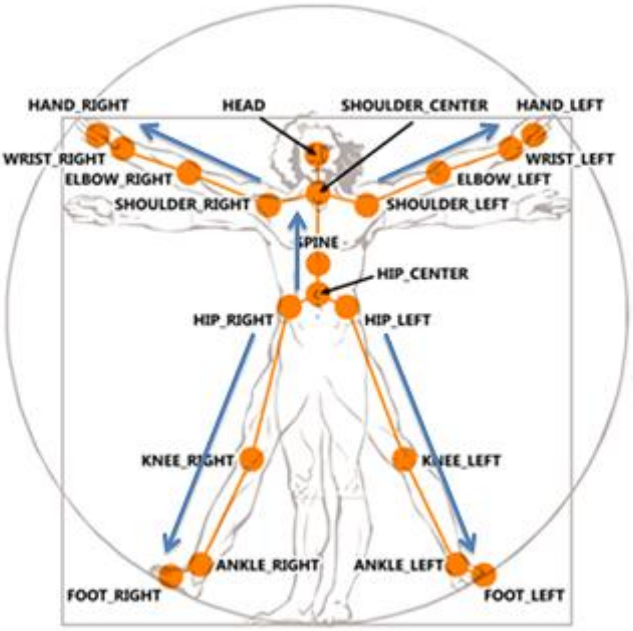
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ
ΠΛΗΡΟΦΟΡΙΚΗΣ & ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Καταγραφή κινήσεις/βημάτων στον χώρο με σύστημα χαμηλού κόστους



Κουτσιούμπας Παναγιώτης

Α.Ε.Μ. : 319

Επιβλ. Καθηγητές: Μηνάς Δασυγένης, Φαχαντίδης Νικόλαος

Κοζάνη, Οκτώβριος 2015

Περιεχόμενα

Περίληψη

Η διπλωματική εργασία επικεντρώνεται στη καταγραφή κίνησης στον χώρο, μέσω του Kinect version 1, και στην απεικόνιση αυτών των κινήσεων γραφικά στην οθόνη του υπολογίστη και καταγραφή των συντεταγμένων σε ένα αρχείο *txt. Μέσω του Microsoft Visual Studio και της γλώσσας προγραμματισμού C++, ολοκληρώθηκε η εφαρμογή αυτή. Το Kinect βασίζεται στην τεχνολογία λογισμικού για την τεχνολογία εύρους κάμερας όπου αναπτύχθηκε ένα σύστημα που μπορεί να ερμηνεύσει συγκεκριμένες χειρονομίες, καθιστώντας αποκλειστικό έλεγχο των ηλεκτρονικών συσκευών χωρίς χέρια. Το Microsoft Visual Studio είναι ένα ολοκληρωμένο περιβάλλον ανάπτυξης (IDE). Χρησιμοποιείται για την ανάπτυξη προγραμμάτων ηλεκτρονικών υπολογιστών, καθώς και εφαρμογών. Το Visual Studio χρησιμοποιεί πλατφόρμες ανάπτυξης λογισμικού της Microsoft, όπως τα Windows API, Windows Forms, Windows Presentation Foundation, το Windows Store και το Microsoft Silverlight. Το Visual Studio υποστηρίζει διαφορετικές γλώσσες προγραμματισμού. Η **C++** είναι μια γενικού σκοπού γλώσσα προγραμματισμού Η/Υ. Θεωρείται μέσου επιπέδου γλώσσα, καθώς περιλαμβάνει έναν συνδυασμό χαρακτηριστικών από γλώσσες υψηλού και χαμηλού επιπέδου.

Λέξεις κλειδιά

kinect sensor, microsoft visual studio, opencv, c++, kinect sdk, api

Abstract

The diploma thesis focuses on motion tracking in space via the Kinect version 1, and display these movements graphically on the computer screen and record the coordinates into a file * txt. Through the Microsoft Visual Studio and the programming language C ++, this application was completed. The Kinect is based on software technology for range camera technology where a system was developed that can interpret specific gestures, making it exclusive control of the electronic device without hands. To Microsoft Visual Studio is an integrated development environment (IDE). Used for developing computer programs and applications. The Visual Studio uses Microsoft software development platforms, including Windows API, Windows Forms, Windows Presentation Foundation, the Windows Store and Microsoft Silverlight. The Visual Studio supports different programming languages. The C ++ is a general purpose PC programming language. Considered medium level language, and includes a combination of characteristics of high and low-level languages.

Keywords

kinect sensor, microsoft visual studio, opencv, c++, kinect sdk, api

Εικόνες

4.3.4 Στιγμιότυπο της εφαρμογής

4.3.5 Στιγμιότυπο της εφαρμογής

4.3.6 Στιγμιότυπο της εφαρμογής

4.3.7 Στιγμιότυπο της εφαρμογής

Ευχαριστίες

Με την ευκαιρία της διπλωματικής εργασίας μου θα ήθελα να ευχαριστήσω θερμά τον κύριο Νικόλαο Φαχαντίδη, τον υπεύθυνο για την εκπόνηση της εργασίας, για όλη την καθοδήγηση και τη βοήθεια που μου παρείχε. Επίσης θα ήθελα να ευχαριστήσω την οικογένεια μου για τη στήριξη όλα αυτά τα χρόνια όπως επίσης και τους συμφοιτητές-φίλους μου για τα χρόνια που περάσαμε μαζί.

1. Εισαγωγή

Η διπλωματική εργασία ασχολείται με την καταγραφή κινήσεων στον χώρο με σύστημα χαμηλού κόστους. Το σύστημα χαμηλού κόστους που χρησιμοποιήθηκε ήταν το Kinect το version 1.

1.1.1 Kinect v1

Το πρόγραμμα που χρησιμοπιήθηκε για την ολοκλήρωση του προγράμματος ήταν το Microsoft visual studio 2010.

Microsoft Visual Studio

Το Microsoft Visual Studio είναι ένα ολοκληρωμένο περιβάλλον ανάπτυξης (IDE) από τη Microsoft. Χρησιμοποιείται για την ανάπτυξη προγραμμάτων ηλεκτρονικών υπολογιστών για τα Microsoft Windows, καθώς και web sites, εφαρμογών και υπηρεσιών web. Το Visual Studio χρησιμοποιεί πλατφόρμες ανάπτυξης λογισμικού της Microsoft, όπως τα Windows API, Windows Forms, Windows Presentation Foundation, το Windows Store και το Microsoft Silverlight. Μπορεί να παράγει τόσο εγγενή κώδικα όσο και διαχειριζόμενο κώδικα.

Το Visual Studio περιλαμβάνει έναν επεξεργαστή κώδικα υποστήριξης IntelliSense (το συστατικό ολοκλήρωσης κώδικα), καθώς και τον κωδικό refactoring. Το ολοκληρωμένο πρόγραμμα εντοπισμού σφαλμάτων

λειτουργεί τόσο ως ένα πρόγραμμα επιπέδου πηγής εντοπισμού σφαλμάτων όσο και ένα πρόγραμμα εντοπισμού σφαλμάτων επιπέδου μηχανής. Άλλα ενσωματωμένα εργαλεία περιλαμβάνουν έναν σχεδιαστή εντύπων για τη δημιουργία εφαρμογών GUI, web designer, σχεδιαστή τάξης, και σχεδιαστή σχήματος βάσης δεδομένων. Δέχεται plug-ins που βελτιώνουν τη λειτουργικότητα σχεδόν σε κάθε επίπεδο, συμπεριλαμβανομένων και την προσθέτη υποστήριξη για συστήματα πηγής ελέγχου (όπως η ανατροπή) και την προσθήκη νέων toolsets όπως εκδότες και visual designers για συγκεκριμένους τομείς γλώσσες ή toolsets για άλλες πτυχές του κύκλου ανάπτυξης λογισμικού.

Το Visual Studio υποστηρίζει διαφορετικές γλώσσες προγραμματισμού και επιτρέπει το πρόγραμμα επεξεργασίας κώδικα και εντοπισμού σφαλμάτων για την υποστήριξη (σε διάφορους βαθμούς) σχεδόν οποιασδήποτε γλώσσας προγραμματισμού, με την προϋπόθεση να υπάρχει μια συγκεκριμένη γλώσσα υπηρεσία. Οι ενσωματωμένες γλώσσες περιλαμβάνουν C, C ++ και C ++ / CLI (μέσω της Visual C ++), VB.NET (μέσω της Visual Basic .NET), C # (μέσω Visual C #), και F # (όπως του Visual Studio 2010). Υποστήριξη για άλλες γλώσσες, όπως η M, Python, Ruby και μεταξύ άλλων είναι διαθέσιμες μέσω των υπηρεσιών των γλώσσα που έχουν εγκατασταθεί ξεχωριστά. Υποστηρίζει, επίσης, XML / XSLT, HTML / XHTML, CSS και JavaScript. Java (και J #) που στηρίχθηκαν στο παρελθόν.

Η Microsoft παρέχει "Community" εκδόσεις το Visual Studio της χωρίς κόστος. Εμπορικές εκδόσεις του Visual Studio μαζί με επιλεγμένες εκδόσεις του παρελθόντος είναι διαθέσιμες δωρεάν στους φοιτητές μέσω του προγράμματος DreamSpark της Microsoft.

Η Microsoft παρέχει μια προεπισκόπηση του Visual Studio Code δωρεάν με μια ιδιόκτητη άδεια. Είναι ένας πηγαίος κώδικας και επεξεργαστής κειμένου, μαζί με άλλα χαρακτηριστικά, για Linux, OS X και Windows.

Η γλώσσα που χρησιμοποιήθηκε για την συγγραφη του προγραμματος ήταν η C++.

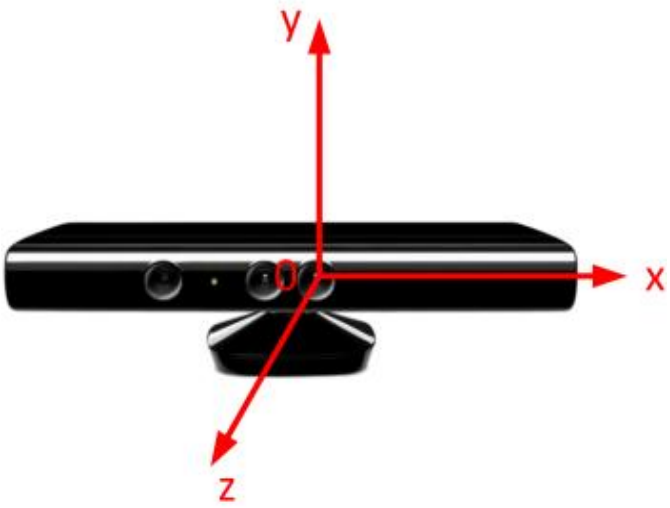
C++

Η **C++** (C Plus Plus, ελληνική προφ. Σι Πλας Πλας, φωνητική [siː plʌs plʌs]) είναι μια γενικού σκοπού γλώσσα προγραμματισμού Η/Υ. Θεωρείται μέσου επιπέδου γλώσσα, καθώς περιλαμβάνει έναν συνδυασμό χαρακτηριστικών από γλώσσες υψηλού και χαμηλού επιπέδου. Είναι μια μεταγλωττιζόμενη γλώσσα πολλαπλών παραδειγμάτων, με τύπους. Υποστηρίζει δομημένο, αντικειμενοστρεφή και γενικό προγραμματισμό. Η C++ κληρονόμησε το μεγαλύτερο μέρος της σύνταξης της C και τον προεπεξεργαστή της C.

Η C++ είναι σχεδιασμένη:

- ως μια γενικής χρήσης γλώσσα με στατικούς τύπους, που είναι όσο αποτελεσματική και φορητή, όσο η C
- ωστε να υποστηρίζει άμεσα και σφαιρικά πολλά είδη προγραμματισμού (δομημένος προγραμματισμός, αντικειμενοστρεφής προγραμματισμός, γενικός προγραμματισμός)
- ωστε να δίνει επιλογές στον προγραμματιστή, ακόμα κι αν του επιτρέπει να επιλέξει λανθασμένα
- με σκοπό να είναι όσο το δυνατόν συμβατή με τη C, διευκολύνοντας έτσι τη μετάβαση από τη C στη C++
- με σκοπό να αποφεύγει χαρακτηριστικά που αναφέρονται σε συγκεκριμένες πλατφόρμες ή δεν είναι γενικής χρήσης
- ώστε να μην δημιουργείται επιπλέον επεξεργαστικό κόστος για χαρακτηριστικά της γλώσσας που δεν χρησιμοποιούνται
- ώστε να λειτουργεί χωρίς κάποιο εξελιγμένο προγραμματιστικό περιβάλλον

Έτσι το πρόγραμμα, αρχικά εντοπίζει έναν ανθρώπινο σκελετό, στην συνέχεια απεικονίζει συγκεκριμένα χαρακτηριστικά όπως γόνατο, αστράγαλος και πατούσα και τέλος καταγράφει τις συντεταγμένες (x,y,z) αυτών των χαρακτηριστικών σε ένα αρχείο σύμφωνα με την παρακάτω εικόνα :



1.1.2 Συντεταγμένες (x,y,z) του Kinect

2. Kinect

Το Kinect (με την κωδική ονομασία στην ανάπτυξη ως Project Natal) είναι μια γραμμή ανίχνευσης κίνησης συσκευές εισόδου από τη Microsoft για Xbox 360 και Xbox One κονσόλες βιντεοπαιχνιδιών και Windows υπολογιστών. Βασισμένο σε ένα περιφερειακό πρόσθετο στυλ κάμερας , επιτρέπει στους χρήστες να ελέγχουν και να αλληλεπιδρούν με την κονσόλα / υπολογιστή τους, χωρίς τη ανάγκη για ένα χειριστήριο, μέσω μιας φυσικής διεπαφής χρήστη με τη χρήση χειρονομιών και ομιλητικών εντολών. Το Kinect πρώτης γενιάς παρουσιάστηκε για πρώτη φορά το Νοέμβριο του 2010, σε μια προσπάθεια να διευρύνει το κοινό του Xbox 360 πέρα από τυπική βάση παίχτη. Μια έκδοση για Windows κυκλοφόρησε την 1η Φεβρουαρίου του 2012. Το Kinect ανταγωνίζεται με διάφορους ελεγκτές κίνησης σε άλλες σπιτικές κονσόλες, όπως το Wii Remote Plus για το Wii και το Wii U, το PlayStation Move / PlayStation Eye για το PlayStation 3, PlayStation και PlayStation κάμερα για 4.

Η Microsoft κυκλοφόρησε το κιτ ανάπτυξης λογισμικού Kinect για Windows 7 στις 16 Ιουνίου του 2011. Η SDK είχε σκοπό να επιτρέψει στους προγραμματιστές να γράψουν εφαρμογές για το Kinect σε C ++ / CLI, C #, ή Visual Basic .NET.

2.1 Τεχνολογία

Το Kinect βασίζεται στην τεχνολογία λογισμικού που αναπτύχθηκε από τη Rare, μια θυγατρική της Microsoft Game Studios που ανήκει στην Microsoft, και για την τεχνολογία εύρους κάμερας από την ισραηλινή προγραμματιστή PrimeSense, η οποία ανέπτυξε ένα σύστημα που μπορεί να ερμηνεύσει συγκεκριμένες χειρονομίες, καθιστώντας αποκλειστικό έλεγχο χωρίς χέρια των ηλεκτρονικών συσκευών που είναι δυνατόν με τη χρήση ενός υπέρυθρου προβολέα και κάμερας και ένα ειδικό μικροτσίπ για την παρακολούθηση της κίνησης των αντικειμένων και των ατόμων σε τρεις διαστάσεις. Αυτό το σύστημα 3D scanner που ονομάζεται Κωδικοποίηση Φωτός χρησιμοποιεί μια παραλλαγή της εικόνας με βάση τη 3D ανακατασκευή.

Ο αισθητήρας Kinect είναι μια οριζόντια μπάρα που συνδέεται σε μια μικρή βάση με ένα μηχανοκίνητο άξονα και έχει σχεδιαστεί για να τοποθετείται κατά μήκος πάνω ή κάτω από την οθόνη απεικόνισης. Η συσκευή διαθέτει "RGB κάμερα, αισθητήρα βάθους και συστοιχία μικροφώνου που τρέχουν απο το ίδιο το λογισμικό του", το οποίο παρέχει κάλυψη 3D κίνησης για όλο το σώμα, αναγνώριση προσώπου και δυνατότητες αναγνώρισης φωνής. Κατά την προώθηση, η αναγνώριση φωνής ήταν διαθέσιμη μόνο στην Ιαπωνία, το Ηνωμένο Βασίλειο, τον Καναδά και τις Ηνωμένες Πολιτείες. Η Ηπειρωτική Ευρώπη έλαβε το χαρακτηριστικό αυτό αργότερα, την άνοιξη του ‘11. Αυτή τη στιγμή, η αναγνώριση φωνής υποστηρίζεται στην Αυστραλία, τον Καναδά, τη Γαλλία, τη Γερμανία, την Ιρλανδία, την Ιταλία, την Ιαπωνία, το Μεξικό, τη Νέα Ζηλανδία, Ηνωμένο Βασίλειο και Ηνωμένες Πολιτείες. Η συστοιχία μικροφώνου του Kinect αισθητήρα δίνει τη δυνατότητα στο Xbox 360 να διεξάγει ακουστικό εντοπισμό πηγής και την καταστολή του θορύβου του περιβάλλοντος.

Ο αισθητήρας βάθους αποτελείται από έναν υπέρυθρου προβολέα λέιζερ σε συνδυασμό με έναν μονόχρωμο αισθητήρα CMOS, ο οποίος καταγράφει τα εικονικά δεδομένα σε 3D κάτω από οποιεσδήποτε συνθήκες φωτισμού του περιβάλλοντος.Το εύρος ανίχνευσης του αισθητήρα βάθους είναι ρυθμιζόμενο, και το λογισμικό του Kinect είναι ικανό για αυτόματη βαθμονόμηση του αισθητήρα που βασίζεται στο παιχνίδι και στο φυσικό περιβάλλον του παίκτη, συμπεριλαμβάνοντας την παρουσία επίπλων ή άλλων εμποδίων.

