Τεχνολογικό Εκπαιδευτικό Ίδρυμα Κρήτης

Σχολή Τεχνολογικών Εφαρμογών
Τμήμα Εφαρμοσμένης Πληροφορικής & Πολυμέσων

Πτυχιακή Εργασία

Ανάπτυξη Εφαρμογής με τη χρήση του Microsoft Kinect Sensor

Ευθύμιος Συντυχάκης (ΑΜ: 2243)

Επιβλέπων καθηγητής: Τριανταφυλλίδης Γεώργιος
Επιτροπή Αξιολόγησης:
Ημερομηνία παρουσίασης:

Ευχαριστίες

Ευχαριστώ τους γονείς μου που με στήριξαν και με στηρίζουν ακόμα, τον κ. Τριανταφυλλίδη και τον κ. Βιδάκη για τη βοήθεια και την υπομονή τους και τέλος όλα μέλη του iSTLab για τη ευχάριστη ατμόσφαιρα και συνεργασία.

Abstract

In this work we represent a multimodal natural user interface called MIDAS (Multimodal Interface Directed by Action Sentences). MIDAS uses body motion, voice commands and other modals of human communication, detecting them with technologies such as gesture, voice and facial expression recognition, interpreting them as words. Users can manipulate the running applications on the computer by constructing action sentences with the words. The device utilized for achieving multimodal input is the Microsoft Kinect which provides RGB, depth and sound input. MIDAS is based on a generic container which runs at the background and serves as an intermediate between multimodal input and active applications running on a computer. We describe the architecture of the system and its representative example using two PC applications.

Keywords:

Multimodal Interaction, Kinect, Multi-application, Action Sentences, MIDAS

Σύνοψη

Σε αυτήν την εργασία παρουσιάζεται μια πολύ-τροπικη φυσική διεπαφή χρήστη με όνομα MIDAS (Multimodal Interface Directed by Action Sentences). Το MIDAS χρησιμοποιεί κινήσεις, φωνητικές εντολές κι άλλους τρόπους ανθρώπινης επικοινωνία, εντοπίζοντάς τους με τεχνολογίες όπως αναγνώριση κίνησης του σώματος, φωνής, έκφρασης – κίνησης προσώπου, ερμηνεύοντας τους ως λέξεις Με τις λέξεις αυτές σχηματίζει προτάσεις με τις οποίες ο χρήστης μπορεί να διαχειριστεί τις τρέχουσες εφαρμογές του υπολογιστή. Η πολύ-τροπικότητα στις εισόδους επιτυγχάνεται με τη χρήση του Microsoft Kinect το οποίο παρέχει εικόνα χρώματος, χάρτη βάθους και ήχο. Το MIDAS βασίζεται σε ένα general container το οποίο τρέχει στο παρασκήνιο λειτουργώντας ως ενδιάμεσος μεταξύ της πολύ-τροπικής εισόδου και των εφαρμογών που τρέχουν σε ένα υπολογιστή.. Εξηγούμε την αρχιτεκτονική του συστήματός μας και παρουσιάζουμε παραδείγματα χρήσης του.

Περιεχόμενα

Εικόνες

Πίνακες και Κώδικας

1 Εισαγωγή

Για περισσότερο από 40 χρόνια, η διεπαφές ανθρώπου – υπολογιστή έχουν εστιάσει στο πληκτρολόγιο και στο ποντίκι. Αν και ο παραπάνω συνδυασμός ήταν επιτυχής, η συνεχής ανάπτυξη της τεχνολογίας η οποία θέλει τους υπολογιστές φορητούς, ενσωματωμένους σε άλλες συσκευές και πανταχού παρόντες, τον καθιστά πολύ περιοριστικό σαν μοντέλο αλληλεπίδρασης. [9]

> "*Το πραγματικό πρόβλημα με μια διεπαφή είναι ότι είναι μια διεπαφή.*
>
> *Οι διεπαφές είναι εμπόδια. Δε θέλω να εστιάζω την προσοχή μου σε μια*
>
> *διεπαφή. Θέλω να συγκεντρώνομαι στη δουλειά μου*." - Donald A. Norman [10]

Ένας τρόπος να εκφράσουμε το στόχο για την απαλλαγή μιας διεπαφής είναι να χαρακτηρίσουμε την καινούρια διεπαφή «φυσική». Ο όρος φυσική διεπαφή χρήστη (Natural User Interface) δεν είναι ακριβής, αλλά αναφέρεται συνήθως σε μια διεπαφή που είναι εύκολη στη χρήση και κάνει τη χρήση του λογισμικού απρόσκοπτη [10]. Για να προσεγγίσουμε μια τέτοια διεπαφή πρέπει να δούμε την ανθρώπινη επικοινωνία.

Η επικοινωνία μεταξύ ανθρώπων προϋποθέτει τη χρήση διαφόρων μέσων έκφρασης. Ο άνθρωπος χρησιμοποιεί μια πλούσια ποικιλία μηχανισμών έκφρασης όπως λόγο, χειρονομίες, βλέμμα, έκφραση προσώπου και στάση σώματος – κάθε ένα ξεχωριστά αλλά και συνδυασμούς αυτών. Δίνοντας τη δυνατότητα στο χρήστη να χρησιμοποιήσει καθημερινούς τρόπους επικοινωνίας για την αλληλεπίδρασή του με τον υπολογιστή, επιτυγχάνουμε έναν πιο ανθρωποκεντρικό σχεδιασμό.

Για να το καταφέρουμε όμως αυτό χρειαζόμαστε την κατάλληλη τεχνολογία η οποία θα πρέπει είναι προσιτή για το μέσο χρήστη. Το μικρόφωνο και η web-camera είναι προσιτές συσκευές εισόδου οι οποίες, με την κατάλληλη επεξεργασία των εισόδων τους, ικανοποιούν κάποιες από τους παραπάνω τρόπους έκφρασης, όπως το λόγο ή την έκφραση προσώπου. Αλλά για τον εντοπισμό της κίνηση του ανθρώπινου σώματος χρειαζόμαστε διαφορετικό εξοπλισμό, πιο εξειδικευμένο, και πιο ακριβό – οπότε και μη προσιτό προς το χρήστη.

Μέχρι τώρα ο εντοπισμός κίνησης του σώματος ήταν προνόμιο μόνο των studio και των ερευνητικών εργαστηρίων, με ακριβές στολές εντοπισμού θέσης των αρθρώσεων. Πρόσφατα όμως κυκλοφόρησαν, προσιτές σε τιμή, συσκευές όπως το Microsoft Kinect και το Asus Xtion Pro LIVE οι οποίες με τον αισθητήρα βάθους τον οποίο έχουν ενσωματωμένο παρέχουν ένα αξιόλογο εντοπισμό κίνησης (Motion Tracking).

Το MS Kinect, κυκλοφόρησε το Νοέμβριο του 2010 ως χειριστήριο για την παιχνιδομηχανή Xbox360, ενώ το Asus Xtion Pro LIVE κυκλοφόρησε για τον υπολογιστή το δεύτερο μισό του 2011. Και οι 2 συσκευές δεν περιορίζονται μονάχα στον αισθητήρα βάθους. Παρέχουν επίσης μικρόφωνα και κάμερα χρώματος κι έτσι οι χρήστες μπορούν να έχουν στο σπίτι τους πια συσκευές πολύ-τροπικής εισόδου, οι οποίες θα δίνουν τη δυνατότητα και την ελευθερία στους προγραμματιστές να αναπτύξουν πιο ανθρωποκεντρικές εφαρμογές.

1.1 Κίνητρο για τη Διεξαγωγή της Εργασίας

Σκοπός τις εργασίας αυτής είναι η παρουσίαση μιας αρχιτεκτονικής η οποία χρησιμοποιεί φυσικές (ανθρώπινες) μεθόδους έκφρασης για την αλληλεπίδραση του χρήστη με τον υπολογιστή, κάνοντας τη διεπαφή πιο «φυσική». Με τον τρόπο αυτό ο βαθμός δυσκολίας προσαρμογής του χρήστη στο σύστημα θα μειωθεί πολύ, ενώ το σύστημα θα είναι αρκετά ευέλικτο για χρήστες με αδυναμίες σε κάποια από τις φυσικές μεθόδους έκφρασης (π.χ. κινητικές δυσκολίες).

1.2 Δομή Εργασίας

Στο επόμενο κεφάλαιο παρουσιάζονται, οι πρόσφατα εμφανιζόμενες, πολύ-τροπικές συσκευές εισόδου Xtion και Kinect, τα APIs τους και εκτενέστερα ο αισθητήρας Kinect και το επίσημο SDK από την Microsoft τα οποία χρησιμοποιήθηκαν για την υλοποίηση. Στο κεφάλαιο 3 θα εξηγήσουμε τι είναι πολύ-τροπικές διεπαφές και θα δούμε πρόσφατες έρευνες. Στο κεφάλαιο 4 περιγράφονται οι βιβλιοθήκες και το λογισμικό που χρησιμοποιήθηκαν και στο κεφάλαιο 5 η αρχιτεκτονική του συστήματός μας. Στο κεφάλαιο 6 η υλοποίηση που παρουσιάζουμε ως παράδειγμα μέσω των δυο σεναρίων χρήσης και στο κεφάλαιο 7 τα συμπεράσματά μας. Στο παράρτημα παραθέτουμε το ερευνητικό άρθρο το οποίο δημοσιεύθηκε στο TEMU 2012.

2 Συσκευές Πολύ-τροπικής Εισόδου

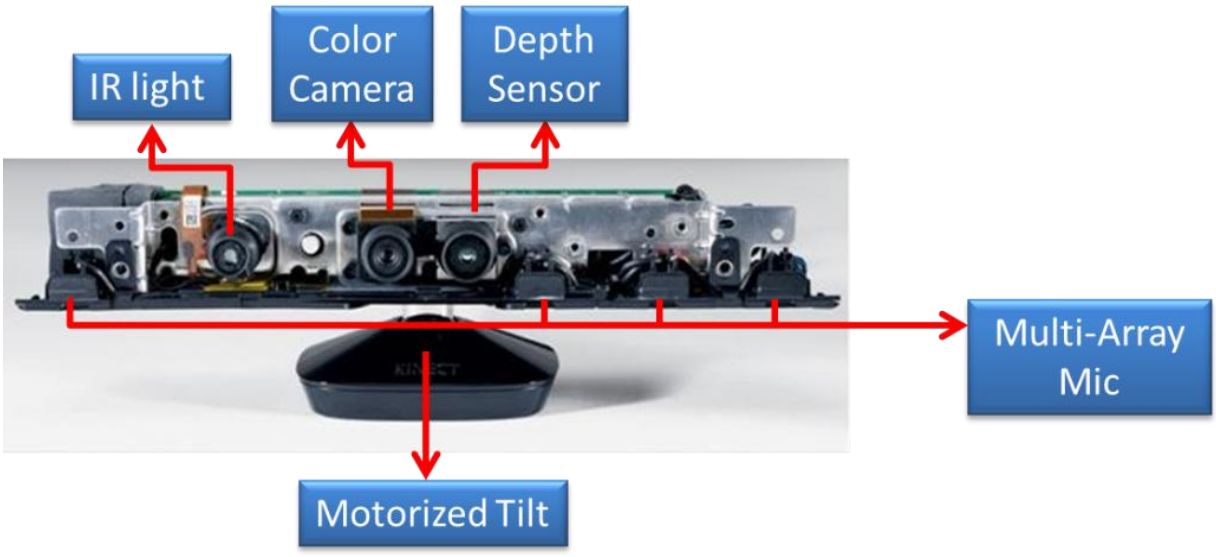
Μέχρι πρότινος, τεχνολογίες όπως καταγραφή βάθους και κίνησης υπήρχαν μόνο σε εργαστήρια και στούντιο animated ταινιών. Αποτελούσαν ακριβό εξοπλισμό εξειδικευμένης χρήσης. Με την εμφάνιση του, χαμηλού σε κόστος, αισθητήρα βάθους από την εταιρία PrimeSense, οι δυνατότητες αυτές έφτασαν στα χέρια του καθημερινού χρήστη. Ερευνητές από διάφορα πεδία χρησιμοποίησαν αυτή την τεχνολογία. Η πρώτη συσκευή η οποία περιείχε τον αισθητήρα βάθους ήταν το Microsoft Kinect, η συσκευή περιείχε επίσης 4 μικρόφωνα, επιταχυνσιόμετρο και μία κλασσική κάμερα χρώματος, δίνοντας έτσι τι δυνατότητα χρήσης παραπάνω από μίας, εισόδων, στον προγραμματιστή. Έπειτα η Asus σε συνεργασία με την PrimeSense κυκλοφόρησε το Asus Xtion Pro, το οποίο περιείχε μονάχα τον αισθητήρα βάθους. Μα η συσκευή δεν είχε τόση απήχηση στην αγορά, το Xtion Pro δεν είχε την πολυμορφικότητα του Kinect. Έτσι η Asus προχώρησε στην δημιουργία του Asus Xtion Pro LIVE, μιας συσκευής η οποία περιείχε τον αισθητήρα βάθους, 2 μικρόφωνα και μια κάμερα χρώματος.

2.1 Microsoft Kinect - Οι αισθητήρες

Εικόνα 1 - Microsoft Kinect για το Xbox 360

Εικόνα 2 - Microsoft Kinect για τα Windows

Στης 1 Ιουνίου του 2009, η Microsoft ανακοίνωσε ένα project με όνομα Project Natal το οποίο θα δίνει τη δυνατότητα αναγνώρισης της κίνησης του παίκτη στο Xbox 360. Το όνομα όπως όλα τα κωδικά ονόματα της Microsoft δόθηκε από μια πόλη. Η πόλη Natal της Βραζιλίας επιλέχθηκε προς τιμήν του Βραζιλιάνου διευθυντή της Microsoft Alex Kipman, ο οποίος ήταν υπεύθυνος για το project, και καταγόταν από εκεί.



Εικόνα 3 – Το εσωτερικό της συσκευής MS Kinect

Ο δεύτερος λόγος για το όνομα αυτό ήταν ότι η λέξη natal σημαίνει και «γέννηση», με το οποίο ήθελαν να συμβολίσουν την γέννηση της καινούριας γενιάς ψυχαγωγίας στο σπίτι. Το όνομα του τελικού προϊόντος ήταν Kinect από τις λέξεις kinetic (κινητικός) και connect (συνδέω). Το Kinect έκανε την εμφάνισή του στην αγορά τον Νοέμβριο του 2010. Ο αισθητήρας βάθους ο οποίος περιέχει σχεδιάστηκε και δημιουργήθηκε από την ισραηλίτικη εταιρία PrimeSense, Ltd. [2]. Δύο μήνες μετά από την κυκλοφορία του, η Microsoft, είχε πουλήσει 8 εκατομμύρια Kinect δίνοντάς της τον τίτλο της ηλεκτρονικής συσκευής με την ταχύτερη πώληση στο βιβλίο Guinness. Έχουν πουληθεί 18 εκατομμύρια Kinect μέχρι τον Ιανουάριο του 2012.

2.1.1 Αισθητήρας Βάθους (Depth Sensor) και Πομπός Υπερύθρων (IR light)

Η τεχνολογία Light Coding ™ (ευρεσιτεχνία των Zalevsky, Z, et al. [1]) επιτρέπει στο Kinect να δημιουργήσει 3D χάρτες βάθους μιας σκηνής, σε πραγματικό χρόνο. Μια δομή από σημεία (σχεδόν) υπέρυθρου φωτός προβάλλεται στον χώρο (βλ. Εικόνα 4) και ένας αισθητήρας εικόνας CMOS (Complementary Metal Oxide Semiconductor) δέχεται τις ανακλώμενες ακτίνες. Το PS1080 SoC (System on a Chip) – τσιπ φτιαγμένο από την PrimeSense που περιέχει το σύστημα του Kinect, ελέγχει την δομή από σημεία φωτός και να επεξεργάζεται τα δεδομένα από τον αισθητήρα CMOS παράγοντας δεδομένα βάθους σε πραγματικό χρόνο [3]. Η μέγιστη ανάλυση της εικόνας βάθους που παράγει το (βλ. Εικόνα 5) PS1080 είναι 640x480, με συχνότητα 30 frames ανά δευτερόλεπτο. Στα 2 μέτρα απόστασης από τον αισθητήρα, έχει τη ακρίβεια 3 χιλιοστών σε ύψος και πλάτος και 1 εκατοστό σε βάθος. Η εμβέλεια ορθής λειτουργίας είναι από 0.8 μέχρι 3.5 μέτρα.

Εικόνα 4 - Η δομή υπερύθρων από τον πομπό του Kinect

Εικόνα 5 - Εικόνα βάθους από το Kinect

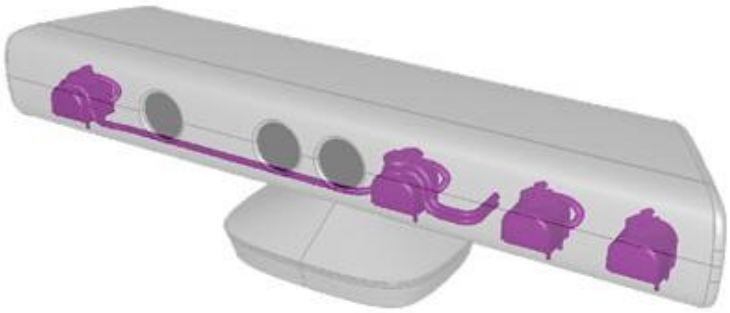
2.1.2 Κάμερα χρώματος (Color Camera)

Το Kinect έχει επίσης μια ενσωματωμένη κάμερα χρώματος (Color CMOS - VNA38209015) με μέγιστη ανάλυση 1280x1024 για να έχουμε την πραγματική εικόνα πέρα από το χάρτη βάθους. Η συχνότητα λήψης της κάμερας είναι 30 fps και η εικόνα που παράγεται είναι αρκετά καλή, ώστε να χρησιμοποιηθεί σε αλγορίθμους αναγνώρισης προσώπου, δακτύλων ή οτιδήποτε άλλο χρειαζόμαστε στην εφαρμογή μας. Παράδειγμα εικόνας από την RGB κάμερα της συσκευής θα δείτε στην Εικόνα 6.

Εικόνα 6 - RGB εικόνα από το Kinect

2.1.3 Μικρόφωνα

Μια σειρά από 4 μικρόφωνα δίνει στο Kinect τη δυνατότητα όχι απλά να δέχεται ήχο, αλλά και να εντοπίζει την γωνία της πηγής του στη σκηνή. Στην Εικόνα 7 βλέπουμε τη θέση των μικροφώνων μέσα στη συσκευή η οποία επεξεργάζεται ξεχωριστά το καθένα από τα τέσσερα κανάλια τα οποία δέχονται 16-bit ήχο με συχνότητα δειγματοληψίας ίση με 16 kHz.



Εικόνα 7 - Η σειρά μικροφώνων μέσα στη συσκευή.

2.1.4 Επιταχυνσιόμετρο (Accelerometer)

Το Kinect παρέχει ένα επιταχυνσιόμετρο τριών αξόνων (KXSD9-1026, βλέπε Εικόνα 8) το οποίο παρέχει την πληροφορία της θέσης της συσκευής. Οι τιμές του X και του Y καθορίζουν την κύλιση και την κλίση, ενώ το Z καθορίζει εάν το Kinect είναι ανάποδα ή όχι. Το επιταχυνσιόμετρο μετρά μονάχα την κλίση, κι όχι τον προσανατολισμό της συσκευής. Το χαρακτηριστικό αυτό του Kinect έχει χρησιμοποιηθεί ευρέως στον τομέα της ρομποτικής.

Εικόνα 8 - Το Επιταχυνσιόμετρο της συσκευής Kinect.

2.1.5 Μηχανοκίνητη Βάση (Motorized Tilt)

Η συσκευή διαθέτει επίσης μια μηχανοκίνητη βάση (βλ. Εικόνα 9) η οποία ρυθμίζεται δυναμικά μέσω κώδικα. Η κίνηση της βάσης προέρχεται από ένα μικρό μοτέρ στο μέγεθος νομίσματος και τρία εύθραυστα πλαστικά γρανάζια, τα οποία είναι ευαίσθητα στη θερμότητα και ίσως αποτελούν πιο αδύναμο σημείο της συσκευής. Η βάση δίνει στο Kinect τη δυνατότητα να στραφεί προς τα πάνω ή κάτω κατά 27°.

Εικόνα 9 - Η μηχανοκίνητη βάση του Kinect

2.2 Asus Xtion Pro και Xtion Pro LIVE

2.2.1 Ιστορικό εξέλιξης

Το τέλος της άνοιξης του 2011 κυκλοφόρησε η πρώτη depth sensing συσκευή της Asus: το Asus Wavi Xtion Pro (Εικόνα 10). Απευθυνόταν καθαρά σε developers και περιείχε μονάχα τον αισθητήρα βάθους της Primesense Ltd.

Εικόνα 10 – Asus Wavi Xtion Pro

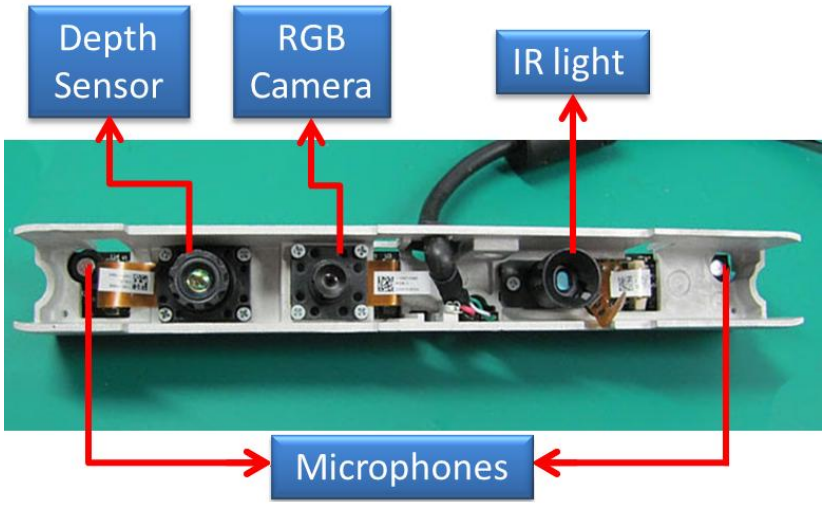
Η σύνδεσή του με τον υπολογιστή γινόταν μέσω USB θύρας, κι ένα μεγάλο πλεονέκτημά του ήταν ότι δεν χρειαζόταν εξωτερική τροφοδοσία. Αυτό το καθιστούσε πιο φορητό από το Kinect και σε συνδυασμό με το χαμηλό του βάρος βοηθούσε πάρα πολύ στο πεδίο της ρομποτικής. Μα το Xtion Pro δεν είχε τόση απήχηση στην αγορά επειδή είχε αποκλειστικά και μόνο τον αισθητήρα βάθους. Έτσι,

μετά από μερικούς μήνες η Asus σταμάτησε την παραγωγή του και κυκλοφόρησε μία νέα συσκευή με το όνομα Asus Xtion Pro Live (βλ. Εικόνα 11).

Εικόνα 11 - Asus Xtion Pro LIVE

2.2.2 Τα χαρακτηριστικά του Xtion Pro Live

Το Xtion Pro Live σχεδιάστηκε έτσι ώστε να μπορεί να συναγωνιστεί το Microsoft Kinect, εξοπλισμένη με πολλά όμοια χαρακτηριστικά (βλ. Πίνακας 1). Παρέχει μια κάμερα χρώματος, ένα αισθητήρα βάθους και 2 μικρόφωνα (βλ. Εικόνα 12).



Εικόνα 12 - Το εσωτερικό της συσκευής Xtion Pro LIVE

Κάποια από τα πλεονεκτήματα που παρέχει η νέα συσκευή της Asus είναι:

- *Δεν χρειάζεται εξωτερική τροφοδοσία* – Όπως και το προηγούμενο μοντέλο, το Xtion Pro LIVE, δε χρειάζεται εξωτερική τροφοδοσία, δουλεύει με την ενέργεια που του παρέχει η USB θύρα.
- *Μικρότερο σε μέγεθος* – Το μέγεθός του (7" x 2" x 1.5") είναι αρκετά πιο εργονομικό από του Kinect.
- *Χαμηλότερο βάρος* – Το βάρος του (0.5 lb.), βοηθά πολύ στο πεδίο της ρομποτικής καθώς το βάρος το οποίο μπορούν να υποστηρίξουν οι ρομποτικές συσκευές είναι περιορισμένο.

