Ευχαριστίες

Θα ήθελα να ευχαριστήσω την οικογένεια και τους φίλους μου οι οποίοι συνέβαλαν τα μέγιστα για να ολοκληρώσω τις σπουδές μου καθώς και τον κ. Μαλάμο ο οποίος επέβλεπε την πτυχιακή μου και τέλος τα μέλη του εργαστηρίου Πολυμέσων τα οποία με άντεξαν όλο αυτό το καιρό.

Abstract

In this thesis, we created an application that transfers data from the kinect device to the user's browser. The application will be extended to be used to handle a game, which will be written in HTML5 and will run in the browser, for smooth user interaction with it. The kinect recognizes the movement of the user while using the application and transfers the information in the browser to handle the game. For the thesis will make use of new, ongoing, technologies such as HTML5, JavaScript, Kinect SDK, WebSockets API(supported by Google Chrome), X3Dom, X3D and WebGL.

Περίληψη

Σε αυτήν την πτυχιακή εργασία, θα δημιουργηθεί μία εφαρμογή η οποία θα μεταφέρει δεδομένα από το kinect, στο browser του χρήστη. Η εφαρμογή θα επεκταθεί έτσι ώστε να χρησιμοποιείται για το χειρισμό ενός παιχνιδιού, το οποίο θα είναι γραμμένο σε HTML5 και θα τρέχει στο browser, με σκοπό την ομαλή αλληλεπίδραση του χρήστη με αυτό. Το kinect θα αναγνωρίζει τη κίνηση του χρήστη ενώ μέσω της εφαρμογής θα μεταφέρεται η πληροφορία στο browser για το χειρισμό του παιχνιδιού. Για την πτυχιακή εργασία θα γίνει χρήση νέων, υπό εξέλιξη, τεχνολογιών όπως HTML5, JavaScript, Kinect SDK, WebSockets API(το οποίο υποστηρίζεται από τον Google Chrome browser), X3Dom, X3D και WebGL αλλά και παλαιότερων τεχνολογιών όπως είναι το ActiveX Control.

Κατάλογος περιεχομένων

Κατάλογος εικόνων

Ευρετήριο πινάκων

1 Εισαγωγή

Για περισσότερο από σαράντα χρόνια, οι 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 περιοστικό σαν μοντέλο αλληλεπίδρασης. 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. Για να προσεγγίσουμε μια τέτοια διεπαφή πρέπει να δούμε την ανθρώπινη επικοινωνία.

Η επικοινωνία μεταξύ ανθρώπων προυποθέτει τη χρήση διάφορων 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 να 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 κάποιους από τους παραπάνω τρόπους έκφρασης, όπως το λόγο ή την έκφραση προσώπου. Αλλά για τον εντοπισμό της κίνησης 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, σε προσιτή 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 Microsoft Kinect κυκλοφόρησε το Νοέμβριο του 2010 ως χειριστήριο για την παιχνιδομηχανή XBOX 360, ενώ το Asus Xtion Pro Live κυκλοφόρησε για τον υπολογιστή το δεύτερο μισό του 2011. Οι δύο αυτές συσκευές δεν περιορίζονται μονάχα στον αισθητήρα βάθους διότι παρέχουν επίσης μικρόφωνο και κάμερα χρώματος κι έτσι οι χρήστες μπορούν να έχουν στο σπίτι τους πια συσκευές πολυτροπικής εισόδου, οι οποίες θα δίνουν τη δυνατότητα και την ελευθερία στους προγραμματιστές να αναπτύξουν πιο ανθροποκεντρικές εφαρμογές.

Οι αφρόκρεμα των hackers περίμεναν το Kinect, γιατί θα είχαν την ευκαιρία να χρησιμοποιήσουν την τεχνολογία μιας μεγάλης εταιρίας και να δουν τι είναι 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. Η 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 δημοφιλή APIs για την αποκωδικοποίηση των δεδομένων που παράγει το Kinect – ανάμερα σε αυτά και το official SDK της Microsoft, τα οποία είναι ελεύθερα προς χρήση: Libfreenect, OpenNI και NITE, CLNUI, Microsoft Kinect SDK, Evoluce SDK (το οποίο είναι βασισμένο στο OpenNI). Στο παράρτημα θα βρείτε πλήρεις οδηγούς για την εγκατάσταση του καθενός από τα παραπάνω APIs.

1.1 Κίνητρο για την διεξαγωγή της εργασίας

Το kinect είναι, σχετικά, μια νέα συσκευή η οποία έφερε την επανάσταση τόσο στον τομέα των video-games αλλά και στην έρευνα για την ανάπτυξη εφαρμογών με βάση το kinect. Μέχρι στιγμής δεν έχει αναπτυχθεί το κατάλληλο λογισμικό για τη χρήση του kinect μέσα από τον browser, έτσι ώστε ο χρήστης μέσα από μία ιστοσελίδα να ενεργοποιήσει το Kinect και να το χρησιμοποιήσει σύμφωνα με τις ανάγκες του. Για αυτόν ακριβώς το λόγο επιλέχτηκε το θέμα αυτής της πτυχιακής εργασίας έτσι ώστε να ανοίξει νέους ορίζοντες στην έρευνα η οποία συνδυάζει το διαδίκτυο με το Kinect. Έτσι με την παραπέρα έρευνα και ανάπτυξη λογισμικού επάνω στο κομμάτι το οποίο προανέφερα μπορούν να δημιουργηθούν εμπορικές εφαρμογές και να αξιοποιηθούν καταλλήλως.

1.2 Σκοπός και στόχος της εργασίας

Σκοπός αυτής της εργασίας είναι να μπορέσει το kinect να ενεργοποιηθεί και να λειτουργήσει στο browser σύμφωνα με τις δικές μας ανάγκες. Στόχος μας είναι αρχικά να πάρουμε τα δεδομένα που μας επιστρέφει το kinect και να τα διαχειριστούμε μέσω του ActiveX Control. Χρησιμοποιούμε ActiveX Control διότι είναι ο μόνος τρόπος για να χρησιμοποιήσουμε συσκευές στο διαδίκτυο μέσα από το κώδικα μιας ιστοσελίδας έτσι ώστε να καταφέρουμε να παίξουμε παιχνίδια στο browser με την αλληλεπίδραση χρήστη-kinect. Έτσι στο τέλος θα καταφέρουμε να γίνει αναγνώριση κίνησης του χρήστη από το kinect και

να χρησιμοποιηθεί καταλλήλως για τον χειρισμό ενός παιχνιδιού το οποίο θα τρέχει με κώδικα γραμμένο σε HTML5.