Περιγράφεται από το προσωπικό της Microsoft ως η κύρια καινοτομία του Kinect, η τεχνολογία λογισμικού επιτρέπει προηγμένης αναγνώρισης χειρονομιών, αναγνώριση προσώπου και αναγνώριση φωνής. Σύμφωνα με τις πληροφορίες που παρέχονται στους λιανοπωλητές, το Kinect είναι ικανό για ταυτόχρονο εντοπισμού μέχρι και έξι ατόμων, μεταξύ των οποίων δύο ενεργοί παίκτες για ανάλυση κίνησης με εξαγωγή 20 χαρακτηριστικών των αρθρώσεων ανά παίκτη. Ωστόσο, η PrimeSense έχει δηλώσει ότι ο αριθμός των ανθρώπων που η συσκευή μπορεί να "δει" (αλλά όχι την επεξεργασία ως παίκτες) περιορίζεται μόνο από το πόσους θα ταιριάζει στο οπτικό πεδίο της κάμερας.

Η αντίστροφη μηχανική έχει διαπιστώσει ότι διάφοροι αισθητήρες εξόδου βίντεο του Kinect κατά ένα ρυθμό καρέ από 9 Hz έως 30 Hz, εξαρτώνται από την ανάλυση. Η προεπιλεγμένη RGB ροή βίντεο χρησιμοποιεί 8-bit VGA ανάλυση (640 × 480 pixels) με ένα Bayer φίλτρο χρώματος, αλλά το υλικό είναι ικανό για αναλύσεις έως και 1280x1024 (με χαμηλότερο ρυθμό καρέ) και άλλες μορφές χρώματος, όπως UYVY. Το μονόχρωμο βάθος ανίχνευσης ροής βίντεο είναι σε ανάλυση VGA (640 × 480 pixels) με βάθος 11-bit, το οποίο παρέχει 2.048 επίπεδα ευαισθησίας. Το Kinect μπορεί επίσης δείχνει τη θέα από κάμερα υπερύθρων άμεσα (δηλαδή: πριν έχει μετατραπεί σε ένα χάρτη βάθους) ως 640x480 βίντεο, ή 1280x1024 σε χαμηλότερο ρυθμό καρέ. Ο αισθητήρας Kinect έχει ένα πρακτικό όριο που κυμαίνεται από 1.2-3.5μ απόσταση, όταν χρησιμοποιείται με το λογισμικό του Xbox. Η περιοχή που απαιτείται για να παίξει το Kinect είναι περίπου 6 τετραφωνικά μέτρα, αν και ο αισθητήρας μπορεί να διατηρήσει την παρακολούθηση μέσω ενός εκτεταμένου εύρος περίπου 0,7 έως 6 μ. Ο αισθητήρας έχει ένα οπτικό πεδίο 57 ° από οριζόντια και κάθετα 43 °, ενώ το μοτέρ περιστροφής είναι ικανή κλίση του αισθητήρα έως 27 ° προς τα πάνω ή

προς τα κάτω. Το οριζόντιο πεδίο του αισθητήρα Kinect στην ελάχιστη απόσταση θέασης του από 0,8μ είναι ως εκ τούτου από 87 εκατοστά, και το κατακόρυφο πεδίο είναι από 63 εκατοστά, με αποτέλεσμα μια ανάλυση μόλις πάνω από 1.3 μμ ανά pixel. Η συστοιχία μικροφώνου διαθέτει τέσσερις κάψουλες μικρόφωνου και λειτουργεί με κάθε κανάλι επεξεργασίας ήχου 16-bit σε ρυθμό δειγματοληψίας 16 kHz.

Επειδή ο μηχανοκίνητος μηχανισμός κλίσης του αισθητήρα Kinect απαιτεί περισσότερη ισχύ από τις θύρες USB του Xbox 360 που μπορούν να προμηθεύσουν, η συσκευή κάνει χρήση μιας ιδιόκτητης υποδοχής USB που συνδυάζει την επικοινωνία με πρόσθετη ισχύ. Τα επανασχεδιασμένα Xbox 360 S μοντέλα περιλαμβάνουν μια ειδική θύρα AUX για την υποδοχή του συνδετήρα, ενώ παλαιότερα μοντέλα απαιτούν ένα ειδικό καλώδιο παροχής ρεύματος (περιλαμβάνεται με τον αισθητήρα) που χωρίζει τη σύνδεση σε ξεχωριστές συνδέσεις USB και συνδέσεις ρεύματος, η ισχύς παρέχεται από το ηλεκτρικό δίκτυο μέσω του μετασχηματιστή AC.

2.2 Kinect για Windows

Στις 21 Φεβρουαρίου του 2011 η Microsoft ανακοίνωσε ότι θα κυκλοφορήσει ένα μη-εμπορικό σετ ανάπτυξης λογισμικού Kinect (SDK) για τα Windows την άνοιξη του 2011, το οποίο κυκλοφόρησε για τα Windows 7 στις 16 Ιουνίου του 2011, σε 12 χώρες. Το SDK περιλαμβάνει Windows 7 συμβατά PC προγράμματα οδήγησης για τη συσκευή Kinect. Παρέχει δυνατότητες του Kinect για τους προγραμματιστές να δημιουργήσουν εφαρμογές με C ++, C #, ή Visual Basic, χρησιμοποιώντας το Microsoft Visual Studio 2010 και περιλαμβάνει τις ακόλουθες δυνατότητες:

1. Πρώτα ρεύματα αισθητήρα: Πρόσβαση σε ρέματα χαμηλού επιπέδου από τον αισθητήρα βάθους, αισθητήρα έγχρωμης κάμερας, και τέσσερα στοιχεία του πίνακα μικροφώνου.
2. Ανίχνευση σκελετού: Η δυνατότητα να καταγράφεται την εικόνα του σκελετού ενός ή δύο ατόμων που κινούνται εντός του οπτικού πεδίου του Kinect για εφαρμογές με γνώμνονα την χειρονομία.
3. Προηγμένες δυνατότητες ήχου: οι δυνατότητες επεξεργασίας ήχου περιλαμβάνουν εξελιγμένο ακουστικό καταστολής του θορύβου και ακύρωσης ηχού, σχηματισμό δέσμης για να προσδιορίσει την τρέχουσα πηγή ήχου, καθώς και ενοποίηση με τα Windows αναγνώρισης ομιλίας API.
4. Δείγμα κώδικα και Τεκμηρίωσης.

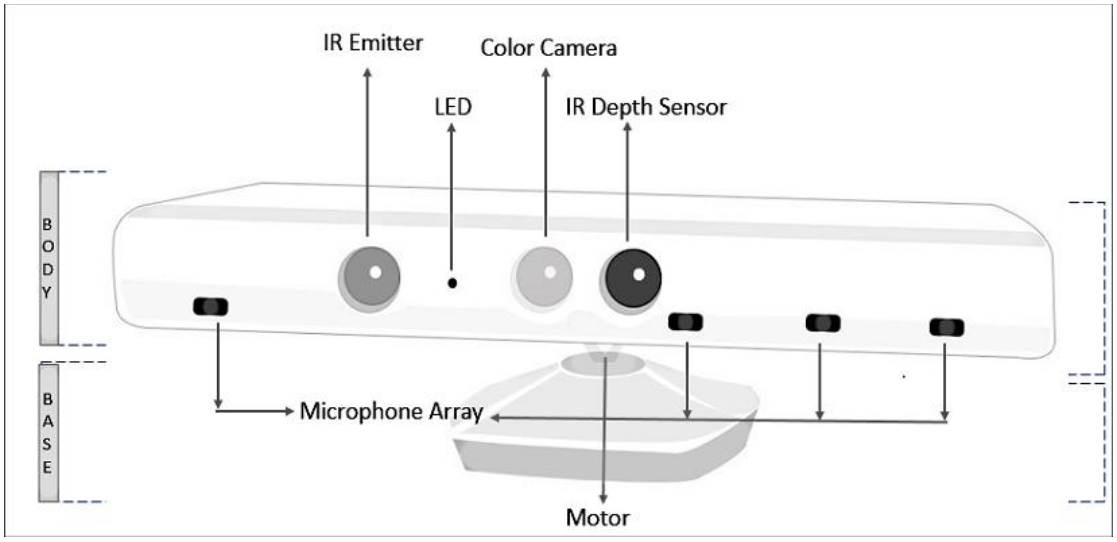
Τον Μάρτιο του 2012, ο Craig Eisler, ο γενικός διευθυντής του Kinect για Windows, δήλωσε ότι σχεδόν 350 εταιρείες εργάζονται με τη Microsoft σχετικά με προσαρμοσμένες εφαρμογές του Kinect για Windows.

2.3 Συνιστώσες του Kinect για Windows

Ο αισθητήρας Kinect περιλαμβάνει τις εξής βασικές συνιστώσες:

- Έγχρωμη κάμερα
- Υπερύθρων (IR)
- IR αισθητήρα βάθους
- Μοτέρ Κλίσης
- Συστοιχία Μικροφώνου
- LED

Η εικόνα που ακολουθεί δείχνει τα διάφορα συστατικά στοιχεία ενός αισθητήρα Kinect:



2.3.1 Οι βασικές συνιστώσες του Kinect

2.4 Kinect v1 vs v2

Το Kinect v2 αναγνώρισης προσώπου, παρακολούθησης κίνησης, και ανάλυσης είναι πολύ πιο ακριβές από ό, τι το v1. Το Kinect v2 χρησιμοποιεί “time of flight” τεχνολογία για να προσδιορίσει τα χαρακτηριστικά και την κίνηση των συγκεκριμένων αντικειμένων. Με τη χρήση αυτής της τεχνολογίας, η Kinect v2 μπορεί να δει καλά σε ένα εντελώς σκοτεινό δωμάτιο όπως ακριβώς και σε ένα καλά φωτισμένο δωμάτιο. Αν και το πρώτο Kinect χρησιμοποιεί παρόμοια τεχνολογία, η Kinect v2 έχει βελτιωθεί σε μεγάλο βαθμό από αυτό. Το Kinect v2 έχει 1080 ανάλυση (HD).

Το Kinect v2 μπορεί να επεξεργαστεί 2 gigabytes δεδομένα ανά δευτερόλεπτο, το USB 3 παρέχει σχεδόν 10 φορές ταχύτερη ευρυζωνική σύνδεση για τη μεταφορά δεδομένων, το 60% του ευρύτερου οπτικού πεδίου, και μπορεί να ανιχνεύσει και να παρακολουθήση 20 αρθρώσεις από 6 ανρώπινα σώματα, συμπεριλαμβανομένων και των αντίχειρων. Σε σύγκριση, ο Kinect v1 θα μπορούσε να παρακολουθήσει μόνο 20 αρθρώσεις από 2 άτομα. Στην κορυφή αυτής, όταν χρησιμοποιούμε το Kinect v2 είμαστε ικανοί να ανιχνεύσουμε τους ρυθμούς της καρδιάς, τις εκφράσεις του προσώπου και τα βάρη στα άκρα, μαζί με πολλά άλλα εξαιρετικά πολύτιμα βιομετρικά δεδομένα. Η συσκευή Kinect v1.0 δεν έχει την πιστότητα για να παρακολουθήση ξεχωριστά τα δάχτυλα και το τέντωμα και τη συρρίκνωση με τα χέρια και τους ώμους, αλλά το Kinect v2 έχει αυτές τις δυνατότητες. Είναι σαφές ότι αυτή η τεχνολογία είναι σίγουρα πολύ πιο ισχυρή και πολύπλοκη από την πρώτη γενιά των Kinect.

Feature	Kinect for Windows 1	Kinect for Windows 2
Color Camera	640 x 480 @30 fps	1920 x 1080 @30 fps
Depth Camera	320 x 240	512 x 424
Max Depth Distance	~4.5 M	~4.5 M
Min Depth Distance	40 cm in near mode	50 cm
Horizontal Field of View	57 degrees	70 degrees
Vertical Field of View	43 degrees	60 degrees
Tilt Motor	yes	no
Skeleton Joints Defined	20 joints	26 joints
Full Skeletons Tracked	2	6
USB Standard	2.0	3.0
Supported OS	Win 7, Win 8	Win 8
Price	$299	TBD

2.4.1 Διαφορές Kinect v1 με v2

3. Εγκατάσταση

3.1 Ανάπτυξη εργαλείων και λογισμικού

Τα παρακάτω είναι το λογισμικό που απαιτείται για την ανάπτυξη του Kinect SDK :

• Microsoft Visual Studio 2010 Express ή υψηλότερες εκδόσεις του Visual Studio

• Microsoft .NET Framework 4.0 ή νεότερη έκδοση

• Kinect για Windows SDK

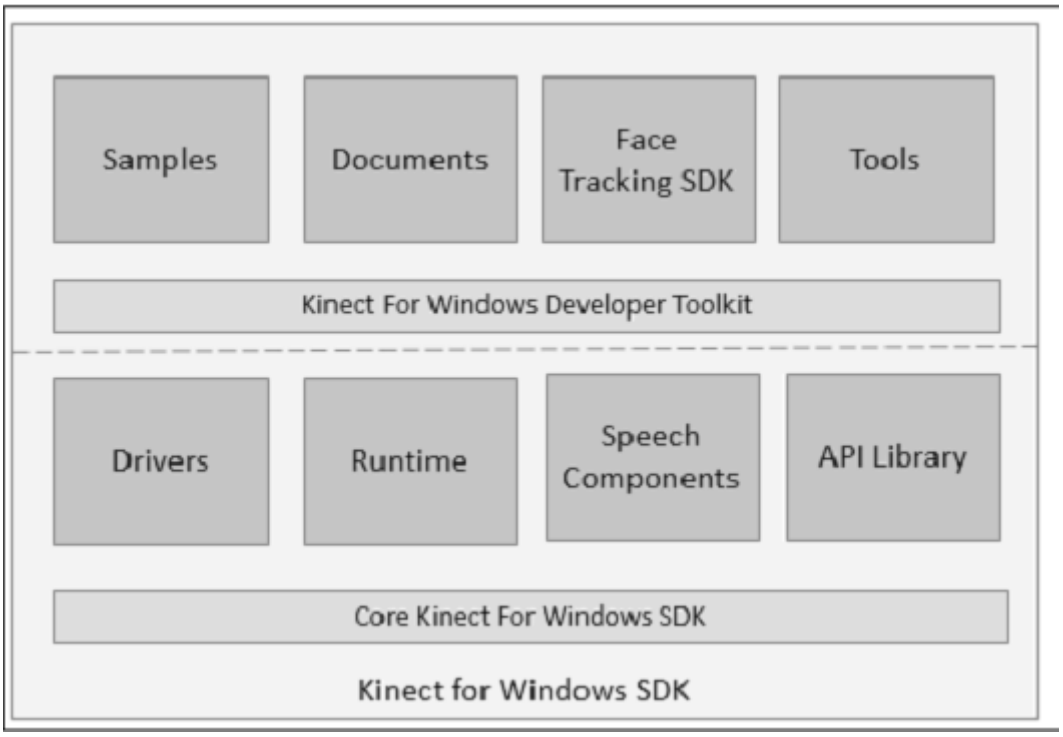
• Κατέβασμα του Opencv 3.0.0

3.2 Κατεβάζοντας το SDK και το Developer Toolkit

Το Kinect SDK και ο Οδηγός για προγραμματιστές είναι διαθέσιμα δωρεάν και μπορεί να τα κατεβάσετε από http://www.microsoft.com/en-us/kinectforwindows/.

Το πρόγραμμα εγκατάστασης θα εγκαταστήσει αυτόματα την έκδοση 64- ή 32-bit του SDK, ανάλογα με το λειτουργικό σας σύστημα. Το Kinect για Windows Developer Toolkit είναι ένα πρόσθετο εγκατάστασης που περιλαμβάνει δείγματα, τα εργαλεία, και άλλες επεκτάσεις ανάπτυξης.

