Abs (q3prev - q32) +
Mathf.Abs  (q4prev  -  q411)  +  Mathf.Abs  (q5prev  -  q511)  +
Mathf.Abs(q6prev - q632);

                                if (sum62 < minSum)
                                {
                                        minSum = sum62;
                                        rotZ1 = q12;
                                        rotZ2 = q26;
                                        rotZ3 = q32;
                                        rotZ4 = q411;
                                        rotZ5 = q511;
                                        rotZ6 = q632;
                                }
                        }

                        if ((q12 >= -160f && q12 <= 160f) &&
(q32 >= -52.5f && q32 <= 232.5f) && (q26 >= -227.5f && q26 <=
47.5f) && (q511 >= -105f && q511 <= 120f) && (q411 >= -270f &&
q411 <= 270f) && (q633 >= -270f && q633 <= 270f))
                        {
                                sum63 = Mathf.Abs (q1prev -
q12) + Mathf.Abs (q2prev - q26) + Mathf.Abs (q3prev - q32) +
Mathf.Abs  (q4prev  -  q411)  +  Mathf.Abs  (q5prev  -  q511)  +
Mathf.Abs(q6prev - q633);

                                if (sum63 < minSum)
                                {
                                        minSum = sum63;
                                        rotZ1 = q12;
                                        rotZ2 = q26;
                                        rotZ3 = q32;
                                        rotZ4 = q411;
                                        rotZ5 = q511;
                                        rotZ6 = q633;
                                }
                        }

                        if ((q12 >= -160f && q12 <= 160f) &&
(q32 >= -52.5f && q32 <= 232.5f) && (q26 >= -227.5f && q26 <=
47.5f) && (q511 >= -105f && q511 <= 120f) && (q412 >= -270f &&
q412 <= 270f) && (q634 >= -270f && q634 <= 270f))
                        {
                                sum64 = Mathf.Abs (q1prev -
q12) + Mathf.Abs (q2prev - q26) + Mathf.Abs (q3prev - q32) +
Mathf.Abs  (q4prev  -  q412)  +  Mathf.Abs  (q5prev  -  q511)  +
Mathf.Abs(q6prev - q634);

                                if (sum64 < minSum)
                                {
                                        minSum = sum64;
                                        rotZ1 = q12;
                                        rotZ2 = q26;
                                        rotZ3 = q32;
                                        rotZ4 = q412;
                                        rotZ5 = q511;
                                        rotZ6 = q634;
                                }
                        }

                        if ((q12 >= -160f && q12 <= 160f) &&
(q32 >= -52.5f && q32 <= 232.5f) && (q26 >= -227.5f && q26 <=
47.5f) && (q511 >= -105f && q511 <= 120f) && (q412 >= -270f &&
q412 <= 270f) && (q635 >= -270f && q635 <= 270f))
                        {
                                sum65 = Mathf.Abs (q1prev -
q12) + Mathf.Abs (q2prev - q26) + Mathf.Abs (q3prev - q32) +
Mathf.Abs  (q4prev  -  q412)  +  Mathf.Abs  (q5prev  -  q511)  +
Mathf.Abs(q6prev - q635);

                                if (sum65 < minSum)
                                {
                                        minSum = sum65;
                                        rotZ1 = q12;
                                        rotZ2 = q26;
                                        rotZ3 = q32;
                                        rotZ4 = q412;
                                        rotZ5 = q511;
                                        rotZ6 = q635;
                                }
                        }

                        if ((q12 >= -160f && q12 <= 160f) &&
(q32 >= -52.5f && q32 <= 232.5f) && (q26 >= -227.5f && q26 <=
47.5f) && (q511 >= -105f && q511 <= 120f) && (q412 >= -270f &&
q412 <= 270f) && (q636 >= -270f && q636 <= 270f))
                        {
                                sum66 = Mathf.Abs (q1prev -
q12) + Mathf.Abs (q2prev - q26) + Mathf.Abs (q3prev - q32) +
Mathf.Abs  (q4prev  -  q412)  +  Mathf.Abs  (q5prev  -  q511)  +
Mathf.Abs(q6prev - q636);

                                if (sum66 < minSum)
                                {
                                        minSum = sum66;
                                        rotZ1 = q12;
                                        rotZ2 = q26;
                                        rotZ3 = q32;
                                        rotZ4 = q412;
                                        rotZ5 = q511;
                                        rotZ6 = q636;
                                }
                        }

                        if ((q12 >= -160f && q12 <= 160f) &&
(q32 >= -52.5f && q32 <= 232.5f) && (q26 >= -227.5f && q26 <=
47.5f) && (q512 >= -105f && q512 <= 120f) && (q411 >= -270f &&
q411 <= 270f) && (q631 >= -270f && q631 <= 270f))
                        {
                                sum67 = Mathf.Abs (q1prev -
q12) + Mathf.Abs (q2prev - q26) + Mathf.Abs (q3prev - q32) +
Mathf.Abs  (q4prev  -  q411)  +  Mathf.Abs  (q5prev  -  q512)  +
Mathf.Abs(q6prev - q631);