	Asus Xtion Pro Live	Microsoft Kinect
Διαστάσεις	7" x 2" x 1.5"	12" x 3" x 2.5"
Βάρος	0.5 λίβρες (226~ γρ.)	3.0 λίβρες (1360~ γρ.)
Κάμερα χρώματος	1280x1024	1280x1024
Αισθητήρας Βάθους	PrimeSense	PrimeSense
Μικρόφωνα	2	4
Επιταχυνσιόμετρο	Όχι	Ναι
Μηχανοκίνητη Βάση	Όχι	Ναι
Εξωτερική τροφοδοσία	Ναι	Όχι

Πίνακας 1 - Χαρακτηριστικά Asus Xtion Pro Live vs. Microsoft Kinect

2.3 Τα διαθέσιμα APIs και τα χαρακτηριστικά τους

Οι αφρόκρεμα των hackers περίμεναν το Kinect, γιατί θα είχαν την ευκαιρία να χρησιμοποιήσουν την τεχνολογία μιας μεγάλης εταιρίας και να δουν είναι τι πραγματικά ικανή να κάνει. Προς μεγάλη έκπληξη των hackers, η Adafruit, μια εταιρία πώλησης ηλεκτρονικών εργαλείων η οποία εδρεύει στην Νέα Υόρκη, ανακοίνωσε τη μέρα της κυκλοφορίας του Microsoft Kinect (4 Νοεμβρίου στις ΗΠΑ) ότι θα δώσει 1000 δολάρια στον προγραμματιστή που θα καταφέρει να χρησιμοποιήσει το Kinect στα Windows, ή άλλο λειτουργικό σύστημα. Η Microsoft έδωσε ακόμα ένα κίνητρο χωρίς να το θέλει. Λίγες ώρες μετά την ανακοίνωση της αμοιβής ανακοίνωσε ότι δεν δέχεται την παραποίηση της συσκευής και θα κινούνταν νομικά ώστε να κρατήσει το Kinect απαραβίαστο.

Η ανακοίνωση της Microsoft λειτούργησε σαν κόκκινο πανί για τους hackers, κι εκείνο το απόγευμα δημοσιεύθηκε στο blog της Adafruit: *"Εντάξει λοιπόν, η αμοιβή μόλις διπλασιάστηκε, 2000 δολάρια"*. Στις 6 Νοεμβρίου ένας hacker με το ψευδώνυμο AlexP κατάφερε να ελέγξει την μηχανοκίνητη βάση του Kinect. Η Microsoft προσπάθησε να αρνηθεί την είδηση ανακοινώνοντας ότι το Kinect δεν έχει χακαριστεί. Οι κοινωνία των hackers πήρε την ανακοίνωση ως προσβολή. Η Adafruit έγραψε στο blog της ότι η ανακοίνωση της Microsoft ήταν χαζή και ανέβασε την αμοιβή στα 3000 δολάρια.

Δύο ημέρες αργότερα (8 Νοεμβρίου) ο AlexP δημοσίευσε ένα βίντεο το οποίο αποδείκνυε ότι είχε τον έλεγχο του αισθητήρα βάθους και της κάμερας του Kinect. Το βραβείο ήταν δικό του εάν δημοσίευε τον κώδικα, μα ο AlexP είχε άλλες προσδοκίες. Απαιτούσε 10000 δολάρια για τη δημοσίευση του κώδικα.

Ωστόσο κι άλλοι δούλευαν για τον ίδιο σκοπό. Η Adafruit δημοσίευσε πακέτα δεδομένων καταγεγραμμένα από το Kinect. Την Τετάρτη 10 Νοεμβρίου στις 10 το πρωί, τη μέρα κυκλοφορίας της συσκευής στην Ευρώπη, ο Héctor Martín, hacker και φοιτητής επιστήμης υπολογιστών, αγόρασε το Kinect και άρχισε να δουλεύει με τα δεδομένα που δημοσίευσε η Adafruit. Λίγο μετά τις 11:00 π.μ. επικοινωνούσε με το Kinect, και μία ώρα αργότερα είχε εικόνα, από το χάρτη βάθους και την κάμερα της συσκευής, στην οθόνη του. Ο Héctor Martín δημοσίευσε τον πηγαίο κώδικα για την χρήση του

Kinect στον υπολογιστή με Linux λογισμικό και η Adafruit τον αναγνώρισε ως νικητή δίνοντας του το έπαθλο των 3000 δολαρίων.

Έπειτα ακολούθησαν κι άλλες εκδοχές αποκωδικοποίησης των δεδομένων του Kinect, με αποτέλεσμα να δημιουργηθούν αρκετά APIs με διαφορετικές δυνατότητες.

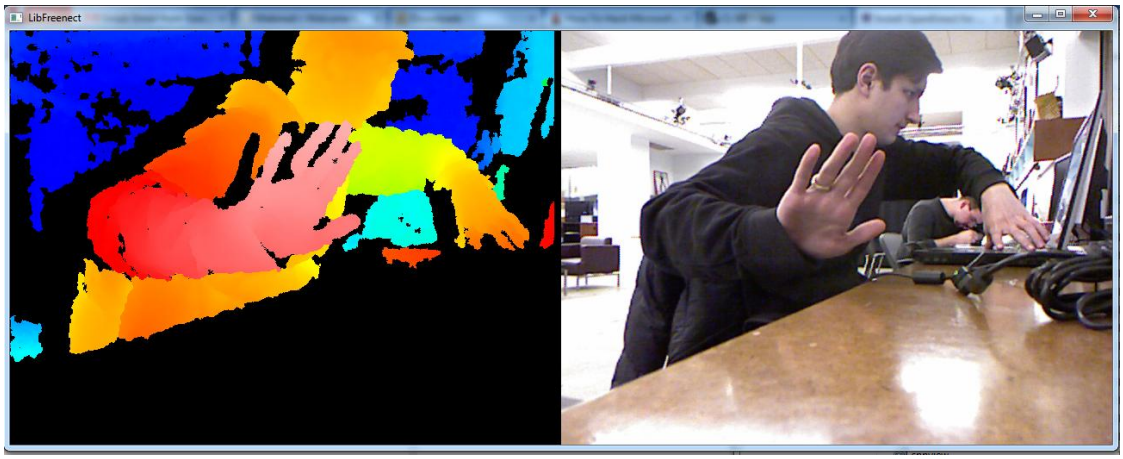
Μέχρι στιγμής υπάρχουν πέντε αρκετά δημοφιλής APIs για την αποκωδικοποίηση των δεδομένων που παράγει το Kinect – ανάμερα σε αυτά και το official SDK της Microsoft, οι οποίες είναι ελεύθερες προς χρήση: Libfreenect [4], OpenNI και NITE [5], CLNUI [6], Microsoft Kinect SDK [7], Evoluce SDK [8] (το οποίο είναι βασισμένο στο OpenNI). Στο παράρτημα θα βρείτε πλήρεις οδηγούς για την εγκατάσταση του καθενός από τα παραπάνω APIs.

Τι είναι API;

Το API είναι μια συντομογραφία του όρου **a**pplication-**p**rogramming **i**nterface (Διεπαφή προγραμματισμού εφαρμογών), το οποίο είναι ένα σύνολο από εργαλεία, πρωτόκολλα και μεθόδους για την επικοινωνία με το λειτουργικό σύστημα ή πρόγραμμα ελέγχου ή συσκευή. Το API είναι διεπαφή ανάμεσα σε εφαρμογές, κι όχι διεπαφή χρήστη. Με τα APIs οι εφαρμογές επικοινωνούν χωρίς να γίνεται αντιληπτό από το χρήστη. Η χρήση ενός API σημαίνει ότι ένα πρόγραμμα οδήγησης ή άλλη εφαρμογή, είναι διαθέσιμο στον υπολογιστή για να κάνει κάποιες διαδικασίες .

2.3.1 Open Kinect – Libfreenect

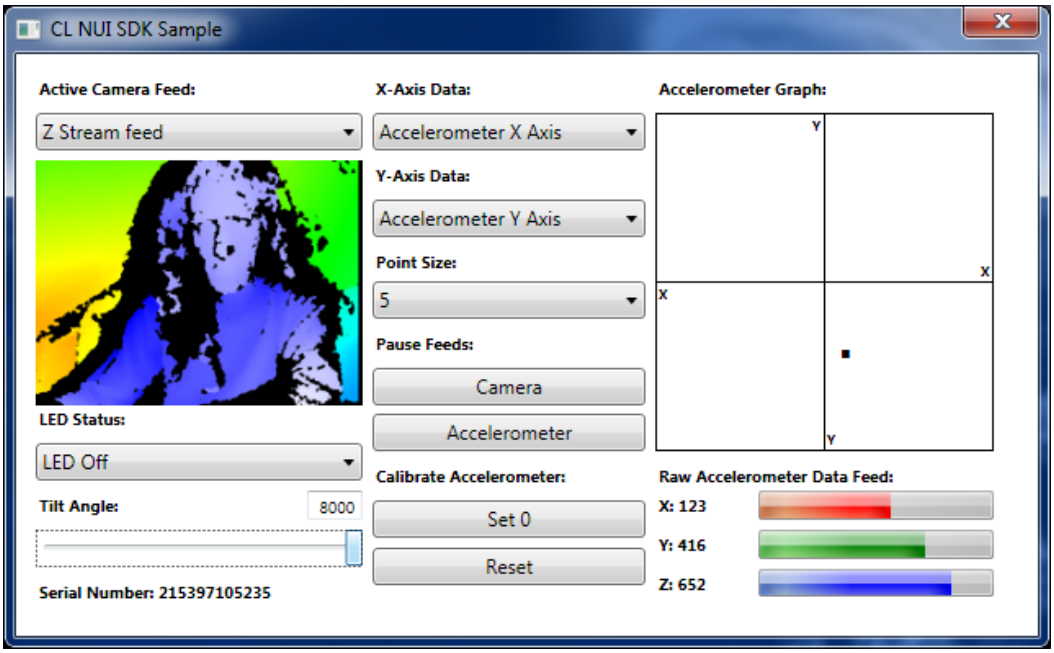
Το Libfreenect ήταν η πρώτη βιβλιοθήκη που δημιουργήθηκε για την αποκωδικοποίηση των δεδομένων του Kinect. Κυκλοφόρησε στις αρχές του Νοεμβρίου του 2010 από το Héctor Martín, λίγο μετά την κυκλοφορία του Kinect στην αγορά. Δημοσιεύθηκε στην ιστοσελίδα του Open Kinect στις 10 Νοεμβρίου όπου αποτέλεσε την αρχή της σύνταξης μιας μεγάλης κοινωνίας προγραμματιστών για τη χρήση του και την εξέλιξή του. Αναπτυγμένο σε C και Python, το Libfreenect παρέχει ένα μεγάλο αριθμό από wrappers, σε διάφορες γλώσσες, και την κατάλληλη βιβλιογραφία για τη χρήση τους. Η βιβλιοθήκη παρέχει πρόσβαση στην κάμερα, στο αισθητήρα βάθους (βλ. Εικόνα 13), στο LED και στη μηχανοκίνητη βάση της συσκευής. Χαρακτηρίζεται API χαμηλού επιπέδου γιατί πέρα από την επικοινωνία με τη συσκευή δεν παρέχει μεθόδους όπως ανίχνευση σκελετού (Skeleton Tracking). Υπάρχει όμως ένας μεγάλος αριθμός από προγραμματιστές που το χρησιμοποιούν, κι αυτό είναι ελεύθερο για εμπορική χρήση με δυνατότητα να τρέξει σε Windows, Mac OSX και Linux.Τέλος η βιβλιοθήκη είναι αρκετά δύσκολη στην εγκατάστασή της γιατί χρειάζεται χειροκίνητη τοποθέτηση των αρχείων της μέσα στους φακέλους του συστήματος του προγραμματιστή.



Εικόνα 13 - Χάρτης βάθους και RGB κάμερα με τη χρήση του Libfreenect

2.3.2 CL NUI

Η πλατφόρμα διαχείρισης των Code Laboratories βασίζεται στον στην αποκωδικοποίηση του AlexP, ο οποίος κατάφερε να αποκωδικοποιήσει τα δεδομένα του Kinect πριν από τον Héctor Martín αλλά δεν δημοσίευσε άμεσα τον κώδικά του. Η πρώτη έκδοση της πλατφόρμας CL NUI εκδόθηκε στις 8 Δεκεμβρίου 2010, και είναι ευρέως γνωστή στον κλάδο της ρομποτικής, γιατί είναι η μόνη βιβλιοθήκη που δίνει τη δυνατότητα πρόσβασης στο επιταχυνσιόμετρο του Kinect. Το CL NUI λειτουργεί μόνο σε Windows XP/Vista και 7 στις 32 αλλά και 64 bit εκδόσεις τους και παρέχει πρόσβαση στην κάμερα χρώματος, στον αισθητήρα βάθους, στο επιταχυνσιόμετρο, στο LED αλλά και στην μηχανοκίνητη βάση του Kinect. Χαρακτηρίζεται και αυτό ως API χαμηλού επιπέδου και απευθύνεται μονάχα στη συσκευή Microsoft Kinect. Το API συνοδεύεται από δύο παραδείγματα, εκ των οποίων το ένα κάνει χρήση του επιταχυνσιόμετρου (βλ. Εικόνα 14).

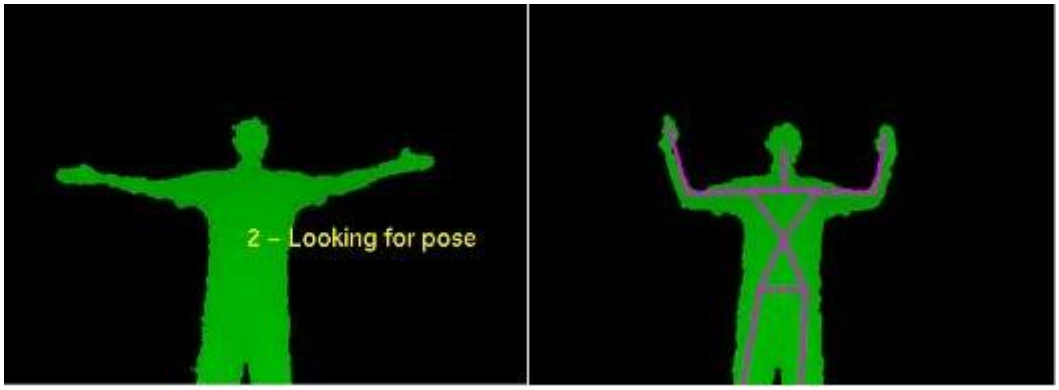


Εικόνα 14 - Παράδειγμα χρήσης επιταχυνσιόμετρου με χρήση του CL NUI.

2.3.3 OpenNI και NITE

Δημιουργήθηκε από έναν αφιλοκερδή οργανισμό ο οποίος απαρτίζεται από διάφορες εταιρίες, συμπεριλαμβανομένου και της PrimeSence Ltd. οι οποίες θέλησαν να θέσουν ένα βιομηχανικό πρότυπο λειτουργικότητας για τις συσκευές φυσική διεπαφής χρήστη – υπολογιστή (Natural User Interaction Devices) και κυκλοφόρησε το Δεκέμβριο του 2010. Το OpenNI αναπτύχθηκε σε C/C++ έτσι ώστε να μπορεί να χρησιμοποιηθεί από διάφορα λειτουργικά συστήματα, όπως Mac OSX, Ubuntu, Windows. Είναι το επίσημο λογισμικό των Xtion συσκευών της Asus, αλλά μπορεί να λειτουργήσει και με το Kinect. Παρέχει επικοινωνία με τον αισθητήρα βάθους, την κάμερα χρώματος, τα μικρόφωνα και τη μηχανοκίνητη βάση. Το OpenNI, όμως, συνοδεύεται και από μια ενδιάμεση βιβλιοθήκη η οποία λέγεται NITE και είναι εξοπλισμένο με τεχνολογίες αναγνώρισης φωνής (Voice Recognition), αναγνώρισης χειρονομιών χεριών (Hand Gesture Recognition), και ανίχνευση σκελετού (Skeleton Tracking). Η ανίχνευση του σκελετού του χρήστη απαιτούσε αρχική στάση βαθμονόμησης

(βλ. Εικόνα 15) αλλά στις πιο πρόσφατες εκδόσεις του, το OpenNI/NITE, δεν χρειάζεται αρχική στάση βαθμονόμησης (χαρακτηριστικό που είχε μόνο το MS SDK).

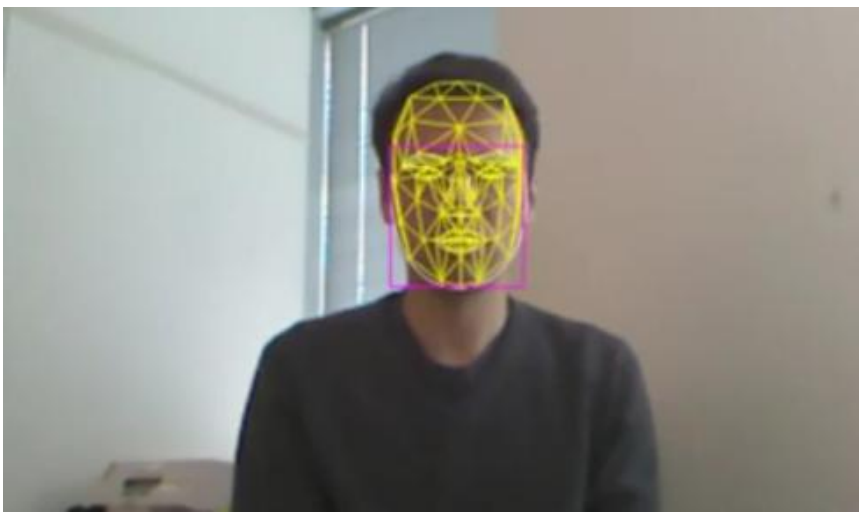


Εικόνα 15 - Η στάση βαθμονόμησης του OpenNI

2.3.4 Microsoft Kinect for Windows SDK

Στις 16 Ιουνίου του 2011 κυκλοφόρησε η beta έκδοση της βιβλιοθήκης της Microsoft για τη χρήση του Kinect στα Windows 7. Το SDK έδινε τη δυνατότητα στου προγραμματιστές να αναπτύξουν εφαρμογές σε C++, C# ή Visual Basic με τη χρήση του Microsoft Visual Studio 2010. Μερικές από της αποκλειστικές δυνατότητες που προσέφερε είναι η επεξεργασία ήχου με δυνατότητα υπολογισμού της πηγής (sound source estimation), αναγνώριση ομιλίας (speech recognition), υπολογισμός του σκελετού του χρήστη χωρίς να χρειάζεται αρχική στάση βαθμονόμησης (skeleton tracking without calibration posture) και τέλος ανιχνεύει περισσότερες αρθρώσεις ανά χρήστη (αστραγάλους και καρπούς). Η Beta έκδοση του SDK δεν παρείχε εμπορική άδεια.

Στις αρχές του 2012 κυκλοφόρησε η πρώτη επίσημη έκδοση του SDK και μαζί της και η καινούρια έκδοση της συσκευής Kinect η οποία προοριζόταν αποκλειστικά για τη χρήση στον υπολογιστή. Το SDK είχε πολλές διαφορές στη σύνταξη του API της, παρείχε όμως το κατάλληλο documentation για να μπορούν οι προγραμματιστές να προσαρμόσουν τον κώδικά τους στα καινούρια δεδομένα. Με την επίσημη έκδοση δόθηκε η δυνατότητα ανάπτυξης εμπορικών εφαρμογών με την προϋπόθεση ότι θα γίνεται χρήση της συσκευής Kinect for Windows. Έτσι πολλές εταιρίες άρχισαν την ανάπτυξη εφαρμογών με τη χρήση του SDK της Microsoft.

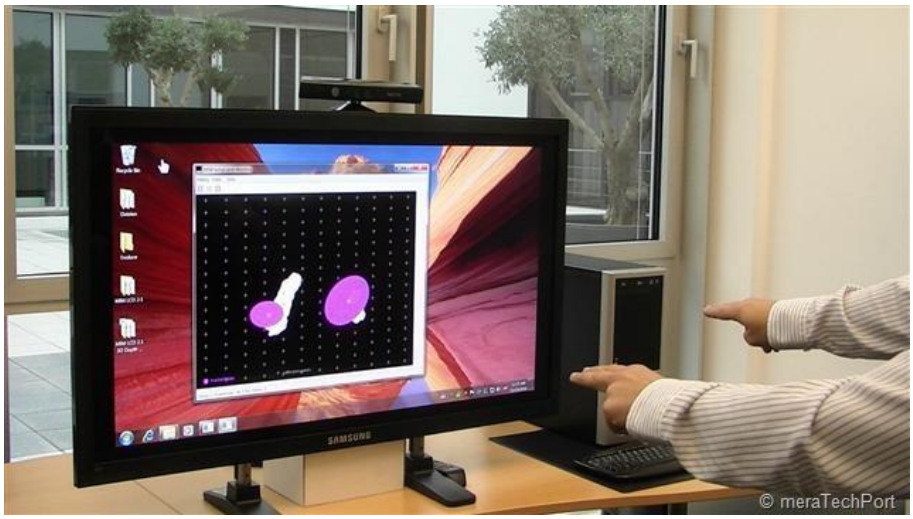


Εικόνα 16 - Ανίχνευση προσώπου με χρήση του Microsoft Kinect SDK.

Το Μάρτιο του 2012 ο Craig Eisler, γενικός διευθυντής του Kinect for Windows ανακοίνωσε ότι σχεδόν 350 εταιρίες συνεργάζονται με τη Microsoft για την ανάπτυξη διάφορων εφαρμογών με το Kinect για τα Windows , επίσης ανακοινώθηκε η 1.5 έκδοση του SDK. Τον Μάιο η Microsoft δημοσίευσε την Version 1.5 του Kinect for Windows SDK. Η έκδοση αυτή περιέχει μια καινούρια βιβλιοθήκη για ανίχνευση προσώπου (Face Detection) (βλ. Εικόνα 16), αναγνώριση φωνής σε τέσσερεις καινούριες γλώσσες – Γαλλικά, Ισπανικά, Ιταλικά και Ιαπωνικά. Επίσης αναγνωρίζει και διαλέκτους των αγγλικών αλλά και των παραπάνω γλωσσών. Όσον αφορά την ανίχνευση κίνησης του χρήστη, η Microsoft προσέθεσε την παράμετρο προσανατολισμού των αρθρώσεων του σκελετού (παράμετρο που παρείχε μέχρι τώρα μόνο το OpenNI) και ένα καινούριο mode το near ή seated mode, το οποίο χρησιμοποιεί κι εμφανίζει μόνο τα χέρια και το κεφάλι του χρήστη.

2.3.5 Evoluce SDK

Η εταιρία Evoluce ανέπτυξε ένα SDK βασισμένο στο OpenNI. Με τη χρήση της βιβλιοθήκης Emgu CV [11] πρόσθεσαν τη δυνατότητα ανίχνευσης των δακτύλων του χρήστη. Το Evoluce SDK δημοσιεύθηκε το Νοέμβριο του 2011 και πέρα από τις ιδιότητες του OpenNI παρέχει αναγνώριση φωνής (Speech Recognition), ανίχνευση χειρονομιών με τα δάκτυλα (Finger Gesture Recognition) (βλ.), υποστήριξη του Surface 2.0. Η χρήση του περιορίζεται μονάχα στα Windows 7.



Εικόνα 17 - Finger gesture recognition με τη χρήση του Evoluce SDK

2.3.6 Σύγκριση Δυνατοτήτων των API's

Μετά τη μελέτη των APIs καταλήξαμε στους παρακάτω πίνακες οι οποίοι διευκολύνουν την επιλογή της βιβλιοθήκης ανάλογα με τις απαιτήσεις της εφαρμογής την οποία θέλουμε να υλοποιήσουμε. Ο Πίνακας 2 μας δείχνει της γλώσσες στις οποίες μπορούμε να προγραμματίσουμε με κάθε API. Με **N** συμβολίζονται οι Native (έμφυτες γλώσσες – γλώσσες οι οποίες χρησιμοποιήθηκαν για την δημιουργία του API), με **W** οι wrapper γλώσσες και με **Κενό** οι γλώσσες που δεν υποστηρίζονται. Ας σημειωθεί ότι όταν χρησιμοποιούμε γλώσσα στην οποία δεν είναι γραμμένη η βιβλιοθήκη, ο κώδικάς μας είναι πιο αργός, επειδή γίνεται η μετάφρασή του κατά την εκτέλεση.