1.3 Δομή της εργασίας

Στο πρώτο κεφάλαιο κάνουμε μια εισαγωγή σχετικά με το kinect και την ιστορία του στον αναγνώστη και στη συνέχεια του αναλύουμε τα κίνητρα για την διεξαγωγή της εργασίας καθώς και τους στόχους και σκοπούς αυτής της εργασίας. Στο δεύτερο κεφάλαιο αναλύουμε τη συσκευή kinect όσον αφορά τη λειτουργία της, τα APIs της αλλά και εφαρμογές οι οποίες ήδη έχουν χρησιμοποιηθεί είτε στο εμπόριο είτε στο πεδίο της έρευνας. Στο τρίτο κεφάλαιο αναφέρουμε τις τεχνολογίες αλλά και γλώσσες προγραμματισμού τις οποίες έχουμε χρησιμοποιήσει σε αυτή την εργασία. Στο τέταρτο κεφάλαιο περιγράφουμε αναλυτικά την υλοποίηση της εφαρμογής που δημιουργήσαμε. Στο πέμπτο κεφάλαιο έχουμε τις οδηγίες χρήσης για την εφαρμογή που δημιουργήσαμε και στο τέλος της εργασίας έχουμε το παράρτημα το οποίο περιέχει οδηγίες για την εγκατάσταση των διάφορων APIs που υπάρχουν.

2 Kinect Device

Εικόνα 1: Συσκευή Kinect

2.1 Εισαγωγή

Το Kinect είναι μία συσκευή αναγνώρισης κίνησης η οποία κατασκευάστηκε από την Microsoft για την κονσόλα(παιχνιδομηχανή) Xbox 360 και στη συνέχεια χρησιμοποιήθηκε από χρήστες ηλεκτρονικών υπολογιστών με λειτουργικό Windows σαν περιφερειακή συσκευή. Για πρώτη φορά ο χρήστης, χρησιμοποιώντας αυτή τη συσκευή, μπόρεσε να αλληλεπιδράσει πλήρως με το εκάστοτε μηχάνημα(Xbox ή PC) χωρίς να χρησιμοποιήσει κάποιο πληκτρολόγιο ή ποντίκι. 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. Φυσικά 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 διότι 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.

2.1.1 Διατέθηκε στην αγορά

Στη 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

Microsoft Alex Kipman, ο οποίος ήταν υπεύθυνος για το project , και καταγόταν από εκεί. 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 του Kinect σχεδιάστηκε και δημιουργήθηκε από την ισραηλινή εταιρία PrimeSense, Ltd. 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.

2.1.2 Kinect και για Windows

Εικόνα 2: Kinect for Windows

Εικόνα 3: Kinect for Xbox 360

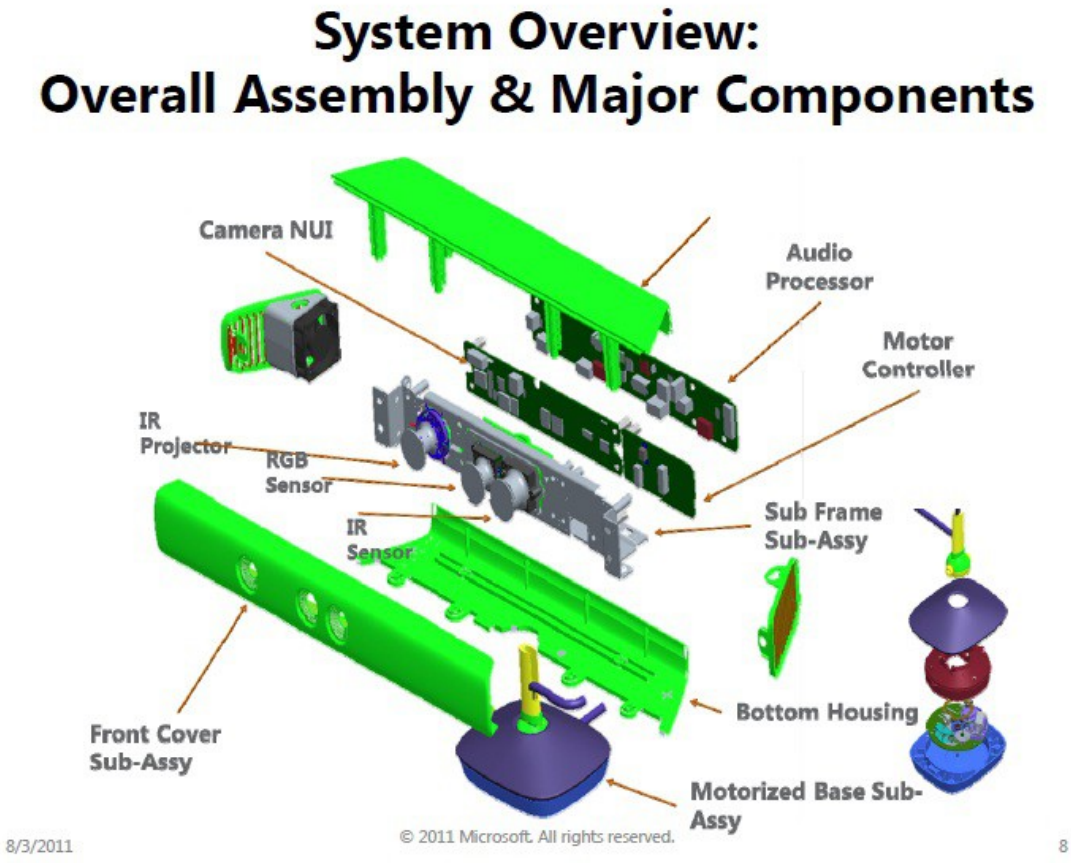
Στις 21 Φεβρουαρίου του 2011 η Microsoft ανακοίνωσε ότι την άνοιξη του ίδιου χρόνου θα διαθέσει στο κοινό το SDK του Kinect, το οποίο τελικά διατέθηκε στο κοινό στις 7 Ιουνίου του 2011 σε 12 χώρες. Το SDK εκτός από τους official drivers του Kinect για windows 7 παρέχει και τη δυνατότητα στους προγραμματιστές να δημιουργήσουν τις δικές τους εφαρμογές σε γλώσσα c++,c# ή Visual Basic χρησιμοποιώντας το Microsoft Visual Studio 2010 καθώς και τις παρακάτω δυνατότητες:

- Raw sensor streams: Πρόσβαση στην ακατέργαστη πληροφορία που παίρνουμε από τους αισθητήρες του Kinect(αισθητήρας βάθους, αισθητήρας κάμερας και αισθητήρες του μικροφώνου).
- Skeletal Tracking: Τη δυνατότητα να παίρνουμε τα σημεία και την εικόνα του σκελετού ενός ατόμου που κινείται εντός της εμβέλειας του Kinect και να τα χρησιμοποιούμε στις εφαρμογές μας.
- Advance audio capabilities: Τη δυνατότητα να επεξεργαζόμαστε τον ήχο και να εξάγουμε εξειδικευμένες πληροφορίες καθώς και να προσδιορίζουμε την τρέχουσα πηγή ήχου.
- Δείγματα κώδικα και Documentation.

Το Μάρτιο του 2012, ο Craig Eisler, ο γενικός διευθυντής του Kinect για τα

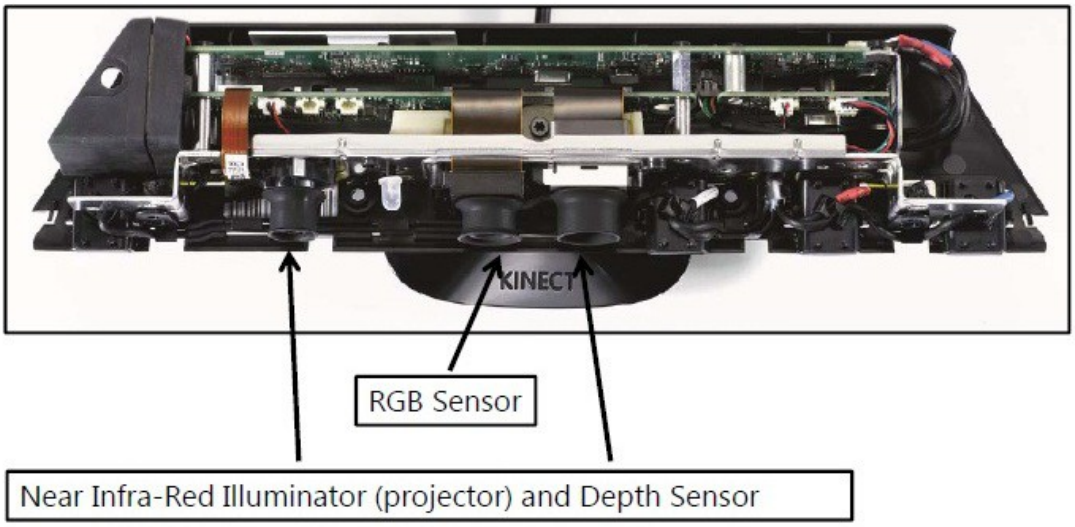
Windows, δήλωσε ότι περίπου 350 εταιρείες συνεργάζονται με τη Microsoft πάνω σε εξειδικευμένες εφαρμογές του Kinect οι οποίες θα τρέχουν σε Windows.

2.2 Οι αισθητήρες του Kinect



Εικόνα 4: Εσωτερικό του Kinect

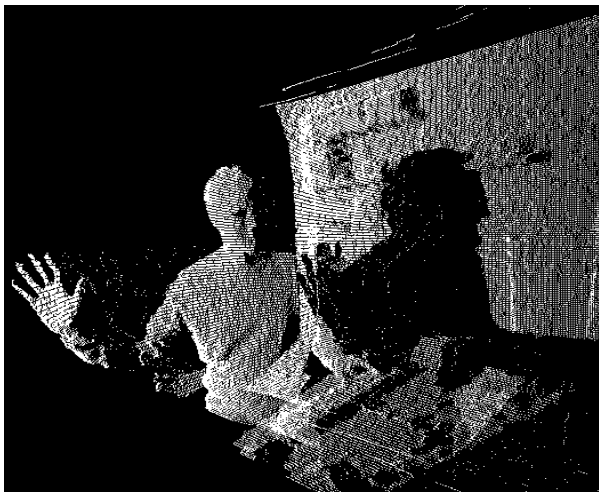
2.2.1 Αισθητήρας Βάθους(Depth Sensor) και πομπός υπερύθρων(IR Sensor)



Εικόνα 5: Θέση των αισθητήρων

Η τεχνολογία Light Coding ™ (ευρεσιτεχνία των Zalevsky, Z) επιτρέπει στο Kinect να δημιουργήσει 3D χάρτες βάθους μιας σκηνής, σε πραγματικό χρόνο. Μια δομή από σημεία (σχεδόν) υπέρυθρου φωτός προβάλλεται στον χώρο και ένας αισθητήρας εικόνας CMOS (Complementary Metal Oxide Semiconductor) δέχεται τις ανακλώμενες ακτίνες(βλ. Εικόνα 6). Το PS 1080 SoC (System on a Chip) – τσιπ φτιαγμένο από την PrimeSense που περιέχει το σύστημα του Kinect, ελέγχει την δομή από σημεία φωτός και επεξεργάζεται τα δεδομένα από τον αισθητήρα CMOS παράγοντας δεδομένα βάθους σε πραγματικό χρόνο. Η μέγιστη ανάλυση της εικόνας βάθους που παράγει το PS 1080 είναι 640x480, με συχνότητα 30 frames ανά δευτερόλεπτο(βλ. Εικόνα 7). 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 6: Kinect IR Image



Εικόνα 7: Kinect Depth Image

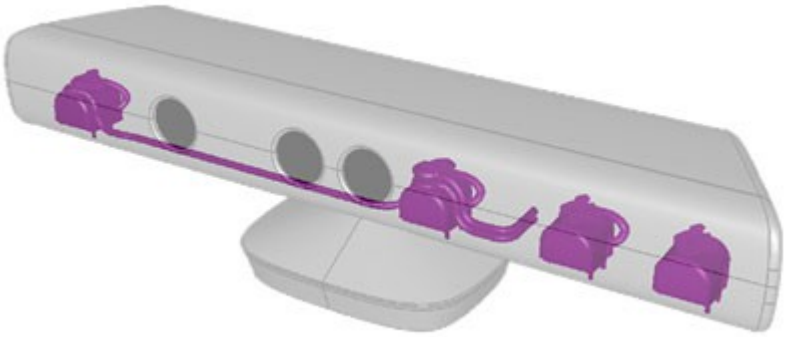
2.2.2 Έγχρωμη κάμερα

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 8.