                                if (sum67 < minSum)
                                {
                                        minSum = sum67;
                                        rotZ1 = q12;
                                        rotZ2 = q26;
                                        rotZ3 = q32;
                                        rotZ4 = q411;
                                        rotZ5 = q512;
                                        rotZ6 = q631;
                                }
                        }

                        if ((q12 >= -160f && q12 <= 160f) &&
(q32 >= -52.5f && q32 <= 232.5f) && (q26 >= -227.5f && q26 <=
47.5f) && (q512 >= -105f && q512 <= 120f) && (q411 >= -270f &&
q411 <= 270f) && (q632 >= -270f && q632 <= 270f))
                        {
                                sum68 = Mathf.Abs (q1prev -
q12) + Mathf.Abs (q2prev - q26) + Mathf.Abs (q3prev - q32) +
Mathf.Abs  (q4prev  -  q411)  +  Mathf.Abs  (q5prev  -  q512)  +
Mathf.Abs(q6prev - q632);

                                if (sum68 < minSum)
                                {
                                        minSum = sum68;
                                        rotZ1 = q12;
                                        rotZ2 = q26;
                                        rotZ3 = q32;
                                        rotZ4 = q411;
```

```
                                        rotZ5 = q512;
                                        rotZ6 = q632;
                                }
                        }

                        if ((q12 >= -160f && q12 <= 160f) &&
(q32 >= -52.5f && q32 <= 232.5f) && (q26 >= -227.5f && q26 <=
47.5f) && (q512 >= -105f && q512 <= 120f) && (q411 >= -270f &&
q411 <= 270f) && (q633 >= -270f && q633 <= 270f))
                        {
                                sum69 = Mathf.Abs (q1prev -
q12) + Mathf.Abs (q2prev - q26) + Mathf.Abs (q3prev - q32) +
Mathf.Abs  (q4prev  -  q411)  +  Mathf.Abs  (q5prev  -  q512)  +
Mathf.Abs(q6prev - q633);

                                if (sum69 < minSum)
                                {
                                        minSum = sum69;
                                        rotZ1 = q12;
                                        rotZ2 = q26;
                                        rotZ3 = q32;
                                        rotZ4 = q411;
                                        rotZ5 = q512;
                                        rotZ6 = q633;
                                }
                        }

                        if ((q12 >= -160f && q12 <= 160f) &&
(q32 >= -52.5f && q32 <= 232.5f) && (q26 >= -227.5f && q26 <=
47.5f) && (q512 >= -105f && q512 <= 120f) && (q412 >= -270f &&
q412 <= 270f) && (q634 >= -270f && q634 <= 270f))
                        {
                                sum70 = Mathf.Abs (q1prev -
q12) + Mathf.Abs (q2prev - q26) + Mathf.Abs (q3prev - q32) +
Mathf.Abs  (q4prev  -  q412)  +  Mathf.Abs  (q5prev  -  q512)  +
Mathf.Abs(q6prev - q634);

                                if (sum70 < minSum)
                                {
                                        minSum = sum70;
                                        rotZ1 = q12;
                                        rotZ2 = q26;
                                        rotZ3 = q32;
                                        rotZ4 = q412;
                                        rotZ5 = q512;
                                        rotZ6 = q634;
                                }
                        }

                        if ((q12 >= -160f && q12 <= 160f) &&
(q32 >= -52.5f && q32 <= 232.5f) && (q26 >= -227.5f && q26 <=
47.5f) && (q512 >= -105f && q512 <= 120f) && (q412 >= -270f &&
q412 <= 270f) && (q635 >= -270f && q635 <= 270f))
                        {
                                sum71 = Mathf.Abs (q1prev -
q12) + Mathf.Abs (q2prev - q26) + Mathf.Abs (q3prev - q32) +
Mathf.Abs  (q4prev  -  q412)  +  Mathf.Abs  (q5prev  -  q512)  +
Mathf.Abs(q6prev - q635);

                                if (sum71 < minSum)
                                {
                                        minSum = sum71;
                                        rotZ1 = q12;
                                        rotZ2 = q26;
                                        rotZ3 = q32;
                                        rotZ4 = q412;
                                        rotZ5 = q512;
                                        rotZ6 = q635;
                                }
                        }

                        if ((q12 >= -160f && q12 <= 160f) &&
(q32 >= -52.5f && q32 <= 232.5f) && (q26 >= -227.5f && q26 <=
47.5f) && (q512 >= -105f && q512 <= 120f) && (q412 >= -270f &&
q412 <= 270f) && (q636 >= -270f && q636 <= 270f))
                        {
                                sum72 = Mathf.Abs (q1prev -
q12) + Mathf.Abs (q2prev - q26) + Mathf.Abs (q3prev - q32) +
Mathf.Abs  (q4prev  -  q412)  +  Mathf.Abs  (q5prev  -  q512)  +
Mathf.Abs(q6prev - q636);

                                if (sum72 < minSum)
                                {
                                        minSum = sum72;
                                        rotZ1 = q12;
                                        rotZ2 = q26;
                                        rotZ3 = q32;
                                        rotZ4 = q412;
                                        rotZ5 = q512;
                                        rotZ6 = q636;
                                }
                        }