	C/C++	C#	Java	Python	Lisp	VB.NET	Actionscript	Javascript
OpenNI	N	N	W	W		W		
Libfreenect	N	W	W	N	W	W	W	W
CL NUI	N	N						
MS Kinect SDK	N	N				N		
Evoluce SDK	N	N						

Πίνακας 2 - Γλώσσες Προγραμματισμού ανά API

Στον Πίνακας 3 έχουμε τις δυνατότητες που παρέχει κάθε API. Κάποιες από αυτές μπορούν να αναπληρωθούν με κώδικα φτιαγμένο από το χρήστη, στον πίνακα περιγράφονται οι δυνατότητες που παρέχουν τα APIs ακριβώς όπως τα κατεβάζουμε.

Συμπερασματικά βλέπουμε ότι το Microsoft Kinect SDK έχει πιο πολλές δυνατότητες πολύ-τροπικής εισόδου τις οποίες μπορούμε να χρησιμοποιήσουμε, παρέχοντας, εκτός της ανίχνευσης σκελετού και φωνής, και αναγνώριση προσώπου. Ένας άλλος λόγος για τον οποίο καταλήξαμε σε αυτή τη βιβλιοθήκη είναι η native γλώσσα προγραμματισμού C# με την οποία είχαμε εξοικειωθεί. Η C# ως native γλώσσα στο Microsoft SDK μας δίνει την δυνατότητα να αποφύγουμε τυχόν delays μετάφρασης των wrappers. Το MS SDK παρέχει, επίσης, βελτιωμένους αλγορίθμους εντοπισμού σκελετού, εντοπίζοντας 20 αρθρώσεις στο ανθρώπινο σώμα με ταχύ και σταθερό αποτέλεσμα. Άλλο ένα χαρακτηριστικό που το κάνει ακόμα πιο προσιτό είναι το Near mode, η δυνατότητα να χρησιμοποιεί ο χρήστης μόνο το πάνω μέρος του σώματος του. Με το Near mode ο χρήστης μπορεί να κάθεται, με το Kinect να εντοπίζει και να χρησιμοποιεί μονάχα τις αρθρώσεις του κεφαλιού και των χεριών. Τέλος πρέπει να αναφέρουμε την βιβλιοθήκη αναγνώρισης ομιλίας την οποία παρέχει σε διάφορες γλώσσες και διαλέκτους.

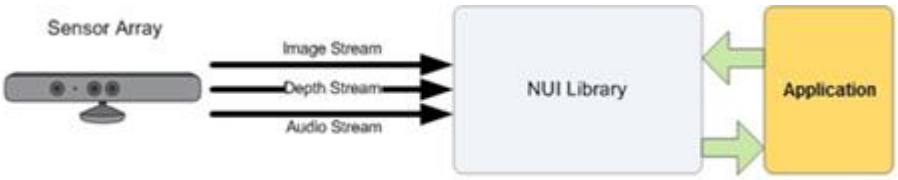
Στην επόμενη ενότητα εξηγούνται αναλυτικά τα πλεονεκτήματα της βιβλιοθήκης MS Kinect SDK. Μέθοδοι και δυνατότητες τις οποίες παρέχει στον προγραμματιστή για την υλοποίηση φυσικών διεπαφών.

		OpenNI/ NITE	Libfreenect	CL NUI	MS Kinect SDK	Evoluce SDK
Low Level API Features	Εικόνα Βάθους	✓	✓	✓	✓	✓
	RGB Εικόνα	✓	✓	✓	✓	✓
	Ήχος	✓	✓		✓	✓
	Επιταχυνσιόμετρο			✓		
	Motor-Base Control	✓	✓	✓	✓	✓
	LED Control		✓	✓		
High Level API Features	Ανίχνευση σκελετού	✓			✓	✓
	Στάση βαθμονόμησης	Όχι			Όχι	Ναι
	Προσανατολισμός αρθρώσεων	✓			✓	✓
	Near Mode				✓	
	Αρθρώσεις ανά σκελετό	15			20	15
	Ανίχνευση Χεριού	✓				✓
	Ανίχνευση Δακτύλων					✓
	Ανίχνευση Προσώπου				✓	
	Αναγνώριση Ομιλίας				✓	✓
	Αναγνώριση Χειρονομιών	✓				✓
	Εύρεση θέσης ηχητικής πηγής				✓	
	Καταγραφή και αναπαραγωγή	✓			✓	
Λοιπά Χαρακτηριστικά	Χρήση MS Kinect	✓	✓	✓	✓	✓
	Χρήση Xtion Pro LIVE	✓				
	Εμπορική άδεια χρήσης	✓	✓	✓	✓	

Πίνακας 3 - Σύγκριση δυνατοτήτων των APIs

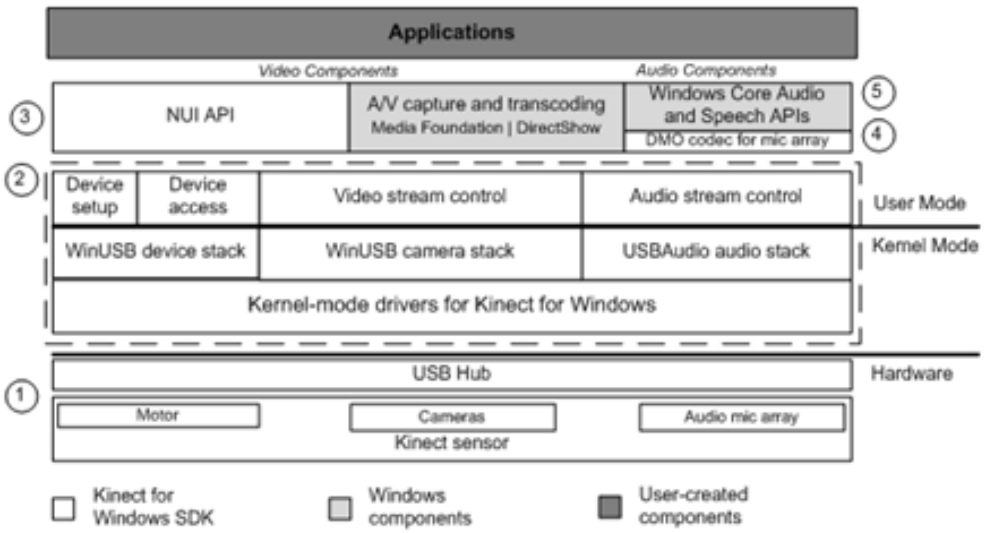
2.4 To Microsoft Kinect for Windows SDK

Στο παράδειγμα εφαρμογής το οποίο υλοποίησα σε αυτή την εργασία χρησιμοποιήσαμε το SDK της Microsoft καθώς παρέχει έτοιμο Speech Recognition API, πολύ γρήγορο Skeleton Tracking και παραπάνω αρθρώσεις ανά χρήστη (20 αρθρώσεις).



Εικόνα 18 - Η επικοινωνία Kinect και εφαρμογών μέσω του SDK

Όπως βλέπουμε στην Εικόνα 18, το Kinect μας δίνει τρεις εξόδους: Εικόνα, Βάθος και Ήχο. Το SDK αποκωδικοποιεί αυτές τις εξόδους και λειτουργεί ως ενδιάμεσος με τις εφαρμογές μας. Παρακάτω βλέπουμε την αρχιτεκτονική του Kinect SDK. Όπως περιγράφεται στο documentation της Microsoft. (βλ. Εικόνα 19).



Εικόνα 19 - Αρχιτεκτονική του Kinect SDK

2.4.1 Hardware Requirements

Για τη χρήση του SDK χρειαζόμαστε τις παρακάτω ελάχιστες απαιτήσεις:

- 32-bit (x86) or 64-bit (x64) processors
- Dual-core, 2.66-GHz or faster processor
- USB 2.0 bus dedicated to the Kinect
- 2 GB of RAM
- Graphics card that supports DirectX 9.0c
- A Microsoft Kinect for Windows Sensor

2.4.2 Software Requirements

- Microsoft Visual Studio 2010 Express or other Visual Studio 2010 edition

- .NET Framework 4 (installed with Visual Studio 2010)
- Windows 7 OS

2.4.3 Depth Stream

Το depth data (δεδομένα βάθους) stream παρέχει frames στα οποία το κάθε pixel περιέχει την καρτεσιανή απόσταση (σε χιλιοστά) από την επιφάνεια της κάμερας μέχρι το κοντινότερο αντικείμενο στις συγκεκριμένες x και y συντεταμένες, στο οπτικό πεδίο του αισθητήρα. Υπάρχουν δύο πιθανές αποστάσεις για τα δεδομένα βάθους: η προεπιλεγμένη (default range) και η κοντινή απόσταση (near range), μια από τις οποίες διαλέγουμε κατά την έναρξη του depth stream. Οι εφαρμογές μπορούν να επεξεργαστούν τα δεδομένα από το depth stream υποστηρίζοντας διάφορα χαρακτηριστικά, όπως παρακολούθηση των κινήσεων του χρήστη (user tracking) και αναγνώριση των αντικειμένων στη σκηνή ώστε να παραβλέπονται εν ώρα παιχνιδιού.

Κάθε pixel στο depth stream χρησιμοποιεί 13 bits για το βάθος και 3 bits για την αναγνώριση του χρήστη. Η τιμή βάθους 0 υποδεικνύει ότι δεν υπάρχουν δεδομένα βάθους για το συγκεκριμένο σημείο, γιατί το αντικείμενο που βρίσκεται σε αυτή τη θέση είναι είτε πολύ κοντά, είτε πολύ μακριά από τον αισθητήρα. Όταν το Skeleton Tracking είναι απενεργοποιημένο, τα 3 bits, τα οποία χρησιμοποιούνται για την αναγνώριση του χρήστη, παίρνουν την τιμή 0.

2.4.4 Color Stream

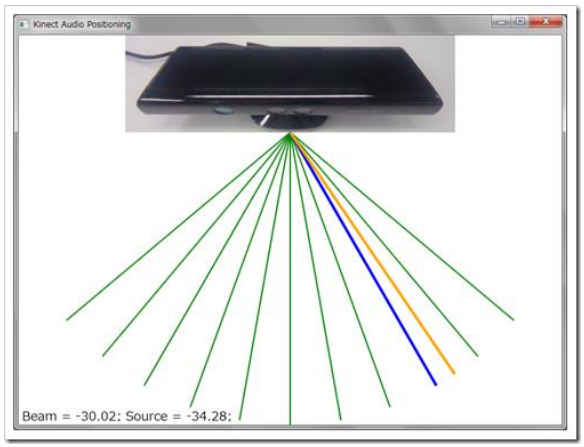
Τα δεδομένα χρώματος είναι διαθέσιμα σε δύο μορφές:

- RGB μορφή χρώματος (RGB Color Format)
  Με αυτή την επιλογή το Kinect παρέχει, ένα 32-bit, γραμμικό X8R8G8B8-μορφοποιημένο bitmap χρώματος, σε sRGB πεδίο χρώματος. Όπου X 8 bits που δε χρησιμοποιούνται, R 8 bits για το χρώμα κόκκινο, G 8 bits για το πράσινο και B 8 bits για το μπλε.
- YUV μορφή χρώματος (YUV Color Format)
  Με την επιλογή αυτή έχουμε ένα 16-bit, gamma-corrected γραμμικό UYVY-μορφοποιημένο bitmap, όπου το gamma-correction στο YUV πεδίο είναι ισοδύναμο με το sRGB gamma στο RGB πεδίο. Επειδή το YUV stream χρησιμοποιεί 16 bits ανά pixel, αυτή η μορφή χρησιμοποιεί λιγότερη μνήμη για την αποθήκευση των δεδομένων του bitmap και δεσμεύει λιγότερη μνήμη στο buffer όταν ανοίγει το color stream. Τα YUV δεδομένα είναι διαθέσιμα μόνο σε 640x480 ανάλυση και μόνο στα 15 FPS.

Και οι δυο μορφές υπολογίζονται από τα ίδια δεδομένα της κάμερας, έτσι ώστε τα δεδομένα YUV και του RGB αντιπροσωπεύουν την ίδια εικόνα. Διαλέγουμε με πια μορφή θα δουλέψουμε κατά την εκκίνηση του Color Stream.

2.4.5 Audio Stream

Το Kinect περιέχει, όπως προαναφέρθηκε μια σειρά από τέσσερα μικρόφωνα, η οποία χρησιμοποιεί 24-bit ADC (Analog-to-digital Converter) και παρέχει τοπική επεξεργασία σήματος, συμπεριλαμβανομένων των: Acoustic Echo Cancellation και Noise Suppression. Οι εφαρμογές που αναπτύσσονται με αυτό το SDK μπορούν να χρησιμοποιήσουν τη σειρά μικροφώνων για υψηλής ποιότητας καταγραφή ήχου, για την εύρεση της ηχητικής πηγής (Source Localization), για την επιλογή λήψης ήχου από μια συγκεκριμένη κατεύθυνση (Beamforming). Με τη χρήση του SDK οι εφαρμογές μας μπορούν επίσης να χρησιμοποιήσουν το Kinect ως input device για τη βιβλιοθήκη αναγνώρισης ομιλίας της Microsoft.

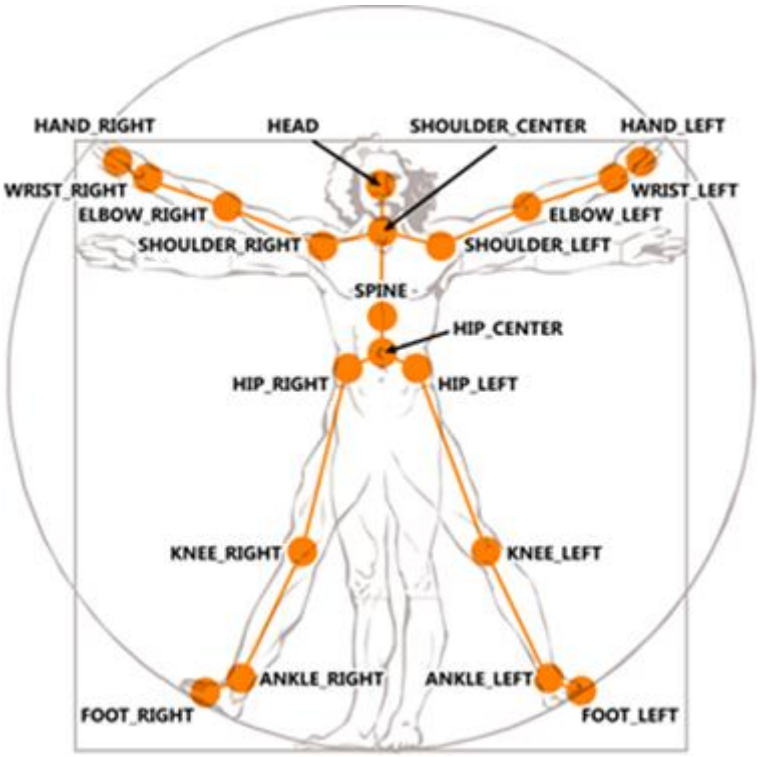


Εικόνα 20 - Εφαρμογή εύρεσης θέσης της ηχητικής πηγής

2.4.6 Skeleton Tracking

Το SDK παρέχει επίσης πληροφορίες για την θέση μέχρι και δυο χρηστών οι οποίοι βρίσκονται μπροστά από το Kinect, με λεπτομερής πληροφορίες για την θέση και τον προσανατολισμό.

Τα δεδομένα παρέχονται στην εφαρμογή ως ένα σύνολο σημείων (skeleton positions), τα οποία συγκροτούν ένα σκελετό, όπως φαίνεται στην Εικόνα 21. Ο σκελετός αυτός αντιπροσωπεύει τη θέση και την στάση του χρήστη. Για να χρησιμοποιήσει το skeleton data, μια εφαρμογή πρέπει να ενεργοποιήσει το skeleton tracking (το οποίο ενεργοποιεί τον αισθητήρα βάθους ανεξάρτητα αν έχουμε ενεργοποιήσει το depth stream).



Εικόνα 21 - Τα σημεία του σκελετού σε σχέση με το ανθρώπινο σώμα

2.5 Εφαρμογές που χρησιμοποιούν το Kinect ή το Xtion

Από την κυκλοφορία του μέχρι σήμερα το Kinect έχει χρησιμοποιηθεί σε διάφορους κλάδους, όπως: Εικονική Πραγματικότητα (Virtual Reality), Ευφυή Περιβάλλοντα (Ambient Intelligent Environments), Τεχνητή Όραση και Ρομποτική (Computer Vision and Robotics), Αναπηρίες (Disabilities) , Αναγνώριση Χειρονομιών (Gesture Recognition), Εικονική Πραγματικότητα (Virtual Reality).

2.5.1 Εικονική Πραγματικότητα (Virtual Reality),

Οι Suma et al. παρουσίασαν μια έρευνα με τίτλο "Sharing Space in Mixed and Virtual Reality Environments" [12], στην οποία περιγράφουν ένα σύστημα που εισάγει το χρήστη σε ένα εικονικό περιβάλλον μέσω ενός head-mounted display. Εφαρμόζοντας το ένα αισθητήρα βάθους πάνω στο κεφάλι του χρήστη, χρησιμοποιεί το χάρτη βάθους για να εισάγει ανθρώπους και αντικείμενα από τον αληθινό κόσμο στον εικονικό. Επιτρέποντας έτσι την επικοινωνία μεταξύ χρηστών οι οποίοι, διαφορετικά, θα ήταν απομονωμένοι στον εικονικό κόσμο της εφαρμογής. Στην Εικόνα 22 βλέπουμε πως ο χρήστης βλέπει ένα άτομο μέσα στο εικονικό περιβάλλον.

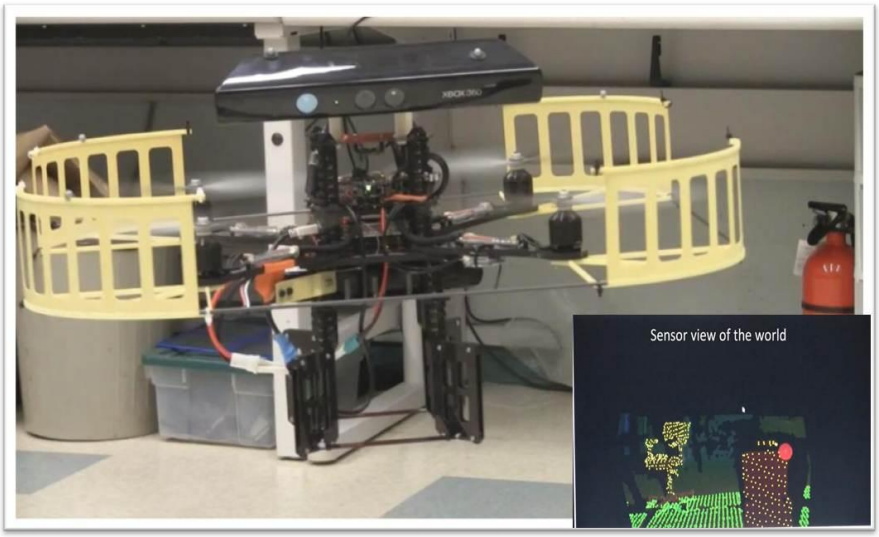
Εικόνα 22 - Εισαγωγή των ανθρώπων που είναι μπροστά από το χρήστη, στον εικονικό κόσμο.

2.5.2 Περιβάλλον Διάχυτης Νοημοσύνης (Ambient Intelligent Environments)

Μια αντιπροσωπευτική έρευνα με τίτλο "Gesture-Based Hybrid Approach for HCI in Ambient Intelligent Environments" από τους Stefano Carrino et al. [13], στην οποία παρουσιάζεται ένα σύστημα με όνομα ARAMIS. Στόχος του συστήματος, η βελτίωση της αλληλεπίδρασης του ανθρώπου με ένα ευφυές περιβάλλον, ακολουθώντας μια υβριδική προσέγγιση. Η προσέγγιση αυτή χαρακτηρίζεται υβριδική επειδή χρησιμοποιείται ένας συνδυασμός από τεχνικές όπως: wearable and pervasive computing paradigms, brute force, fuzzy and ML methods, virtual and real worlds, optical and non-optical sensing technologies. Επίσης για να αξιολογήσουν την προσέγγιση αυτή, δημιούργησαν ένα πολύ-τροπικό (multimodal) framework, στο οποίο οι χειρονομίες (gestures) επιλέχθηκαν ως κύριος τρόπος αλληλεπίδρασης. Στόχος του framework είναι πρώτον η συγχώνευση και χρήση πληροφοριών από ετερογενής αισθητήρες και δεύτερον να προσφέρουν ένα απλό εργαλείο για την σύνδεση τέτοιων συσκευών.

2.5.3 Τεχνητή Όραση και Ρομποτική

Ο ερευνητής Patrick Bouffard, από το Hybrid Systems Lab του πανεπιστημίου του Berkeley (EECS department) [31], αποφάσισε να χρησιμοποιήσει το Kinect επάνω σε ένα ελικόπτερο quadrotor. Χρησιμοποιώντας τον αισθητήρα βάθους, το ελικόπτερο, δεχόταν στον ενσωματωμένο υπολογιστή του ένα σύνολο από σημεία (Point Cloud), όπως βλέπουμε και στην Εικόνα 23, και μετά από μια επεξεργασία των δεδομένων αυτών μπορούσε να πετάει αυτόνομα και να αποφεύγει εμπόδια. Για την επικοινωνία του με το Kinect χρησιμοποιήθηκε η βιβλιοθήκη Open Kinect.



Εικόνα 23 - Το ελικοπτεράκι quadrotor με το Kinect

2.5.4 Αναπηρίες (Disabilities)

Οι Chang Y et al. παρουσίασαν μια έρευνα με τίτλο Kinerehab [15], η οποία είναι ένα σύστημα βασισμένο στο Kinect για φυσιοθεραπεία. Μια έρευνα για νεαρούς ενήλικες με κινητικές αναπηρίες ανεπτυγμένη με απλή αναγνώριση χειρονομιών (gesture recognition), για να κάνει την θεραπεία πιο ενδιαφέρουσα και προσβάσιμη για τον ασθενή. Χρησιμοποιώντας το Kinect ο ασθενής δε χρειάζεται να φορά κανένα αισθητήρα πάνω του, επίσης τα αποτελέσματα αποθηκεύονται αυτόματα κι έτσι ο γιατρός μπορεί να αφοσιωθεί περισσότερο στον ασθενή.

2.5.5 Αναγνώριση Χειρονομιών (Gesture Recognition)

Μια έρευνα η οποία αναπτύχθηκε για τον τομέα της Ιατρικής ήταν αυτή των Gallo et al. [16]. Το σύστημά τους επιτρέπει στο χρήστη να αλληλεπιδράσει με ιατρικές εικόνες χωρίς να φορά ή να κρατάει κάποια συσκευή (βλ. Εικόνα 24). Με τη χρήση του OpenNI ως βιβλιοθήκη επικοινωνίας με το Kinect, και το Open CV – μια βιβλιοθήκη επεξεργασίας εικόνας, ο χρήστης έχει την δυνατότητα να πλοηγηθεί, να επιλέξει, να γυρίσει, να κινήσει, να εστιάσει, να μεγαλώσει ή να μικρύνει και να διαγράψει μια εικόνα από την οθόνη.

Εικόνα 24 - Πλοήγηση σε εικόνα εγκεφάλου

Το project FAAST (The Flexible Action and Articulated Toolkit) [30], το οποίο αναπτύχθηκε από τους Suma et al., είναι ένα ενδιάμεσο λογισμικό (Middleware) που μπορεί να χρησιμοποιήσει ο προγραμματιστής για τη χρήση βασικών χειρονομιών (Basic gestures). Το FAAST χρησιμοποιεί το Open NI για την επικοινωνία με το Kinect (ή άλλο υποστηριζόμενο αισθητήρα βάθους) και δίνει τη δυνατότητα χρήσης ολόκληρου του σώματος του χρήστη ως μέσω αλληλεπίδρασης με εφαρμογές εικονικής πραγματικότητας και βιντεοπαιχνίδια. Χωρίζοντας τις πληροφορίες που λαμβάνει από τον αισθητήρα σε δύο κατηγορίες: Actions και Articulated Skeletons, και χρησιμοποιώντας ένα Virtual Reality Peripheral Network (VRPN) μεταφέρει τις πληροφορίες ως και τεσσάρων σκελετών σε ένα δίκτυο. Το FAAST μπορεί επίσης να μιμηθεί το πάτημα ενός κουμπιού από το πληκτρολόγιο αναγνωρίζοντας κάποια κίνηση ή χειρονομία από το χρήστη. Στην Εικόνα 25 βλέπουμε τον χρήστη να χρησιμοποιεί το Kinect ως εναλλακτική είσοδο για την αλληλεπίδρασή του με το παιχνίδι World of Warcraft.