Το ακόλουθο διάγραμμα δείχνει αυτά τα στοιχεία:



3.2.1 Kinect for Windows - Developer Toolkit

3.3 Εγκατάσταση Kinect για Windows SDK

Πριν από την εκτέλεση της εγκατάστασης, βεβαιωθείτε για τα ακόλουθα:

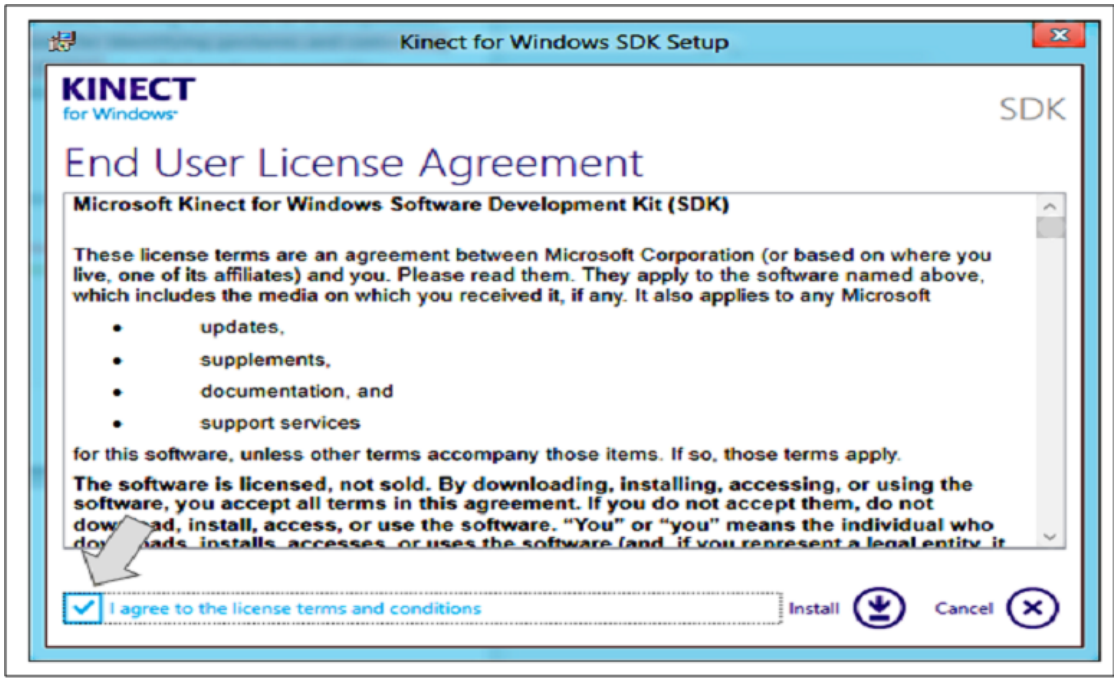
• Έχετε απεγκαταστήσει όλες τις προηγούμενες εκδόσεις του Kinect για Windows SDK

• Ο αισθητήρας Kinect δεν έχει συνδεθεί στη θύρα USB του υπολογιστή σας

• Δεν υπάρχουν περιπτώσεις που τρέχουν σε Visual Studio

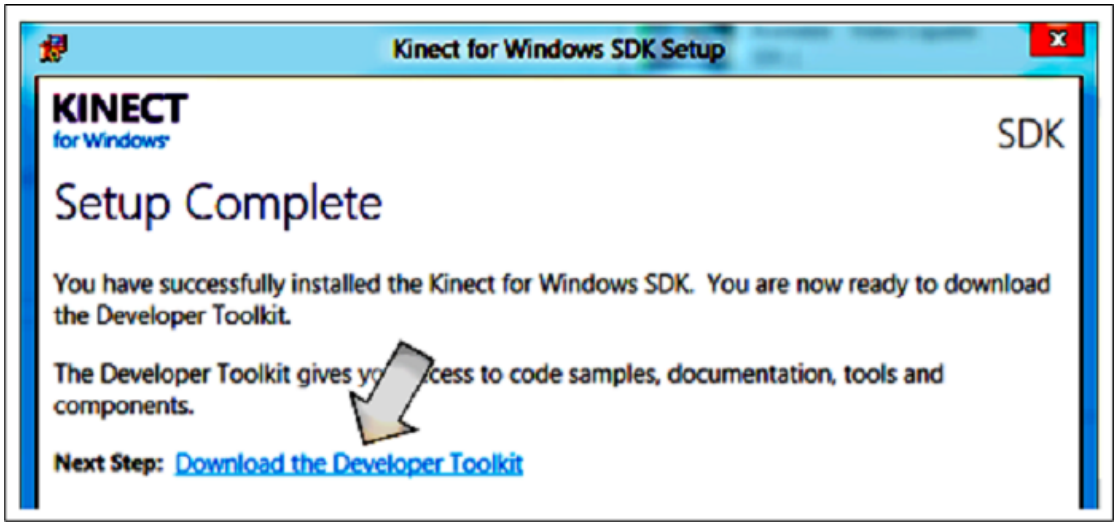
Ξεκινήστε το πρόγραμμα εγκατάστασης, το οποίο θα εμφανιστεί στην οθόνη εκκίνησης ως End User License Agreement. Θα πρέπει να διαβάσετε και να αποδεχτείτε αυτήν τη συμφωνία για να προχωρήσει η εγκατάσταση.

Το παρακάτω screenshot δείχνει την άδεια χρήσης:



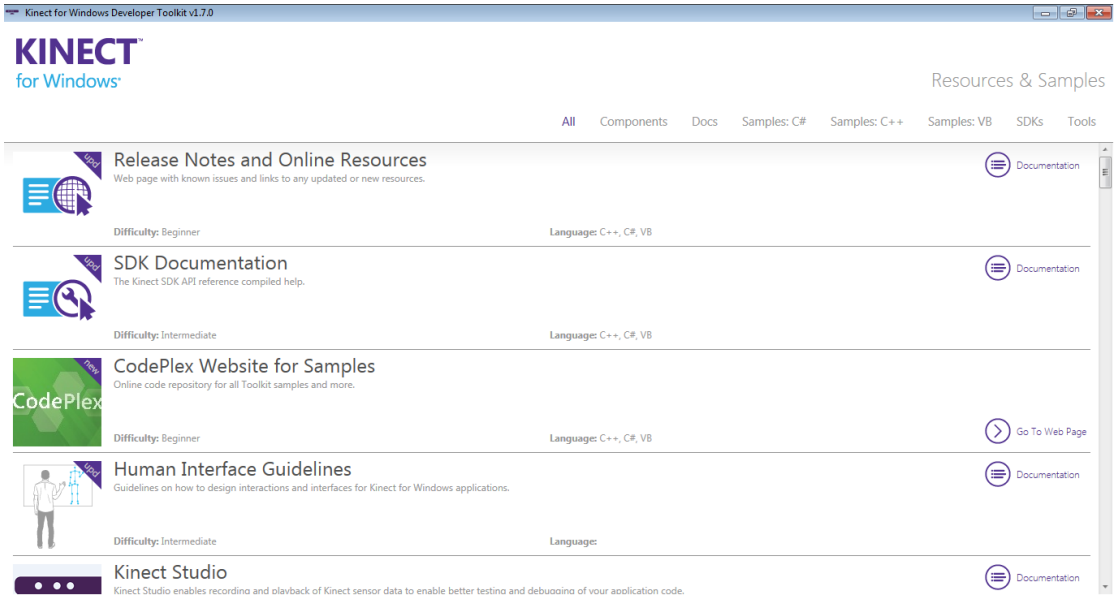
3.3.1 Kinect SDK

Μόλις τελειώσει η εγκατάσταση, θα ενημερωθείτε για μια νέα επιλογή για την εγκατάσταση του Developer Toolkit, όπως φαίνεται στην επόμενη εικόνα:



3.3.2 Kinect SDK setup

Και μετά το τέλος της εγκατάστασης του Developer Toolkit, μπορείτε να το τρέξετε και θα έχετε το εξής αποτέλεσμα:



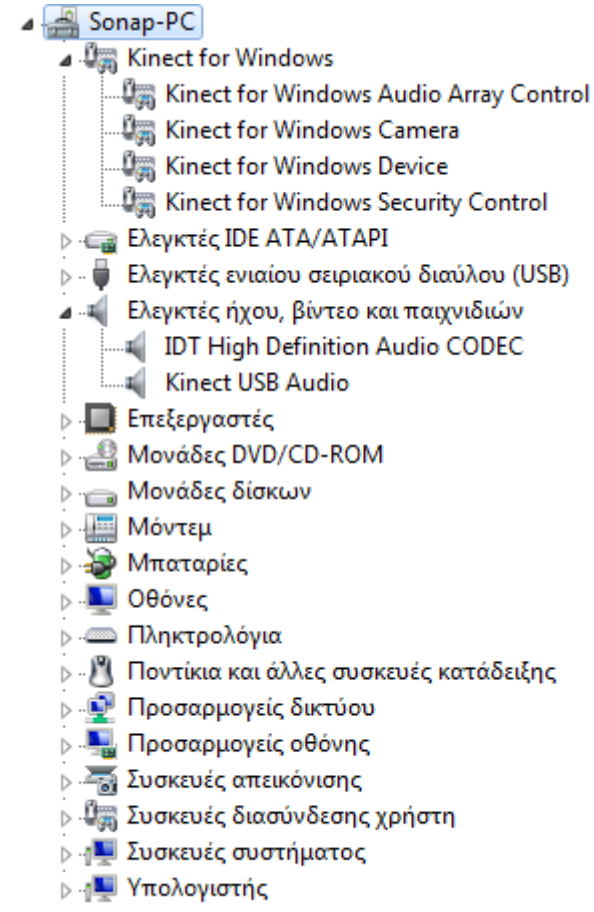
3.3.3 Developer Toolkit

3.4 Σύνδεση του αισθητήρα με το σύστημα

Τώρα που έχουμε εγκαταστήσει το SDK, μπορούμε να συνδέσετε τη συσκευή Kinect στο PC σας. Την πρώτη φορά που θα συνδέσετε τη συσκευή στο σύστημά σας, θα παρατηρήσετε την ενδεικτική λυχνία LED του αισθητήρα Kinect να αναβοσβήνει σταθερά κόκκινη και το σύστημα θα ξεκινήσει αυτόματα την εγκατάστασητα προγράμματων οδήγησης. Οι οδηγοί θα φορτωθούν μόνο μετά την πλήρη εγκατάσταση του SDK είναι. Η διαδικασία αυτή ελέγχει για τις πιο πρόσφατες ενημερώσεις των Windows σε USB προγράμματα οδήγησης, γι 'αυτό είναι καλό να συνδεθεί με το Internet.

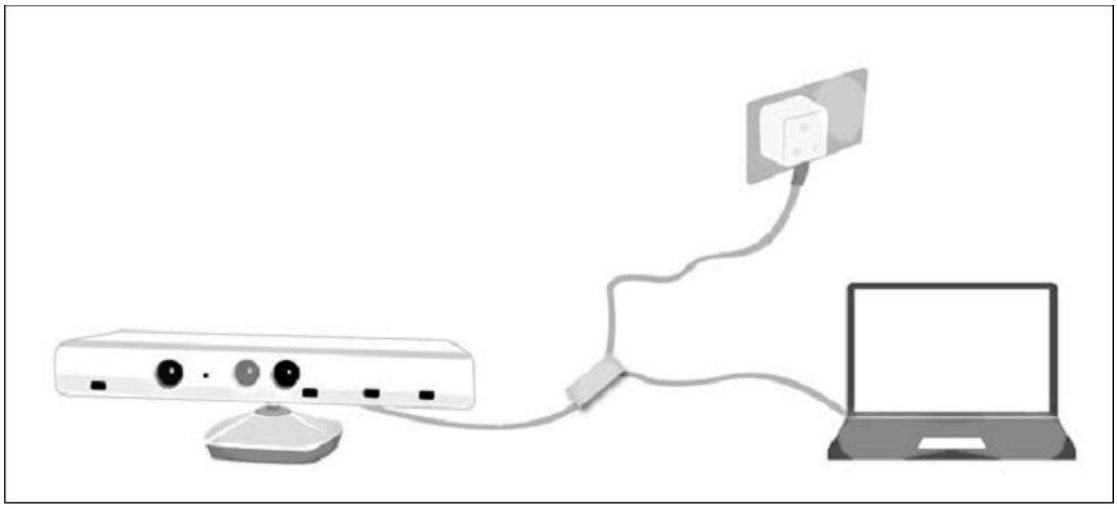
3.5 Επαλήθευση των εγκατεστημένων προγραμμάτων οδήγησης

Αυτό είναι συνήθως μια διαδικασία αντιμετώπισης προβλημάτων σε περίπτωση που αντιμετωπίσετε κάποιο πρόβλημα. Επίσης, η διαδικασία επαλήθευσης θα σας βοηθήσει να καταλάβετε πώς οι οδηγοί της συσκευής έχουν εγκατασταθεί στο σύστημά σας. Για να βεβαιωθείτε ότι οι οδηγοί έχουν εγκατασταθεί σωστά, ανοίξτε τον Πίνακα Ελέγχου και επιλέξτε Διαχείριση συσκευών, στη συνέχεια, αναζητήστε το κόμβο Kinect for Windows. Θα βρείτε την επιλογή Kinect for Windows Device όπως φαίνεται στην επόμενη εικόνα:



3.5.1 Kinect Drivers

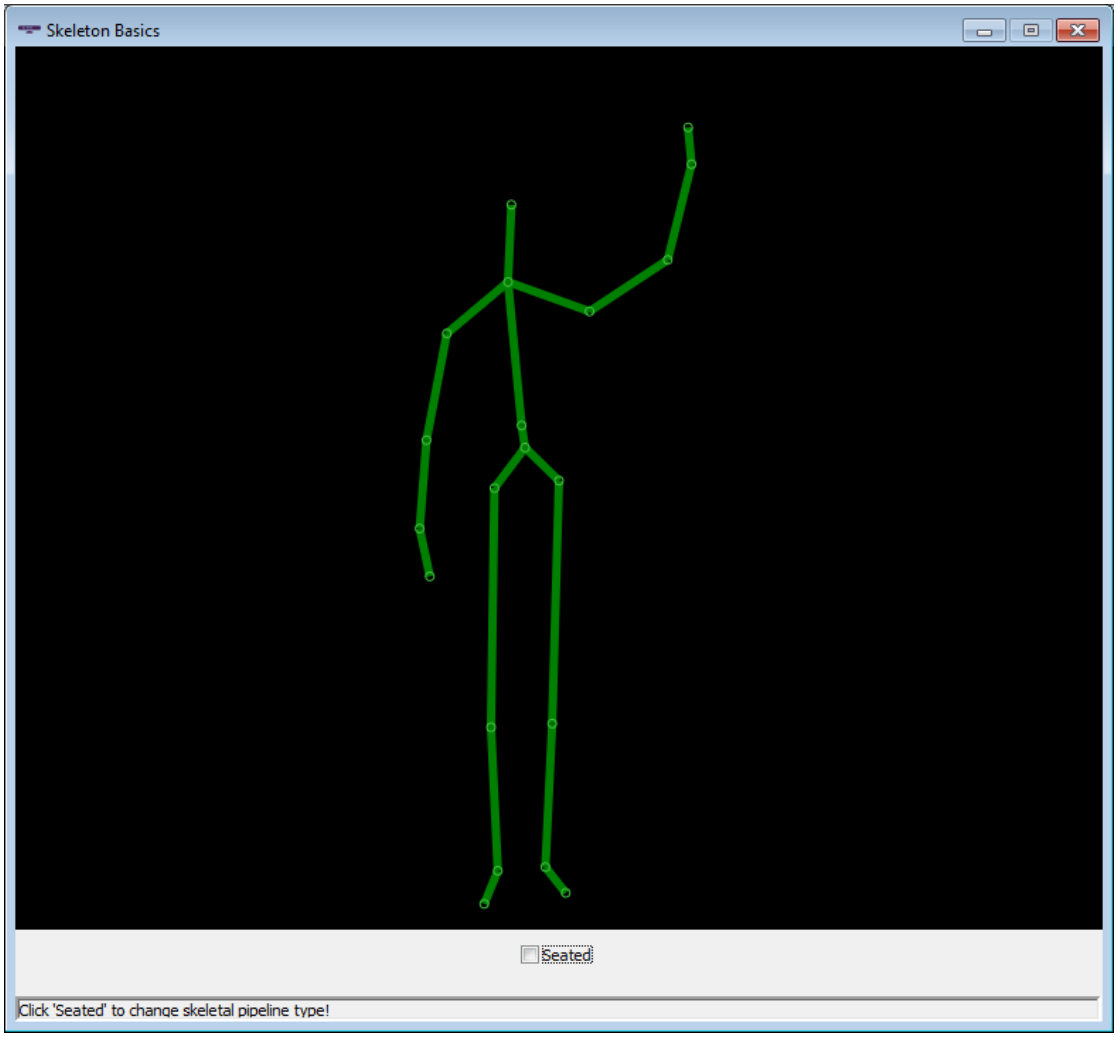
Η επόμενη εικόνα δείχνει τον αισθητήρα Kinect με υποδοχή USB και τροφοδοτικό, και το πώς έχουν χρησιμοποιηθεί:



3.5.2 Στιγμιότυπο με Kinect, Laptop και τροφοδοσία

3.6 Έλεγχος Συσκευής

Το Developer Toolkit έχει ένα σύνολο από δείγματα εφαρμογών όπου μπορείτε να επιλέξετε κάποια από αυτά για να δοκιμάσετε τη συσκευή σας. Για το επόμενο στιγμιότυπο χρησιμοποιήθηκε το παράδειγμα «Skeleton Basics - D2D»



3.6.1 Έλεγχος Kinect

4. Ξεκινώντας το «χτίσιμο» της εφαρμογής

4.1 Λειτουργίες Kinect

Για την ανάπτυξη της κάθε εφαρμογής Kinect, υπάρχουν ορισμένες κοινές λειτουργίες που πρέπει να εκτελεστούν,όπως:

• Η εφαρμογή πρέπει να εντοπίζει τη συνδεδεμένη συσκευή Kinect και πρέπει να την ξεκινήσει.

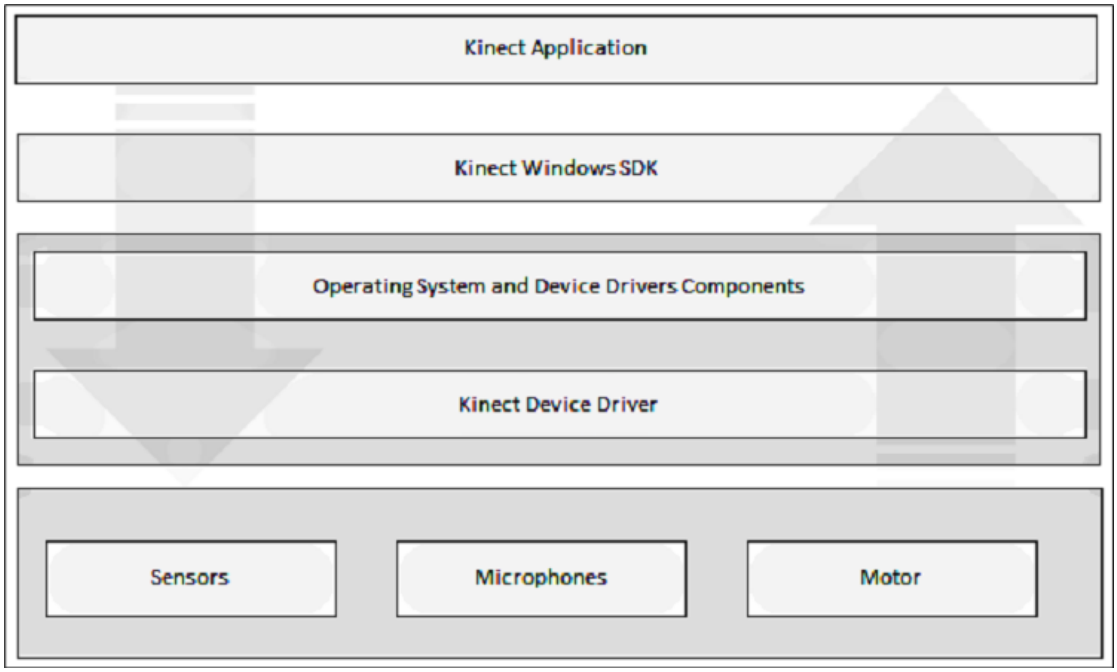
• Μόλις ξεκινήσει ο αισθητήρας, η εφαρμογή πρέπει να προετοιμάσει και να κάνει εγγραφή το είδος των στοιχείων που απαιτούνται από τον αισθητήρα.