                        if ((q12 >= -160f && q12 <= 160f) &&
(q33 >= -52.5f && q33 <= 232.5f) && (q27 >= -227.5f && q27 <=
47.5f) && (q513 >= -105f && q513 <= 120f) && (q413 >= -270f &&
q413 <= 270f) && (q637 >= -270f && q637 <= 270f))
                        {
                                sum73 = Mathf.Abs (q1prev -
q12) + Mathf.Abs (q2prev - q27) + Mathf.Abs (q3prev - q33) +
Mathf.Abs  (q4prev  -  q413)  +  Mathf.Abs  (q5prev  -  q513)  +
Mathf.Abs(q6prev - q637);

                                if (sum73 < minSum)
                                {
                                        minSum = sum73;
                                        rotZ1 = q12;
                                        rotZ2 = q27;
                                        rotZ3 = q33;
                                        rotZ4 = q413;
                                        rotZ5 = q513;
                                        rotZ6 = q637;
                                }
                        }

                        if ((q12 >= -160f && q12 <= 160f) &&
(q33 >= -52.5f && q33 <= 232.5f) && (q27 >= -227.5f && q27 <=
47.5f) && (q513 >= -105f && q513 <= 120f) && (q413 >= -270f &&
q413 <= 270f) && (q638 >= -270f && q638 <= 270f))
                        {
                                sum74 = Mathf.Abs (q1prev -
q12) + Mathf.Abs (q2prev - q27) + Mathf.Abs (q3prev - q33) +
Mathf.Abs  (q4prev  -  q413)  +  Mathf.Abs  (q5prev  -  q513)  +
Mathf.Abs(q6prev - q638);

                                if (sum74 < minSum)
                                {
                                        minSum = sum74;
                                        rotZ1 = q12;
                                        rotZ2 = q27;
                                        rotZ3 = q33;
                                        rotZ4 = q413;
                                        rotZ5 = q513;
                                        rotZ6 = q638;
                                }
                        }

                        if ((q12 >= -160f && q12 <= 160f) &&
(q33 >= -52.5f && q33 <= 232.5f) && (q27 >= -227.5f && q27 <=
47.5f) && (q513 >= -105f && q513 <= 120f) && (q413 >= -270f &&
q413 <= 270f) && (q639 >= -270f && q639 <= 270f))
                        {
                                sum75 = Mathf.Abs (q1prev -
q12) + Mathf.Abs (q2prev - q27) + Mathf.Abs (q3prev - q33) +
Mathf.Abs  (q4prev  -  q413)  +  Mathf.Abs  (q5prev  -  q513)  +
Mathf.Abs(q6prev - q639);

                                if (sum75 < minSum)
                                {
                                        minSum = sum75;
                                        rotZ1 = q12;
                                        rotZ2 = q27;
                                        rotZ3 = q33;
                                        rotZ4 = q413;
                                        rotZ5 = q513;
                                        rotZ6 = q639;
                                }
                        }


                        if ((q12 >= -160f && q12 <= 160f) &&
(q33 >= -52.5f && q33 <= 232.5f) && (q27 >= -227.5f && q27 <=
47.5f) && (q513 >= -105f && q513 <= 120f) && (q414 >= -270f &&
q414 <= 270f) && (q640 >= -270f && q640 <= 270f))
                        {
                                sum76 = Mathf.Abs (q1prev -
q12) + Mathf.Abs (q2prev - q27) + Mathf.Abs (q3prev - q33) +
Mathf.Abs  (q4prev  -  q414)  +  Mathf.Abs  (q5prev  -  q513)  +
Mathf.Abs(q6prev - q640);

                                if (sum76 < minSum)
                                {
                                        minSum = sum76;
                                        rotZ1 = q12;
                                        rotZ2 = q27;
                                        rotZ3 = q33;
                                        rotZ4 = q414;
                                        rotZ5 = q513;
                                        rotZ6 = q640;
                                }
                        }

                        if ((q12 >= -160f && q12 <= 160f) &&
(q33 >= -52.5f && q33 <= 232.5f) && (q27 >= -227.5f && q27 <=
47.5f) && (q513 >= -105f && q513 <= 120f) && (q414 >= -270f &&
q414 <= 270f) && (q641 >= -270f && q641 <= 270f))
                        {
                                sum77 = Mathf.Abs (q1prev -
q12) + Mathf.Abs (q2prev - q27) + Mathf.Abs (q3prev - q33) +
Mathf.Abs  (q4prev  -  q414)  +  Mathf.Abs  (q5prev  -  q513)  +
Mathf.Abs(q6prev - q641);

                                if (sum77 < minSum)
                                {
                                        minSum = sum77;
                                        rotZ1 = q12;
                                        rotZ2 = q27;
                                        rotZ3 = q33;
                                        rotZ4 = q414;
                                        rotZ5 = q513;
                                        rotZ6 = q641;
                                }
                        }

                        if ((q12 >= -160f && q12 <= 160f) &&
(q33 >= -52.5f && q33 <= 232.5f) && (q27 >= -227.5f && q27 <=
47.5f) && (q513 >= -105f && q513 <= 120f) && (q414 >= -270f &&
q414 <= 270f) && (q642 >= -270f && q642 <= 270f))
                        {
                                sum78 = Mathf.Abs (q1prev -
q12) + Mathf.Abs (q2prev - q27) + Mathf.Abs (q3prev - q33) +
Mathf.Abs  (q4prev  -  q414)  +  Mathf.Abs  (q5prev  -  q513)  +
Mathf.Abs(q6prev - q642);