Εικόνα 25 - Χρήση του Kinect μέσω του FAAST ως είσοδο για το παιχνίδι World of Warcraft (κανονική συσκευή εισόδου: πληκτρολόγιο)

3 Πολύ-τροπικές Διεπαφές - Multimodal Interfaces

Οι πολύ-τροπικές διεπαφές συνδυάζουν διάφορους τρόπους εισόδου για τον χρήστη, όπως φωνή, γραφίδα, αφή, χειρονομίες, βλέμμα, και κίνηση κεφαλιού και σώματος. Αντιπροσωπεύουν μια καινούρια κατεύθυνση, παρεκκλίνοντας από τις τυπικές γραφικές διεπαφές, επειδή περιλαμβάνουν τεχνολογίες αναγνώρισης σχεδιασμένες να χειρίζονται συνεχόμενα και ταυτόχρονα εισόδους από παράλληλα εισερχόμενα κανάλια, και να επεξεργάζονται τις ασάφειες τον εισόδων με πιθανολογικές μεθόδους, κατανεμημένη επεξεργασία και αρχιτεκτονικές βασισμένες στο χρόνο. Αυτό το νέο είδος διεπαφών στοχεύει στην αναγνώριση της ανθρώπινης επικοινωνίας έτσι ώστε η αλληλεπίδραση με τον υπολογιστή να είναι πιο φυσική για τον χρήστη. Κάποια από τα πλεονεκτήματα των διεπαφών αυτών είναι:

Ευχρηστία (Robustness): Ο πλεονασμός στην πολύ-τροπική είσοδο αυξάνει την ποιότητα της επικοινωνίας ανάμεσα στο χρήστη και στο σύστημα, εκχωρώντας παρόμοιες ή συσχετιζόμενες πληροφορίες χρησιμοποιώντας διαφορετικές εισόδους αυξάνουμε την πιθανότητα της αναγνώρισης [17]. Στρατηγικές αμοιβαίας αποσαφήνισης και ενοποίησης δεδομένων εισόδου, στις οποίες οι ενσωματωμένες πολύ-τροπικές είσοδοι επεξεργάζονται μαζί, μπορούν να βοηθήσουν στην αύξηση της επίδοσης του συστήματος.

Φυσικότητα (Naturalness): Η πολύ-τροπική επικοινωνία μπορεί να οδηγήσει σε ένα μεγάλο βαθμό φυσικότητας, αξιοποιώντας τις καθιερωμένες πρακτικές της ανθρώπινης επικοινωνίας [17]. Πολύπλοκες διαδικασίες μπορούν να διευκολυνθούν από την πολύ-τροπική αλληλεπίδραση, επειδή το παράδειγμα αυξάνει αποτελεσματικά το εύρος ζώνης της επικοινωνίας μεταξύ του χρήστη και του συστήματος αυξάνοντας το επίπεδο της εκφραστικότητας εισόδου.

Ελαστικότητα (Flexibility): Ένα μεγάλο πλεονέκτημα των πολύ-τροπικών διεπαφών, είναι ότι επιτρέπει στους χρήστες, να αντιλαμβάνονται και να δομούν την επικοινωνία με διάφορους τρόπους ανάλογα με το περιβάλλον. Οι χρήστες διαλέγουν ποιους τρόπους (modalities) θα χρησιμοποιήσουν και πως θα διαρθρωθούν με βάση τους σημασιολογικούς, χρονικούς και συντακτικούς κανόνες. Οι χρήστες έχουν δείξει επίσης προτίμηση στις πολύ-τροπικές εισόδους για τις εφαρμογές που έχουν να κάνουν με προσανατολισμό, ωστόσο, τείνουν να εναλλάσσουν μόνο-τροπική και πολύ-τροπική αλληλεπίδραση όπου αυτοί θεωρούν βολική [19].

Ελαχιστοποίηση Σφαλμάτων (Minimizing Errors): Έχει αποδειχθεί ότι, οι πολύ-τροπικές διεπαφές, αυξάνουν την απόδοση του συστήματος ελαττώνοντας τον αριθμό των σφαλμάτων (μέσω αποφυγής σφαλμάτων) και των προβλημάτων που δημιουργεί η αυθόρμητη μη ευφράδεια λόγου, σε σύγκριση με τις διεπαφές οι οποίες χρησιμοποιούν μονάχα αναγνώριση φωνής, για διαδραστικές εφαρμογές που έχουν να κάνουν με χώρο [19].

3.1 Τεχνολογία μέχρι σήμερα

Τα συστήματα πολύ-τροπικής εισόδου μέχρι σήμερα, είναι ικανά να επεξεργαστούν δυο με τρείς κανάλια εισόδου, είναι ειδικευμένα για κάποιες εφαρμογές και έχουν περιορισμένης αλληλεπίδρασης λεξιλόγια και γραμματικές [21]. Πολλά θέματα που σχετίζονται με τον συγχρονισμό, την σύνθεση των πολύ-τροπικών εισόδων (ερμηνεία των εισόδων), ο διαχωρισμός των εξόδων (παρουσίαση εξόδων) και η αρχιτεκτονική της αλληλεπίδρασης ερευνούνται ακόμη.

Οι δύο ποιο διαδεδομένοι τύποι συστημάτων πολύ-τροπικών εισόδων, στα οποία το ποντίκι και το πληκτρολόγιο αντικαταστάθηκαν πλήρως, είναι αυτά που συνδυάζουν ομιλία και γραφίδα [21] ή ομιλία και κίνηση χειλιών [22][23][24]. Για τα συστήματα που κάνουν χρήση ομιλίας και γραφίδας, η

ομιλία επεξεργάζεται μερικές φορές παράλληλα με περίπλοκες κινήσεις της γραφίδας, με αποτέλεσμα εκατοντάδων διαφορετικών συμβολικών ερμηνειών πέραν του κλασσικού pointing [25]. Στα συστήματα που κάνουν χρήση ομιλίας και κίνησης χειλιών, η ομιλία επεξεργάζεται μαζί με την κίνηση των χειλιών κατά τη διάρκεια της φυσικής αλληλεπίδρασης εικόνας και ήχου. Οι ερευνητές έχουν στραφεί προς το ποσοτικό μοντέλο αλληλεπίδρασης, στη συγχρονισμένη επεξεργασία των χαρακτηριστικών των δύο εισόδων, και έχουν αναπτύξει καινοτόμες αρχιτεκτονικές με βάση το χρόνο για την επεξεργασία αυτών τον μοντέλων με εύχρηστους τρόπους.

Τα πολύ-τροπικά συστήματα τα οποία περιλαμβάνουν συνεχή 3-Δ χειρονομίες και ομιλία έχουν κάνει την εμφάνισή τους, ωστόσο, δεν είναι τόσο ώριμα σε σύγκριση με της τεχνολογίες που χρησιμοποιούνται για 2-Δ εισόδου με γραφίδα και μελάνι [26][27]. Όπως είναι φυσικό, εμφανίζονται νέες προκλήσεις με την χρήση 3-Δ χειρονομιών στις οποίες περιλαμβάνεται ο διαχωρισμός (segmentation), η εξαγωγή χαρακτηριστικών (Feature extraction) και η ερμηνεία αυθόρμητων κινήσεων (spontaneous movement trajectories), τα οποία πρέπει να επιλυθούν πριν τα συστήματα αυτά γίνουν πιο διαδεδομένα. Η εισαγωγή άλλων τρόπων εισόδου τα οποία βασίζονται σε συσκευές λήψης εικόνας, όπως βλέμμα, εκφράσεις προσώπου, κινήσεις με το κεφάλι και κινήσεις του σώματος αποκτούν μεγάλο ενδιαφέρον στα νέα είδη αλληλεπίδρασης, τα οποία είναι ακόμα σε εμβρυακό επίπεδο [27][28][29].

4 Τεχνολογίες Υλοποίησης

Σε αυτό το κεφάλαιο θα παρουσιαστούν σύντομα οι βιβλιοθήκες, οι οποίες χρησιμοποιήθηκαν για την ανάπτυξη της εφαρμογής.

4.1 XML

Η XML (Extensible Markup Language) είναι μια γλώσσα σήμανσης, που περιέχει ένα σύνολο κανόνων για την ηλεκτρονική κωδικοποίηση κειμένων. Προέρχεται από την SGML (Standard Generalized Markup Language) από την οποία προέρχεται και η HTML.

Στο παράδειγμά μου χρησιμοποίησα απλά αρχεία XML για την αποθήκευση των gestures, των voice command, των προτάσεων και των ικανών εφαρμογών.

4.2 Microsoft Visual C#

Η C# είναι μια αντικειμενοστραφής γλώσσα προγραμματισμού που δίνει τη δυνατότητα ανάπτυξης διαφόρων ασφαλών και εύχρηστων εφαρμογών οι οποίες τρέχουν με το .NET Framework. Μπορεί να χρησιμοποιηθεί για τη δημιουργία κλασσικών εφαρμογών Windows, διαδικτυακές εφαρμογές με XML, client-server εφαρμογές, εφαρμογές με τη χρήση βάσεων δεδομένων, κ.α. Η C# είναι ιδιαίτερα εκφραστική, ωστόσο είναι απλή και εύκολη στην εκμάθηση. Η σύνταξη με τη χρήση άγκιστρων μπορεί εύκολα να αναγνωριστεί από οποιονδήποτε είναι οικείος με C, C++ ή Java. Προγραμματιστές που ξέρουν οποιαδήποτε από αυτές τις γλώσσες είναι τυπικά έτοιμοι να δουλέψουν παραγωγικά στη C# μέσα σε μικρό χρονικό διάστημα. Η σύνταξή της απλοποιεί πολλές από τις πολυπλοκότητες της C++ και παρέχει ισχυρά χαρακτηριστικά όπως nullable τύπους μεταβλητών, άμεση πρόσβαση στη μνήμη τα οποία δεν υποστηρίζει η Java.

4.3 Microsoft XNA Framework

Το XNA είναι μια συλλογή από εργαλεία με ένα runtime περιβάλλον ανεπτυγμένα από την Microsoft, με σκοπό την ανάπτυξη και διαχείριση παιχνιδιών. Στόχος της ανάπτυξης αυτής της εργαλειοθήκης είναι να απαλλάξει τους προγραμματιστές παιχνιδιών από την συγγραφή ίδιων κομματιών κώδικα πολλές φορές και να ενώσει διάφορες δυνατότητες της παραγωγής παιχνιδιών σε ένα σύστημα. Το όνομα "XNA" προήλθε από το όνομα που είχε κατά την ανάπτυξή του το οποίο ήταν: Xbox New Architecture. Αντί όμως να δημοσιευθεί με το όνομα του Xbox, το Xbox 360 δημοσιεύθηκε, κι έτσι το XNA τώρα πια σημαίνει "**X**NA's **N**ot **A**cronymed".

5 Αρχιτεκτονική του Συστήματος

Η αρχιτεκτονική του συστήματός αποτελείται από τρία βασικά κομμάτια τα οποία συνεργάζονται για (α) να δεχθούν δεδομένα από διάφορες μορφές εισόδου (input modalities) όπως ήχο, έγχρωμη εικόνα και εικόνα βάθους, (β) να συντάσσουν τις εντολές που εντοπίστηκαν και (γ) να προσδιορίσουν ποιες εφαρμογές είναι ενεργοποιημένες και μπορούν να χρησιμοποιήσουν τις συνταγμένες εντολές. Περιφερειακά στο σύστημά μας είναι τα APIs των συσκευών εισόδου τα οποία παρέχουν τις διάφορες μορφές εισόδου (multimodal inputs) και το λειτουργικό σύστημα το οποίο δίνει πληροφορίες στο σύστημα μας για τις τρέχον εφαρμογές. Τα κομμάτια αυτά της αρχιτεκτονικής είναι: ο "Sentence Compiler", ο "Action Sentences Modulator" και ο "Context Information Manager".

5.1 Sentence Compiler

Ο "Sentence Compiler" (βλ. Εικόνα 5) είναι υπεύθυνος για την συλλογή των εντολών από τα διάφορα είδη εισόδου (input modalities) και την σύνταξη των προτάσεων δράσης (action sentences). Αποτελείται από δυο μέρη, το "Qualifier Input Control" (Ελεγκτής λέξεων εισόδου) και το "Sentence Syntax Evaluation" (Αξιολόγηση σύνταξης της πρότασης). Το πρώτο δέχεται τις διάφορες εισόδους παράλληλα και συνεχόμενα. Όταν, για παράδειγμα, εντοπιστεί μια χειρονομία ή μια φωνητική εντολή, αρχίζει μια διαδικασία σύνταξης μιας πρότασης δράσης. Η εντολή που εντοπίστηκε περνά στο "Sentence Syntax Evaluation" ως εν δυνάμει πρόταση για να ελεγχθεί εάν υπάρχει στη Grammar database (μία βάση δεδομένων για την γραμματική μου, υλοποιημένη σε XML, στη οποία περιέχονται δομημένες προτάσεις) είτε ως ολοκληρωμένη πρόταση, είτε ως τμήμα ολοκληρωμένης πρότασης. Αν υπάρχει τουλάχιστον σε μία δομημένη πρόταση, αλλά η πρόταση χρειάζεται κι άλλες εντολές για τη σύνταξή της τότε κρατάμε την εντολή/ες που έχουμε και γυρίζουμε πίσω στο "Qualifier Input Control" περιμένοντας μια καινούρια λέξη-εντολή. Όταν συμπληρωθεί και αναγνωρισθεί μια ολοκληρωμένη πρόταση, από το "Sentence Compiler", στέλνεται στον "Action Sentences Manipulator".

5.1.1 Ιδιότητες του Sentence Compiler

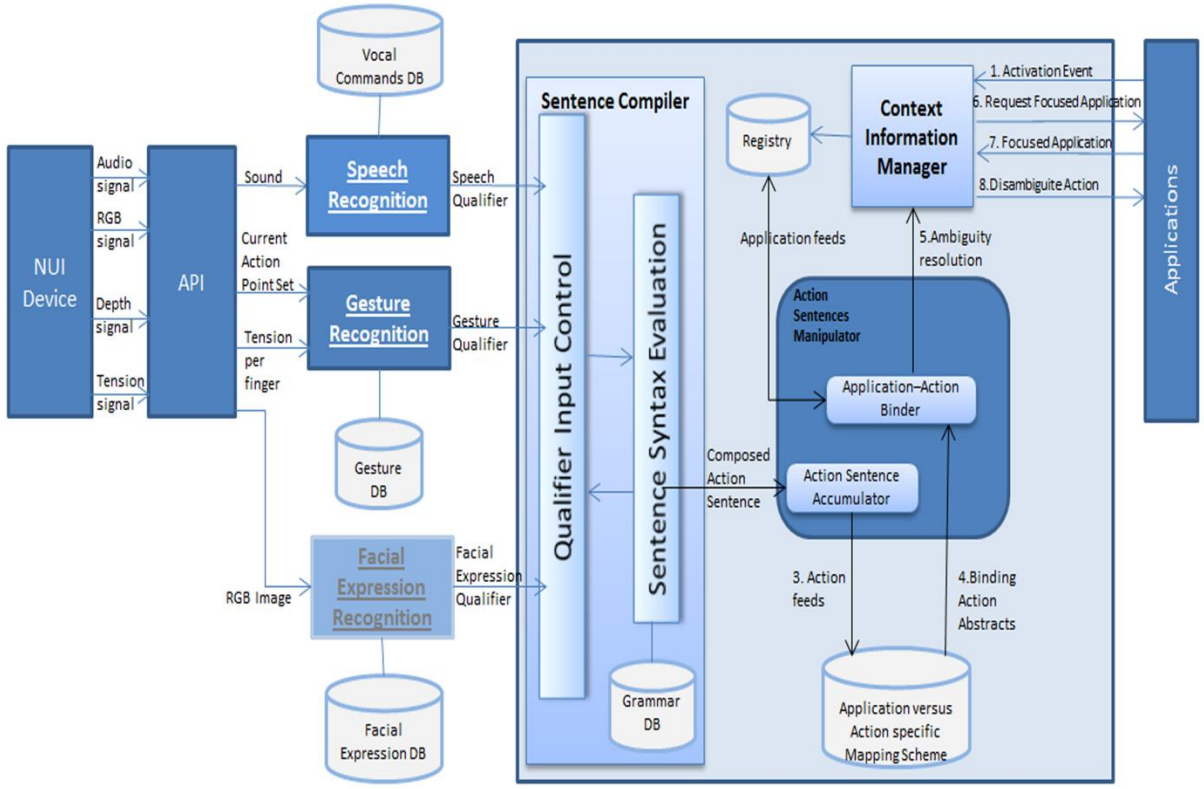
- Εάν ο Sentence Compiler λάβει μια εντολή που δεν υπάρχει σε καμία πρόταση της γραμματικής του (Grammar Database) τότε το Input Control καθαρίζει τη μνήμη του και περιμένει καινούρια λέξη για να ξεκινήσει τη σύνταξη από την αρχή.
- Εάν κατά τη σύνταξη μιας πρότασης (ενώ δηλαδή υπάρχουν εντολές μέσα στη μνήμη του Input Control) ληφθεί μια λέξη που καθιστά την εν δυνάμει πρόταση άγνωστη προς το σύστημα, τότε το Input Control καθαρίζει τη μνήμη του και περιμένει καινούρια λέξη για να ξεκινήσει τη σύνταξη από την αρχή.

5.2 Action Sentence Manipulator

Ο "Action Sentences Manipulator" (βλ. Εικόνα 5) αποτελείται από τον "Action Sentence Accumulator" και τον "Application-Action Binder". Είναι υπεύθυνος για την εύρεση των εφαρμογών οι οποίες μπορούν να ανταποκριθούν στην πρόταση δράσης (Action Sentence) την οποία δέχθηκε. Για να γίνει αυτό, ο "Action Sentence Accumulator" συγκρίνει την πρόταση που δέχθηκε με τον "Sentence Compiler", με τις πληροφορίες που βρίσκονται στο "Application versus Action specific Scheme" (και αυτό υλοποιημένο σε XML), φιλτράροντας τις εφαρμογές οι οποίες είναι συμβατές ικανές να τη χρησιμοποιήσουν. Έπειτα ο "Application-Action Binder" δέχεται τις εφαρμογές αυτές και σε συνδυασμό με ποιες εφαρμογές είναι ανοικτές, ενημερώνει τον "Context Information Manager".

5.3 Context Information Manager

Ο "Context Information Manager" (CIM) του συστήματος, είναι υπεύθυνος για την επικοινωνία του λειτουργικού συστήματος και την παροχή πληροφοριών σε σχέση με τις τρέχουσες εφαρμογές (π.χ. ενεργές εφαρμογές, ποια εφαρμογή είναι focused, ποιες minimized, etc.). Κάθε φορά που ενεργοποιείται μία εφαρμογή, καταγράφεται από τον CIM. Έπειτα ο CIM στέλνει τη λίστα με τις τρέχουσες εφαρμογές στο Action Sentence Manipulator. Επίσης είναι υπεύθυνος για την αποσαφήνιση της Action Sentence της οποίας δέχεται από τον Action Sentence Manipulator. Όταν ο CIM δέχεται μια πρόταση (και τις εφαρμογές στις οποίες μπορεί να εκτελεστεί), κάνει ένα ερώτημα στο λειτουργικό σύστημα για να πάρει την εφαρμογή η οποία είναι εστιασμένη εκείνη τη στιγμή. Όταν πάρει απάντηση, ελέγχει εάν η εφαρμογή υπάρχει στη λίστα των ικανών εφαρμογών. Εάν υπάρχει, η πρόταση μεταφράζεται στην εντολή που αντιστοιχεί στη συγκεκριμένη εφαρμογή και εκτελείται. Εάν δεν υπάρχει τότε η πρόταση μεταφράζεται για όλες τις εφαρμογές τις λίστας και ο χρήστης έρχεται να επιλέξει σε ποια/ες θέλει να την εκτελέσει.



Εικόνα 26 - Η Αρχιτεκτονική του Συστήματος

5.4 Συσκευές εισόδου και APIs

Οι συσκευές που δέχεται το σύστημά μας δεν είναι συγκεκριμένες, είναι συσκευές που καταγράφουν ένα ή παραπάνω χαρακτηριστικά της ανθρώπινης επικοινωνίας (όπως ομιλία, κινήσεις χεριών, βλέμμα, κίνηση χειλιών κ.α.) και μέσω των API τους παίρνουμε τις κατάλληλες πληροφορίες ως εισόδους στους αλγόριθμους αναγνώρισης του κάθε χαρακτηριστικού. Τέτοιες συσκευές θα μπορούσαν να είναι η κάμερα, ο αισθητήρας βάθους, γάντια εντοπισμού κίνησης των δακτύλων, γραφίδα, κ.α., το κάθε ένα από αυτά και με ένα διαφορετικό API για την λειτουργία του. Τα APIs είναι βιβλιοθήκες οι οποίες χρησιμοποιούνται για την ενεργοποίηση, την απενεργοποίηση και γενικά το χειρισμό μιας συσκευής από το σύστημα. Με τη χρήση των μεθόδων τους που περιέχουν χειριζόμαστε τα input events και τα προωθούμε ως εισόδους στα πρότυπα αναγνώρισης του συστήματος.

5.5 Ανίχνευση και αναγνώριση σημάτων εισόδου

Για να αντλήσουμε τις πληροφορίες τις οποίες χρειαζόμαστε από τα κανάλια εισόδου που χρησιμοποιούμε, χρειαζόμαστε αλγόριθμους αναγνώρισης και ανίχνευσης. Για παράδειγμα για πάρουμε την πληροφορία ότι ο χρήστης χαμογελά, χρειάζεται πρώτα να βρούμε το κεφάλι του (ή μεμονωμένα τα χείλη του) κι έπειτα να αναγνωρίσουμε εάν χαμογελά ή όχι. Οι αλγόριθμοι αναγνώρισης ποικίλουν και χρησιμοποιούν διάφορες τεχνικές, ένα όμως χαρακτηριστικό που απαιτεί το σύστημά μας είναι η απόδοση των αποτελεσμάτων σε λέξεις. Αναγνωρίζοντας ένα χαμόγελο, μια κίνηση του χεριού ή μια κίνηση της γραφίδας (προκαθορισμένη από εμάς) πρέπει ο αλγόριθμος να είναι σε θέση να αποδίδει την αντίστοιχη λέξη (Qualifier) στο σύστημα ώστε να μπορεί να το εντάξει σε μία πρόταση.

5.5.1 Αναγνώριση Χειρονομίας (Gesture Recognition) και Gesture DB

Μια χειρονομία (Gesture) είναι ένα είδος μη-λεκτικής επικοινωνίας στην οποία κινήσεις του σώματος αποδίδουν συγκεκριμένα μηνύματα, είτε αντικαθιστώντας την ομιλία, είτε παράλληλα με αυτή. Στην επικοινωνία μεταξύ ανθρώπων, οι χειρονομίες περιλαμβάνουν κινήσεις με τα χέρια, ή άλλα μέρη του σώματος. Οι χειρονομίες δίνουν τη δυνατότητα στους ανθρώπους να μεταδώσουν διάφορα συναισθήματα, από περιφρόνηση και εχθρότητα μέχρι έγκριση και αγάπη, με τη γλώσσα του σώματος.

Η επιστήμη υπολογιστών χρησιμοποιεί μαθηματικούς αλγορίθμους για την αναγνώριση αυτών των χειρονομιών, με σκοπό την ανάπτυξη πιο φυσικών τρόπων επικοινωνίας χρήστη – υπολογιστή. Η αναγνώριση μπορεί να πραγματοποιηθεί με τεχνικές από τα επιστημονικά πεδία της τεχνητής όρασης και της επεξεργασίας εικόνας.

Στην αρχιτεκτονική μας η αναγνώριση χειρονομιών μπορεί να περιλαμβάνει κινήσεις χεριών, ή άλλων μερών του σώματος, δακτύλων ή ότι άλλο θα μπορούσε να χρησιμοποιήσει ο χρήστης ώστε να εκφράσει τι θέλει να κάνει χωρίς να χρησιμοποιήσει την ομιλία.

Βασικό χαρακτηριστικό η απόδοση του Gesture ως λέξη σαν είσοδο στο Sentence Compiler του συστήματος. Πληροφορίες για τα Gesture και οι αντίστοιχες λέξεις (Gesture Qualifiers) υπάρχουν στο **Gesture DB**, το οποίο είναι σε μορφή XML.