• Κατά τη διάρκεια του συνολικού κύκλου εκτέλεση της εφαρμογής, ένας αισθητήρας μπορεί να αλλάξει την κατάστασή του. Η εφαρμογή πρέπει να παρακολουθεί τις αλλαγές στην κατάσταση για τη συνδεδεμένη συσκευή και να το χειρίζεται κατάλληλα.

• Όταν η εφαρμογή τερματίζεται / τελειώνει, θα πρέπει να κλείσει σωστά τη συσκευή.

Πώς εφαρμογές αλληλεπιδρούν με το Kinect αισθητήρα

Το Kinect για Windows SDK λειτουργεί ως διεπαφή μεταξύ της συσκευής Kinect και την εφαρμογή. Όταν χρειαστεί να αποκτήσουμε πρόσβαση στον αισθητήρα, η εφαρμογή στέλνει μια κλήση API στον οδηγό. Οι έλεγχοι οδηγών του Kinect έχουν πρόσβαση στα δεδομένα των αισθητήρων.



4.1.1 Η εφαρμογή συνδέεται με τον αισθητήρα

Κατανοώντας την ταξινόμηση των APIs SDK

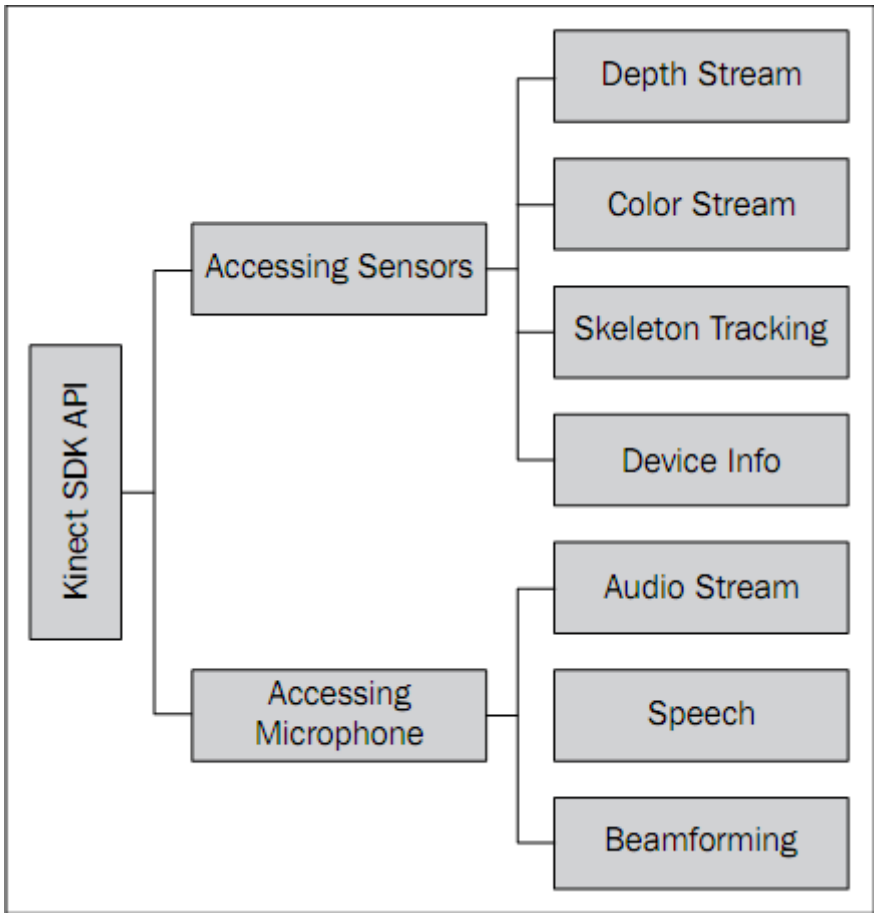
Για να κατανοήσουμε τη λειτουργία των διαφόρων APIs και να γνωρίζουν τη χρήση τους, είναι πάντα καλό να έχουμε μια σαφή εικόνα του τρόπου με τον οποίο δουλεύουν.

Μπορούμε να ταξινομήσει τις βιβλιοθήκες SDK στις δύο ακόλουθες κατηγορίες:

- Εκείνους που ελέγχουν και έχουν πρόσβαση στους αισθητήρες Kinect.
- Εκείνους που έχουν πρόσβαση σε μικρόφωνα και έλεγχο ήχου.

Η πρώτη κατηγορία αφορά τους αισθητήρες με τη σύλληψη του χρώματος, τις υπέρυθρες ροές δεδομένων, και το βάθος, παρακολουθώντας ανθρώπινους σκελετούς και να αναλαμάνοντας τον έλεγχο της προετοιμασίας του αισθητήρα. Ένα σύνολο από APIs σε αυτή την κατηγορία επικοινωνεί άμεσα με το υλικό του αισθητήρα, ενώ μερικά APIs στην επεξεργασία εφαρμόζουν τα δεδομένα που καταγράφονται από τον αισθητήρα.

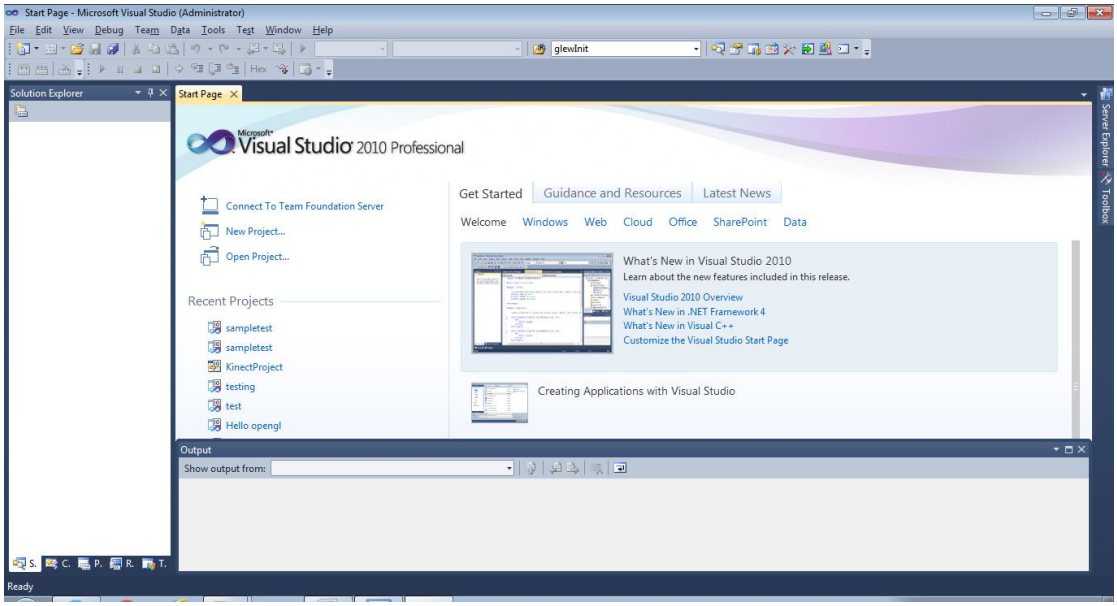
Από την άλλη πλευρά, ο ήχος των APIs ελέγχει τη συστοιχία μικροφώνου του Kinect και βοηθά να συλλάβει τη ροή ήχου από τους αισθητήρες, τον έλεγχο της πηγής ήχου και τη δυνατότητα αναγνώρισης ομιλίας, και ούτω καθεξής. Το παρακάτω διάγραμμα δείχνει μια υψηλού επιπέδου API ταξινόμηση βασισμένο στον τύπο της εργασίας που το API εκτελείται:



4.1.2 APIs SDK

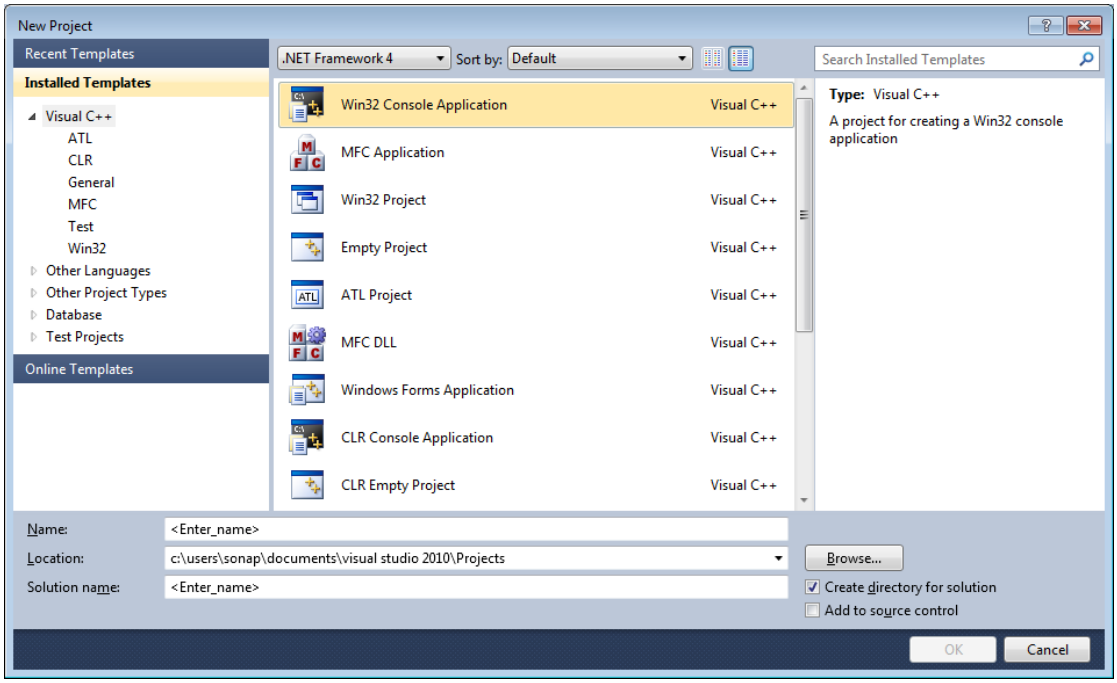
4.2 Δημιουργία νέου project στο Visual Studio

Ανοίγουμε την εφαρμογή Microsoft Visual Studio και ξεκινάμε μια νέα συνεδρία.



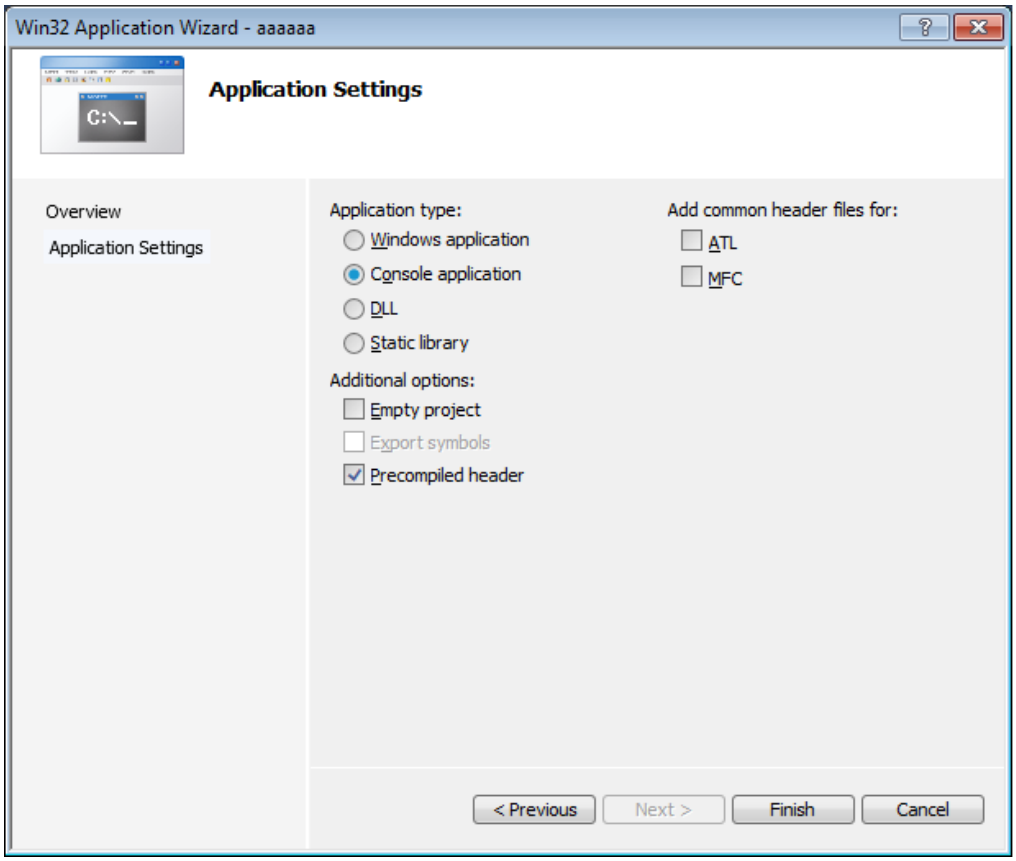
4.2.1 Microsoft Visual Studio Start up

Δημιουργούμε ένα νέο project απο το File -> New Project. Αυτό θα ανοίξει ένα νέο παράθυρο με το καινούργιο project. Επιλέγουμε C++ και Win32 Console Project, όπως φαίνεται και στο παρακάτω στιγμιότυπο:



4.2.2 Επιλογή γλώσσας και Κονσόλας

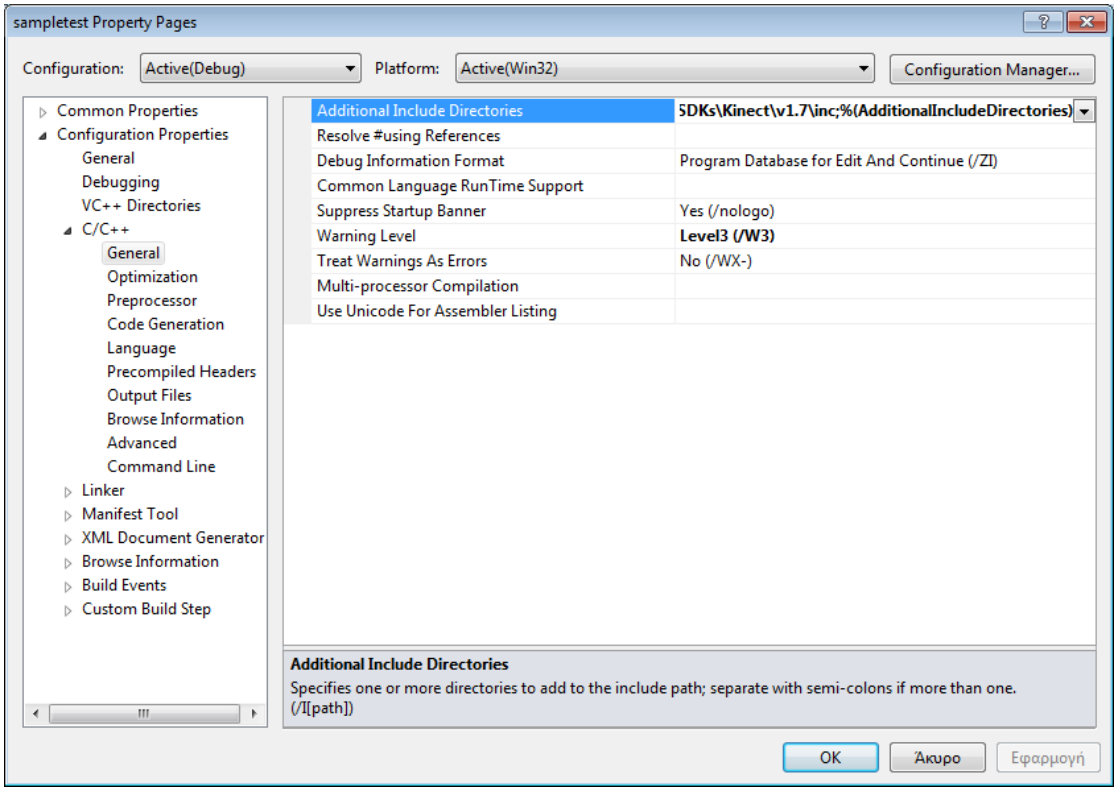
Στην επόμενη οθόνη επιλέγουμε το Console application και Precompiled header, όπως φαίνεται και παρακάτω:



4.2.3 Ρυθμίσεις Project

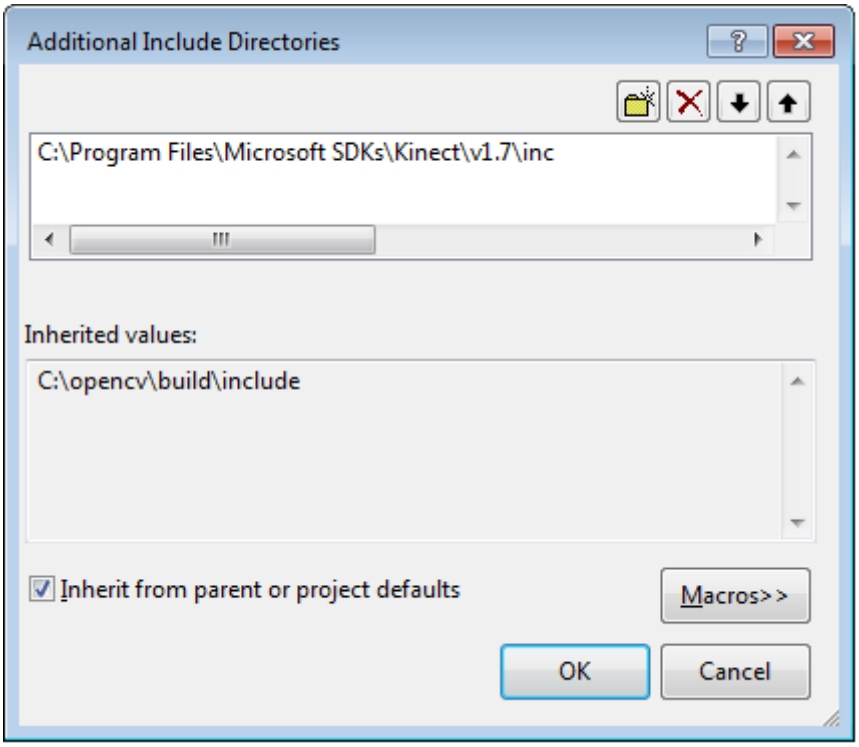
Στην συνέχεια θα προσπαθήσουμε να συνδέσουμε το Kinect sensor και το opencv με το project που δημιουργήσαμε, έτσι ώστε να μπορέσουμε να χρησιμοποιήσουμε τις κατάλληλες βιβλιοθήκες για το πρόγραμμα.