                                if (sum78 < minSum)
                                {
                                        minSum = sum78;
                                        rotZ1 = q12;
                                        rotZ2 = q27;
                                        rotZ3 = q33;
                                        rotZ4 = q414;
                                        rotZ5 = q513;
                                        rotZ6 = q642;
                                }
                        }

                        if ((q12 >= -160f && q12 <= 160f) &&
(q33 >= -52.5f && q33 <= 232.5f) && (q27 >= -227.5f && q27 <=
47.5f) && (q514 >= -105f && q514 <= 120f) && (q413 >= -270f &&
q413 <= 270f) && (q637 >= -270f && q637 <= 270f))
                        {
                                sum79 = Mathf.Abs (q1prev -
q12) + Mathf.Abs (q2prev - q27) + Mathf.Abs (q3prev - q33) +
Mathf.Abs  (q4prev  -  q413)  +  Mathf.Abs  (q5prev  -  q514)  +
Mathf.Abs(q6prev - q637);

                                if (sum79 < minSum)
                                {
                                        minSum = sum79;
                                        rotZ1 = q12;
                                        rotZ2 = q27;
                                        rotZ3 = q33;
                                        rotZ4 = q413;
                                        rotZ5 = q514;
```

```
                                        rotZ6 = q637;
                                }
                        }

                        if ((q12 >= -160f && q12 <= 160f) &&
(q33 >= -52.5f && q33 <= 232.5f) && (q27 >= -227.5f && q27 <=
47.5f) && (q514 >= -105f && q514 <= 120f) && (q413 >= -270f &&
q413 <= 270f) && (q638 >= -270f && q638 <= 270f))
                        {
                                sum80 = Mathf.Abs (q1prev -
q12) + Mathf.Abs (q2prev - q27) + Mathf.Abs (q3prev - q33) +
Mathf.Abs  (q4prev  -  q413)  +  Mathf.Abs  (q5prev  -  q514)  +
Mathf.Abs(q6prev - q638);

                                if (sum80 < minSum)
                                {
                                        minSum = sum80;
                                        rotZ1 = q12;
                                        rotZ2 = q27;
                                        rotZ3 = q33;
                                        rotZ4 = q413;
                                        rotZ5 = q514;
                                        rotZ6 = q638;
                                }
                        }

                        if ((q12 >= -160f && q12 <= 160f) &&
(q33 >= -52.5f && q33 <= 232.5f) && (q27 >= -227.5f && q27 <=
47.5f) && (q514 >= -105f && q514 <= 120f) && (q413 >= -270f &&
q413 <= 270f) && (q639 >= -270f && q639 <= 270f))
                        {
                                sum81 = Mathf.Abs (q1prev -
q12) + Mathf.Abs (q2prev - q27) + Mathf.Abs (q3prev - q33) +
Mathf.Abs  (q4prev  -  q413)  +  Mathf.Abs  (q5prev  -  q514)  +
Mathf.Abs(q6prev - q639);

                                if (sum81 < minSum)
                                {
                                        minSum = sum81;
                                        rotZ1 = q12;
                                        rotZ2 = q27;
                                        rotZ3 = q33;
                                        rotZ4 = q413;
                                        rotZ5 = q514;
                                        rotZ6 = q639;
                                }
                        }

                        if ((q12 >= -160f && q12 <= 160f) &&
(q33 >= -52.5f && q33 <= 232.5f) && (q27 >= -227.5f && q27 <=
47.5f) && (q514 >= -105f && q514 <= 120f) && (q414 >= -270f &&
q414 <= 270f) && (q640 >= -270f && q640 <= 270f))
                        {
                                sum82 = Mathf.Abs (q1prev -
q12) + Mathf.Abs (q2prev - q27) + Mathf.Abs (q3prev - q33) +
Mathf.Abs  (q4prev  -  q414)  +  Mathf.Abs  (q5prev  -  q514)  +
Mathf.Abs(q6prev - q640);

                                if (sum82 < minSum)
                                {
                                        minSum = sum82;
                                        rotZ1 = q12;
                                        rotZ2 = q27;
                                        rotZ3 = q33;
                                        rotZ4 = q414;
                                        rotZ5 = q514;
                                        rotZ6 = q640;
                                }
                        }

                        if ((q12 >= -160f && q12 <= 160f) &&
(q33 >= -52.5f && q33 <= 232.5f) && (q27 >= -227.5f && q27 <=
47.5f) && (q514 >= -105f && q514 <= 120f) && (q414 >= -270f &&
q414 <= 270f) && (q641 >= -270f && q641 <= 270f))
                        {
                                sum83 = Mathf.Abs (q1prev -
q12) + Mathf.Abs (q2prev - q27) + Mathf.Abs (q3prev - q33) +
Mathf.Abs  (q4prev  -  q414)  +  Mathf.Abs  (q5prev  -  q514)  +
Mathf.Abs(q6prev - q641);