5.5.2 Αναγνώριση Ομιλίας (Speech Recognition) και Voice Commands DB

Στην επιστήμη υπολογιστών, η αναγνώριση ομιλίας είναι η μετάφραση των λέξεων από προφορικό σε γραπτό λόγο. Δύο βασικές κατηγορίες συστημάτων είναι τα "ανεξάρτητα από τον ομιλητή" (Speaker Independent) και τα "εξαρτημένα από τον ομιλητή" (Speaker Dependent) συστήματα. Τα Speaker Dependent χρησιμοποιούν "εκπαίδευση" του συστήματος κατά την οποία ο χρήστης διαβάζει κομμάτια κειμένου, έτσι ώστε το σύστημα να αναλύσει την φωνή του και να τη χρησιμοποιήσει για να βελτιώσει την αναγνώριση ομιλίας του σε σχέση μ' αυτόν. Τα Speaker Independent δεν χρησιμοποιούν "εκπαίδευση", δεν απαιτούν λοιπόν τόσο χρόνο προσαρμογής αλλά δεν έχουν και τις ίδιες επιδόσεις με τα Dependent.

Στην αρχιτεκτονική μας οι λέξης που δέχεται το σύστημά μας είναι αποθηκευμένες στο **Vocal Commands DB**, το οποίο περιέχει και τις αντίστοιχες λέξεις που στέλνονται στο Sentence Compiler. Μια λέξη (Speech qualifier) μπορεί να αντιστοιχεί σε πολλές φωνητικές εντολές, για παράδειγμα το Yes μπορεί να ανιχνευθεί λέγοντας Yeah, Yep, Yes. Χρησιμοποιούμε Speaker Independent συστήματα αναγνώρισης φωνής, για να αποφύγουμε τη χρονοβόρα διαδικασία εκπαίδευσης του συστήματος.

5.5.3 Αναγνώριση Έκφρασης Προσώπου (Facial Expression Recognition) και Facial Expression DB

Η έκφραση του προσώπου είναι μία ή περισσότερες κινήσεις ή θέσεις των μυών του προσώπου. Αυτές οι κινήσεις μεταδίδουν τα συναισθήματα του ατόμου στους παρατηρητές. Οι εκφράσεις του προσώπου είναι άλλο ένα είδος μη-λεκτικής επικοινωνίας.

Στην αρχιτεκτονική μας με τον όρο Facial Expression εννοούμε κάθε πληροφορία που μπορούμε να αντλήσουμε από το πρόσωπο του χρήστη, όπως κίνηση χειλιών, βλέμμα, έκφραση κλπ. Η **Facial Expression DB** υποστηρίζει τους αλγόριθμους αναγνώρισής μας έχοντας τις εκφράσεις που μπορούμε να αναγνωρίσουμε και τις αντίστοιχες λέξεις (Facial Expression Qualifiers) τις οποίες στέλνουν στον Sentence Compiler.

5.5.4 Application versus Action Specific Action Scheme

Το Application vs. Action Specific Action Scheme είναι ένα XML αρχείο. Στο αρχείο απαρτίζεται από προτάσεις και ικανές εφαρμογές, δηλαδή εφαρμογές που μπορούν να εκτελέσουν την αντίστοιχη ολοκληρωμένη πρόταση. Το XML αυτό, χρησιμοποιείται και από τα δύο κομμάτια του Action Sentence Manipulator.

6 Υλοποίηση

Η εφαρμογή μας χρησιμοποιεί το Microsoft Kinect ως συσκευή πολύ-τροπικής εισόδου (multimodal input device) το οποίο παρέχει ήχο, εικόνα, και χάρτη βάθους. Με τη χρήση του Microsoft Kinect SDK, όπως προανέφερα, επεξεργαζόμαστε τις πληροφορίες βάθους για να πάρουμε τον σκελετό του χρήστη ως Action Points και τη φωνή του χρήστη ως ηχητική είσοδο. Η θέση των Action Points ανανεώνεται με συχνότητα 30 φορές το δευτερόλεπτο, όσο είναι και το frame rate του skeleton tracking.

Χρησιμοποιούμε αυτά τα Action Points για να εφαρμόσουμε ένα απλό Gesture Recognition, το οποίο ελέγχει αν ο χρήστης κάνει κάποια χειρονομία από τη βιβλιοθήκη μας. Το gesture recognition σε αυτή την εφαρμογή περιορίζεται σε swing left/ right/up/down. Επειδή έχουμε μία χειρονομία με διαφορετικές κατευθύνσεις, στο συγκεκριμένο παράδειγμα, παραλείψαμε την υλοποίηση του Gesture XML αρχείου της αρχιτεκτονικής. Ελέγχουμε την θέση των action points ανά 10 frames, και συγκρίνουμε τα αποτελέσματα με τους κανόνες που έχουμε θεσπίσει για τα swing gestures για να αποδώσουμε την κατάλληλη λέξη.

Η επεξεργασία ήχου γίνεται σε πραγματικό χρόνο με τη χρήση του Microsoft Speech Recognition SDK, το οποίο ανιχνεύει λέξεις που είπε ο χρήστης και έχουμε θεσπίσει εμείς. Το Vocal Command XML παραλείφθηκε στο συγκεκριμένο παράδειγμα επειδή το μικρό λεξιλόγιο της εφαρμογής ήταν εύκολο να περαστεί αμέσως μέσα στο πρόγραμμα.

Στο παράδειγμα έτρεξα δύο διαφορετικά σενάρια χρήσης του συστήματος. Στο πρώτο υπάρχουν ενεργοποιημένες κάποιες εφαρμογές, κάποιες από αυτές είναι minimized (όπως τα Skype, Mozilla κλπ), και δύο από αυτές είναι ανοικτές, οι οποίες είναι κι αυτές που μπορούν να ανταποκριθούν στην πολύ-τροπική μας είσοδο. Μια από τις ανοικτές εφαρμογές είναι εστιασμένη (focused), με όνομα Puzzle application, ενώ η άλλη η οποία είναι μια παρουσίαση του PowerPoint είναι απλά ανοικτή. Το δεύτερο σενάριο κάνει χρήση μιας και μόνο εφαρμογής, η οποία είναι το Puzzle και είναι ανοικτή και εστιασμένη (focused), και χρησιμοποιεί μια πρόταση τριών εντολών.

6.1 Χειρονομίες (Gestures)

Οι χειρονομίες (Gestures) χωρίζονται σε στατικές και μη στατικές. Στην εφαρμογή αυτή χρησιμοποίησα μη στατικές. Όπως αναφέραμε, η εφαρμογή αυτή αναγνωρίζει τέσσερα Gestures τα οποία είναι: Swing Right, Swing Left, Swing Up, Swing Down. Η Swing χειρονομία είναι ουσιαστικά μία, αυτό που αλλάζει είναι η κατεύθυνση. Επειδή λοιπόν στο συγκεκριμένο παράδειγμα έχω μόνο ένα gesture δε χρειάστηκε η υλοποίηση του gesture XML.

Ανά 10 frames παίρνω τη θέση του point of reference (π.χ. το joint του δεξιού καρπού του χρήστη) το αποθηκεύω σε μια μεταβλητή Vector3D. Αρχικά ελέγχουμε την απόσταση μεταξύ των δύο σημείων χρησιμοποιώντας την μέθοδο Distance3D της βιβλιοθήκης `System.Windows.Media.Media3D`.

```
……….
if (accCounter > 10)
{
  dist = Distance3D(curPointPos.X, curPointPos.Y, curPointPos.Z,
                    prevPointPos.X, prevPointPos.Y, prevPointPos.Z);
  Vector3D cur = prevPointPos - curPointPos;
……….
```

Κώδικας 1 - Συνθήκη μέτρησης των frames και πράξεις για την απόσταση.

Έπειτα αφαιρούμε το ένα σημείο από το άλλο και χρησιμοποιώντας την απόλυτη τιμή των αποτελεσμάτων (|x|,|y|,|z|) βρίσκουμε την κατεύθυνση της κίνησης (βλ. Κώδικας 1). Ελέγχοντας την

απόσταση και την κατεύθυνση της κίνησης αναγνωρίζουμε το gesture και αποστέλλουμε την κατάλληλη λέξη στον Sentence Compiler.

```
……….
if (dist > .5 && Math.Abs(cur.X) > Math.Abs(cur.Y) && Math.Abs(cur.X) > Math.Abs(cur.Z))
{
  if(cur.X < 0)
    rsltGesture = "SwingLeft";
  else
    rsltGesture = "SwingRight";

  accCounter = 0;
  prevRightHand = Kinect_Device.skelMan.getScldSkeleton().ElementAt(11).Value;
  gestureCooldown();
  return rsltGesture;
}
else if (dist > .5 && Math.Abs(cur.Y) > Math.Abs(cur.X) && Math.Abs(cur.Y) > Math.Abs(cur.Z))
{
  if (cur.Y > 0)
    rsltGesture = "SwingUp";
  else
    rsltGesture = "SwingDown";

  accCounter = 0;
  prevRightHand = Kinect_Device.skelMan.getScldSkeleton().ElementAt(11).Value;
  gestureCooldown();
  return rsltGesture;
}
……..
```

Κώδικας 2- Αναγνώριση του Swing Gesture και της κατεύθυνσης του.

Επειδή η φυσική κίνηση του χρήστη μετά από κάποιο Swing είναι να επιστρέψει το χέρι του στην αρχική θέση, έτσι το σύστημα εντόπιζε για παράδειγμα το Swing Right (βλ. Εικόνα 27Εικόνα 27 – Η Swing Right κίνηση του χρήστη.), ο χρήστης επέστρεφε στην αρχική του θέση και το σύστημα εντόπιζε και το Swing Right. Για την αντιμετώπιση αυτού του προβλήματος προσέθεσα ένα δευτερόλεπτο sleep time στο Gesture Recognition το οποίο επιτρέπει στο χρήστη να επιστρέψει στην αρχική του θέση χωρίς να το σύστημα να επηρεαστεί. Μόλις λοιπόν το σύστημα αναγνωρίσει ένα Gesture καλεί την gestureCooldown() η οποία δίνει χρόνο στο χρήστη να φέρει το χέρι του στην αρχική του θέση, χωρίς να προκαλέσει πρόβλημα στην αλληλεπίδρασή του με το σύστημα (βλ. Κώδικας 2).

Εικόνα 27 – Η Swing Right κίνηση του χρήστη.

6.2 Φωνητικές εντολές (Vocal Commands)

Για τον εντοπισμό των φωνητικών εντολών του χρήστη, χρησιμοποίησα το Microsoft Speech Recognition SDK. Στον παρακάτω κώδικα (βλ. Κώδικας 3) βλέπουμε τις απαραίτητες εντολές για τη δημιουργία νέας γραμματικής. Επίσης, ορίζουμε και τις μεθόδους οι οποίες θα αναλάβουν να χειριστούν τα εισερχόμενα events.

```
……….
var grammar = new Choices();
            grammar.Add("page");
            grammar.Add("next");
            grammar.Add("previous");
            grammar.Add("open");
            grammar.Add("close");
            grammar.Add("move");
            grammar.Add("this");
            grammar.Add("there");

            var gb = new GrammarBuilder { Culture = ri.Culture };
            gb.Append(grammar);

            var g = new Grammar(gb);

            sre.LoadGrammar(g);
            sre.SpeechRecognized += this.SreSpeechRecognized;
            sre.SpeechHypothesized += this.SreSpeechHypothesized;
            sre.SpeechRecognitionRejected +=
this.SreSpeechRecognitionRejected;
……….
```

Κώδικας 3 - Θέσπιση των λέξεων της γραμματικής μας και ορισμός των Event Handlers

Κατά την αναγνώριση μιας λέξης έχουμε, από μεθόδους του SDK, μια confidence παράμετρο. Η παράμετρος αυτή μας βοηθά να ελέγξουμε το βαθμό ελαστικότητας που θα έχει το Speech Recognition μας στην σωστή προφορά της λέξης. Εάν δεχόμαστε λέξεις με χαμηλό confidence θα δημιουργηθεί ένα χάος στο σύστημά μας, ακόμα περισσότερο αν υπάρχουν λέξεις ή φράσεις που διαφέρουν πολύ λίγο μεταξύ τους. Όταν αναγνωριστεί μια εντολή από τη λίστα των Voice Commands αποθηκεύεται στη μεταβλητή recostring. Όταν η recostring παίρνει μια τιμή, ενεργοποιείται ένα event (βλ. Κώδικας 4) το οποίο δέχεται ο Sentence Compiler, παίρνοντας την λέξη η οποία εντοπίστηκε, αρχίζει την αναζήτηση μέσα στο Grammar XML.

```
private void SreSpeechRecognized(object sender, SpeechRecognizedEventArgs
e)
        {
            recostring = e.Result.Text;
            NotifyPropertyChanged("recostring");
        }
```

Κώδικας 4 - Αποθήκευση της λέξης που βρέθηκε στην recostring και ενεργοποίηση του event.

6.3 Συνδυασμός εισόδων - Οι εντολές This και There

Το “this” και το “there” είναι δύο ιδιόμορφες εντολές σε ένα σύστημα το οποίο δουλεύει με Graphical User Interface. Ο χρήστης μπορεί να δείχνει σε ένα παράθυρό, σε ένα αντικείμενο, σε μια εφαρμογή.

Για την υλοποίηση των δύο εντολών, ακολούθησα δύο προσεγγίσεις από τις οποίες επέλεξα την πιο βολική για το χρήστη.

6.3.1 Α' Προσέγγιση – Push Gesture

Η πρώτη προσέγγιση για το πώς θα δείχνει ο χρήστης ένα αντικείμενο στην οθόνη ήταν να προσθέσουμε ένα gesture "Push", το Push θα μετατρεπόταν σε this ή there ανάλογα με το context της πρότασης. Θέλαμε οι προτάσεις μας να διατυπώνονται και να αναγνωρίζονται με οποιαδήποτε σειρά διατυπωθούν, δηλαδή θέλαμε να παράγεται η ίδια πρόταση αν ο χρήστης δώσει σαν είσοδο "Put, This, There" και "There, Put, This" (και όλους τους υπόλοιπους δυνατούς συνδυασμούς).

Για την υλοποίηση του "Push" χρειαζόμουν ένα στατικό σημείο αναφοράς. Μια επιλογή ήταν το Kinect. Όταν η απόσταση ανάμεσα στο χέρι του χρήστη και το Kinect ήταν μικρότερη από Α, τότε ο χρήστης έκανε "Push" (βλ. Εικόνα 28). Αυτή η υλοποίηση όμως, περιόριζε την απόσταση του χρήστη από το Kinect. Ο χρήστης ήταν καθηλωμένος σε ένα συγκεκριμένο, βολικό για τον κώδικα, σημείο.

Εικόνα 28 - Push Gesture με στατικό σημείο αναφοράς το Kinect Device (αρχή των XYZ αξόνων).

Το στατικό σημείο αναφοράς δεν έπρεπε να είναι στατικό ως προς το χρήστη, αλλά στατικό ως προς το χέρι του χρήστη. Έτσι επέλεξα το κεφάλι του, ως στατικό σημείο αναφοράς (βλ. Εικόνα 29). Όποια κι αν είναι η θέση του χρήστη στο χώρο, η απόσταση του χεριού του από το κεφάλι του είναι η ίδια. Ο χρήστης έχει ελευθερία κίνησης έτσι ώστε να μπορεί είτε να πλησιάσει, είτε να απομακρυνθεί από την συσκευή (πάντα μέσα στα όρια που αναφέρει ο κατασκευαστής).

Εικόνα 29 - Push Gesture με στατικό σημείο αναφοράς το κεφάλι του χρήστη.

Ένα πρόβλημα που παρουσιάστηκε και στις δύο εκδοχές του "Push" ήταν ότι χρήστης κινώντας το χέρι του προς τα μπρός (για να κάνει το Push gesture) έχανε το συγκεκριμένο σημείο που ήθελε να επιλέξει. Αυτή η προσέγγιση λοιπόν απορρίφθηκε λόγω έλλειψης ακρίβειας.

6.3.2 Β' Προσέγγιση – Συνδυασμός εισόδων

Ένας δεύτερος τρόπος, ο οποίος φάνηκε αρκετά βολικός και εύχρηστος, ήταν ο συνδυασμός των δύο εισόδων που είχαμε στη διάθεσή μας. Μια πολύ φυσική προσέγγιση ήθελε απλά το χρήστη να δείχνει προς το σημείο που θέλει και να λέει τη λέξη This ή There. Έτσι μπορεί ο χρήστης να κρατά σταθερό το χέρι του στο σημείο επιλογής και καθώς το σύστημα αναγνωρίζει την εντολή this ή there να αποθηκεύει την θέση την οποία δείχνει ο χρήστης. Ο τρόπος αυτός είναι πιο βολικός για τον χρήστη αλλά και για τον προγραμματιστή, καθώς δεν χρειάζεται πια το context evaluation για την αποκωδικοποίηση του Push. Επίσης, απλοποιώντας έτσι τον κώδικα έχουμε κάποια βελτίωση και στο χρόνο ανταπόκρισης (response time) της εφαρμογής.

6.4 Οι XML βάσεις δεδομένων

6.4.1 Grammar XML

Το Grammar XML απαρτίζεται από ολοκληρωμένες προτάσεις τις οποίες δέχεται το σύστημά μας. Έχοντας τις προτάσεις, ο Sentence Compiler ταυτοποιεί ή απορρίπτει τις εν δυνάμει προτάσεις που συντάσσει με τις λέξεις που δέχεται.

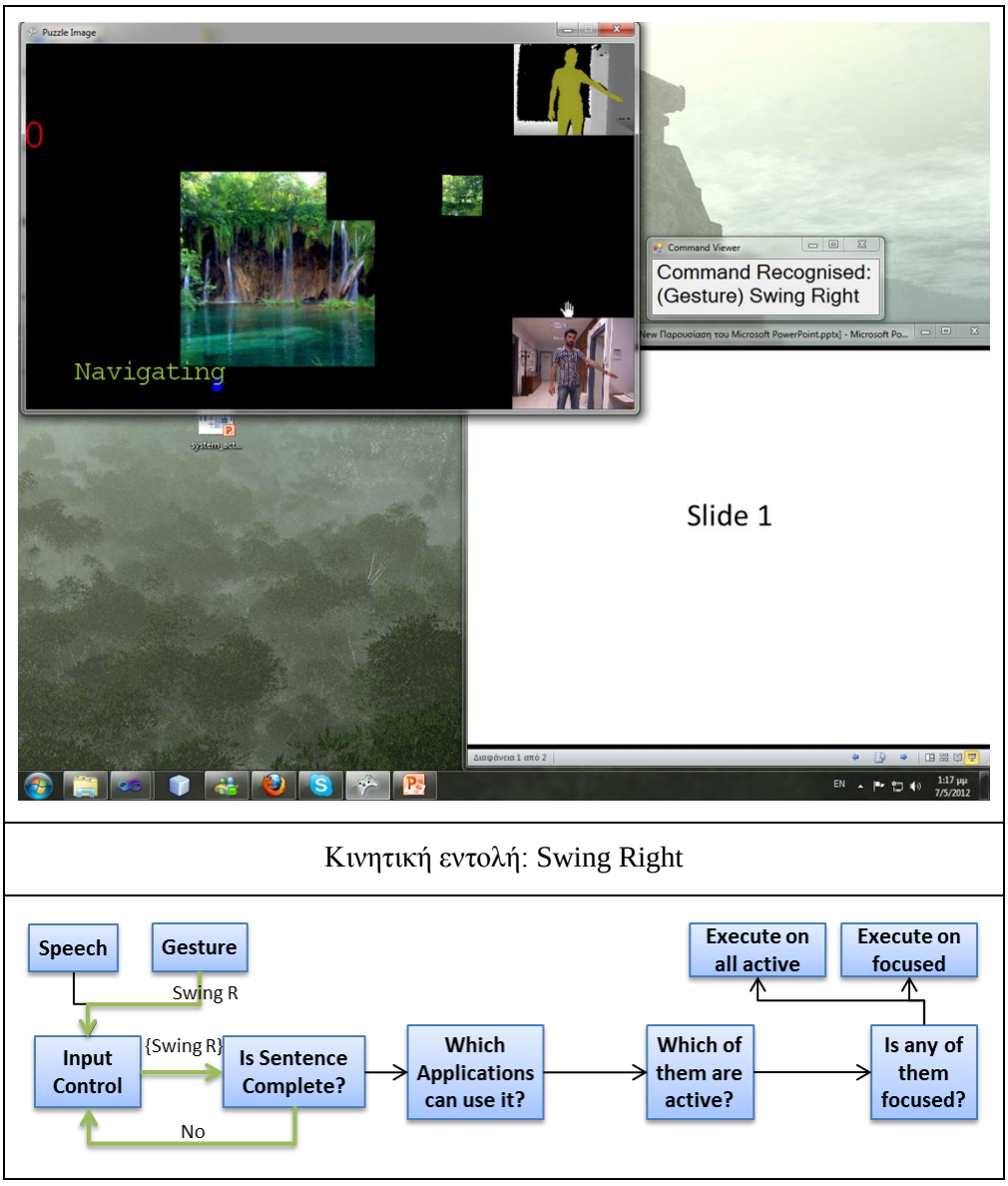
```
<?xml version="1.0" encoding="utf-8" ?>
<sentences>
  <sentence>Page SwingRight</sentence>
  <sentence>Page SwingLeft</sentence>
       …
       …
  <sentence>Move This There</sentence>
</sentences>
```

6.5 Σενάριο 1 - Σύνταξη πρότασης με δύο λέξεις, επιλογή εφαρμογής για τη χρήση της.

Οι εικόνες (Εικόνα 30,Εικόνα 31) δείχνουν το πρώτο σενάριο χρήσης, ενώ τα flowcharts περιγράφουν τις διεργασίες του συστήματός μου. Στο γραφικό περιβάλλον μας βλέπουμε ότι έχουμε ανοικτή μια εφαρμογή με όνομα "Puzzle Image" και μια παρουσίαση με το PowerPoint. Επίσης μπορείτε να δείτε το Command View Window το οποίο μας δείχνει ποια εντολή ανιχνεύει το σύστημα κάθε φορά. Η ανίχνευση των εντολών μπορεί να γίνει και παράλληλα, απλά για λόγους κατανόησης διατυπώθηκαν με τέτοιο τρόπο ώστε να είναι εμφανείς στις εικόνες.

6.5.1 Πρώτη εντολή – Swing Right

Ο χρήστης κινεί το χέρι του προς τα δεξιά και το σύστημα, μέσω του Gesture Recognition, εντοπίζει την χειρονομία Swing Right και στέλνει την αντίστοιχη λέξη στο Input Control του Sentence Compiler. Δεν έχει ληφθεί άλλη λέξη κι έτσι το σύστημα προχωρά ,με την μονολεκτική εν δυνάμει πρόταση που έχει καταγράψει, στο Sentence Syntax Evaluation. Σε αυτή τη φάση ελέγχεται εάν η πρόταση που έχουμε υπάρχει, έστω ως κομμάτι μιας πρότασης, στο Grammar XML μας. Η εν δυνάμει πρότασή μας υπάρχει στο XML, αλλά δεν συμπληρώνει κάποια ολοκληρωμένη πρόταση. Έτσι κρατώντας αποθηκευμένη την εν δυνάμει πρότασή μας επιστρέφουμε στο Input Control περιμένοντας καινούρια είσοδο.