Πηγαίνοντας στις ιδιότητες του project που δημιουργήσαμε, στην καρτέλα C/C++ στο General και στο Additional Include Directories, όπως φαίνεται:



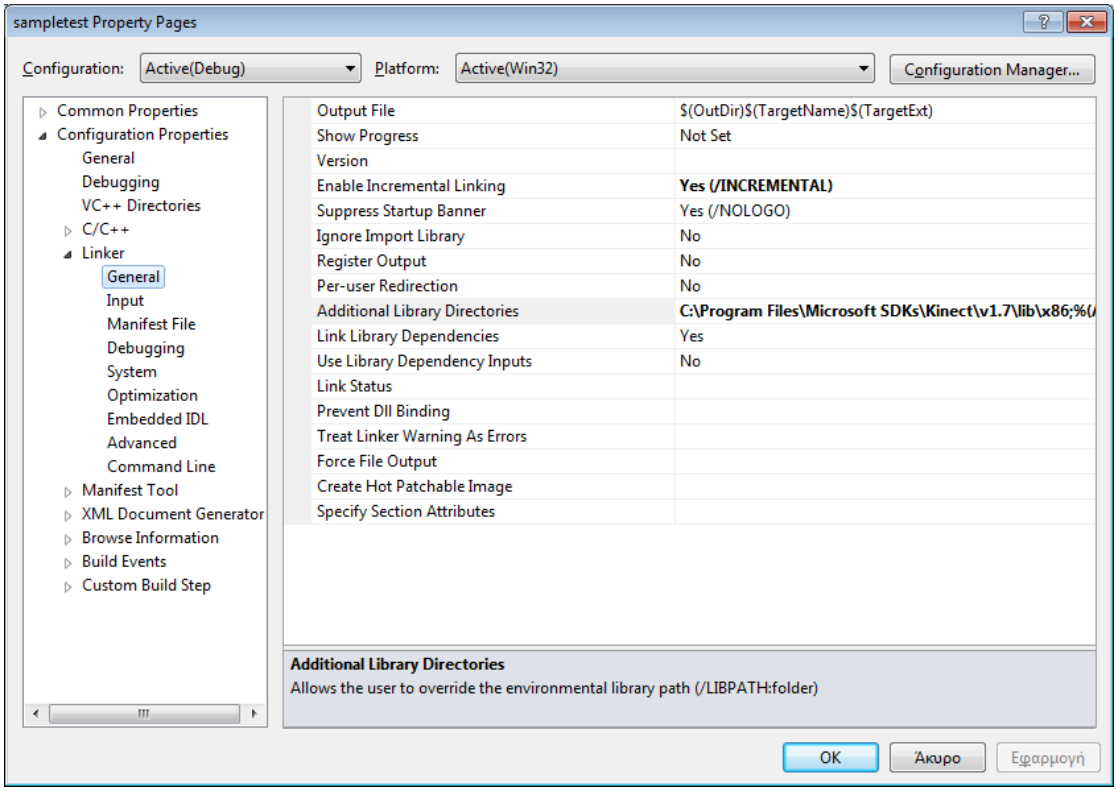
4.2.4 Συμπεριλανβάνουμε τα αρχεία KInect και Opencv

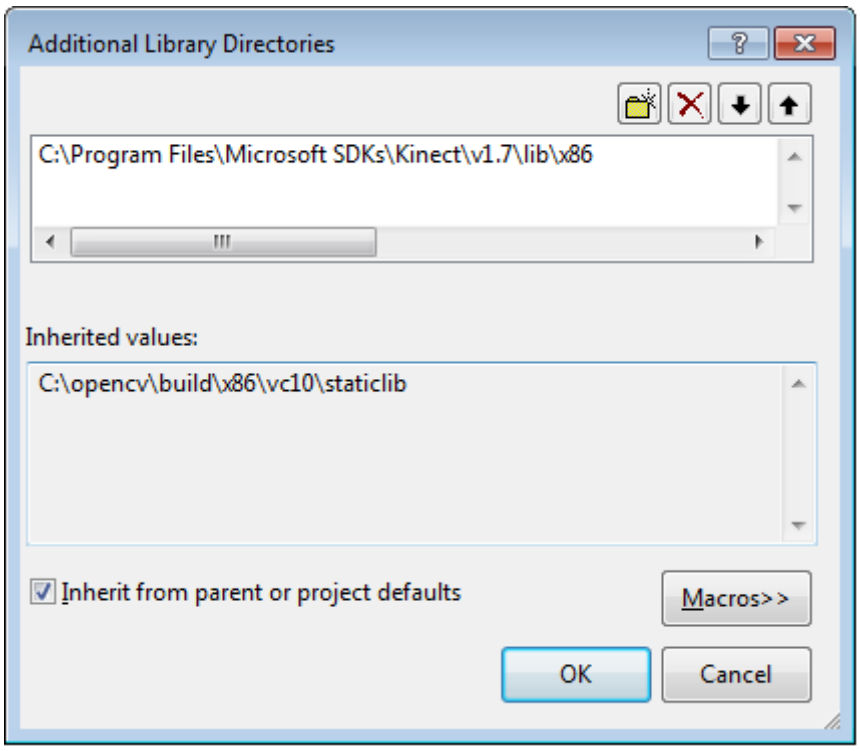
Από την επιλογή Edit, συμπληρώνουμε την διαδρομή για συμπεριλαμβανόμενα αρχεία του Kinect και του Opencv, στην περίπτωση μου, συμπλήρωσα τα εξής:



4.2.5 Συμπεριλανβάνουμε τα αρχεία KInect και Opencv

Έπειτα, στην καρτέλα Linker -> General, συμπληρώνουμε με τον ίδιο τρόπο την διαδρομή για τα αρχεία βιβλιοθηκών του Kinect και του Opencv, ώς εξής:



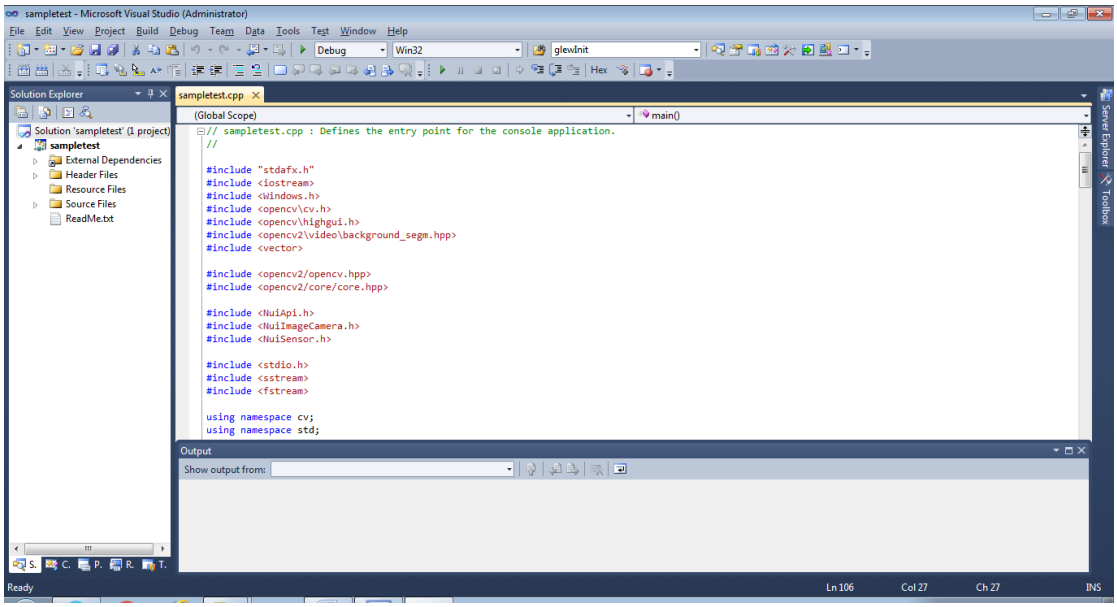


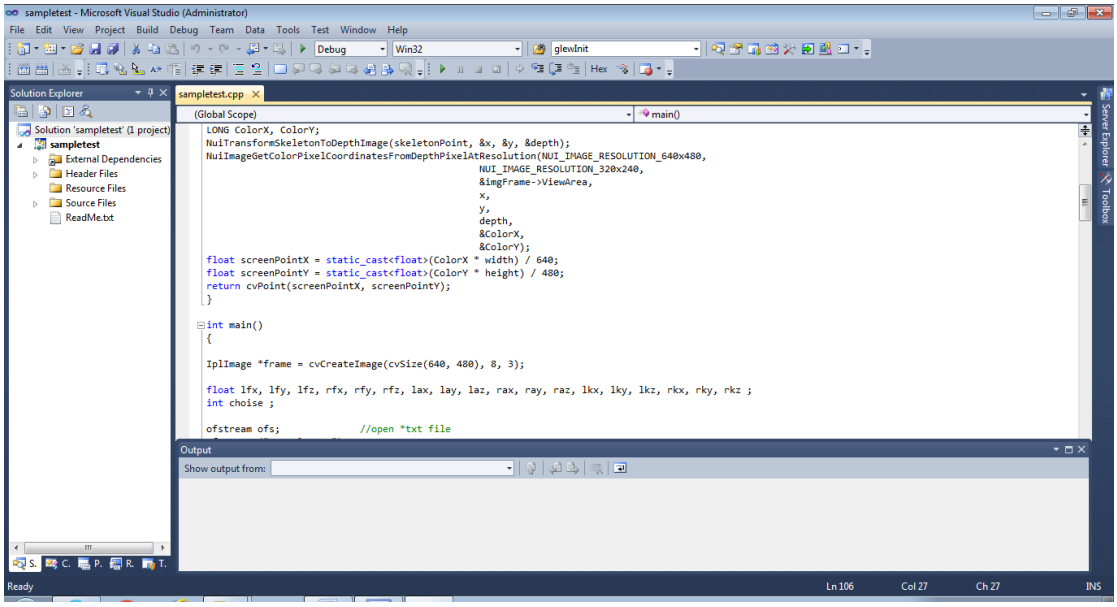
4.2.6 Συμπεριλανβάνουμε βιβλιοθήκες KInect και Opencv

Και τέλος, μετά από όλες αυτές τις ενέργειες αρχίζουμε την συγγραφή του κώδικα για την ολοκλήρωση του προγράμματος.

4.3 Η Εφαρμογή

Στιγμιότυπα του κώδικα:





4.3.1 Στιγμιότυπα του κώδικα

Οι βιβλιοθήκες που χρησιμοποιήθηκαν:

```
#include "stdafx.h"
#include <iostream>
#include <Windows.h>
#include <opencv\cv.h>
#include <opencv\highgui.h>
#include <opencv2\video\background_segm.hpp>
#include <vector>

#include <opencv2/opencv.hpp>
#include <opencv2/core/core.hpp>

#include <NuiApi.h>
#include <NuiImageCamera.h>
#include <NuiSensor.h>

#include <stdio.h>
#include <sstream>
#include <fstream>
```

Οι συναρτήσεις που χρησιμοποιήθηκαν:

NuiInitialize(NUI_INITIALIZE_FLAG_USES_SKELETON |NUI_INITIALIZE_FLAG_USES_COLOR);

Η συνάρτηση ενημερώνει το λογισμικό τι χρειαζόμαστε.

NUI_SKELETON_FRAME skeletonFrame = {0};
θα δημιουργήσει μια θέση στη μνήμη για να κρατήσει τα δεδομένα που το Kinect στέλνει σε μας.

NuiSkeletonGetNextFrame(0, &skeletonFrame);
Η NuiSkeletonGetNextFrame είναι μια λειτουργία που αποθηκεύει δεδομένα στη δεύτερη μεταβλητή. Η πρώτη μεταβλητή είναι ένα ρυθμιστικό, χιλιοστά του δευτερολέπτου για να περιμένετε, για το οποίο δεν χρειάζεται να ανησυχείτε αυτή τη στιγμή. Χρειάζεται και "& ourframe" στη δεύτερη μεταβλητή, έτσι ώστε το πρόγραμμα να πηγαίνει στην τοποθεσία της μεταβλητής αυτής και να αποθηκευεί δεδομένα. Το πλαίσιο θα είναι γεμάτο απο δεδομένα, ακόμα και αν αυτά είναι μεδέν.

```
NUI_SKELETON_TRACKING_STATE trackingState =
skeletonFrame.SkeletonData[i].eTrackingState;
        if (NUI_SKELETON_TRACKED == trackingState)
```

το ourframe είναι μια δομή με έναν αριθμό μελών, η σημαντικότερη εκ των οποίων είναι το μέλος "SkeletonData." Είναι μια σειρά από έξι

NUI_SKELETON_DATA δομών. Υπάρχουν έξι, επειδή το Kinect μπορεί να παρακολουθεί μέχρι και έξι ανθρώπους σε μια στιγμή. Κάθε ένα από αυτά έχει ένα μέλος,
NUI_SKELETON_TRACKING _STATE eTrackingState,
που είναι ένα από μια σειρά τιμών. Ο ένας που μας ενδιαφέρει είναι NUI_SKELETON_TRACKED, το οποίο μας λέει για εκείνα τα στοιχεία σχετικά με τη θέση του ατόμου σε αυτή την περίπτωση. Έτσι, για να τα τοποθετήσουμε όλα μαζί, θέλουμε να δούμε σε ourframe.SkeletonData [i] .eTrackingState και να δούμε αν αυτό είναι ίσο με NUI_SKELETON_TRACKED.

skeletonFrame.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITION_ANKLE_RIGHT].y ;

Εδώ, ψάχνουμε στο ίδιο μέρος, ourframe.SkeletonData [i], αλλά τώρα θέλουμε ένα διαφορετικό μέλος της SkeletonData. Μέσα στο SkeletonData είναι μια σειρά από 20 δομές Vector4, οι οποίες περιέχουν δεδομένα σχετικά με ένα σημείο σε 3D χώρο (x, y, z). Αυτός ο πίνακας είναι σε ευρετήριο του ενός αριθμητικού τύπου, NUI_SKELETON_POSITION_INDEX, το οποίο σας δίνει έναν εύκολο τρόπο να αναφερθούμε σε κάθε μία από κοινού. Εμείς απλά κοιτάζοντας το δεξί χέρι, NUI_SKELETON_POSITION_HAND_RIGHT, αυτή τη στιγμή, αλλά θα μπορούσαμε να το αλλάξουμε για οτιδήποτε στη λίστα. Προσθέτουμε ένα .y να πάρει το y συντεταγμένων, ή .x για τους χ, κλπ Γνωρίζοντας όλα αυτά, μπορούμε να το θέσουμε και πάλι μαζί για να πάρει τη γραμμή.