                                if (sum83 < minSum)
                                {
                                        minSum = sum83;
                                        rotZ1 = q12;
                                        rotZ2 = q27;
                                        rotZ3 = q33;
                                        rotZ4 = q414;
                                        rotZ5 = q514;
                                        rotZ6 = q641;
                                }
                        }

                        if ((q12 >= -160f && q12 <= 160f) &&
(q33 >= -52.5f && q33 <= 232.5f) && (q27 >= -227.5f && q27 <=
47.5f) && (q514 >= -105f && q514 <= 120f) && (q414 >= -270f &&
q414 <= 270f) && (q642 >= -270f && q642 <= 270f))
                        {
                                sum84 = Mathf.Abs (q1prev -
q12) + Mathf.Abs (q2prev - q27) + Mathf.Abs (q3prev - q33) +
Mathf.Abs  (q4prev  -  q414)  +  Mathf.Abs  (q5prev  -  q514)  +
Mathf.Abs(q6prev - q642);

                                if (sum84 < minSum)
                                {
                                        minSum = sum84;
                                        rotZ1 = q12;
                                        rotZ2 = q27;
                                        rotZ3 = q33;
                                        rotZ4 = q414;
                                        rotZ5 = q514;
                                        rotZ6 = q642;
                                }
                        }

                        if ((q12 >= -160f && q12 <= 160f) &&
(q34 >= -52.5f && q34 <= 232.5f) && (q28 >= -227.5f && q28 <=
47.5f) && (q515 >= -105f && q515 <= 120f) && (q415 >= -270f &&
q415 <= 270f) && (q643 >= -270f && q643 <= 270f))
                        {
                                sum85 = Mathf.Abs (q1prev -
q12) + Mathf.Abs (q2prev - q28) + Mathf.Abs (q3prev - q34) +
Mathf.Abs  (q4prev  -  q415)  +  Mathf.Abs  (q5prev  -  q515)  +
Mathf.Abs(q6prev - q643);

                                if (sum85 < minSum)
                                {
                                        minSum = sum85;
                                        rotZ1 = q12;
                                        rotZ2 = q28;
                                        rotZ3 = q34;
                                        rotZ4 = q415;
                                        rotZ5 = q515;
                                        rotZ6 = q643;
                                }
                        }

                        if ((q12 >= -160f && q12 <= 160f) &&
(q34 >= -52.5f && q34 <= 232.5f) && (q28 >= -227.5f && q28 <=
47.5f) && (q515 >= -105f && q515 <= 120f) && (q415 >= -270f &&
q415 <= 270f) && (q644 >= -270f && q644 <= 270f))
                        {
                                sum86 = Mathf.Abs (q1prev -
q12) + Mathf.Abs (q2prev - q28) + Mathf.Abs (q3prev - q34) +
Mathf.Abs  (q4prev  -  q415)  +  Mathf.Abs  (q5prev  -  q515)  +
Mathf.Abs(q6prev - q644);

                                if (sum86 < minSum)
                                {
                                        minSum = sum86;
                                        rotZ1 = q12;
                                        rotZ2 = q28;
                                        rotZ3 = q34;
                                        rotZ4 = q415;
                                        rotZ5 = q515;
                                        rotZ6 = q644;
                                }
                        }

                        if ((q12 >= -160f && q12 <= 160f) &&
(q34 >= -52.5f && q34 <= 232.5f) && (q28 >= -227.5f && q28 <=
47.5f) && (q515 >= -105f && q515 <= 120f) && (q415 >= -270f &&
q415 <= 270f) && (q645 >= -270f && q645 <= 270f))
                        {
                                sum87 = Mathf.Abs (q1prev -
q12) + Mathf.Abs (q2prev - q28) + Mathf.Abs (q3prev - q34) +
Mathf.Abs  (q4prev  -  q415)  +  Mathf.Abs  (q5prev  -  q515)  +
Mathf.Abs(q6prev - q645);

                                if (sum87 < minSum)
                                {
                                        minSum = sum87;
                                        rotZ1 = q12;
                                        rotZ2 = q28;
                                        rotZ3 = q34;
                                        rotZ4 = q415;
                                        rotZ5 = q515;
                                        rotZ6 = q645;
                                }
                        }

                        if ((q12 >= -160f && q12 <= 160f) &&
(q34 >= -52.5f && q34 <= 232.5f) && (q28 >= -227.5f && q28 <=
47.5f) && (q515 >= -105f && q515 <= 120f) && (q416 >= -270f &&
q416 <= 270f) && (q646 >= -270f && q646 <= 270f))
                        {
                                sum88 = Mathf.Abs (q1prev -
q12) + Mathf.Abs (q2prev - q28) + Mathf.Abs (q3prev - q34) +
Mathf.Abs  (q4prev  -  q416)  +  Mathf.Abs  (q5prev  -  q515)  +
Mathf.Abs(q6prev - q646);

                                if (sum88 < minSum)
                                {
                                        minSum = sum88;
                                        rotZ1 = q12;
                                        rotZ2 = q28;
                                        rotZ3 = q34;
                                        rotZ4 = q416;
                                        rotZ5 = q515;
                                        rotZ6 = q646;
                                }
                        }