Εικόνα 8: RGB εικόνα από το Kinect

2.2.3 Μικρόφωνα του Kinect

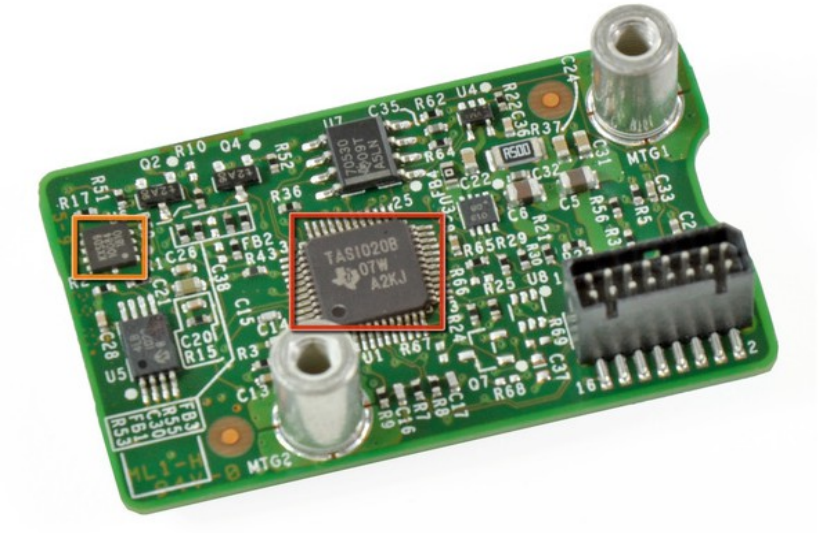
Μια σειρά από 4 μικρόφωνα δίνει στο Kinect τη δυνατότητα όχι απλά να δέχεται ήχο, αλλά και να εντοπίζει την γωνία της πηγής του στη σκηνή. Στην Εικόνα 8 βλέπουμε τη θέση των μικροφώνων μέσα στη συσκευή η οποία επεξεργάζεται ξεχωριστά το καθένα από τα τέσσερα κανάλια τα οποία δέχονται 16-bit ήχο με συχνότητα δειγματοληψίας ίση με 16 kHz.



Εικόνα 9: Kinect Microphones

2.2.4 Επιταχυνσιόμετρο (Accelerometer)

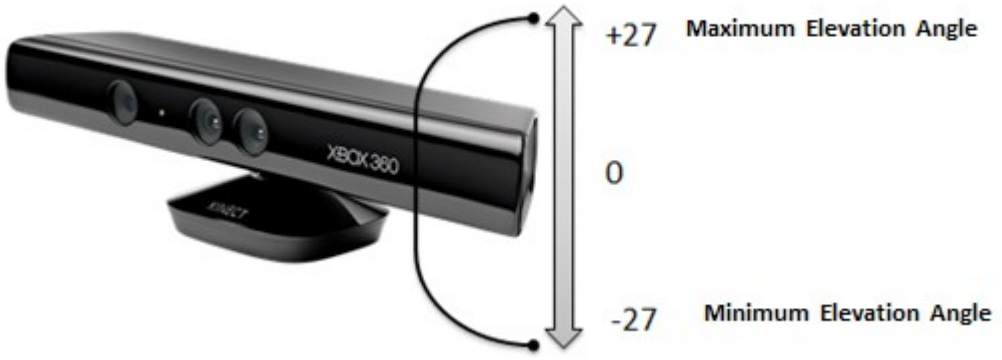
Το Kinect παρέχει ένα επιταχυνσιόμετρο τριών αξόνων (KXSD9-1026, βλέπε Εικόνα 9) το οποίο παρέχει την πληροφορία της θέσης της συσκευής. Οι τιμές του X και του Y καθορίζουν την κύλιση και την κλίση, ενώ το Z καθορίζει εάν το Kinect είναι ανάποδα ή όχι. Το επιταχυνσιόμετρο μετρά μονάχα την κλίση, κι όχι τον προσανατολισμό της συσκευής. Το χαρακτηριστικό αυτό του Kinect έχει χρησιμοποιηθεί ευρέως στον τομέα της ρομποτικής.



Εικόνα 10: Accelerometer(orange rectangular) του Kinect

2.2.5 Μηχανοκίνητη βάση(Motorized Tilt) του Kinect

Η συσκευή διαθέτει επίσης μια μηχανοκίνητη βάση (βλ. Εικόνα 10) 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°.