4.4 Όλος ο κώδικας τους προγράμματος

```
// sampletest.cpp : Defines the entry point for the console
application.
//

#include "stdafx.h"
#include <iostream>
#include <Windows.h>
#include <opencv\cv.h>
#include <opencv\highgui.h>
#include <opencv2\video\background_segm.hpp>
#include <vector>

#include <opencv2/opencv.hpp>
#include <opencv2/core/core.hpp>

#include <NuiApi.h>
#include <NuiImageCamera.h>
#include <NuiSensor.h>

#include <stdio.h>
#include <sstream>
#include <fstream>

using namespace cv;
using namespace std;

CvPoint SkeletonToScreen(Vector4 skeletonPoint, int width, int
height, const
NUI_IMAGE_FRAME *imgFrame)
{
LONG x, y;
USHORT depth;
LONG ColorX, ColorY;
NuiTransformSkeletonToDepthImage(skeletonPoint, &x, &y, &depth);
NuiImageGetColorPixelCoordinatesFromDepthPixelAtResolution(NUI_IMAGE
_RESOLUTION_640x480,

NUI_IMAGE_RESOLUTION_320x240,
                                                    &imgFrame->ViewArea,
                                                    x,
                                                    y,
                                                    depth,
                                                    &ColorX,
                                                    &ColorY);
float screenPointX = static_cast<float>(ColorX * width) / 640;
float screenPointY = static_cast<float>(ColorY * height) / 480;
return cvPoint(screenPointX, screenPointY);
}

int main()
{
```

```
IplImage *frame = cvCreateImage(cvSize(640, 480), 8, 3);

float lfx, lfy, lfz, rfx, rfy, rfz, lax, lay, laz, rax, ray, raz,
lkx, lky, lkz, rkx, rky, rkz ;
int choise ;

ofstream ofs;               //open *txt file
ofs.open ("example.txt");

HRESULT hr;
hr = NuiInitialize(NUI_INITIALIZE_FLAG_USES_SKELETON |
NUI_INITIALIZE_FLAG_USES_COLOR);  //Get Skeleton

if(hr != S_OK)
    return hr;

HANDLE SkeletonEvent = CreateEventW(NULL, TRUE, FALSE, NULL);
hr = NuiSkeletonTrackingEnable(SkeletonEvent, 0);
if(hr != S_OK)
    return hr;

HANDLE ColorEvent = CreateEventW(NULL, TRUE, FALSE, NULL);
HANDLE h2 = NULL;

hr = NuiImageStreamOpen(NUI_IMAGE_TYPE_COLOR,
NUI_IMAGE_RESOLUTION_640x480, 0, 2, ColorEvent, &h2);
if(hr != S_OK)
    return hr;

NUI_SKELETON_FRAME skeletonFrame = {0};  //Declare our frame

for (;;) {

      cout << "What joint do you want to track ?" << endl <<
                  "1. ANKLE" << endl <<
                  "2. KNEE" << endl <<
                  "3. FOOT" << endl <<
                  "4. ANKLE and KNEE" << endl <<
                  "5. ANKLE and FOOT" << endl <<
                  "6. KNEE and FOOT" << endl <<
                  "7. ANKLE and KNEE and FOOT" << endl ;   //Choose

      cin >> choise ;

      if ( choise == 1)
      {
            ofs << " Recorded Ankle " << endl << " Sample Frequency
(Hz) " << endl << " 100 Hz / 0,1 sec " << endl << endl ;

while (1) //For all of time
{

    WaitForSingleObject(SkeletonEvent, INFINITE);
```

```
      //get skeleton data
    hr = NuiSkeletonGetNextFrame(0, &skeletonFrame);    //Get a frame
and stuff it into ourframe
    if( hr != S_OK )
        continue;

    NuiTransformSmooth(&skeletonFrame, NULL);

    //get color image data
    const NUI_IMAGE_FRAME * pImageFrame = NULL;

      hr = NuiImageStreamGetNextFrame(h2, 0, &pImageFrame );

      if( hr != S_OK )
        continue;

    INuiFrameTexture *pTexture = pImageFrame->pFrameTexture;
    NUI_LOCKED_RECT LockedRect;
    pTexture->LockRect( 0, &LockedRect, NULL, 0 );

      //Draw skeleton points on frame
    for(int i=0; i<6; i++){
        NUI_SKELETON_TRACKING_STATE trackingState =
skeletonFrame.SkeletonData[i].eTrackingState;
        if (NUI_SKELETON_TRACKED == trackingState)


cvCircle(frame,SkeletonToScreen(skeletonFrame.SkeletonData[i].Skelet
onPositions[NUI_SKELETON_POSITION_ANKLE_RIGHT],640,
480,pImageFrame), 4, CV_RGB(255,0,0),2);

      cvCircle(frame,SkeletonToScreen(skeletonFrame.SkeletonData[i].
SkeletonPositions[NUI_SKELETON_POSITION_ANKLE_LEFT],640,
480,pImageFrame), 4, CV_RGB(151,13,220),2);

                        ray =
skeletonFrame.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITIO
N_ANKLE_RIGHT].y ;
                        lay =
skeletonFrame.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITIO
N_ANKLE_LEFT].y ;
                        rax =
skeletonFrame.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITIO
N_ANKLE_RIGHT].x ;
                        lax =
skeletonFrame.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITIO
N_ANKLE_LEFT].x ;
                        raz =
skeletonFrame.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITIO
N_ANKLE_RIGHT].z ;
                        laz =
skeletonFrame.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITIO
N_ANKLE_LEFT].z ;
```

```
                                        cout << "Right Ankle.y: " ;
                                        cout << ray << endl ;
                                        cout << "Left Ankle.y: " ;
                                        cout << lay << endl ;
                                        cout << "Right Ankle.x: " ;
                                        cout << rax << endl ;
                                        cout << "Left Ankle.x: " ;
                                        cout << lax << endl ;
                                        cout << "Right Ankle.z: " ;
                                        cout << raz << endl ;
                                        cout << "Left Ankle.z: " ;
                                        cout << laz << endl ;

                                        ofs << "Right Ankle.y: "
<< ray << endl ;
                                        ofs << "Left Ankle.y: "  <<
lay << endl ;
                                        ofs << "Right Ankle.x: "
<< rax << endl ;
                                        ofs << "Left Ankle.x: "  <<
lax << endl ;
                                        ofs << "Right Ankle.z: "
<< raz << endl ;
                                        ofs << "Left Ankle.y: "  <<
laz << endl ;
                                        ofs << endl ;
    }

       cvShowImage("Frame",frame);
    NuiImageStreamReleaseFrame( h2, pImageFrame );
    cvWaitKey(30);

      Sleep(100);

      }
}

      else if ( choise == 2)
      {

            ofs << " Recorded Knee " << endl << " Sample Frequency
(Hz) " << endl << " 100 Hz / 0,1 sec " << endl << endl ;

while(1)
{

    WaitForSingleObject(SkeletonEvent, INFINITE);
    //get skeleton data
    hr = NuiSkeletonGetNextFrame(0, &skeletonFrame);
    if( hr != S_OK )
        continue;
    NuiTransformSmooth(&skeletonFrame, NULL);
    //get color image data
```

```
    const NUI_IMAGE_FRAME * pImageFrame = NULL;
    hr = NuiImageStreamGetNextFrame(h2, 0, &pImageFrame );
    if( hr != S_OK )
        continue;

    INuiFrameTexture *pTexture = pImageFrame->pFrameTexture;
    NUI_LOCKED_RECT LockedRect;
    pTexture->LockRect( 0, &LockedRect, NULL, 0 );

      //Draw skeleton points on frame
    for(int i=0; i<6; i++){
        NUI_SKELETON_TRACKING_STATE trackingState =
skeletonFrame.SkeletonData[i].eTrackingState;
        if (NUI_SKELETON_TRACKED == trackingState)


cvCircle(frame,SkeletonToScreen(skeletonFrame.SkeletonData[i].Skelet
onPositions[NUI_SKELETON_POSITION_KNEE_RIGHT],640, 480,pImageFrame),
4, CV_RGB(255,0,0),2);

      cvCircle(frame,SkeletonToScreen(skeletonFrame.SkeletonData[i].
SkeletonPositions[NUI_SKELETON_POSITION_KNEE_LEFT],640,
480,pImageFrame), 4, CV_RGB(151,13,220),2);

                        rky =
skeletonFrame.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITIO
N_KNEE_RIGHT].y ;
                        lky =
skeletonFrame.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITIO
N_KNEE_LEFT].y ;
                        rkx =
skeletonFrame.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITIO
N_KNEE_RIGHT].x ;
                        lkx =
skeletonFrame.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITIO
N_KNEE_LEFT].x ;
                        rkz =
skeletonFrame.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITIO
N_KNEE_RIGHT].z ;
                        lkz =
skeletonFrame.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITIO
N_KNEE_LEFT].z ;

                                        cout << "Right Knee.y: " ;
                                        cout << rky << endl ;
                                        cout << "Left Knee.y: " ;
                                        cout << lky << endl ;
                                        cout << "Right Knee.x: " ;
                                        cout << rkx << endl ;
                                        cout << "Left Knee.x: " ;
                                        cout << lkx << endl ;
                                        cout << "Right Knee.z: " ;
                                        cout << rkz << endl ;
                                        cout << "Left Knee.z: " ;
```

```
                                                cout << lkz << endl ;

                                                ofs << "Right Knee.y: "  <<
rky << endl ;
                                                ofs << "Left Knee.y: "  <<
lky << endl ;
                                                ofs << "Right Knee.x: "  <<
rkx << endl ;
                                                ofs << "Left Knee.x: "  <<
lkx << endl ;
                                                ofs << "Right Knee.z: "  <<
rkz << endl ;
                                                ofs << "Left Knee.y: "  <<
lkz << endl ;
                                                ofs << endl ;

    }
    cvShowImage("Frame",frame);
    NuiImageStreamReleaseFrame( h2, pImageFrame );
    cvWaitKey(30);

      Sleep(100);

      }
}

      else if ( choise == 3)
      {

            ofs << " Recorded Foot " << endl << " Sample Frequency
(Hz) " << endl << " 100 Hz / 0,1 sec " << endl << endl ;

while(1)
{

    WaitForSingleObject(SkeletonEvent, INFINITE);
    //get skeleton data
    hr = NuiSkeletonGetNextFrame(0, &skeletonFrame);
    if( hr != S_OK )
        continue;
    NuiTransformSmooth(&skeletonFrame, NULL);
    //get color image data
    const NUI_IMAGE_FRAME * pImageFrame = NULL;
    hr = NuiImageStreamGetNextFrame(h2, 0, &pImageFrame );
    if( hr != S_OK )
        continue;

    INuiFrameTexture *pTexture = pImageFrame->pFrameTexture;
    NUI_LOCKED_RECT LockedRect;
    pTexture->LockRect( 0, &LockedRect, NULL, 0 );

      //Draw skeleton points on frame
    for(int i=0; i<6; i++){
```

```
        NUI_SKELETON_TRACKING_STATE trackingState =
skeletonFrame.SkeletonData[i].eTrackingState;
        if (NUI_SKELETON_TRACKED == trackingState)


cvCircle(frame,SkeletonToScreen(skeletonFrame.SkeletonData[i].Skelet
onPositions[NUI_SKELETON_POSITION_FOOT_RIGHT],640, 480,pImageFrame),
4, CV_RGB(255,0,0),2);

      cvCircle(frame,SkeletonToScreen(skeletonFrame.SkeletonData[i].
SkeletonPositions[NUI_SKELETON_POSITION_FOOT_LEFT],640,
480,pImageFrame), 4, CV_RGB(151,13,220),2);

                          rfy =
skeletonFrame.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITIO
N_FOOT_RIGHT].y ;
                          lfy =
skeletonFrame.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITIO
N_FOOT_LEFT].y ;
                          rfx =
skeletonFrame.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITIO
N_FOOT_RIGHT].x ;
                          lfx =
skeletonFrame.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITIO
N_FOOT_LEFT].x ;
                          rfz =
skeletonFrame.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITIO
N_FOOT_RIGHT].z ;
                          lfz =
skeletonFrame.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITIO
N_FOOT_LEFT].z ;

                                        cout << "Right Foot.y: " ;
                                              cout << rfy << endl ;
                                              cout << "Left Foot.y: " ;
                                              cout << lfy << endl ;
                                              cout << "Right Foot.x: " ;
                                              cout << rfx << endl ;
                                              cout << "Left Foot.x: " ;
                                              cout << lfx << endl ;
                                              cout << "Right Foot.z: " ;
                                              cout << rfz << endl ;
                                              cout << "Left Foot.z: " ;
                                              cout << lfz << endl ;

                                              ofs << "Right Foot.y: "  <<
rfy << endl ;
                                              ofs << "Left Foot.y: "  <<
lfy << endl ;
                                              ofs << "Right Foot.x: "  <<
rfx << endl ;
                                              ofs << "Left Foot.x: "  <<
lfx << endl ;
```

```
                                        ofs << "Right Foot.z: "  <<
rfz << endl ;
                                        ofs << "Left Foot.y: "  <<
lfz << endl ;
                                        ofs << endl ;
    }
    cvShowImage("Frame",frame);
    NuiImageStreamReleaseFrame( h2, pImageFrame );
    cvWaitKey(30);

      Sleep(100);

      }
}

else if ( choise == 4)
      {

            ofs << " Recorded Ankle and Knee " << endl << " Sample
Frequency (Hz) " << endl << " 100 Hz / 0,1 sec " << endl << endl ;

while(1)
{

    WaitForSingleObject(SkeletonEvent, INFINITE);
    //get skeleton data
    hr = NuiSkeletonGetNextFrame(0, &skeletonFrame);
    if( hr != S_OK )
        continue;
    NuiTransformSmooth(&skeletonFrame, NULL);
    //get color image data
    const NUI_IMAGE_FRAME * pImageFrame = NULL;
    hr = NuiImageStreamGetNextFrame(h2, 0, &pImageFrame );
    if( hr != S_OK )
        continue;

    INuiFrameTexture *pTexture = pImageFrame->pFrameTexture;
    NUI_LOCKED_RECT LockedRect;
    pTexture->LockRect( 0, &LockedRect, NULL, 0 );

      //Draw skeleton points on frame
    for(int i=0; i<6; i++){
        NUI_SKELETON_TRACKING_STATE trackingState =
skeletonFrame.SkeletonData[i].eTrackingState;
        if (NUI_SKELETON_TRACKED == trackingState)


cvCircle(frame,SkeletonToScreen(skeletonFrame.SkeletonData[i].Skelet
onPositions[NUI_SKELETON_POSITION_KNEE_RIGHT],640, 480,pImageFrame),
4, CV_RGB(255,0,0),2);

      cvCircle(frame,SkeletonToScreen(skeletonFrame.SkeletonData[i].
SkeletonPositions[NUI_SKELETON_POSITION_KNEE_LEFT],640,
480,pImageFrame), 4, CV_RGB(151,13,220),2);
```

```
      cvCircle(frame,SkeletonToScreen(skeletonFrame.SkeletonData[i].
SkeletonPositions[NUI_SKELETON_POSITION_ANKLE_RIGHT],640,
480,pImageFrame), 4, CV_RGB(255,0,0),2);

      cvCircle(frame,SkeletonToScreen(skeletonFrame.SkeletonData[i].
SkeletonPositions[NUI_SKELETON_POSITION_ANKLE_LEFT],640,
480,pImageFrame), 4, CV_RGB(151,13,220),2);

                        rky =
skeletonFrame.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITIO
N_KNEE_RIGHT].y ;
                        lky =
skeletonFrame.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITIO
N_KNEE_LEFT].y ;
                        rkx =
skeletonFrame.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITIO
N_KNEE_RIGHT].x ;
                        lkx =
skeletonFrame.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITIO
N_KNEE_LEFT].x ;
                        rkz =
skeletonFrame.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITIO
N_KNEE_RIGHT].z ;
                        lkz =
skeletonFrame.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITIO
N_KNEE_LEFT].z ;

                        ray =
skeletonFrame.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITIO
N_ANKLE_RIGHT].y ;
                        lay =
skeletonFrame.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITIO
N_ANKLE_LEFT].y ;
                        rax =
skeletonFrame.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITIO
N_ANKLE_RIGHT].x ;
                        lax =
skeletonFrame.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITIO
N_ANKLE_LEFT].x ;
                        raz =
skeletonFrame.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITIO
N_ANKLE_RIGHT].z ;
                        laz =
skeletonFrame.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITIO
N_ANKLE_LEFT].z ;

                                        cout << "Right Knee.y: " ;
                                        cout << rky << endl ;
                                        cout << "Left Knee.y: " ;
                                        cout << lky << endl ;
                                        cout << "Right Knee.x: " ;
                                        cout << rkx << endl ;
```

```
                                                cout << "Left Knee.x: " ;
                                                cout << lkx << endl ;
                                                cout << "Right Knee.z: " ;
                                                cout << rkz << endl ;
                                                cout << "Left Knee.z: " ;
                                                cout << lkz << endl ;

                                                cout << "Right Ankle.y: " ;
                                                cout << ray << endl ;
                                                cout << "Left Ankle.y: " ;
                                                cout << lay << endl ;
                                                cout << "Right Ankle.x: " ;
                                                cout << rax << endl ;
                                                cout << "Left Ankle.x: " ;
                                                cout << lax << endl ;
                                                cout << "Right Ankle.z: " ;
                                                cout << raz << endl ;
                                                cout << "Left Ankle.z: " ;
                                                cout << laz << endl ;

                                                ofs << "Right Knee.y: "  <<
rky << endl ;
                                                ofs << "Left Knee.y: "  <<
lky << endl ;
                                                ofs << "Right Knee.x: "  <<
rkx << endl ;
                                                ofs << "Left Knee.x: "  <<
lkx << endl ;
                                                ofs << "Right Knee.z: "  <<
rkz << endl ;
                                                ofs << "Left Knee.y: "  <<
lkz << endl ;
                                                ofs << endl ;


                                                ofs << "Right Ankle.y: "
<< ray << endl ;
                                                ofs << "Left Ankle.y: "  <<
lay << endl ;
                                                ofs << "Right Ankle.x: "
<< rax << endl ;
                                                ofs << "Left Ankle.x: "  <<
lax << endl ;
                                                ofs << "Right Ankle.z: "
<< raz << endl ;
                                                ofs << "Left Ankle.y: "  <<
laz << endl ;
                                                ofs << endl ;

    }
    cvShowImage("Frame",frame);
    NuiImageStreamReleaseFrame( h2, pImageFrame );
    cvWaitKey(30);

      Sleep(100);
```

```
      }
}

else if ( choise == 5)
      {

            ofs << " Recorded Ankle and Foot " << endl << " Sample
Frequency (Hz) " << endl << " 100 Hz / 0,1 sec " << endl << endl ;

while(1)
{

    WaitForSingleObject(SkeletonEvent, INFINITE);
    //get skeleton data
    hr = NuiSkeletonGetNextFrame(0, &skeletonFrame);
    if( hr != S_OK )
        continue;
    NuiTransformSmooth(&skeletonFrame, NULL);
    //get color image data
    const NUI_IMAGE_FRAME * pImageFrame = NULL;
    hr = NuiImageStreamGetNextFrame(h2, 0, &pImageFrame );
    if( hr != S_OK )
        continue;

    INuiFrameTexture *pTexture = pImageFrame->pFrameTexture;
    NUI_LOCKED_RECT LockedRect;
    pTexture->LockRect( 0, &LockedRect, NULL, 0 );

      //Draw skeleton points on frame
    for(int i=0; i<6; i++){
        NUI_SKELETON_TRACKING_STATE trackingState =
skeletonFrame.SkeletonData[i].eTrackingState;
        if (NUI_SKELETON_TRACKED == trackingState)


cvCircle(frame,SkeletonToScreen(skeletonFrame.SkeletonData[i].Skelet
onPositions[NUI_SKELETON_POSITION_FOOT_RIGHT],640, 480,pImageFrame),
4, CV_RGB(255,0,0),2);

      cvCircle(frame,SkeletonToScreen(skeletonFrame.SkeletonData[i].
SkeletonPositions[NUI_SKELETON_POSITION_FOOT_LEFT],640,
480,pImageFrame), 4, CV_RGB(151,13,220),2);


      cvCircle(frame,SkeletonToScreen(skeletonFrame.SkeletonData[i].
SkeletonPositions[NUI_SKELETON_POSITION_ANKLE_RIGHT],640,
480,pImageFrame), 4, CV_RGB(255,0,0),2);

      cvCircle(frame,SkeletonToScreen(skeletonFrame.SkeletonData[i].
SkeletonPositions[NUI_SKELETON_POSITION_ANKLE_LEFT],640,
480,pImageFrame), 4, CV_RGB(151,13,220),2);
```

```
                        rfy =
skeletonFrame.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITIO
N_FOOT_RIGHT].y ;
                        lfy =
skeletonFrame.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITIO
N_FOOT_LEFT].y ;
                        rfx =
skeletonFrame.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITIO
N_FOOT_RIGHT].x ;
                        lfx =
skeletonFrame.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITIO
N_FOOT_LEFT].x ;
                        rfz =
skeletonFrame.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITIO
N_FOOT_RIGHT].z ;
                        lfz =
skeletonFrame.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITIO
N_FOOT_LEFT].z ;

                        ray =
skeletonFrame.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITIO
N_ANKLE_RIGHT].y ;
                        lay =
skeletonFrame.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITIO
N_ANKLE_LEFT].y ;
                        rax =
skeletonFrame.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITIO
N_ANKLE_RIGHT].x ;
                        lax =
skeletonFrame.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITIO
N_ANKLE_LEFT].x ;
                        raz =
skeletonFrame.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITIO
N_ANKLE_RIGHT].z ;
                        laz =
skeletonFrame.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITIO
N_ANKLE_LEFT].z ;

                                        cout << "Right Foot.y: " ;
                                        cout << rfy << endl ;
                                        cout << "Left Foot.y: " ;
                                        cout << lfy << endl ;
                                        cout << "Right Foot.x: " ;
                                        cout << rfx << endl ;
                                        cout << "Left Foot.x: " ;
                                        cout << lfx << endl ;
                                        cout << "Right Foot.z: " ;
                                        cout << rfz << endl ;
                                        cout << "Left Foot.z: " ;
                                        cout << lfz << endl ;

                                        cout << "Right Ankle.y: " ;
                                        cout << ray << endl ;
                                        cout << "Left Ankle.y: " ;
```

```
                                        cout << lay << endl ;
                                        cout << "Right Ankle.x: " ;
                                        cout << rax << endl ;
                                        cout << "Left Ankle.x: " ;
                                        cout << lax << endl ;
                                        cout << "Right Ankle.z: " ;
                                        cout << raz << endl ;
                                        cout << "Left Ankle.z: " ;
                                        cout << laz << endl ;

                                        ofs << "Right Foot.y: "  <<
rfy << endl ;
                                        ofs << "Left Foot.y: "  <<
lfy << endl ;
                                        ofs << "Right Foot.x: "  <<
rfx << endl ;
                                        ofs << "Left Foot.x: "  <<
lfx << endl ;
                                        ofs << "Right Foot.z: "  <<
rfz << endl ;
                                        ofs << "Left Foot.y: "  <<
lfz << endl ;
                                        ofs << endl ;


                                        ofs << "Right Ankle.y: "
<< ray << endl ;
                                        ofs << "Left Ankle.y: "  <<
lay << endl ;
                                        ofs << "Right Ankle.x: "
<< rax << endl ;
                                        ofs << "Left Ankle.x: "  <<
lax << endl ;
                                        ofs << "Right Ankle.z: "
<< raz << endl ;
                                        ofs << "Left Ankle.y: "  <<
laz << endl ;
                                        ofs << endl ;

    }
    cvShowImage("Frame",frame);
    NuiImageStreamReleaseFrame( h2, pImageFrame );
    cvWaitKey(30);

      Sleep(100);

      }
}

else if ( choise == 6)
      {

            ofs << " Recorded Knee and Foot " << endl << " Sample
Frequency (Hz) " << endl << " 100 Hz / 0,1 sec " << endl << endl ;
```

```
while(1)
{

    WaitForSingleObject(SkeletonEvent, INFINITE);
    //get skeleton data
    hr = NuiSkeletonGetNextFrame(0, &skeletonFrame);
    if( hr != S_OK )
        continue;
    NuiTransformSmooth(&skeletonFrame, NULL);
    //get color image data
    const NUI_IMAGE_FRAME * pImageFrame = NULL;
    hr = NuiImageStreamGetNextFrame(h2, 0, &pImageFrame );
    if( hr != S_OK )
        continue;

    INuiFrameTexture *pTexture = pImageFrame->pFrameTexture;
    NUI_LOCKED_RECT LockedRect;
    pTexture->LockRect( 0, &LockedRect, NULL, 0 );

      //Draw skeleton points on frame
    for(int i=0; i<6; i++){
        NUI_SKELETON_TRACKING_STATE trackingState =
skeletonFrame.SkeletonData[i].eTrackingState;
        if (NUI_SKELETON_TRACKED == trackingState)


cvCircle(frame,SkeletonToScreen(skeletonFrame.SkeletonData[i].Skelet
onPositions[NUI_SKELETON_POSITION_FOOT_RIGHT],640, 480,pImageFrame),
4, CV_RGB(255,0,0),2);

      cvCircle(frame,SkeletonToScreen(skeletonFrame.SkeletonData[i].
SkeletonPositions[NUI_SKELETON_POSITION_FOOT_LEFT],640,
480,pImageFrame), 4, CV_RGB(151,13,220),2);


      cvCircle(frame,SkeletonToScreen(skeletonFrame.SkeletonData[i].
SkeletonPositions[NUI_SKELETON_POSITION_KNEE_RIGHT],640,
480,pImageFrame), 4, CV_RGB(255,0,0),2);

      cvCircle(frame,SkeletonToScreen(skeletonFrame.SkeletonData[i].
SkeletonPositions[NUI_SKELETON_POSITION_KNEE_LEFT],640,
480,pImageFrame), 4, CV_RGB(151,13,220),2);

                        rfy =
skeletonFrame.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITIO
N_FOOT_RIGHT].y ;
                        lfy =
skeletonFrame.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITIO
N_FOOT_LEFT].y ;
                        rfx =
skeletonFrame.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITIO
N_FOOT_RIGHT].x ;
```

```
                        lfx =
skeletonFrame.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITIO
N_FOOT_LEFT].x ;
                        rfz =
skeletonFrame.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITIO
N_FOOT_RIGHT].z ;
                        lfz =
skeletonFrame.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITIO
N_FOOT_LEFT].z ;

                        rky =
skeletonFrame.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITIO
N_KNEE_RIGHT].y ;
                        lky =
skeletonFrame.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITIO
N_KNEE_LEFT].y ;
                        rkx =
skeletonFrame.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITIO
N_KNEE_RIGHT].x ;
                        lkx =
skeletonFrame.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITIO
N_KNEE_LEFT].x ;
                        rkz =
skeletonFrame.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITIO
N_KNEE_RIGHT].z ;
                        lkz =
skeletonFrame.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITIO
N_KNEE_LEFT].z ;

                                        cout << "Right Foot.y: " ;
                                        cout << rfy << endl ;
                                        cout << "Left Foot.y: " ;
                                        cout << lfy << endl ;
                                        cout << "Right Foot.x: " ;
                                        cout << rfx << endl ;
                                        cout << "Left Foot.x: " ;
                                        cout << lfx << endl ;
                                        cout << "Right Foot.z: " ;
                                        cout << rfz << endl ;
                                        cout << "Left Foot.z: " ;
                                        cout << lfz << endl ;

                                        cout << "Right Knee.y: " ;
                                        cout << rky << endl ;
                                        cout << "Left Knee.y: " ;
                                        cout << lky << endl ;
                                        cout << "Right Knee.x: " ;
                                        cout << rkx << endl ;
                                        cout << "Left Knee.x: " ;
                                        cout << lkx << endl ;
                                        cout << "Right Knee.z: " ;
                                        cout << rkz << endl ;
                                        cout << "Left Knee.z: " ;
                                        cout << lkz << endl ;
```