                        if ((q12 >= -160f && q12 <= 160f) &&
(q34 >= -52.5f && q34 <= 232.5f) && (q28 >= -227.5f && q28 <=
47.5f) && (q515 >= -105f && q515 <= 120f) && (q416 >= -270f &&
q416 <= 270f) && (q647 >= -270f && q647 <= 270f))
                        {
                                sum89 = Mathf.Abs (q1prev -
q12) + Mathf.Abs (q2prev - q28) + Mathf.Abs (q3prev - q34) +
Mathf.Abs  (q4prev  -  q416)  +  Mathf.Abs  (q5prev  -  q515)  +
Mathf.Abs(q6prev - q647);

                                if (sum89 < minSum)
                                {
                                        minSum = sum89;
                                        rotZ1 = q12;
                                        rotZ2 = q28;
                                        rotZ3 = q34;
                                        rotZ4 = q416;
                                        rotZ5 = q515;
                                        rotZ6 = q647;
                                }
                        }

                        if ((q12 >= -160f && q12 <= 160f) &&
(q34 >= -52.5f && q34 <= 232.5f) && (q28 >= -227.5f && q28 <=
47.5f) && (q515 >= -105f && q515 <= 120f) && (q416 >= -270f &&
q416 <= 270f) && (q648 >= -270f && q648 <= 270f))
                        {
                                sum90 = Mathf.Abs (q1prev -
q12) + Mathf.Abs (q2prev - q28) + Mathf.Abs (q3prev - q34) +
Mathf.Abs  (q4prev  -  q416)  +  Mathf.Abs  (q5prev  -  q515)  +
Mathf.Abs(q6prev - q648);

                                if (sum90 < minSum)
                                {
                                        minSum = sum90;
                                        rotZ1 = q12;
                                        rotZ2 = q28;
                                        rotZ3 = q34;
                                        rotZ4 = q416;
                                        rotZ5 = q515;
                                        rotZ6 = q648;
```

```
                        }
                    }

                    if ((q12 >= -160f && q12 <= 160f) &&
(q34 >= -52.5f && q34 <= 232.5f) && (q28 >= -227.5f && q28 <=
47.5f) && (q516 >= -105f && q516 <= 120f) && (q415 >= -270f &&
q415 <= 270f) && (q643 >= -270f && q643 <= 270f))
                    {
                        sum91 = Mathf.Abs (q1prev -
q12) + Mathf.Abs (q2prev - q28) + Mathf.Abs (q3prev - q34) +
Mathf.Abs  (q4prev  -  q415)  +  Mathf.Abs  (q5prev  -  q516)  +
Mathf.Abs(q6prev - q643);

                        if (sum91 < minSum)
                        {
                            minSum = sum91;
                            rotZ1 = q12;
                            rotZ2 = q28;
                            rotZ3 = q34;
                            rotZ4 = q415;
                            rotZ5 = q516;
                            rotZ6 = q643;
                        }
                    }

                    if ((q12 >= -160f && q12 <= 160f) &&
(q34 >= -52.5f && q34 <= 232.5f) && (q28 >= -227.5f && q28 <=
47.5f) && (q516 >= -105f && q516 <= 120f) && (q415 >= -270f &&
q415 <= 270f) && (q644 >= -270f && q644 <= 270f))
                    {
                        sum92 = Mathf.Abs (q1prev -
q12) + Mathf.Abs (q2prev - q28) + Mathf.Abs (q3prev - q34) +
Mathf.Abs  (q4prev  -  q415)  +  Mathf.Abs  (q5prev  -  q516)  +
Mathf.Abs(q6prev - q644);

                        if (sum92 < minSum)
                        {
                            minSum = sum92;
                            rotZ1 = q12;
                            rotZ2 = q28;
                            rotZ3 = q34;
                            rotZ4 = q415;
                            rotZ5 = q516;
                            rotZ6 = q644;
                        }
                    }

                    if ((q12 >= -160f && q12 <= 160f) &&
(q34 >= -52.5f && q34 <= 232.5f) && (q28 >= -227.5f && q28 <=
47.5f) && (q516 >= -105f && q516 <= 120f) && (q415 >= -270f &&
q415 <= 270f) && (q645 >= -270f && q645 <= 270f))
                    {
                        sum93 = Mathf.Abs (q1prev -
q12) + Mathf.Abs (q2prev - q28) + Mathf.Abs (q3prev - q34) +
Mathf.Abs  (q4prev  -  q415)  +  Mathf.Abs  (q5prev  -  q516)  +
Mathf.Abs(q6prev - q645);

                        if (sum93 < minSum)
                        {
                            minSum = sum93;
                            rotZ1 = q12;
                            rotZ2 = q28;
                            rotZ3 = q34;
                            rotZ4 = q415;
                            rotZ5 = q516;
                            rotZ6 = q645;
                        }
                    }

                    if ((q12 >= -160f && q12 <= 160f) &&
(q34 >= -52.5f && q34 <= 232.5f) && (q28 >= -227.5f && q28 <=
47.5f) && (q516 >= -105f && q516 <= 120f) && (q416 >= -270f &&
q416 <= 270f) && (q646 >= -270f && q646 <= 270f))
                    {
                        sum94 = Mathf.Abs (q1prev -
q12) + Mathf.Abs (q2prev - q28) + Mathf.Abs (q3prev - q34) +
Mathf.Abs  (q4prev  -  q416)  +  Mathf.Abs  (q5prev  -  q516)  +
Mathf.Abs(q6prev - q646);