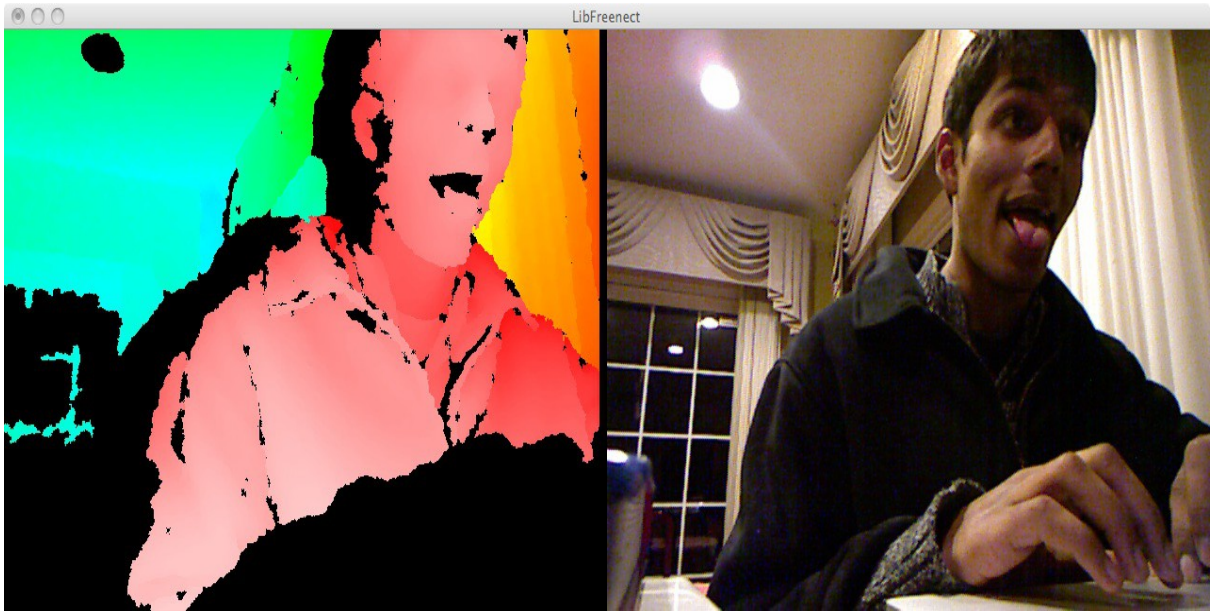
Εικόνα 11: Μηχανοκίνητη βάση στήριξης του Kinect

2.3 Τα διαθέσιμα API 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 τι είναι 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 δημοφιλή APIs για την αποκωδικοποίηση των δεδομένων που παράγει το Kinect – ανάμερα σε αυτά και το official SDK της Microsoft, τα οποία είναι ελεύθερα προς χρήση: Libfreenect, OpenNI και NITE, CLNUI, Microsoft Kinect SDK, Evoluce SDK (το οποίο είναι βασισμένο στο OpenNI). Στο παράρτημα θα βρείτε πλήρεις οδηγούς για την εγκατάσταση του καθενός από τα παραπάνω APIs.

2.3.1 OpenKinect – Libfreenect

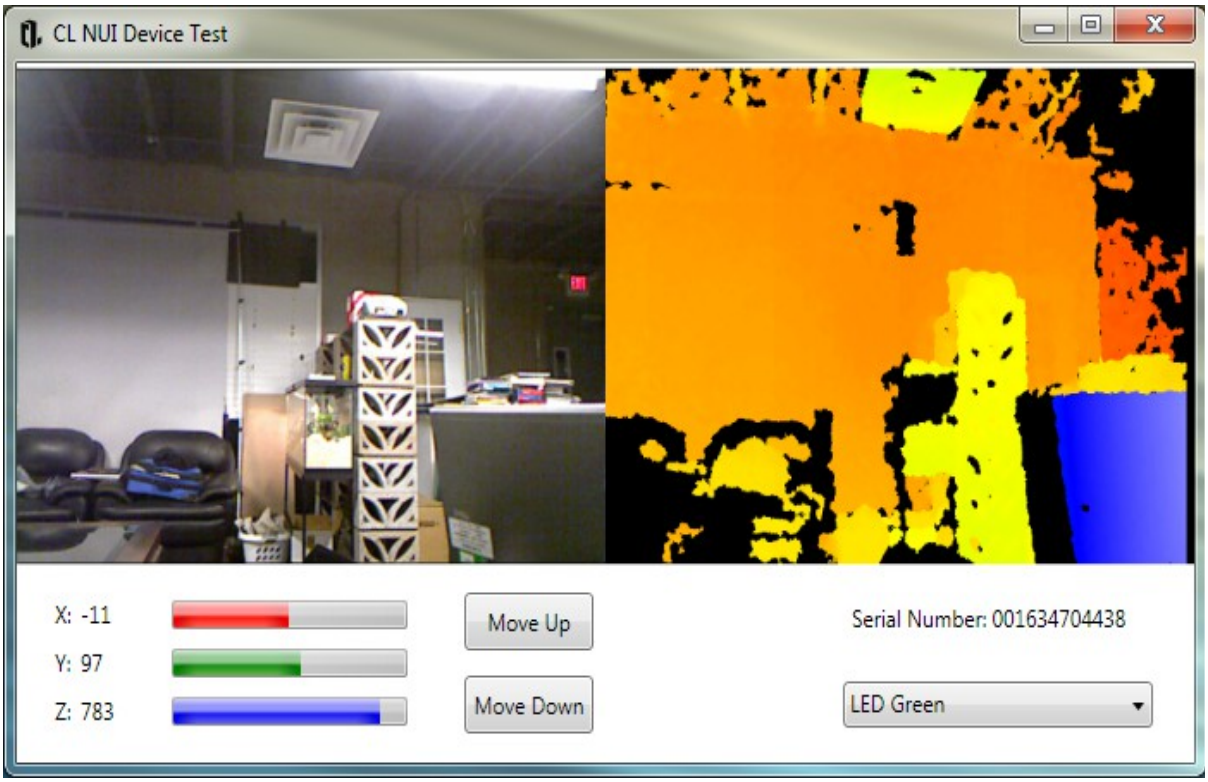
Το Libfreenect ήταν η πρώτη βιβλιοθήκη που δημιουργήθηκε για την αποκωδικοποίηση των δεδομένων του Kinect. Κυκλοφόρησε στις αρχές του Νοεμβρίου του 2010 από το Héctor Martín, λίγο μετά την κυκλοφορία του Kinect στην αγορά. Δημοσιεύθηκε στην ιστοσελίδα του Open Kinect στις 10 Νοεμβρίου όπου αποτέλεσε 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 11), 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, διότι 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 12: Libfreenect – Απεικόνιση βάθους(αριστερά) και RGB εικόνας(δεξιά)