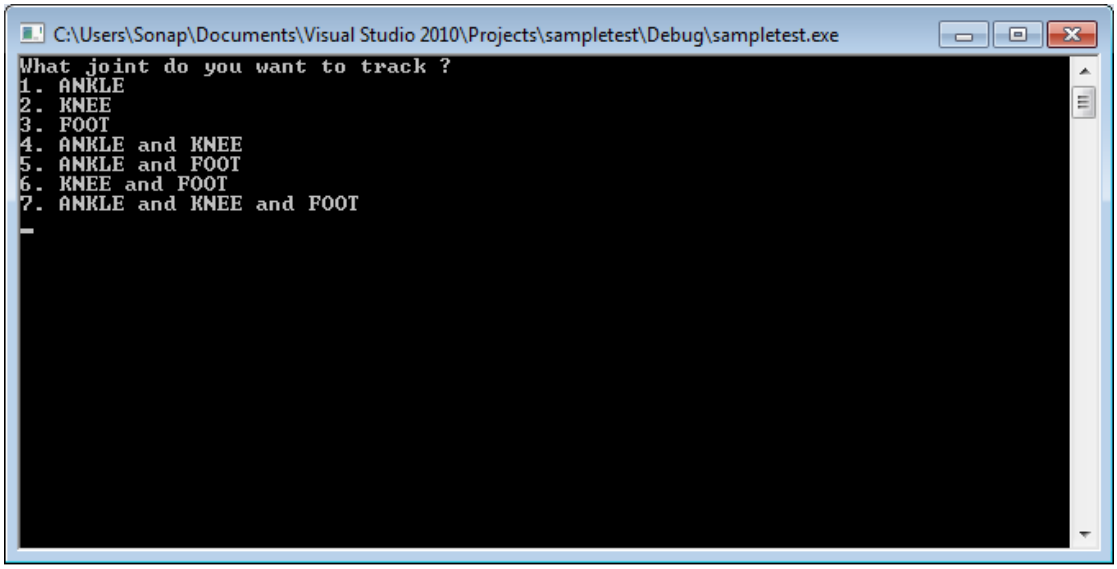
```
                                                ofs << "Right Foot.y: "  <<
rfy << endl ;
                                                ofs << "Left Foot.y: "  <<
lfy << endl ;
                                                ofs << "Right Foot.x: "  <<
rfx << endl ;
                                                ofs << "Left Foot.x: "  <<
lfx << endl ;
                                                ofs << "Right Foot.z: "  <<
rfz << endl ;
                                                ofs << "Left Foot.y: "  <<
lfz << endl ;
                                                ofs << endl ;


                                                ofs << "Right Knee.y: "  <<
rky << endl ;
                                                ofs << "Left Knee.y: "  <<
lky << endl ;
                                                ofs << "Right Knee.x: "  <<
rkx << endl ;
                                                ofs << "Left Knee.x: "  <<
lkx << endl ;
                                                ofs << "Right Knee.z: "  <<
rkz << endl ;
                                                ofs << "Left Knee.y: "  <<
lkz << endl ;
                                                ofs << endl ;

    }
    cvShowImage("Frame",frame);
    NuiImageStreamReleaseFrame( h2, pImageFrame );
    cvWaitKey(30);

      Sleep(100);

      }
}

else if ( choise == 7)
      {

            ofs << " Recorded Knee and Foot " << endl << " Sample
Frequency (Hz) " << endl << " 100 Hz / 0,1 sec " << endl << endl ;

while(1)
{

    WaitForSingleObject(SkeletonEvent, INFINITE);
    //get skeleton data
    hr = NuiSkeletonGetNextFrame(0, &skeletonFrame);
    if( hr != S_OK )
        continue;
    NuiTransformSmooth(&skeletonFrame, NULL);
```

```
    //get color image data
    const NUI_IMAGE_FRAME * pImageFrame = NULL;
    hr = NuiImageStreamGetNextFrame(h2, 0, &pImageFrame );
    if( hr != S_OK )
        continue;

    INuiFrameTexture *pTexture = pImageFrame->pFrameTexture;
    NUI_LOCKED_RECT LockedRect;
    pTexture->LockRect( 0, &LockedRect, NULL, 0 );

      //Draw skeleton points on frame
    for(int i=0; i<6; i++){
        NUI_SKELETON_TRACKING_STATE trackingState =
skeletonFrame.SkeletonData[i].eTrackingState;
        if (NUI_SKELETON_TRACKED == trackingState)


cvCircle(frame,SkeletonToScreen(skeletonFrame.SkeletonData[i].Skelet
onPositions[NUI_SKELETON_POSITION_FOOT_RIGHT],640, 480,pImageFrame),
4, CV_RGB(255,0,0),2);

      cvCircle(frame,SkeletonToScreen(skeletonFrame.SkeletonData[i].
SkeletonPositions[NUI_SKELETON_POSITION_FOOT_LEFT],640,
480,pImageFrame), 4, CV_RGB(151,13,220),2);


      cvCircle(frame,SkeletonToScreen(skeletonFrame.SkeletonData[i].
SkeletonPositions[NUI_SKELETON_POSITION_KNEE_RIGHT],640,
480,pImageFrame), 4, CV_RGB(255,0,0),2);

      cvCircle(frame,SkeletonToScreen(skeletonFrame.SkeletonData[i].
SkeletonPositions[NUI_SKELETON_POSITION_KNEE_LEFT],640,
480,pImageFrame), 4, CV_RGB(151,13,220),2);


      cvCircle(frame,SkeletonToScreen(skeletonFrame.SkeletonData[i].
SkeletonPositions[NUI_SKELETON_POSITION_ANKLE_RIGHT],640,
480,pImageFrame), 4, CV_RGB(255,0,0),2);

      cvCircle(frame,SkeletonToScreen(skeletonFrame.SkeletonData[i].
SkeletonPositions[NUI_SKELETON_POSITION_ANKLE_LEFT],640,
480,pImageFrame), 4, CV_RGB(151,13,220),2);

                        rfy =
skeletonFrame.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITIO
N_FOOT_RIGHT].y ;
                        lfy =
skeletonFrame.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITIO
N_FOOT_LEFT].y ;
                        rfx =
skeletonFrame.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITIO
N_FOOT_RIGHT].x ;
```