                        if (sum94 < minSum)
                        {
                            minSum = sum94;
                            rotZ1 = q12;
                            rotZ2 = q28;
                            rotZ3 = q34;
                            rotZ4 = q416;
                            rotZ5 = q516;
                            rotZ6 = q646;
                        }
                    }

                    if ((q12 >= -160f && q12 <= 160f) &&
(q34 >= -52.5f && q34 <= 232.5f) && (q28 >= -227.5f && q28 <=
47.5f) && (q516 >= -105f && q516 <= 120f) && (q416 >= -270f &&
q416 <= 270f) && (q647 >= -270f && q647 <= 270f))
                    {
                        sum95 = Mathf.Abs (q1prev -
q12) + Mathf.Abs (q2prev - q28) + Mathf.Abs (q3prev - q34) +
Mathf.Abs  (q4prev  -  q416)  +  Mathf.Abs  (q5prev  -  q516)  +
Mathf.Abs(q6prev - q647);

                        if (sum95 < minSum)
                        {
                            minSum = sum95;
                            rotZ1 = q12;
                            rotZ2 = q28;
                            rotZ3 = q34;
                            rotZ4 = q416;
                            rotZ5 = q516;
                            rotZ6 = q647;
                        }
                    }

                    if ((q12 >= -160f && q12 <= 160f) &&
(q34 >= -52.5f && q34 <= 232.5f) && (q28 >= -227.5f && q28 <=
47.5f) && (q516 >= -105f && q516 <= 120f) && (q416 >= -270f &&
q416 <= 270f) && (q648 >= -270f && q648 <= 270f))
                    {
                        sum96 = Mathf.Abs (q1prev -
q12) + Mathf.Abs (q2prev - q28) + Mathf.Abs (q3prev - q34) +
Mathf.Abs  (q4prev  -  q416)  +  Mathf.Abs  (q5prev  -  q516)  +
Mathf.Abs(q6prev - q648);

                        if (sum96 < minSum)
                        {
                            minSum = sum96;
                            rotZ1 = q12;
                            rotZ2 = q28;
                            rotZ3 = q34;
                            rotZ4 = q416;
                            rotZ5 = q516;
                            rotZ6 = q648;
                        }
                    }

                    // Οδηγηση του ρομποτ στην επομενη θεση
                    if                          (Input.GetKey
(KeyCode.JoystickButton7))
                    {
                        CalculateDoReady();
                    }
                    else
                    {
                        CalculateTarget ();
                    }


                    if (Mathf.Abs (q3prev - rotZ3) < 85f)
                    {
                        MoveToTarget ();

                        //Καταχωρηση της νεας θεσης
ως προηγουμενη
                        q1prev = rotZ1;
                        q2prev = rotZ2;
                        q3prev = rotZ3;
                        q4prev = rotZ4;
                        q5prev = rotZ5;
                        q6prev = rotZ6;



            if(Input.GetKeyDown(KeyCode.JoystickButton5))
                        {

            ppPoints.WriteLine("DO     DECOMPOSE     J_V["     +
decomposeIter + "]" + "=#PPOINT(" + rotZ1 + "," + rotZ2 + "," +
rotZ3 + "," + rotZ4 + "," + rotZ5 + "," + rotZ6 + ")");
                                                decomposeIter
+= 6;

                        }
                    }

                    else
                    {
                        rotZ1 = q1prev;
                        rotZ2 = q2prev;
                        rotZ3 = q3prev;
                        rotZ4 = q4prev;
                        rotZ5 = q5prev;
                        rotZ6 = q6prev;
                    }

                    endEffectorLabPosition                =
CoordinateSystemChange     (endEffectorUnityPosision.position)-
CoordinateSystemChange (referenceFrameUnityPosition.position);
                    wristLabPosition                      =
CoordinateSystemChange           (wristUnityPosition.position)-
CoordinateSystemChange (referenceFrameUnityPosition.position);


            }

            // Calculates "Do Ready" position of the robot
            void CalculateDoReady()
            {
                rotZ1 = initRotZ1;
                rotZ2 = initRotZ2;
                rotZ3 = initRotZ3;
                rotZ4 = initRotZ4;
                rotZ5 = initRotZ5;
                rotZ6 = initRotZ6;

                j1Rot = j1Def;
                j2Rot = j2Def;
                j3Rot = j3Def;
                j4Rot = j4Def;
                j5Rot = j5Def;
                j6Rot = j6Def;

                q1prev = rotZ1;
                q2prev = rotZ2;
                q3prev = rotZ3;
                q4prev = rotZ4;
                q5prev = rotZ5;
                q6prev = rotZ6;

                pwx = 0f;
                pwy = 0f;
                pwz = 1.1f;
                ax = 0f;
                ay = 0f;
                az = 0f;