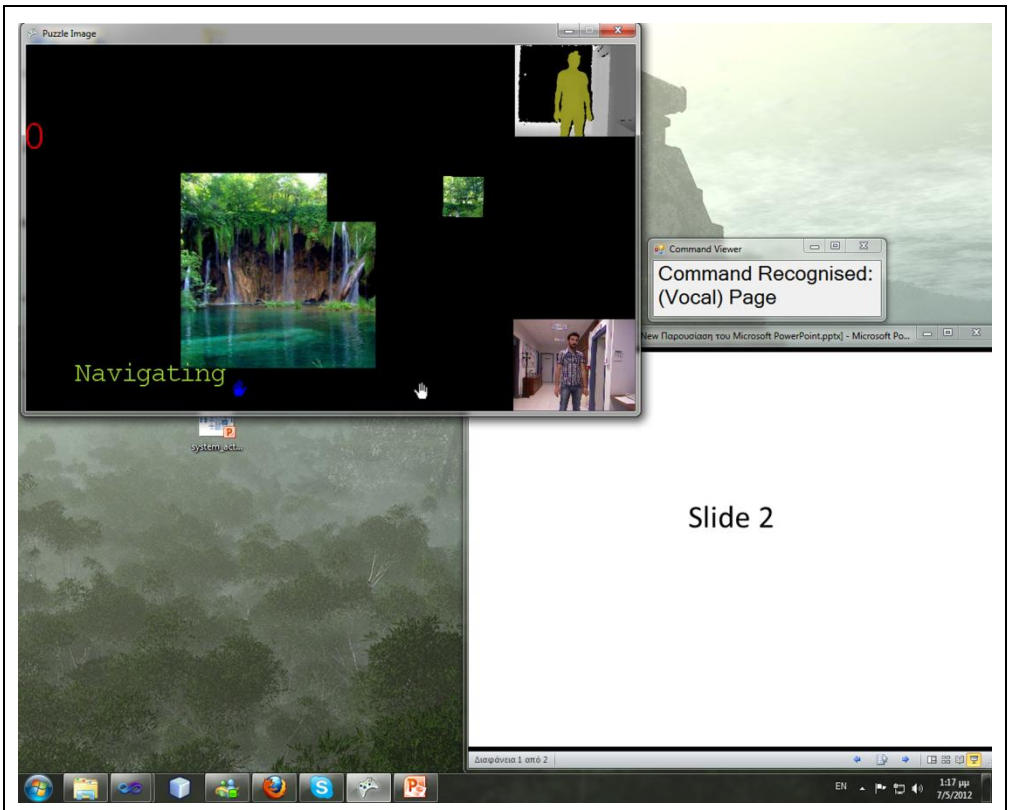
Κινητική εντολή: Swing Right

Εικόνα 30 - Εντοπισμός εντολής Swing Right

6.5.2 Δεύτερη εντολή - Page

Ο χρήστης λέει τη λέξη Page και το σύστημα, μέσω του Speech Recognition, εντοπίζει την λέξη και στέλνει την αντίστοιχη στο Input Control του Sentence Compiler. Το Input Control προσθέτει τη καινούρια λέξη στην εν δυνάμει πρόταση και τη στέλνει ξανά για Syntax Evaluation. Ελέγχεται εάν η εν δυνάμει πρόταση <"SwingRight", "Page"> υπάρχει στο Grammar XML και αναγνωρίζεται από το σύστημα ως ολοκληρωμένη πρόταση. Ο Sentence Compiler στέλνει την ολοκληρωμένη πρόταση στον Action Sentence Manipulator. Εκεί παραλαμβάνει την πρόταση ο Action Sentence Accumulator, ο οποίος είναι υπεύθυνος να φιλτράρει τις εφαρμογές οι οποίες βρίσκονται στο Application versus Action Specific Mapping Scheme έτσι ώστε ο Application-Action Binder να ενημερωθεί για το ποιες εφαρμογές στο σύστημά μας μπορούν να εκτελέσουν την ολοκληρωμένη πρόταση. Η λίστα των ικανών εφαρμογών στην περίπτωση μας είναι: {Microsoft PowerPoint, Microsoft Word}. Ο Application-Action Binder, έχοντας τη λίστα με τις ικανές εφαρμογές και τη λίστα με τις εφαρμογές που τρέχουν στον υπολογιστή μας – η οποία είναι: {Puzzle Image, Microsoft PowerPoint, Microsoft Visual Studio 2010, ….}, στέλνει ως αποτέλεσμα στον Context Information Manager τη λίστα με τις

ενεργές εφαρμογές οι οποίες είναι ικανές να ανταποκριθούν στην πρότασή μας. Στην περίπτωσή μας η λίστα αυτή έχει μονάχα ένα αντικείμενο: {Microsoft PowerPoint}. Μόλις ο Context Information Manager πάρει τη λίστα, ζητά από το λειτουργικό σύστημα την εφαρμογή η οποία είναι εστιασμένη (focused) εκείνη τη στιγμή και η απάντηση που λαμβάνει είναι: η εφαρμογή Puzzle Image (όπως βλέπουμε και στην Εικόνα 31). Η εφαρμογή αυτή, όμως, δεν υπάρχει στη λίστα, οπότε το σύστημά μας εκτελεί την πρόταση σε όλες τις εφαρμογές οι οποίες βρίσκονται στη λίστα. Στην περίπτωσή μας στο PowerPoint. Κι έτσι πηγαίνουμε στην δεύτερη διαφάνεια της παρουσίασης.



Εικόνα 31 - Εντοπισμός εντολής Page

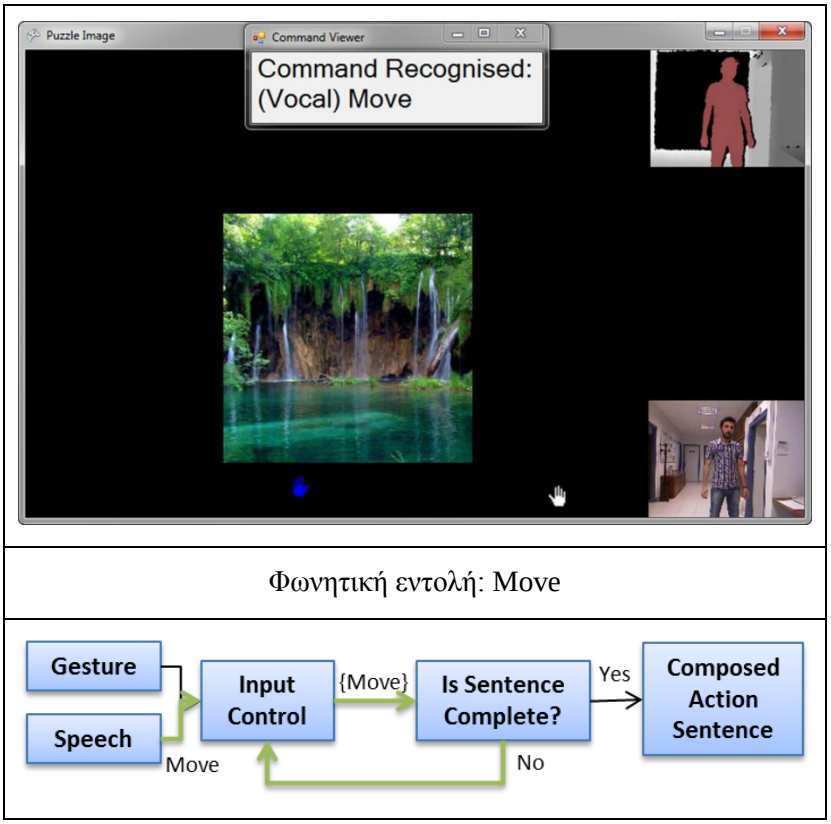
6.6 Σενάριο 2 – Σύνταξη πρότασης τριών λέξεων.

Οι εικόνες (Εικόνα 32,Εικόνα 33,Εικόνα 34) δείχνουν το δεύτερο σενάριο χρήσης του συστήματός μου. Τα flowcharts περιγράφουν τις διεργασίες του συστήματός μου μέχρι και την ολοκλήρωση της πρότασης. Στο γραφικό περιβάλλον μας βλέπουμε ότι έχουμε ανοικτή μια μόνο εφαρμογή με όνομα "Puzzle Image". Επίσης υπάρχει κι εδώ το Command View Window το οποίο μας δείχνει ποια

εντολή ανιχνεύει το σύστημα κάθε φορά. Η ανίχνευση των εντολών μπορεί να γίνει και παράλληλα, απλά για λόγους κατανόησης διατυπώθηκαν με τέτοιο τρόπο ώστε να είναι εμφανείς στις εικόνες.

6.6.1 Πρώτη εντολή – Move

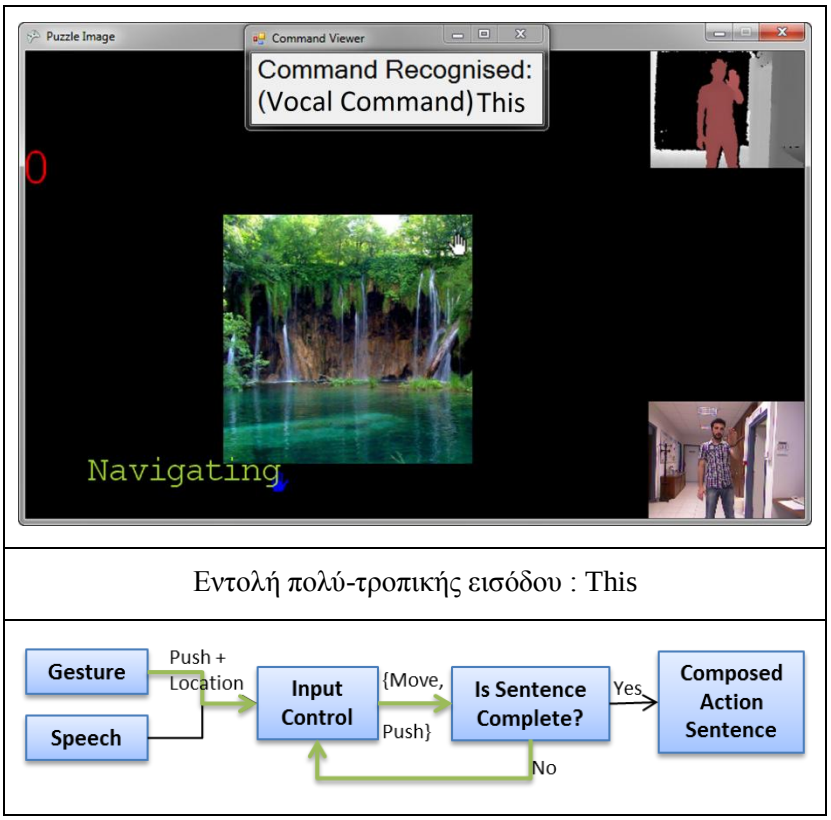
Ο χρήστης λέει τη λέξη Move και το σύστημα, μέσω του Speech Recognition, εντοπίζει την λέξη και στέλνει την αντίστοιχη στο Input Control του Sentence Compiler. Δεν έχει ληφθεί άλλη λέξη κι έτσι το σύστημα προχωρά ,με την μονολεκτική εν δυνάμει πρόταση που έχει καταγράψει, στο Sentence Syntax Evaluation. Σε αυτή τη φάση ελέγχεται εάν η πρόταση που έχουμε υπάρχει, έστω ως κομμάτι μιας πρότασης, στο Grammar XML μας. Η εν δυνάμει πρότασή μας υπάρχει στο XML, αλλά δεν συμπληρώνει κάποια ολοκληρωμένη πρόταση. Έτσι κρατώντας αποθηκευμένη την εν δυνάμει πρότασή μας επιστρέφουμε στο Input Control περιμένοντας καινούρια είσοδο.



Εικόνα 32 - Εντοπισμός εντολής Move

6.6.2 Δεύτερη εντολή – This

Ο χρήστης λέει τη λέξη This και το σύστημα, μέσω του Speech Recognition, εντοπίζει την λέξη και στέλνει την αντίστοιχη στο Input Control του Sentence Compiler, ενώ παράλληλα αποθηκεύει την θέση του χεριού του χρήστη (βλ. 6.3.2). Το Input Control προσθέτει τη καινούρια λέξη στην εν δυνάμει πρόταση και τη στέλνει ξανά για Syntax Evaluation. Ελέγχεται εάν η εν δυνάμει πρόταση <"Move", "This"> υπάρχει στο Grammar XML. Η εν δυνάμει πρότασή μας υπάρχει στο XML, αλλά δεν συμπληρώνει κάποια ολοκληρωμένη πρόταση, οπότε αποθηκεύεται και επιστρέφουμε στο Input Control περιμένοντας για καινούρια είσοδο.

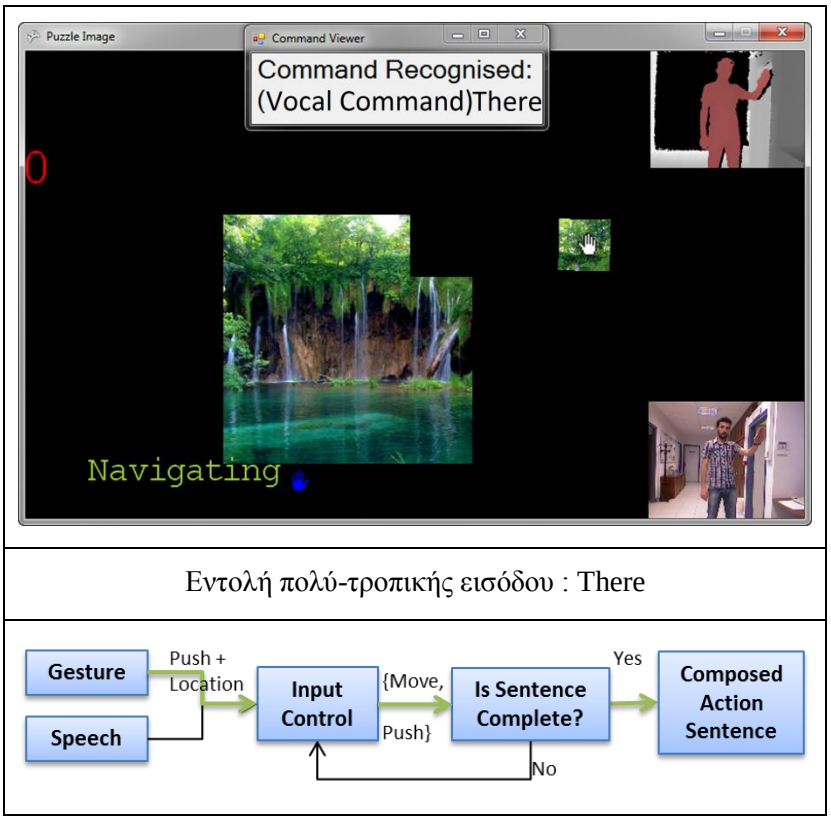


Εντολή πολύ-τροπικής εισόδου : This

Εικόνα 33 - Εντοπισμός εντολής This

6.6.3 Τρίτη εντολή – There

Ο χρήστης λέει τη λέξη This και το σύστημα, μέσω του Speech Recognition, εντοπίζει την λέξη και στέλνει την αντίστοιχη στο Input Control του Sentence Compiler, ενώ παράλληλα αποθηκεύει την θέση του χεριού του χρήστη (βλ. 6.3.2). Το Input Control προσθέτει τη καινούρια λέξη στην εν δυνάμει πρόταση και τη στέλνει ξανά για Syntax Evaluation. Ελέγχεται εάν η εν δυνάμει πρόταση <"Move", "This", "There">υπάρχει στο Grammar XML και αναγνωρίζεται από το σύστημα ως ολοκληρωμένη πρόταση. Ο Sentence Compiler στέλνει την ολοκληρωμένη πρόταση στον Action Sentence Manipulator. Εκεί παραλαμβάνει την πρόταση ο Action Sentence Accumulator, ο οποίος είναι υπεύθυνος να φιλτράρει τις εφαρμογές οι οποίες βρίσκονται στο Application versus Action Specific Mapping Scheme έτσι ώστε ο Application-Action Binder να ενημερωθεί για το ποιες εφαρμογές στο σύστημά μας μπορούν να εκτελέσουν την ολοκληρωμένη πρόταση. Η λίστα των ικανών εφαρμογών στην περίπτωση μας είναι: {Puzzle Image}. Ο Application-Action Binder, έχοντας τη λίστα με τις ικανές εφαρμογές και τη λίστα με τις εφαρμογές που τρέχουν στον υπολογιστή μας – η οποία είναι: {Puzzle Image}, στέλνει ως αποτέλεσμα στον Context Information Manager τη λίστα με τις ενεργές εφαρμογές οι οποίες είναι ικανές να ανταποκριθούν στην πρότασή μας. Στην περίπτωσή μας η λίστα αυτή έχει μονάχα ένα αντικείμενο: {Puzzle Image}. Μόλις ο Context Information Manager πάρει τη λίστα, ζητά από το λειτουργικό σύστημα την εφαρμογή η οποία είναι εστιασμένη (focused) εκείνη τη στιγμή και η απάντηση που λαμβάνει είναι: η εφαρμογή Puzzle Image (όπως βλέπουμε και στην Εικόνα 34). Η εφαρμογή αυτή είναι εστιασμένη κι έτσι το κομμάτι της εικόνα μετακινείται στην θέση την οποία διάλεξε ο χρήστης.



Εντολή πολύ-τροπικής εισόδου : There

Εικόνα 34 - Εντοπισμός εντολής There

7 Συμπεράσματα, Μελλοντική Εργασία και Επεκτάσεις

Σε αυτή την εργασία παρουσιάσαμε μια πολύ-τροπική φυσική διεπαφή χρήστη, η οποία χρησιμοποιεί το Microsoft Kinect ως πολύ-τροπική συσκευή εισόδου.

Περιγράψαμε την τεχνολογία την οποία χρησιμοποιήσαμε και είδαμε τις πρόσφατες έρευνες στις οποίες χρησιμοποιήθηκε. Επίσης εξηγήσαμε την έννοια της πολύ-τροπικής εισόδου και αναφέραμε πρόσφατα παραδείγματα ερευνών με διάφορους συνδυασμούς εισόδων.

Παρουσιάσαμε την αρχιτεκτονική του συστήματος MIDAS το οποίο χρησιμοποιεί τον ήχο, την εικόνα και το χάρτη βάθους ως διαφορετικές εισόδους. Η αρχιτεκτονική του σχεδιάστηκε με τέτοιο τρόπο ώστε να μπορούμε μελλοντικά να χρησιμοποιήσουμε κι άλλες εισόδους όπως συσκευές μέτρησης πίεσης, έντασης κ.α. Για την διαχείριση των εφαρμογών δημιουργήθηκε ένας general container ο οποίος τρέχει στο παρασκήνιο και λειτουργεί ως ενδιάμεσος μεταξύ εφαρμογών και συσκευών πολύ-τροπικής εισόδου.

Δείξαμε δύο αντιπροσωπευτικά σενάρια λειτουργίας του συστήματος με δύο διαφορετικές εφαρμογές, οι οποίες ανταποκρίνονταν σε πολύ-τροπικές εντολές: μια εφαρμογή πάζλ, και μια παρουσίαση του PowerPoint.

Όσον αφορά την μελλοντική εργασία, μερικά από τα θέματα που προκύπτουν είναι η χρήση πιο πολύπλοκων αλγορίθμων αναγνώρισης, η χρήση περισσότερων εισόδων όπως πίεση, αναγνώριση προσώπου κλπ. και η διεύρυνση του φάσματος των εφαρμογών οι οποίες μπορούν να ανταποκριθούν σε πολύ-τροπικές εισόδους.

Βιβλιογραφία

[1] Ευρεσιτεχνία αισθητήρα βάθους, ZALEVSKY, Z. et al., http://patentscope.wipo.int/search/en/detail.jsf?docId=WO2007043036&recNum=1&maxRec=&office=&prevFilter=&sortOption=&queryString=&tab=PCT+Biblio

[2] PrimeSense Ltd, «The PrimeSensor(TM) Reference Design 1.08.»

[3] Z. S. A. M. A. e. a. Zalevsky, «Method and System for Object Reconstruction». USA Ευρεσιτεχνία 991,994, 14 Μάρτιος 2006

[4] OPENKINECT *OpenKinect Main Page*. http://openkinect.org/ (Last Accessed: January 2012)

[5] OPENNI *OpenNI*. http://openni.org/ (Last Accessed: January 2012)

[6] LABORATORIES, C. *About: CL NUI Platform*. Code Laboratories, http://codelaboratories.com/kb/nui (Last Accessed:January 2012)

[7] MICROSOFT KINECT SDK, http://www.microsoft.com/en-us/kinectforwindows/

[8] EVOLUCE SDK, http://www.evoluce.com/en/software/sdk-for-kinect.php

[9] Yale Song, et al., Continuous body and hand gesture recognition for natural human-computer interaction, ACM, Transactions on Interactive Intelligent Systems(TiiS),vol 2,issue 1, March 2012

[10] Donald A. Norman, Why interfaces don't work. In: *The Art of Human-Computer Interface Design*, B. Laurel (Ed.), Addison-Wesley, 1990,209-219.

[11] Emgu CV, OpenCV wrapper http://www.emgu.com/wiki/index.php/Main_Page

[12] Suma, E.A.; Krum, D.M.; Bolas, M.; , "Sharing space in mixed and virtual reality environments using a low-cost depth sensor," VR Innovation (ISVRI), 2011 IEEE International Symposium on , vol., no., pp.349-350, 19-20 March 2011

[13] Carrino, Stefano; Mugellini, Elena; Khaled, Omar Abou; Ingold, Rolf; , "Gesture-based hybrid approach for HCI in ambient intelligent environments," Fuzzy Systems (FUZZ), 2011 IEEE International Conference on , vol., no., pp.86-93, 27-30 June 2011

[14] Stowers, J.; Hayes, M.; Bainbridge-Smith, A.; , "Altitude control of a quadrotor helicopter using depth map from Microsoft Kinect sensor," *Mechatronics (ICM), 2011 IEEE International Conference on* , vol., no., pp.358-362, 13-15 April 2011

[15] Chang,Y.-J,etal.AKinect-based system for physical rehabilitation: A pilot study for young adults with motor disabilities. Research in Developmental Disabilities (2011)

[16] Gallo, L.; Placitelli, A.P.; Ciampi, M.; , "Controller-free exploration of medical image data: Experiencing the Kinect," *Computer-Based Medical Systems (CBMS), 2011 24th International Symposium on* , vol., no., pp.1-6, 27-30 June 2011

[17] Mark T. Maybury and Wolfgang Wahlster, editors. Readings in Intelligent User Interfaces. Morgan Kaufmann Publishers, 1998.

[18] Philip Cohen, David McGee, and Josh Clow. The efficiency of multimodal interaction for a map-based task. In Proceedings of the sixth conference on Applied natural language processing, pages 331–338, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[19] Sharon Oviatt. Ten myths of multimodal interaction. Communications of the ACM, 42(11):74–81, 1999.

[20] Natural Interaction Research Team of TEI of Crete https://blog.teicrete.gr/nirt/

[21] Sharon Oviatt and Philip Cohen. Perceptual user interfaces: multimodal interfaces that process what comes naturally. Commun. ACM, 43(3):45–53, 2000.

[22] C. Benoit, J.-C. Martin, C. Pelachaud, L. Schomaker, and B. Suhm. Audiovisual and multimodal speech-based systems. In R. Moore, editor, Hand-book of Multimodal and Spoken

Dialogue Systems:Resources, Terminology and Product Evaluation, pages 102–203. Kluwer Academic Publishers, Boston, MA, 2000.

[23] G. Potamianos, C. Neti, G. Gravier, and A. Garg. Automatic recognition of audio-visual speech: Recent progress and challenges. In Proceedings of the IEEE, volume 91. IEEE, September 2003.

[24] D. G. Stork and M. E. Hennecke, editors. Speechreading by Humans and Machines. Springer Verlag, New York, 1995.

[25] Sharon L. Oviatt. Multimodal interfaces. In J. Jacko and A. Sears, editors, Handbook of Human-Computer Interaction: Fundamentals, Evolving Technologies and Emerging Applications(revised 2nd sdition), pages 413–432. Lawrence Erlbaum Associates, Mahwah, New Jersey, 2 edition, 2008.

[26] L. Miguel Encarnacao and Lawrence J. Hettinger. Perceptual multimodal interfaces. IEEE Computer Graphics and Applications, 23(5), 2003.

[27] J.L. Flanagan and T.S. Huang. Scanning the issue special issue on human computer multimodal interface. Proceedings of the IEEE, 91(9):1267–1271, Sept. 2003.

[28] L-P. Morency, C. Sidner, C. Lee, and T. Darrell. Contextual recognition of head gestures. In Proceedings of the Seventh International Conference on Multimodal Interfaces, page 18, New York. ACM, ACM Press.

[29] Matthew Turk and George Robertson. Perceptual user interfaces (introduction). Commun. ACM, 43(3):32–34, 2000.