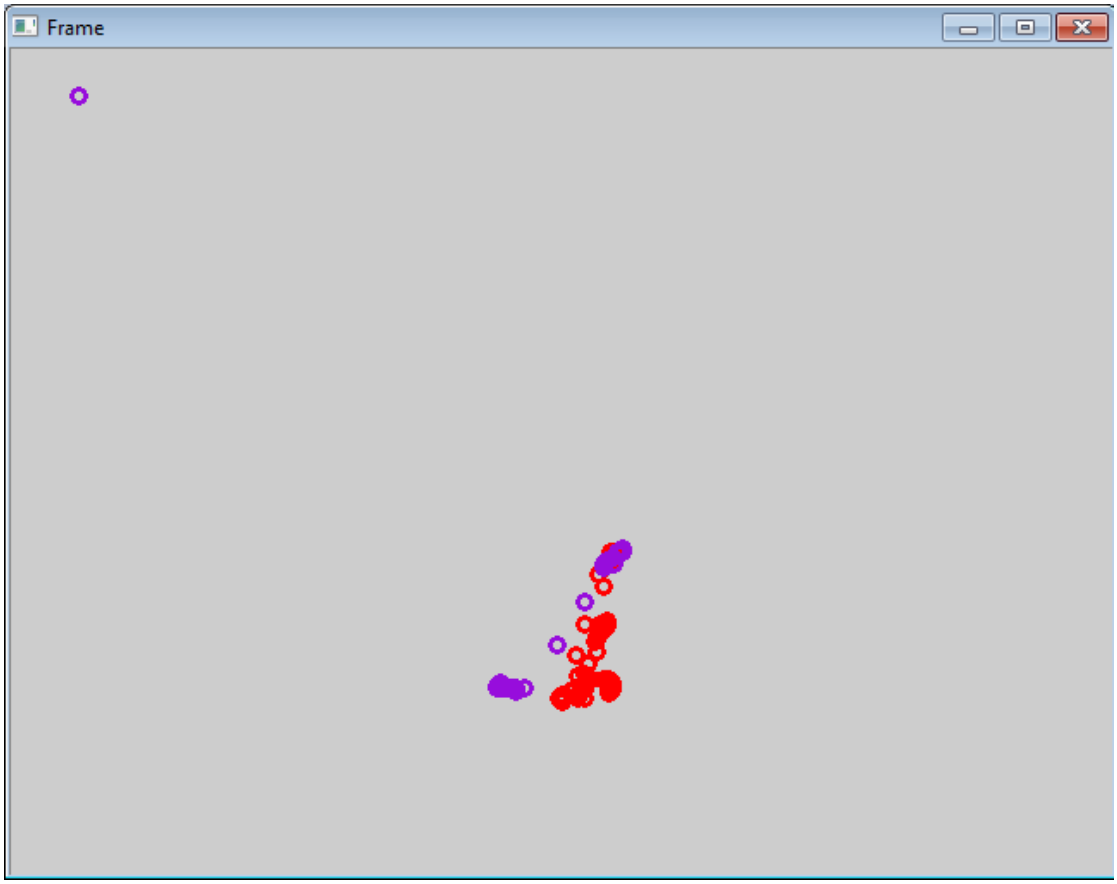
```
                        lfx =
skeletonFrame.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITIO
N_FOOT_LEFT].x ;
                        rfz =
skeletonFrame.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITIO
N_FOOT_RIGHT].z ;
                        lfz =
skeletonFrame.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITIO
N_FOOT_LEFT].z ;

                        rky =
skeletonFrame.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITIO
N_KNEE_RIGHT].y ;
                        lky =
skeletonFrame.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITIO
N_KNEE_LEFT].y ;
                        rkx =
skeletonFrame.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITIO
N_KNEE_RIGHT].x ;
                        lkx =
skeletonFrame.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITIO
N_KNEE_LEFT].x ;
                        rkz =
skeletonFrame.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITIO
N_KNEE_RIGHT].z ;
                        lkz =
skeletonFrame.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITIO
N_KNEE_LEFT].z ;

                        ray =
skeletonFrame.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITIO
N_ANKLE_RIGHT].y ;
                        lay =
skeletonFrame.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITIO
N_ANKLE_LEFT].y ;
                        rax =
skeletonFrame.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITIO
N_ANKLE_RIGHT].x ;
                        lax =
skeletonFrame.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITIO
N_ANKLE_LEFT].x ;
                        raz =
skeletonFrame.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITIO
N_ANKLE_RIGHT].z ;
                        laz =
skeletonFrame.SkeletonData[i].SkeletonPositions[NUI_SKELETON_POSITIO
N_ANKLE_LEFT].z ;

                                                cout << "Right Foot.y: " ;
                                                cout << rfy << endl ;
                                                cout << "Left Foot.y: " ;
                                                cout << lfy << endl ;
                                                cout << "Right Foot.x: " ;
                                                cout << rfx << endl ;
```

```
                                        cout << "Left Foot.x: " ;
                                        cout << lfx << endl ;
                                        cout << "Right Foot.z: " ;
                                        cout << rfz << endl ;
                                        cout << "Left Foot.z: " ;
                                        cout << lfz << endl ;

                                        cout << "Right Knee.y: " ;
                                        cout << rky << endl ;
                                        cout << "Left Knee.y: " ;
                                        cout << lky << endl ;
                                        cout << "Right Knee.x: " ;
                                        cout << rkx << endl ;
                                        cout << "Left Knee.x: " ;
                                        cout << lkx << endl ;
                                        cout << "Right Knee.z: " ;
                                        cout << rkz << endl ;
                                        cout << "Left Knee.z: " ;
                                        cout << lkz << endl ;

                                        cout << "Right Ankle.y: " ;
                                        cout << ray << endl ;
                                        cout << "Left Ankle.y: " ;
                                        cout << lay << endl ;
                                        cout << "Right Ankle.x: " ;
                                        cout << rax << endl ;
                                        cout << "Left Ankle.x: " ;
                                        cout << lax << endl ;
                                        cout << "Right Ankle.z: " ;
                                        cout << raz << endl ;
                                        cout << "Left Ankle.z: " ;
                                        cout << laz << endl ;

                                        ofs << "Right Foot.y: "  <<
rfy << endl ;

                                        ofs << "Left Foot.y: "  <<
lfy << endl ;

                                        ofs << "Right Foot.x: "  <<
rfx << endl ;

                                        ofs << "Left Foot.x: "  <<
lfx << endl ;

                                        ofs << "Right Foot.z: "  <<
rfz << endl ;

                                        ofs << "Left Foot.y: "  <<
lfz << endl ;

                                        ofs << endl ;


                                        ofs << "Right Knee.y: "  <<
rky << endl ;

                                        ofs << "Left Knee.y: "  <<
lky << endl ;

                                        ofs << "Right Knee.x: "  <<
rkx << endl ;
```

```
                                                ofs << "Left Knee.x: "  <<
lkx << endl ;
                                                ofs << "Right Knee.z: "  <<
rkz << endl ;
                                                ofs << "Left Knee.y: "  <<
lkz << endl ;
                                                ofs << endl ;


                                                ofs << "Right Ankle.y: "
<< ray << endl ;
                                                ofs << "Left Ankle.y: "  <<
lay << endl ;
                                                ofs << "Right Ankle.x: "
<< rax << endl ;
                                                ofs << "Left Ankle.x: "  <<
lax << endl ;
                                                ofs << "Right Ankle.z: "
<< raz << endl ;
                                                ofs << "Left Ankle.y: "  <<
laz << endl ;
                                                ofs << endl ;

    }
    cvShowImage("Frame",frame);
    NuiImageStreamReleaseFrame( h2, pImageFrame );
    cvWaitKey(30);

      Sleep(100);

      }
}

else
{
      cout << "Press a number between 1-7!" << endl;
      continue;
}

}

      NuiShutdown();  //shut down the Kinect

      ofs.close();

      return 0;
}
```

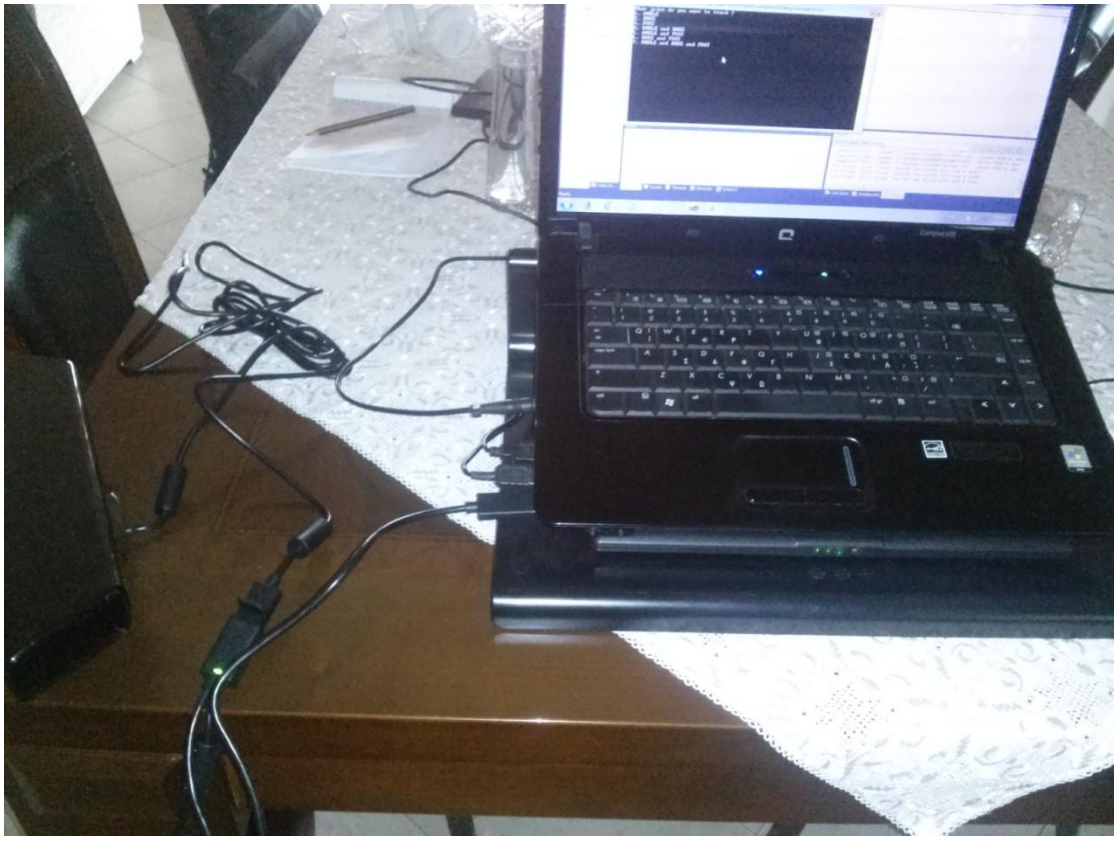


4.3.2 Στιγμιότυπο της εφαρμογής

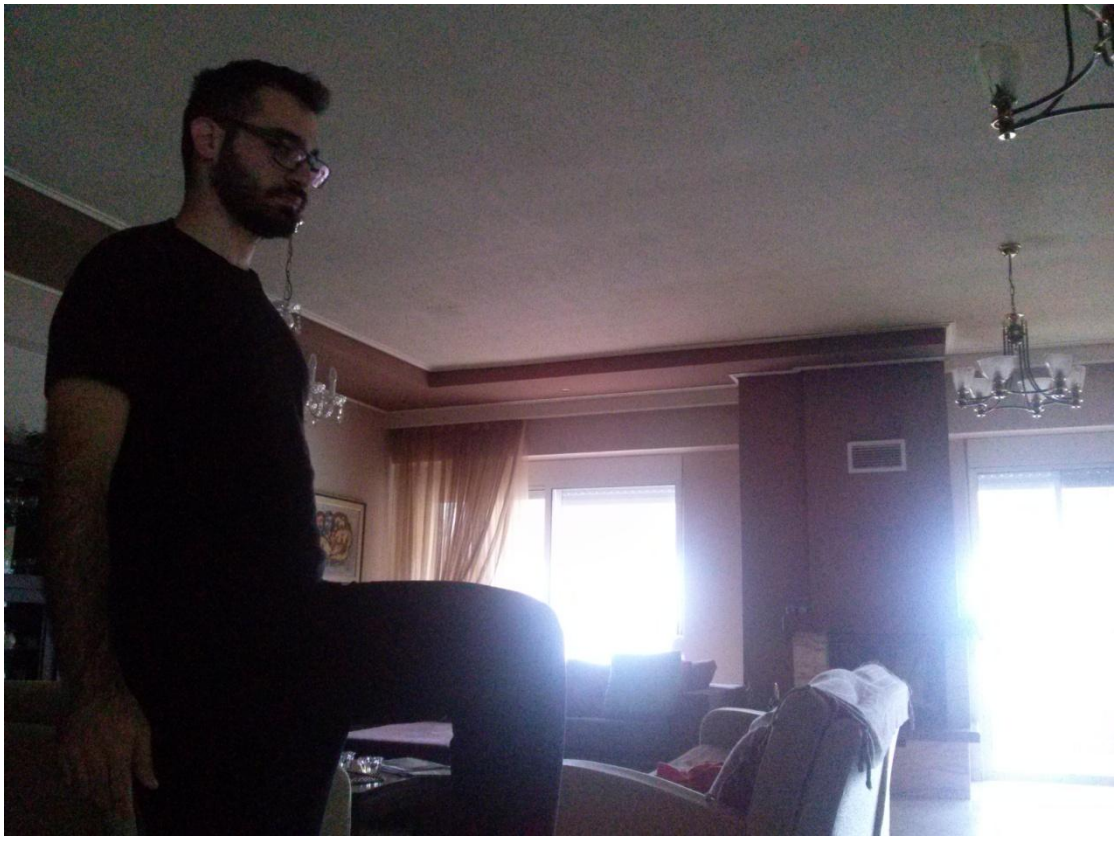


4.3.3 Στιγμιότυπο της εφαρμογής

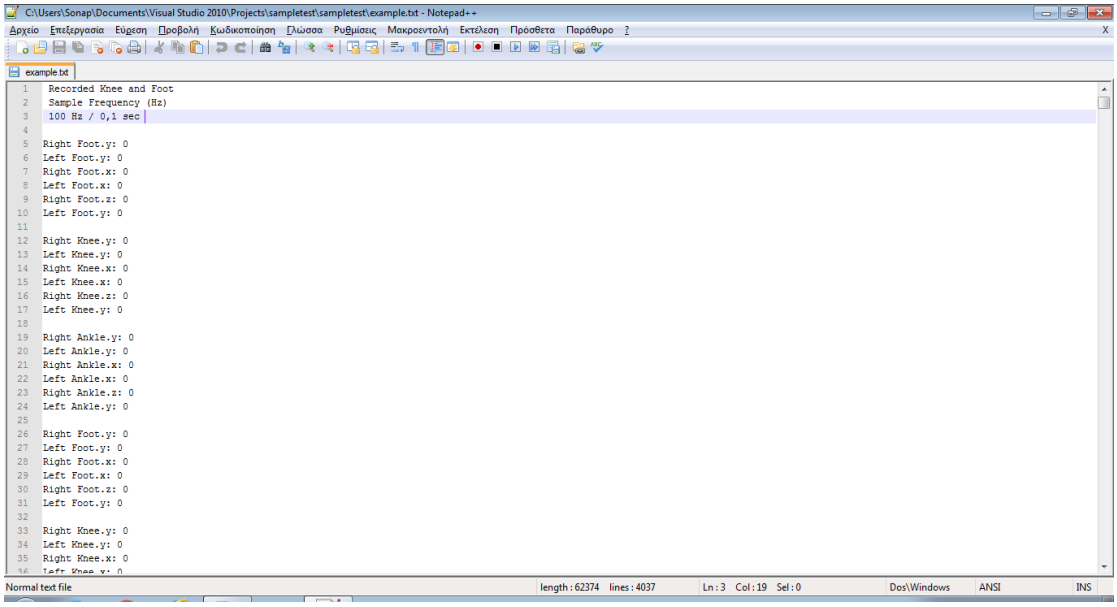
4.3.4 Στιγμιότυπο της εφαρμογής



4.3.5 Στιγμιότυπο της εφαρμογής



4.3.6 Στιγμιότυπο της εφαρμογής



4.3.7 Στιγμιότυπο της εφαρμογής

5.Άλλες Εφαρμογές με το Kinect

Το Kinect εκτός απο τη λειτουργία και τη χρησιμότητά του στις ηλεκτρονικές κονσόλες (όπως xBox) έχει και άλλες δυνατότητες. Με το Kinect μπορούμε να δημιουργήσουμε πάρα πολλές εφαρμογές, σε διάφορες γλώσσες προγραμματισμού. Κάποιες απο αυτές θα σας τις αναφέρω παρακάτω:

- Ανίχνευση χεριών με OpenCV, το Kinect SDK και OpenGL
  Παράθυρο σφαίρα δείχνει την 3D αναπαράσταση του χεριού

- Eyetracking με Facetracklib χρησιμοποιώντας το Kinect (Kinect SDK από 1,7)
  Επεξεργάζεται την eyetracking χρησιμοποιώντας αναφορά Kinect

- Δοκιμές Face Tracking και αναγνώριση χειρονομιών με το νέο Kinect SDK σε Epitech!

- Εντοπισμός αντικειμένων χρησιμοποιώντας Goblin XNA και το Kinect SDK

- Χέρι στο ποντίκι χρησιμοποιώντας το Kinect SDK / OpenCV

- Φωτιστικά εφέ χρησιμοποιώντας το Kinect SDK / OpenCV

- Μετακίνηση χεριού χρησιμοποιώντας cvDistTransform (με βάση το Kinect SDK / OpenCV)

- Ανίχνευση χεριού βασισμένο σε άμορφη μάζα (Kinect SDK και OpenCV)

- Τμηματοποίηση χρώματος χρησιμοποιώντας το Kinect SDK / OpenCV

- Προγραμματισμός παιχνιδιού με Micorosoft Kinect SDK

- Kinect εγγραφής βάθους στην εικόνα
- Kinect SDK Point Cloud
- Παρακολούθηση δαχτύλων με Kinect SDK
- Mocap Animation συστήματος χρησιμοποιώντας το Microsoft Kinect SDK

6. Επίλογος

Η εργασία επικεντρώνεται στην καταγραφή βημάτων αλλά το Kinect είναι ένα εργαλείο με το οποίο μπορούμε να δημιουργήσουμε πολλές και διάφορες εφαρμογές. Είναι ένα εργαλείο χωρίς όρια και περιορισμούς, έτσι με πολύ μελέτη και εργασία μπορούν να γίνουν πολλά πράγματα.

Τέλος, θα ήθελα να κλείσω αναφέροντας τις προσωπικές μου εμπειρίες κατα τη διάρκεια της διπλωματικής αυτής εργασίας. Αρχικά θα ήθελα να αναφερθώ στην πρώτη επαφή που είχα με τον αισθητήρα Kinect και ποσό ενδιαφέρον ήταν το να μάθω πως λειτουργεί, τη χρησιμότητα του και τις δυνατότητες του. Επείτα θα ήθελα να αναφερθώ και στην γλώσσα προγραμματισμού C++, που μέσα απο την εργασία βελτίωσα τις γνώσεις μου πανω σε αυτή.

Βιβλιογραφία

- http://docs.opencv.org/doc/tutorials/introduction/windows_visual_studio_Opencv/windows_visual_studio_Opencv.html
- https://msdn.microsoft.com/en-us/library/hh973085.aspx
- https://msdn.microsoft.com/en-us/library/jj131025.aspx
- https://msdn.microsoft.com/en-us/library/hh855386.aspx
- http://zugara.com/how-does-the-kinect-2-compare-to-the-kinect-1
- https://en.wikipedia.org/wiki/Kinect
- https://en.wikipedia.org/wiki/Microsoft_Visual_Studio
- http://www.microsoft.com/en-us/download/confirmation.aspx?id=36996
- https://msdn.microsoft.com/en-us/library/hh855356.aspx
- http://www.cs.princeton.edu/~edwardz/tutorials/kinect/kinect0_sdl.html
- http://rozzles.com/kinect-sdk-tutorial/
- http://meta-guide.com/videography/100-best-kinect-sdk-videos/
- https://el.wikipedia.org/wiki/C%2B%2B
- https://electroniclunch.wordpress.com/2013/01/14/kinect-tutorial-c/
- https://www.visualstudio.com/en-us/downloads#d-2010-express
- Kinect for Windows SDK Programming Guide, Abhijit Jana
- Learn the Kinect APIs, Rob Miles
- How does the Kinect work? , John MacCormick
- Programming with the Kinect for Windows – Software Development Kit, David Catuhe
- Beginning Kinect Programming with the Microsoft Kinect SDK, Jarrett Webb and James Ashley