2.3.2 CL – NUI

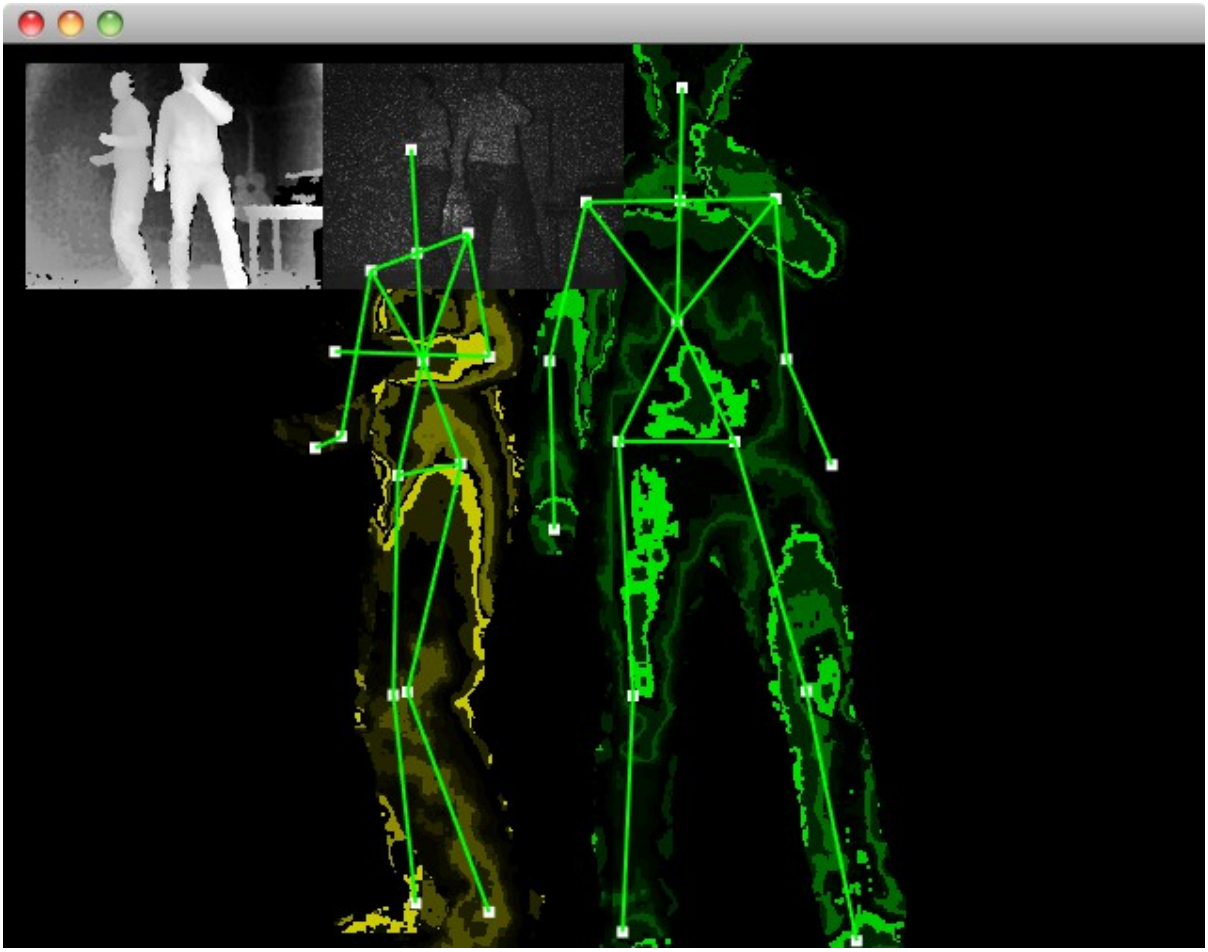
Η πλατφόρμα διαχείρισης των Code Laboratories βασίζεται στην αποκωδικοποίηση του AlexP, ο οποίος κατάφερε να αποκωδικοποιήσει τα δεδομένα του Kinect πριν από τον Héctor Martín αλλά δεν δημοσίευσε άμεσα τον κώδικά του. Η πρώτη έκδοση της πλατφόρμας CL NUI εκδόθηκε στις 8 Δεκεμβρίου 20 10, και είναι ευρέως 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 12).



Εικόνα 13: CL NUI user interface

2.3.3 OpenNI , NITE

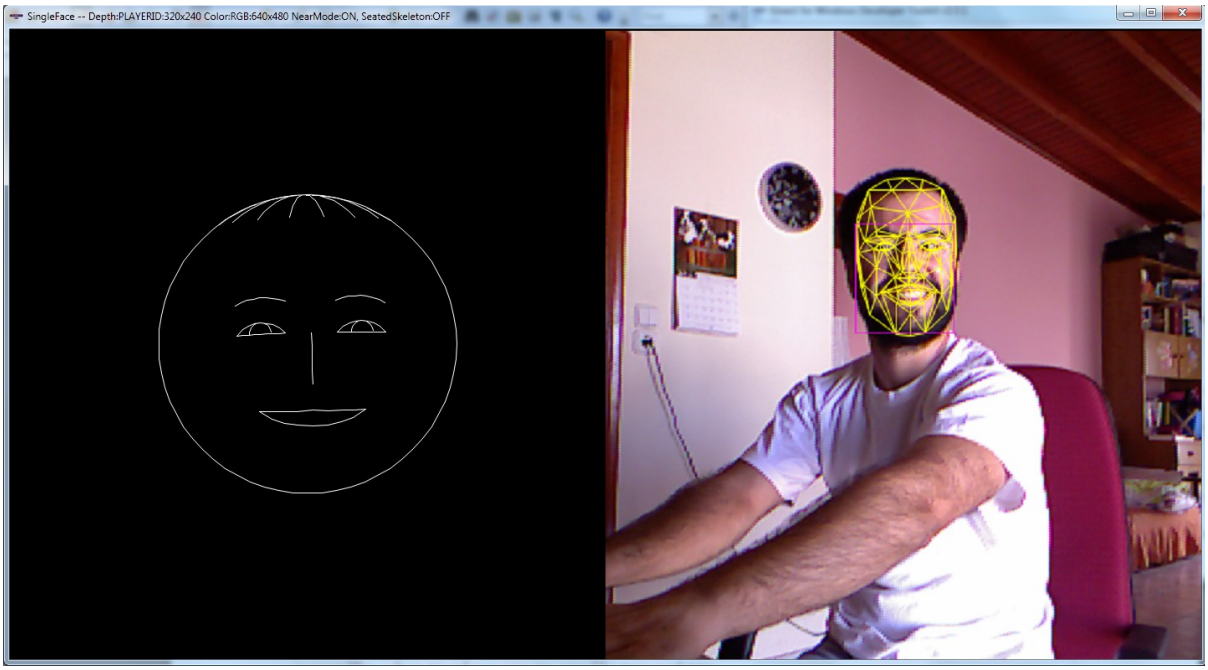
Δημιουργήθηκε από έναν αφιλοκερδή οργανισμό ο οποίος απαρτίζεται από διάφορες εταιρίες, συμπεριλαμβανομένου και της PrimeSence Ltd. οι οποίες θέλησαν να θέσουν ένα βιομηχανικό πρότυπο λειτουργικότητας για τις συσκευές φυσική διεπαφής χρήστη – υπολογιστή (Natural User Interaction Devices) και κυκλοφόρησε 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 13) αλλά στις πιο πρόσφατες εκδόσεις του, το OpenNI/NITE, δεν χρειάζεται αρχική στάση βαθμονόμησης (χαρακτηριστικό που είχε μόνο το MS SDK).