[30] Suma, E.A.; Lange, B.; Rizzo, A.; Krum, D.M.; Bolas, M.; , "FAAST: The Flexible Action and Articulated Skeleton Toolkit," Virtual Reality Conference (VR), 2011 IEEE , vol., no., pp.247-248, 19-23 March 2011

[31] Bouffard, P. Quadrotor and Kinect Project, http://hybrid.eecs.berkeley.edu/~bouffard/kinect.html

Παράρτημα

1 Εγκατάσταση των APIs

1.1 Οδηγός εγκατάστασης του OpenNI/NITE

- **Βήμα 1**
  Κάνουμε απεγκατάσταση όλων των προγραμμάτων οδήγησης (Drivers) του Kinect/ Xtion. Αυτό το βήμα χρειάζεται εάν ήδη υπάρχουν Drivers των συσκευών στον υπολογιστή μας οι οποίοι είναι πολύ πιθανό να προκαλέσουν επιπλοκές στην εγκατάσταση του OpenNI
- **Βήμα 2**

  Κατεβάζουμε και εγκαθιστούμε τους Drivers της συσκευής μας.

  https://github.com/avin2/SensorKinect

  Οι Drivers για κάθε λογισμικό βρίσκονται στο φάκελο Bin του SensorKinect αρχείου που θα κατεβάσετε.

- **Βήμα 3**

  Κατεβάζουμε και εγκαθιστούμε τις νεότερες Stable ή Unstable εκδόσεις των:

  OpenNI Binaries

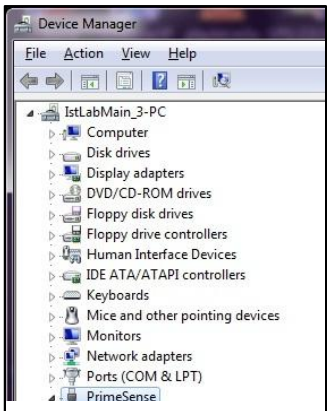
  OpenNI Compliant Middleware Binaries

  Από το παρακάτω link:

  http://75.98.78.94/Downloads/OpenNIModules.aspx

- **Βήμα 4**

  Ανοίγουμε το Device Manager για να βεβαιωθούμε ότι τα παραπάνω βήματά μας ήταν επιτυχημένα. Αν η εγκατάσταση ήταν σωστή θα δούμε κάτι παρόμοιο με την εικόνα δεξιά. (Εικονα X)



Εικόνα 35 - Η εγκατεστημένη μας συσκευή στον Device Manager

- **Βήμα 5**

  Πηγαίνουμε στην παρακάτω διεύθυνση όπου υπάρχουν τα samples του OpenNI:

  C:\Program Files\OpenNI\Samples\Bin\Release

  και στην παρακάτω όπου υπάρχουν τα samples του NITE:

  C:\Program Files\Prime Sense\NITE\Samples\Bin\Release

- **Βήμα 6**

  Εάν τα παραπάνω λειτουργούν τότε εγκαταστήσαμε με επιτυχία το OpenNI/NITE. Μπορείτε να βρείτε τις βιβλιοθήκες στις εξής διευθύνσεις:

  OpenNI: ***C:\Program Files\OpenNI\Bin***

  NITE: ***C:\Program Files\Prime Sense\NITE\Bin***

1.2 Οδηγός εγκατάστασης του Libfreenect

Δουλεύοντας με όλα τα API του Kinect και ανταλλάσοντας εντυπώσεις και με άλλα άτομα που πειραματίστηκαν με αυτό, μέσω της ερευνητικής ομάδας Natural Interaction Research Team (NIRTeam) του ΤΕΙ Κρήτης [20], κατάλαβα ότι το Libfreenect είναι μεν το πιο δύσκολο για εγκατάσταση API, αλλά είναι πολύ ευέλικτο και συνεργάσιμο με πολλά διαφορετικά λογισμικά και γλώσσες. Παρακάτω παραθέτω τον ελληνικό οδηγό εγκατάστασης της βιβλιοθήκης για τα windows έτσι ώστε να διευκολύνω τους αρχάριους χρήστες και να αποτρέψω την τυχών απογοήτευση τους την οποία είχα καθώς προσπαθούσα άκαρπα να το εγκαταστήσω.

Στο συγκεκριμένο παράδειγμα εγκατάστασης θα χρησιμοποιήσουμε Windows 7, Microsoft Visual Studio 2010 και το Cmake για να κάνουμε build τη βιβλιοθήκη μας.

- **Βήμα 1**
  Κάνουμε απεγκατάσταση όλων των προγραμμάτων οδήγησης (Drivers) του Kinect/ Xtion. Αυτό το βήμα χρειάζεται εάν ήδη υπάρχουν Drivers των συσκευών στον υπολογιστή μας οι οποίοι είναι πολύ πιθανό να προκαλέσουν επιπλοκές στην εγκατάσταση του OpenNI
- **Βήμα 2**
  Πλοηγηθείτε στο project του Open Kinect που ακολουθεί:
  https://github.com/OpenKinect/libfreenect
  Επιλέγουμε branch (Master ή Unstable δείτε παρακάτω για επεξήγηση) και κάνουμε κλικ στο κουμπί Download. Έπειτα αποσυμπιέζουμε σε μια βολική θέση στον υπολογιστή μας.

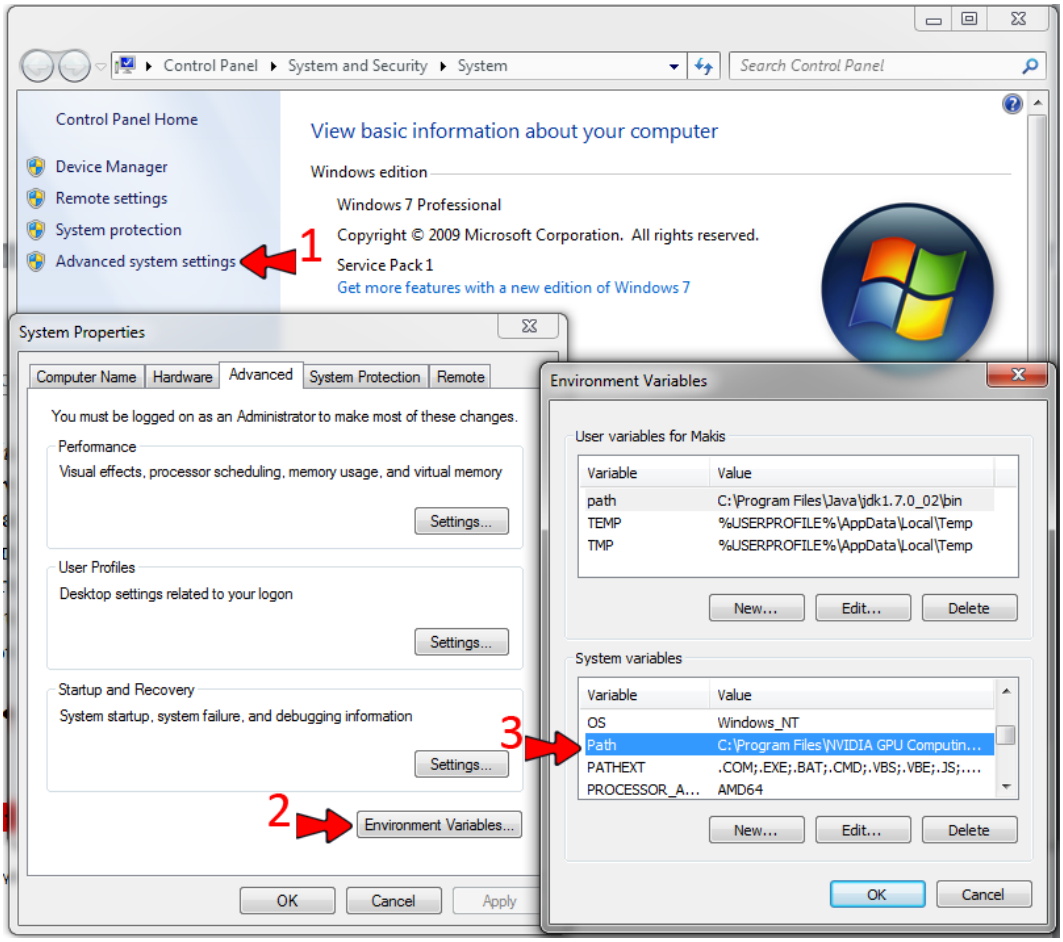
  Master και Unstable branch
  Εάν επιθυμείτε να αλλάξετε το Libfreenect, να συντάξετε wrappers, ή θέλετε να δοκιμάσετε πιο πρόσφατες δυνατότητες, τότε είναι πιθανό να θέλετε το unstable branch. Το Unstable έχει τις πιο πρόσφατες αλλαγές που δεν έχουν ελεγχθεί ακόμη. Η άλλη επιλογή είναι το Master branch το οποίο περιέχει ελεγμένες μεθόδους και δυνατότητες, οι οποίες είναι σταθερές και δεν υπάρχει κίνδυνος δημιουργίας προβλημάτων στον κώδικά σας.

- **Βήμα 3**
  Αφού κατεβάσαμε το αρχείο του πηγαίου κώδικα θα προχωρήσουμε στην εγκατάσταση των αρχείο που χρειαζόμαστε για να κάνουμε build την βιβλιοθήκη μας.

  Libusb-win32 – κατεβάστε την έκδοση της επιλογή σας, αλλά προσέξτε να είναι από 1.2.5.0 και πάνω. Μπορείτε να τη βρείτε εδώ:
  http://sourceforge.net/projects/libusb-win32/files/libusb-win32-releases/
  Θα χρησιμοποιήσουμε το Libusb αργότερα με το Cmake.

  pthreads-win32 – κατεβάστε και αποσυμπιέστε την πιο πρόσφατη έκδοση. Μπορείτε να το βρείτε εδώ: http://sourceware.org/pthreads-win32/
  Μετά την αποσυμπίεση βρείτε το /lib φάκελο μέσα στο αρχείο και αντιγράψτε τα κατάλληλα .dll. Επειδή εμείς χρησιμοποιούμε το Visual Studio 2010, θα χρειαστούμε το *phtreadVC2.dll* το οποίο θα αντιγράψουμε στο φάκελο /windows ή /windows/system32 του συστήματός μας.

Glut – κατεβάστε κι αποσυμπιέστε την πιο πρόσφατη έκδοση του Glut από το παρακάτω σύνδεσμο: http://user.xmission.com/~nate/glut.html (επιλέξτε το ***bin.zip, δε χρειάζεστε όλο τον πηγαίο κώδικα)
Έπειτα βρείτε το glut32.dll και αντιγράψτε το στο /windows/system ή κάποιο άλλο directory το οποίο περιέχεται στο PATH (PATH είναι μια global μεταβλητή του συστήματος που περιέχει διευθύνσεις από directories να χρησιμοποιούνται τα περιεχόμενά τους από το σύστημα). Μπορείτε να βρείτε το System Path, ακολουθώντας τα τρία βήματα της εικόνας X.



Εικόνα 36 - Εύρεση του System PATH

- **Βήμα 4**

Αφού τελειώσαμε με τα παραπάνω προχωρήσουμε στην εγκατάσταση του προγράμματος οδήγησης του Kinect. Ο παρακάτω τρόπος λειτουργεί σε Windows 7 αλλά και XP.

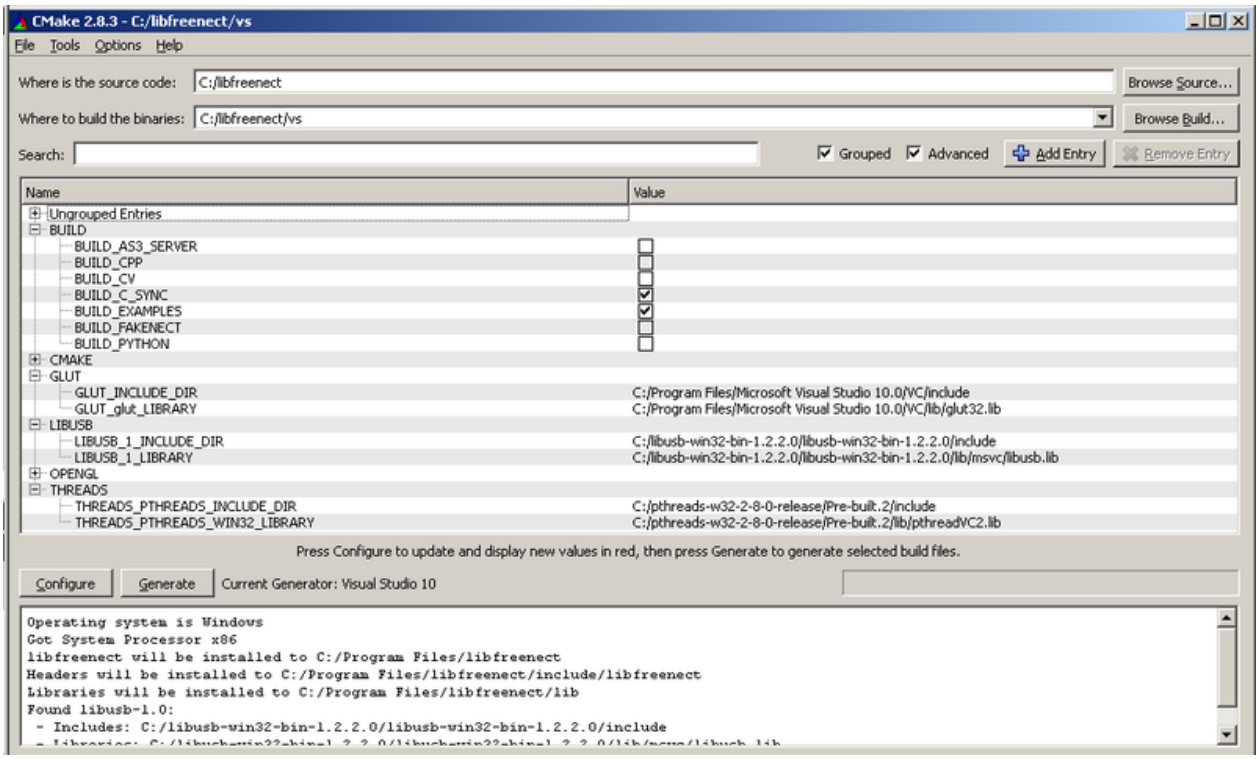
1) Συνδέστε το Kinect στον υπολογιστή. Τα Windows θα σας ειδοποιήσουν ότι βρέθηκε μια καινούρια συσκευή αλλά δεν υπάρχουν διαθέσιμα προγράμματα οδήγησης (το LED της συσκευής θα παραμείνει κλειστό). Εάν τα Windows εμφανίσουν ένα μήνυμα το οποίο θα ζητάει άδεια να ψάξει για προγράμματα οδήγησης, απλά πατήστε Άκυρο / Cancel.
2) Ανοίξτε την Διαχείριση Συσκευών (Device Manager) από τον πίνακα ελέγχου ακολουθώντας την παρακάτω προτεινόμενη διαδρομή: **Start** >> **Control Panel** >> **System and Security** >> **System** >> **Device Manager**

3) Μια συσκευή που θα λέγεται "Xbox NUI Motor" θα πρέπει να βρίσκεται εκεί (πιθανόν στο "Άλλες Συσκευές (Other devices)") με ένα κίτρινο τρίγωνο με θαυμαστικό στο πάνω μέρος του εικονιδίου. Κάντε δεξί κλικ στο εικονίδιο, επιλέξτε "Ενημέρωση Προγραμμάτων Οδήγησης" (Update Driver Software) και κάντε κλικ στο "Αναζήτηση προγραμμάτων οδήγησης στον υπολογιστή μου" (Browse my computer for driver software)
4) Επιλέξτε "Αναζήτηση" (Browse) και βρείτε το φάκελο στον οποίο υπάρχει το "Xbox_NUI_Motor.inf" (/platform/windows/inf μέσα στο Libfreenect φάκελό σας). Επιλέξτε "Επόμενο" (Next), αν τα Windows σας ειδοποιήσουν ότι ένα μη-πιστοποιημένο πρόγραμμα οδήγησης πρόκειται να εγκατασταθεί, απλά δώστε την άδεια να εγκατασταθεί.
5) Μόλις η εγκατάσταση τελειώσει το σύστημα θα μπορεί να αναγνωρίσει τη συσκευή και το LED θα αρχίσει να αναβοσβήνει με πράσινο χρώμα. Τώρα θα υπάρχουν 2 παραπάνω συσκευές στο Device Manager: το "Xbox NUI Camera" και "Xbox NUI Audio". Επαναλάβετε το 3 και το 4 και για αυτά.

- **Βήμα 5**

Αφού τελειώσουμε το Βήμα 4, προχωράμε στο build της βιβλιοθήκης μας.

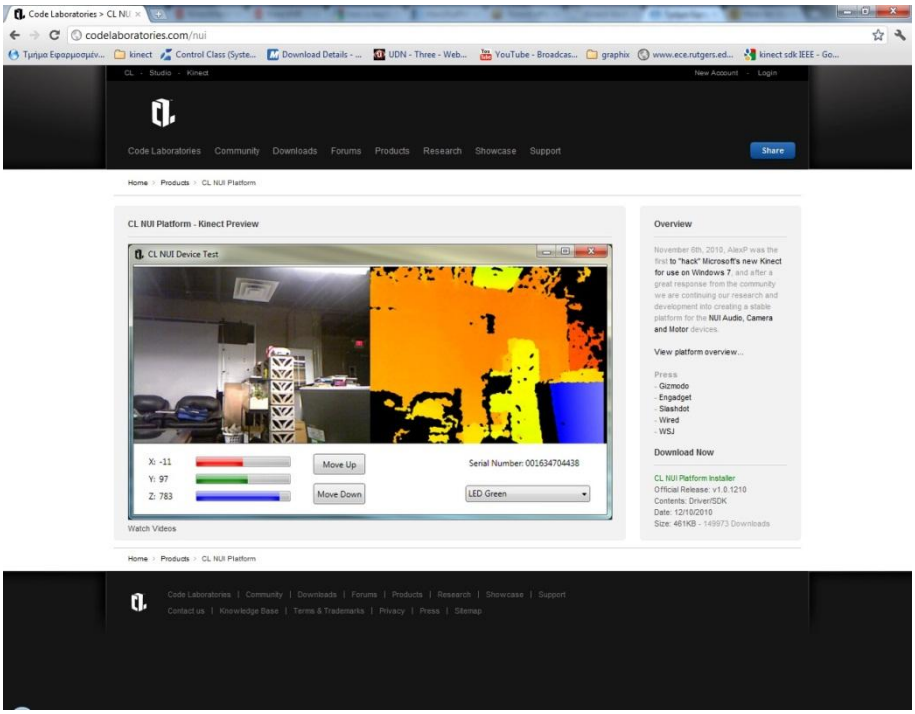
1) Κατεβάζουμε το Cmake (Cross-Platform Make) από τον παρακάτω σύνδεσμο: http://www.cmake.org/cmake/resources/software.html
(Εδώ θα χρειαστούμε και ένα C compiler, όπως προανέφερα στην εγκατάσταση αυτή χρησιμοποιούμε Visual Studio 2010)
2) Ανοίγουμε το Cmake και επιλέγουμε ως πηγή (source) το /libfreenect φάκελό μας, και ένα φάκελο στον οποίο θέλουμε να αποθηκευτεί το αποτέλεσμα του build. Προσέξτε να είναι επιλεγμένα τα checkboxes "advanced" και "grouped", έπειτα πατήστε "Configure".
3) Επιλέξτε το C compiler που θα χρησιμοποιήσετε. (Στην περίπτωσή μας Visual Studio 10)
4) Επιλέξτε μονάχα τα EXAPLES (παραδείγματα) και C_SYNC. Όπως δείχνει η εικόνα X.
5) Οι εξαρτήσεις οι οποίες έχουν πρόβλημα θα γίνουν κόκκινες. Βάλτε τις διευθύνσεις που λείπουν με βάση την παραπάνω εικόνα και πατήστε ξανά το κουμπί Configure.
    a. Οι μεταβλητές *_LIBRARY πρέπει να δείχνουν σε σε .lib αρχεία, όχι φακέλους.
    b. Οι μεταβλητές INCLUDE πρέπει να δείχνουν σε include φακέλους.
6) Όταν όλα τα προβλήματα (κόκκινα σημεία) έχουν λυθεί, τότε πατήστε το κουμπί Generate για να δημιουργήσετε της βιβλιοθήκες για τα προγράμματά σας.



Εικόνα 37- Τελική όψη ρυθμίσεων του CMake

1.3 Οδηγός εγκατάστασης του CL NUI

Για την εγκατάσταση του CL NUI δεν έχουμε παρά να πλοηγηθούμε στην σελίδα http://codelaboratories.com/nui και να κατεβάσουμε τον CL NUI Platform Installer.



Εικόνα 38 - Το site των Code Laboratories.

1.4 Οδηγός εγκατάστασης του MS Kinect SDK for Windows

Για την εγκατάσταση του Microsoft Kinect SDK θα χρειαστεί να πλοηγηθούμε στον παρακάτω σύνδεσμο: http://www.microsoft.com/en-us/kinectforwindows/develop/developer-downloads.aspx

1.5 Οδηγός εγκατάστασης του MS Kinect SDK for Windows

Για την εγκατάσταση του Evoluce SDK δεν έχουμε παρά να πλοηγηθούμε στην σελίδα http://www.evoluce.com/_win-and-i/en/software/overview/index.php?we_objectID=55 και να κατεβάσουμε το SDK.

2 Δημοσίευση του συστήματος στο TEMU 2012 (Accepted)

Multimodal Natural User Interaction for Multiple Applications: The Gesture – Voice Example

N. Vidakis, M. Syntychakis, G. Triantafyllidis and D. Akoumianakis
Applied Informatics and Multimedia
Technological Educational Institute of Crete
Heraklion, Greece
e-mail: vidakis@epp.teicrete.gr

***Abstract*—In this paper we present a natural user interface system. The device utilized for achieving natural interaction is the MS-Kinect which provides RGB, depth & audio signal. Our system is based on the theory of multimodal interaction and provides the ability to the user to interact simultaneously with different applications using vocal commands and gesture in conjunction. The difference of our system from other similar efforts is that it uses a multimodal interaction approach to manipulate multi-applications by employing a generic container, which runs at the background and serves as an intermediate between multimodal input and active applications running on a computer. We describe the architecture of the system based on multimodal interaction and its representative example using two PC applications.**

Keywords-component; Multimodal Interaction; Multi-Application Interaction; MS-Kinect

3 Introduction

Since 1963 when Ivan Sutherland has used a pointing device in Sketchpad [1] to directly manipulate visible objects on the screen, human-computer interaction has been focused on the keyboard and pointing devices such as mouse, pen etc. Although this has been successful and served well for many years, as computation becomes increasingly mobile, embedded, and ubiquitous, these interaction modalities become more constraining as an interaction model. Current research trends, suggest that natural-based interaction is the wave of the future [2], with considerable attention from both the research community (see recent survey articles by Mitra & Acharya [3] and by Weinland et al. [4]) and the industry (see MS-Kinect[5], Asus Xtion PRO LIVE[6]). Evidence can also be found in a wide range of potential application areas, such as medical devices [7], video gaming [8], robotics [9], [10], ambient intelligent environments [11], and physical rehabilitation [12] etc.

In HCI, several researchers are addressing the design of natural user interfaces allowing the machine to understand the natural language of human beings [13]. People naturally communicate through voice-speech, gestures, facial expressions and movements and they express themselves using one of those modes or a combination of them. In order to establish a Natural Human-Computer Interaction, the system has to take into consideration the above communication means not separately, but as an integral set whose purpose is to convey the interaction act. Natural interaction modalities, as defined above, have a number of clear advantages compared to the keyboard and mouse modalities. They use "equipment" (i.e. hands, face, head, voice etc.) that humans have always available, as they are part of the human body. In addition, they can be designed to facilitate acts that are natural and intuitive to users and thus minimize cognition overhead and let the user concentrate on the task itself, as it should be, not on the interaction modality [2].

Recently, new devices have been released that are capable of capturing body movement and sound, to allow a more natural interaction of users with computers in the context of a Natural User Interface. Kinect and Xtion PRO are such devices. Kinect is a new game controller technology introduced by Microsoft in November 2010. Since its launch date, it was evident that Microsoft's device is transforming not only computer gaming but also many other applications like robotics and virtual reality, thanks to its ability to track movement and voice, and even identify faces, without the need for any additional devices [14].

In this paper, we present initial findings and work in progress with regards to multi-application, multimodal interaction using the Kinect device as a two modal source. For supporting multi-applications, a generic container has been designed which runs at the background and serves as an intermediate between multimodal input and active applications - processes running on a computer. To illustrate the concept we present an example with two applications responding to multimodal commands: a simple puzzle application and the MS-PowerPoint.

The rest of the paper is structured as follows. The next section reviews the state of the art in multimodal multi-

application interaction. Then we present our system which uses both voice and gesture as input to interact with the currently active applications of a computer. Finally, we summarize to some conclusions and quote on-going and future work.