                directionCoding = 0;
            }

            // Calculates target position of the robot
            void CalculateTarget()
            {
                //Η νεα γωνια προκυπτει απο την παλια +
την διαφορα της καινουριας απο την αρχικη με αντιθετο προσημο
ωστε να ταιριαζει στο συστημα του unity.
                j1Rot    =    Quaternion.Euler    (rotX1,
rotYAll, initRotZ1 -(rotZ1-initRotZ1));
                j2Rot    =    Quaternion.Euler    (rotX2,
rotYAll, initRotZ2 -(rotZ2-initRotZ2));
                j3Rot    =    Quaternion.Euler    (rotX3,
rotYAll, initRotZ3 -(rotZ3-initRotZ3));
                j4Rot    =    Quaternion.Euler    (rotX4,
rotYAll, initRotZ4 -(rotZ4-initRotZ4));
                j5Rot    =    Quaternion.Euler    (rotX5,
rotYAll, initRotZ5 -(rotZ5-initRotZ5));
```

```
                        j6Rot    =    Quaternion.Euler    (rotX6,
rotYAll, initRotZ6 -(rotZ6-initRotZ6));
            }

            // Moves robot to next position from the previous
position
            void MoveToTarget()
            {
                        joint1.localRotation  =  Quaternion.Lerp
(Quaternion.Euler       (joint1.localEulerAngles),       j1Rot,
rotationSpeed*Time.deltaTime);
                        joint2.localRotation  =  Quaternion.Lerp
(Quaternion.Euler       (joint2.localEulerAngles),       j2Rot,
rotationSpeed*Time.deltaTime);
                        joint3.localRotation  =  Quaternion.Lerp
(Quaternion.Euler       (joint3.localEulerAngles),       j3Rot,
rotationSpeed*Time.deltaTime);
                        joint4.localRotation  =  Quaternion.Lerp
(Quaternion.Euler       (joint4.localEulerAngles),       j4Rot,
rotationSpeed*Time.deltaTime);
                        joint5.localRotation  =  Quaternion.Lerp
(Quaternion.Euler       (joint5.localEulerAngles),       j5Rot,
rotationSpeed*Time.deltaTime);
                        joint6.localRotation  =  Quaternion.Lerp
(Quaternion.Euler       (joint6.localEulerAngles),       j6Rot,
rotationSpeed*Time.deltaTime);
            }

            // Calculates the lab Tool Coordinates given the
Unity coordinates
            public    Vector3    CoordinateSystemChange(Vector3
frame)
            {
                        Vector3 unityFrame = frame;
                        Vector3  labFrame  =  new  Vector3  (-
unityFrame.x, -unityFrame.z, unityFrame.y);
                        return labFrame;
            }

            void OnApplicationQuit()
            {
                        ppPoints.Close ();
            }

}
```

SecondCameraFollow.cs

```
using UnityEngine;
using System.Collections;

public class SecondCameraFollow : MonoBehaviour
{
            private IKMover ikMoverInstance;
            private Transform endEffectorUnityPosision;
            private Vector3 offset;
            private int dirCoding;
            private float axis6;
            private float mappedAxis;
            private float smooth = 15f;

            void Start()
            {
                        ikMoverInstance    =    GameObject.Find
("Staubli RX90L").GetComponent<IKMover> ();
                        endEffectorUnityPosision              =
GameObject.Find         ("(#11508)         RX90BL         TOOL
FLANGE").GetComponent<Transform> ();

                        dirCoding                             =
ikMoverInstance.directionCoding;

                        offset = new Vector3(0.5f, 0.5f, 0.5f);
            }

            void Update ()
            {
                        dirCoding                             =
ikMoverInstance.directionCoding;

                        axis6 = Input.GetAxis ("3rd_Axis");
                        mappedAxis = Map (axis6);

                        transform.position    =    Vector3.Lerp
(transform.position,
endEffectorUnityPosision.transform.position + offset, smooth *
Time.deltaTime);

                        switch (dirCoding)
                        {
                        case 0:
                                    offset  =  new  Vector3(0.5f,
0.1f, 0.5f);
                                    transform.eulerAngles = new
Vector3(30f, 256.523f, 0f);
                                    break;
                        case 1:
                                    offset = new Vector3(0.5f -
axis6 * 0.5f, axis6 * 0.5f, 0f);
                                    transform.eulerAngles = new
Vector3(mappedAxis, -90f, 0f);
                                    break;
                        case 2:
                                    offset  =  new  Vector3(0f,
axis6 * 0.5f, 0.5f - axis6 * 0.5f);
                                    transform.eulerAngles = new
Vector3(mappedAxis, -180f, 0f);
                                    break;
                        case 3:
                                    offset = new Vector3(0.5f -
axis6 * 0.5f, axis6 * 0.5f, 0f);
                                    transform.eulerAngles = new
Vector3(mappedAxis, -90f, 0f);
                                    break;
                        case 4:
                                    offset  =  new  Vector3(0f,
axis6 * 0.5f, 0.5f - axis6 * 0.5f);
                                    transform.eulerAngles = new
Vector3(mappedAxis, -180f, 0f);
                                    break;
                        }
            }

            private float Map(float rawValue)
            {
                        float fixedValue = rawValue * 90f;
                        return fixedValue;
            }
}
```