Εικόνα 14: OpenNI - Ταυτόχρονη απεικόνιση βάθους και σειμείων του σκελετού

2.3.4 Official Microsoft Kinect SDK

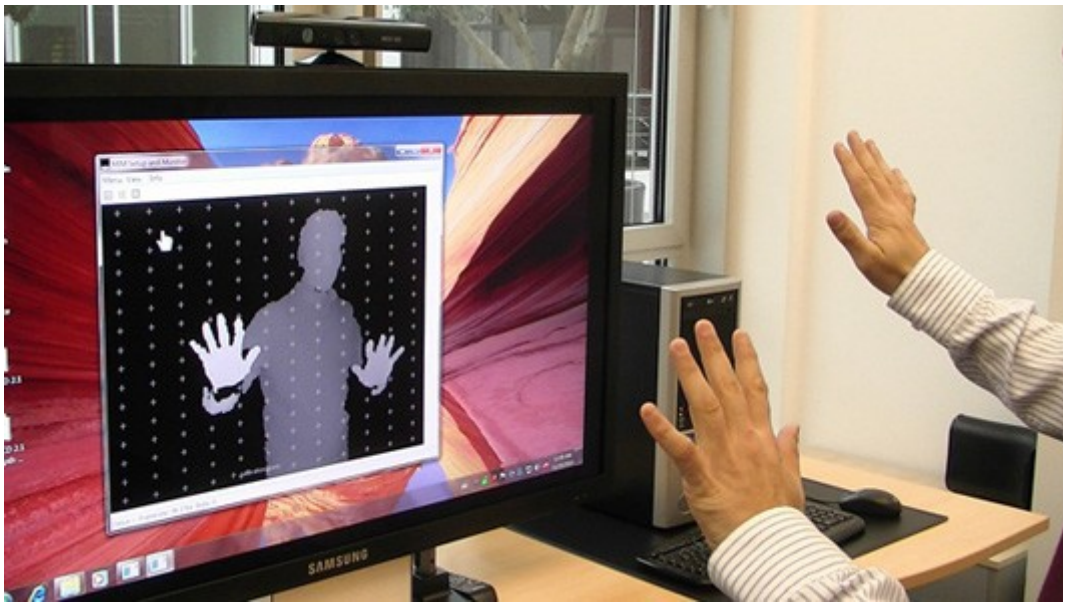
Στις 16 Ιουνίου του 2011 κυκλοφόρησε η beta έκδοση της βιβλιοθήκης της Microsoft για τη χρήση του Kinect στα Windows 7. Το SDK έδινε τη δυνατότητα στου προγραμματιστές να αναπτύξουν εφαρμογές σε C++, C# ή Visual Basic με τη χρήση του Microsoft Visual Studio 2010. Μερικές από της αποκλειστικές δυνατότητες 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. 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 14), 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.



Εικόνα 15: Official Microsoft Kinect SDK – Face Detection

2.3.5 Evoluce SDK

Η εταιρία Evoluce ανέπτυξε ένα SDK βασισμένο στο OpenNI. Με τη χρήση της βιβλιοθήκης Emgu CV 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. Εικόνα 15), υποστήριξη του Surface 2.0. Η χρήση του περιορίζεται μονάχα στα Windows 7.



Εικόνα 16: Evoluce SDK

2.3.6 Σύγκριση Δυνατοτήτων των API

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 1 μας δείχνει τις 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 1: Kinect APIs και γλώσσες προγραμματισμού

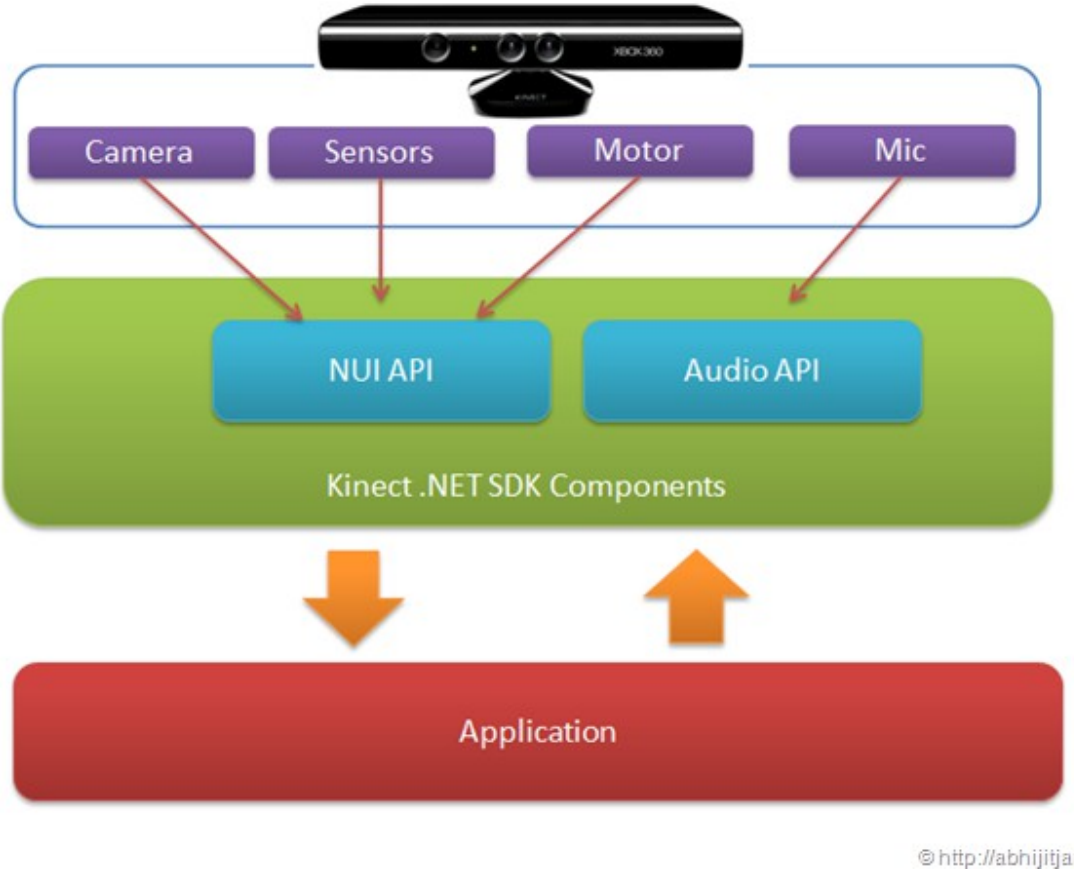
Στον Πίνακα 2 έχουμε τις δυνατότητες που παρέχει κάθε API. Κάποιες από αυτές μπορούν να αναπληρωθούν με κώδικα φτιαγμένο από το χρήστη, στον πίνακα περιγράφονται οι δυνατότητες που παρέχουν τα APIs ακριβώς όπως τα κατεβάζουμε. Συμπερασματικά βλέπουμε ότι το Microsoft Kinect SDK έχει πιο πολλές δυνατότητες 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 και 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 2: Διαφορές στα χαρακτηριστικά των APIs