4 Multimodal – multi-application Interaction

The main advantage of multiple input modalities is increased usability: the weaknesses of one modality are countervailing by the strengths of another.

Current research on natural interaction, combined with devices like Kinect, define multimodality as: (a) every distinct human body section is considered as different modality and (b) depth and RGB signal are considered as different modalities. Limited or no consideration has been given to the multi-application interaction in conjunction with multimodal input. The next paragraphs present three different systems that use Kinect as a multimodal input device in order to offer natural interaction to their users. These systems are representing different research disciplines such as medicine, virtual reality and robotics.

Gallo's et al. developed a controller-free exploration of medical image data [7]. The project allows users to interact at a distance through hand and arm gestures, giving them the opportunity to explore medical images. With the use of OpenNI as a Kinect communication library, and Open CV [15] Computer Vision library for image processing, users have the ability to navigate, click, rotate, translate, zoom, scale and erase an image from the screen. The gestures were one to one assigned to a command and could not be changed by the users.

Suma et al. developed a middleware called "The Flexible Action and Articulated Skeleton Toolkit (FAAST)" [8] to facilitate integration of full-body control using OpenNI-compliant depth sensors (currently ASUS Xtion and Microsoft Kinect). FAAST incorporates a VRPN server for streaming the user's skeleton joints over a network, which provides a convenient interface for custom virtual reality applications and games. Although it is flexible at assigning different commands to the user's movement, it can only use one type of input.

John Stowers et al. on their project use the Kinect device built on a quad rotor, to control its altitude by using the information received from both the depth and image sensors [9]. Another similar approach is developed by Michael Van den Bergh et al. They use the Kinect device to direct a robot with human 3D hand gestures real time [10] by using Depth and RGB sensor information.

5 The System

Our system, unlike other systems, is designed to facilitate multimodal and multi-application natural interaction of users with computers. The architecture satisfies multimodality by supporting audio, depth and RGB input devices as different modals while remaining open to future additions of other input modalities such as tension, pressure, facial etc. devices. We have developed a generic container which runs at the background and serves as an intermediate between input from different modalities and information, concerning application status, from the operating system.

5.1 The System Architecture

The system architecture consists of three distinct components that collaborate together (a) to receive data from different input modalities, such as Audio, RGB and depth devices, (b)to compile commands and (c) to determine and select active applications that can accept and perform the compiled commands. Peripheral to the system architecture are the input device's APIs which provide the multimodal input and the operating system, which provides information for the currently running applications. The components of our architecture are the "Sentence Compiler", the "Action Sentences Manipulator" and the "Context Information Manager".

The "Sentence Compiler" component (see Figure 1.) of the system is responsible for collecting commands from the different input modalities and compiling action sentences. It consists of two components, namely the "Qualifier Input Control" and the Sentence Syntax Evaluation. The Qualifier Input Control "listens" to the input modalities continuously. When, for example, a gesture or vocal command is detected, an action sentence composition process is initiated, named the composition phase. The detected command is passed on to the Sentence Syntax Evaluation component for syntax check. Evaluation is done with the help of a Grammar database (in XML), that contains structured sentences. If the command doesn't fulfill the syntax of a sentence in the database then the sentence is considered uncompleted and the system returns to the composition phase waiting for a new input qualifier. When a complete sentence is composed and recognized by the system it is then sent to the Action Sentence Manipulator.

The "Action Sentence Manipulator" component (see Figure 1.) of the system consists of the "Action Sentence Accumulator" and the "Application-Action Binder". It is responsible for matching action sentences and applications. In other words, it determines which application/s can respond to an action sentence. To do so, the "Action Sentence Accumulator" component compares the sentence received from the sentence compiler, with the information stored in an Action Sentence-Application database, concerning the ability of applications to respond to sentence actions. The outcome of this comparison is passed on to the Application-Action Binder component. The Application-Action Binder component, having on one hand the information about which application/s are capable of handling the action sentence and on the other hand which application/s are up and running, informs the "Context Information Manager" which in turn passes the action sentence to the corresponding application/s.

The "Context Information Manager" component (see Figure 1.) of the system, is responsible for communicating with the operating system and provide information about the current running applications (i.e. active application/s, focused no focused, iconized minimized, etc.). Each time an application is activated it is registered into the Context Information Manager. The Context Information Manager then sends the list of active applications to the Action Sentence Manipulator

(ASM). ASM asks the Mapping Scheme which application/s can use the composed Action Sentence. ASM then semi-disambiguates the abstraction of the Action by filtering the results with the active applications' information provided. It then sends the Action and the possible applications to the Context Information Manager who in consideration of which application is the focused one, disambiguates the Abstract Action and sends the command to the specified application. If there is no application focused but there are active applications that support the action, the command is performed in all of them.

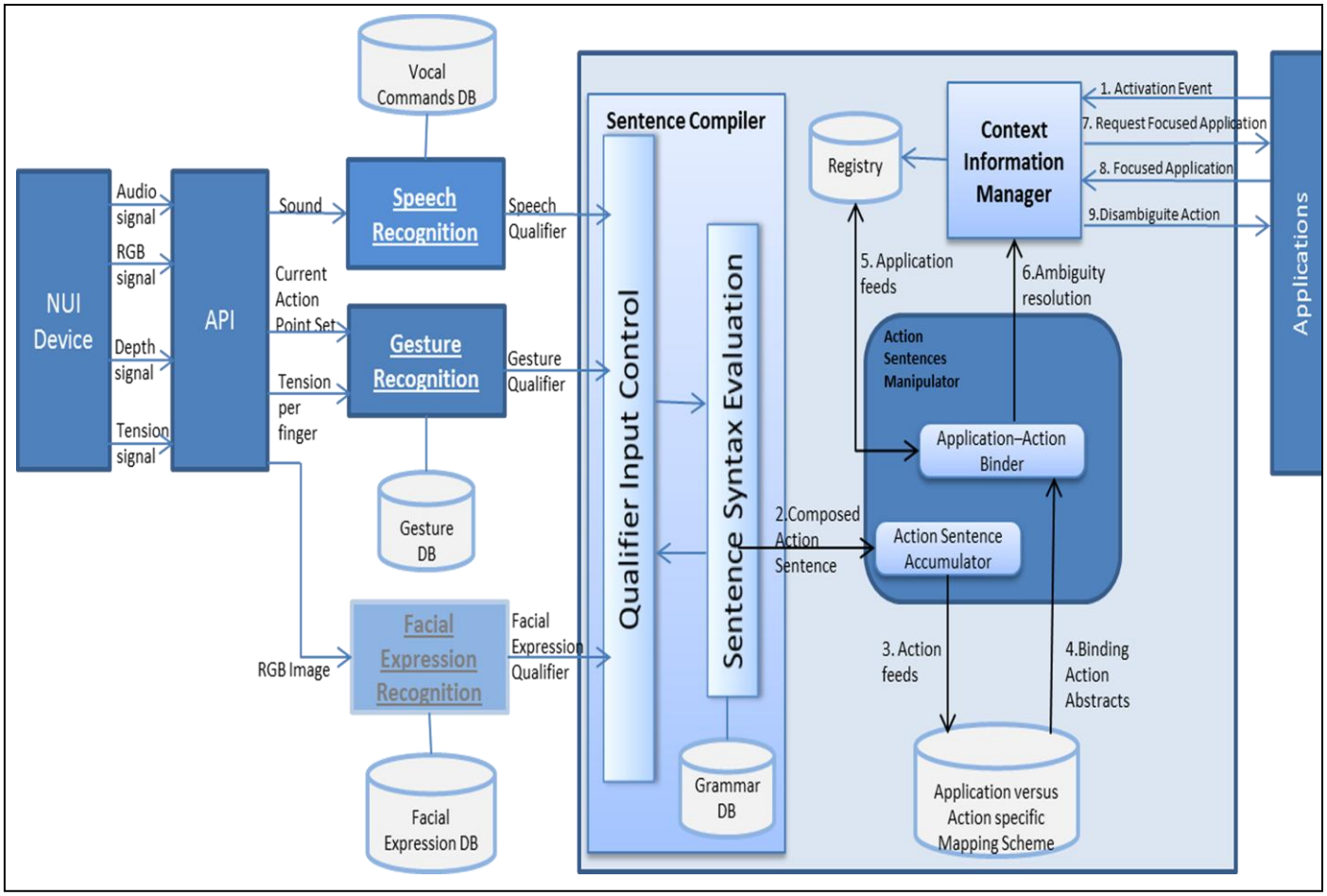


Figure 1. **The System Arcitecture**

B. The Multi Application Example

Our example application uses Microsoft Kinect as a multimodal input device that provides Audio, RGB and Depth Image. We use Microsoft's official SDK for Kinect to process the incoming information to get the user's skeleton joints as our action points and the user's voice as input sound. Action points' position is refreshed at the rate of 30 times per second, which is Kinect's frame rate for skeleton tracking.

We use skeleton tracking to get the users action points (in this example the joints) and apply simple gesture recognition to detect any gesture from our database.

Our gesture recognition in this example is limited to push, swing left, right, up and down. We check the change of action points position every 10 frames, comparing the results with the XML gesture database which contains the gesture traits such as Distance, Orientation and Action Point of Reference. Sound is processed in real-time with the use of Microsoft Speech Recognition SDK, which searches for a match within our Vocal Command XML database. The database consists of commands and corresponding speech qualifiers.

We have run two different use case scenarios of our system: The first scenario sets the operating system to have a few applications running. Some of them are minimized (as icons e.g. skype, Mozilla etc.) and two of them are open which are the ones actually respond to multimodal input. One of the open applications has the focus, namely the Puzzle application, where the PowerPoint application is just active. The second use case scenario uses one application, the Puzzle which is active and focused and a three-command sentence.

Figure 2, demonstrates the first running use case scenario of how our system works. In the left upper image we show that our system detects a gesture command (Swing Right) and sends the Gesture Qualifier to the Sentence Compiler. The Compiler checks the sentence syntax, recognize the qualifier in its library (see Input Control step of flowchart in bottom left

side of Figure 2) but needs more input to compose a complete sentence (see the NO branch of the "Is Sentence Complete" of the flowchart of Figure 2), so the qualifier input control waits for another input. Then system detects a voice command (Page), and sends a Speech Qualifier to the Sentence Compiler (see Input Control step of flowchart in bottom right side of Figure 2) Qualifier Input Control sends two words for syntax evaluation to the Sentence Syntax Evaluation component which compares the two commands with the data stored in the Grammar database and finds a match (see the YES branch of the "Is Sentence Complete" of the flowchart of Figure 2). The composed sentence is sent to Action Sentence Manipulator.

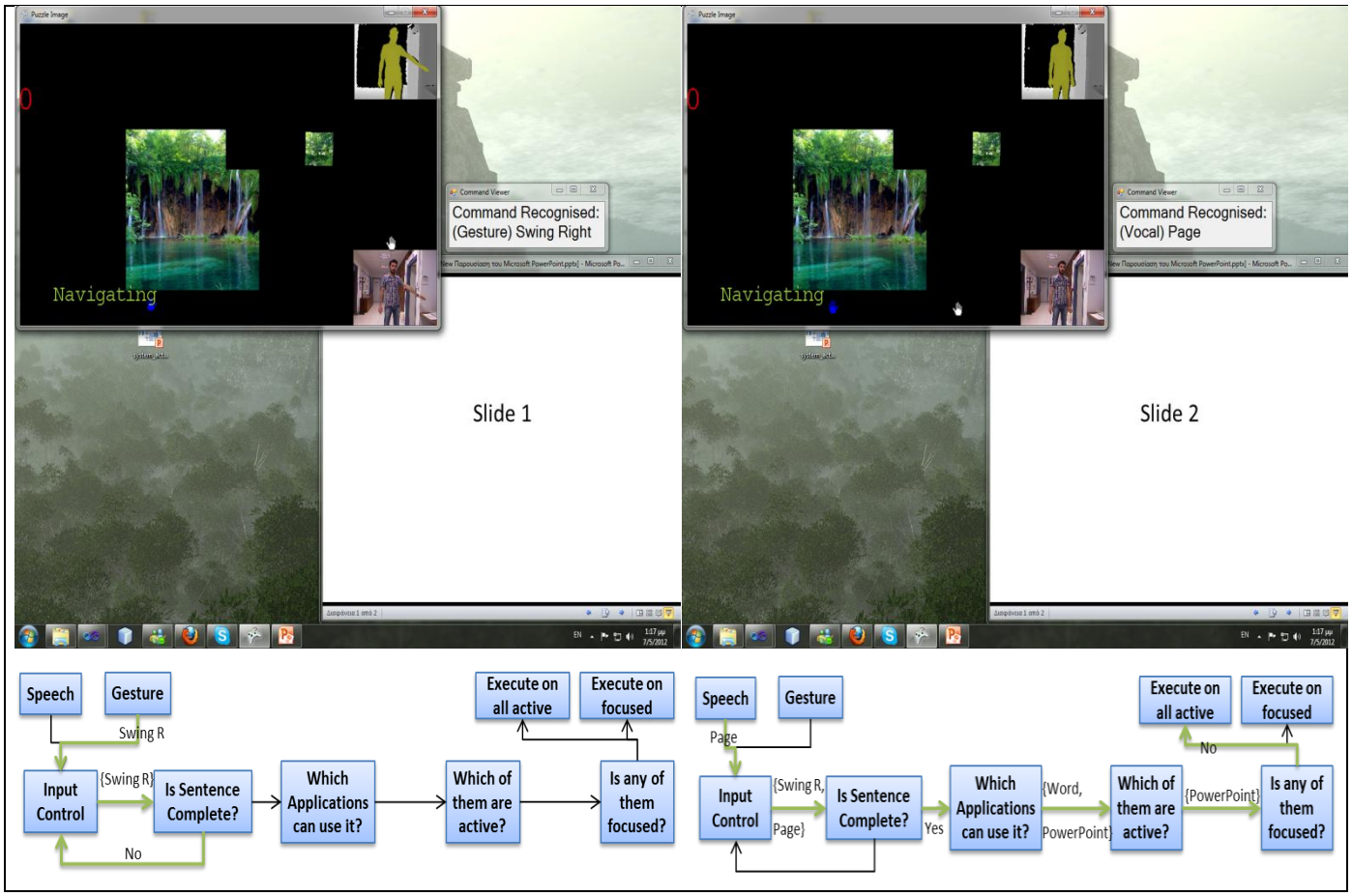


Figure 2. **Two Applications Multimodal (Gesture - Voice) Example**

The Manipulator checks which application/s can use this "Swing Right, Page" action sentence (see "Which Application can use it" of the flowchart of Figure 2). In our use case scenario, only PowerPoint can respond to such an action sentence. Then, it filters the results with those applications that are active, sending the list of applications to Context Information Manager. Context Information Manager gets the current focused application, in our case the Puzzle Image application. If the focused application can use the action sentence it disambiguates the sentence and sends the command to be executed to the application. Puzzle Image application though cannot respond to the specific command (Swing Right & Page) and thus, the action sentence is send for execution to every active application that can use it. In our use case scenario, PowerPoint is the application capable of executing such a command (Swing Right & Page). The action sentence is executed by PowerPoint and changes slide as seen at the right upper image of Figure 2.

The one application and three-command sentence scenario is demonstrated in Figure 3. In the left upper image we show that our system detects a vocal command (Move) and sends the Gesture Qualifier to the Sentence Compiler. The Compiler checks the sentence syntax, recognize the qualifier in its library (see Input Control step with the MOVE output of flowchart in bottom left side of Figure 3) but needs more input to compose a complete sentence (see the NO branch of the "Is Sentence Complete" of the flowchart of Figure 3), so the qualifier input control waits for another input. After that the system detects a gesture command (Push, which is translated into "This"), and sends a Gesture Qualifier to the Sentence Compiler (see Input Control step of flowchart in bottom middle side of Figure 3) but once more needs more input to compose a complete sentence (see the NO branch of the "Is Sentence Complete" of the bottom middle flowchart of Figure 3), so the qualifier input control waits for another input. Then the system detects a second gesture command (Push which is translated into "There"), and sends a Gesture Qualifier to the Sentence

Compiler (see Input Control step of flowchart in bottom right side of Figure 3)

Qualifier Input Control sends three words for syntax evaluation to the Sentence Syntax Evaluation component which compares the three commands with the data stored in the Grammar database and finds a match (see the YES branch of the "Is Sentence Complete" of the bottom right flowchart of Figure 3). The composed sentence is sent to the ASM. The ASM in his turn, checks which application/s can use this "Move-This-There" action sentence. It, then, filters the results with those applications that are active, sending the list of applications to Context Information Manager. The Context Information Manager receives the list and as a result the

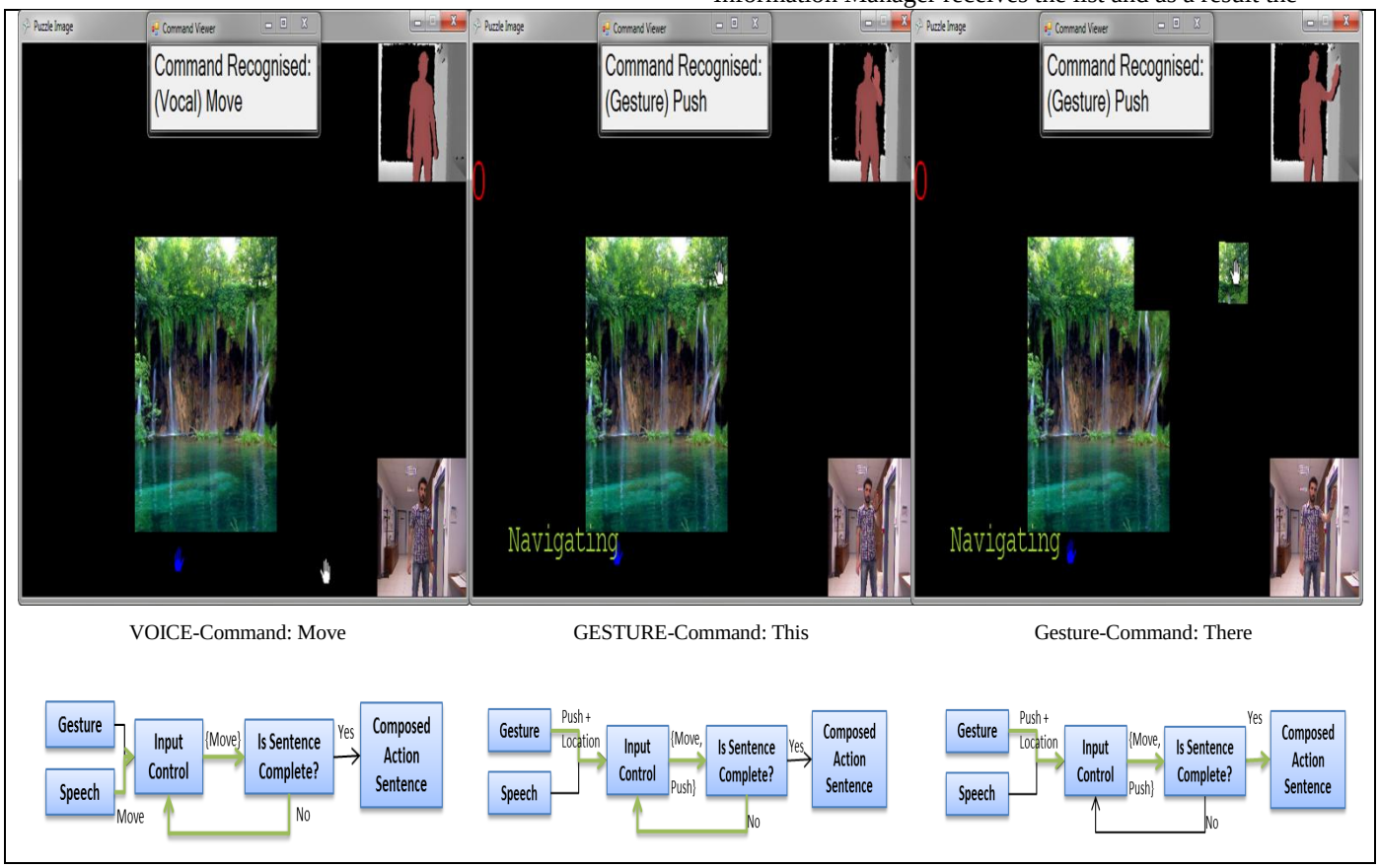


Figure 3. **One Application Multimodal (Voice - Gesture - Gesture) Example**

current focused application –in our case the Puzzle Image application which can respond to such an action sentence. If the focused application can use the action sentence it disambiguates the sentence and sends the command to be executed to the application. Since Puzzle Image application can respond to the specific command (Move-This-There), the action sentence is executed and moves the upper right piece of the puzzle to a different position, further to the right. (see right image of Figure 3)

6 CONCLUSION & Future Work

This paper has described an approach for multimodal – multi-application natural interaction of users with computers using the Kinect device as a two modal source.

We have described our system, and its architecture which satisfies multimodality by supporting audio, depth and RGB input devices as different modals. Its architecture is designed in such a way that remains open to future additions of other input modalities such as tension, pressure, facial etc. devices.

For the multi-applications perspective of our system, a generic container has been developed which runs at the background and serves as an intermediate between multimodal input devices and information from the operating system concerning running applications status.

To illustrate the concept we presented two examples with two different applications responding to multimodal commands: a simple puzzle application and a PowerPoint file. To test the idea we have designed and run two use case scenarios: one with two applications and another with a three-command sentence.

However, on-going work exploits matters concerning (a) grammar databases to enrich the ability of the system to respond to complex sentences, (b) other input modalities such as pressure and facial recognition and (c) further application types that will respond to multimodal input.

6.1.1.1.1 References

[1] Sutherland, I.E. "SketchPad: A Man-Machine Graphical Communication System," in AFIPS Spring Joint Computer Conference. 1963. 23. pp. 329-346.

[2] Yale Song, et al., Continuous body and hand gesture recognition for natural human-computer interaction, ACM, Transactions on Interactive Intelligent Systems(TiiS),vol 2,issue 1, March 2012

[3] Mitra, S. and Acharya, T. 2007. Gesture recognition: A survey. IEEE Trans. Syst. Man, Cybernet. C: Appl. Rev. 37, 3, 311–324.

[4] Weinland, D., Ronfard, R., and Boyer, E. 2011. A survey of vision-based methods for action representation, segmentation and recognition. Comput. Vis. Image Understand. 115, 2, 224–241.

[5] Microsoft Kinect, http://www.microsoft.com/en-us/kinectforwindows/

[6] Asus Xtion PRO LIVE, http://www.asus.com/Multimedia/Motion_Sensor/Xtion_PRO_LIVE/

[7] Gallo, L.; Placitelli, A.P.; Ciampi, M.; , "Controller-free exploration of medical image data: Experiencing the Kinect," Computer-Based Medical Systems (CBMS), 2011 24th International Symposium on , vol., no., pp.1-6, 27-30 June 2011

[8] Suma, E.A.; Lange, B.; Rizzo, A.; Krum, D.M.; Bolas, M.; , "FAAST: The Flexible Action and Articulated Skeleton Toolkit," Virtual Reality Conference (VR), 2011 IEEE , vol., no., pp.247-248, 19-23 March 2011

[9] Stowers, J.; Hayes, M.; Bainbridge-Smith, A.; , "Altitude control of a quadrotor helicopter using depth map from Microsoft Kinect sensor," Mechatronics (ICM), 2011 IEEE International Conference on , vol., no., pp.358-362, 13-15 April 2011

[10] Van den Bergh, M.; Carton, D.; De Nijs, R.; Mitsou, N.; Landsiedel, C.; Kuehnlenz, K.; Wollherr, D.; Van Gool, L.; Buss, M.; , "Real-time 3D hand gesture interaction with a robot for understanding directions from humans," RO-MAN, 2011 IEEE , vol., no., pp.357-362, July 31 2011-Aug. 3 2011

[11] Carrino, Stefano; Mugellini, Elena; Khaled, Omar Abou; Ingold, Rolf; , "Gesture-based hybrid approach for HCI in ambient intelligent environmments," Fuzzy Systems (FUZZ), 2011 IEEE International Conference on , vol., no., pp.86-93, 27-30 June 2011

[12] Chang,Y.-J,etal.AKinect-based system for physical rehabilitation: A pilot study for young adults with motor disabilities. Research in Developmental Disabilities (2011)

[13] OpenNi, http://www.openni.org/.

[14] J. Giles. Inside the race to hack the Kinect. The New Scientist, 208(2789):22–23, 2010.

[15] OpenCV Computer Vision, http://opencv.willowgarage.com/wiki/