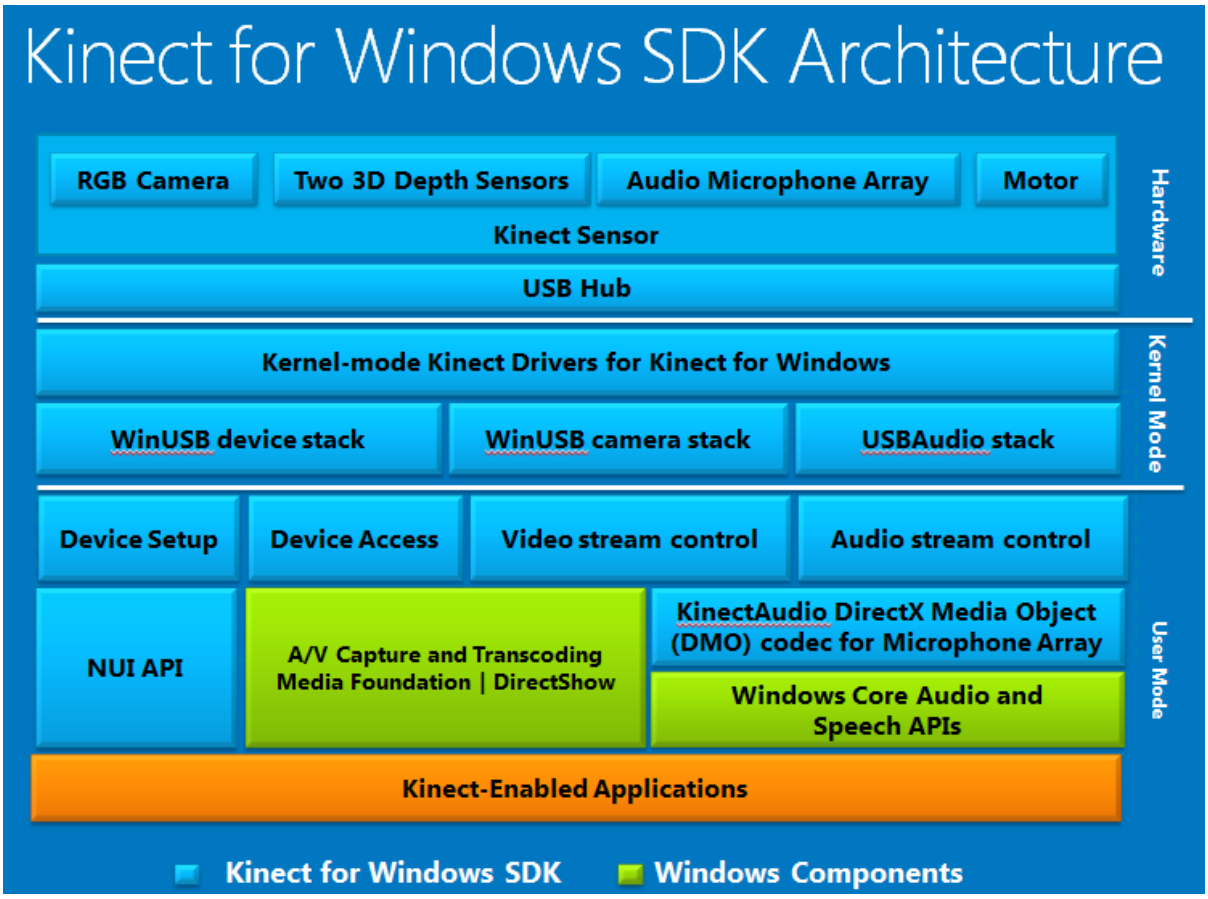
2.4 Official Microsoft Kinect SDK for Windows

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



Εικόνα 17: 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 18: Αρχιτεκτονική του Kinect SDK

2.4.1 Απαιτήσεις Συστήματος

Υποστηριζόμενα λειτουργικά συστήματα

- Windows 7, επίσης έχει τεσταριστεί στα Windows 8 Developer Preview

Hardware Requirements

- 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

Software Requirements

- Microsoft Visual Studio 2010 Express ή κάποια άλλη έκδοση του Visual Studio 2010
- .NET Framework(εγκαθίσταται με το Visual Studio 2010)
- Για να προγραμματίσεις χρησιμοποιώντας την ανίχνευση φωνής, πρέπει να εγκαταστήσεις το Microsoft Speech Platform SDK v11

2.4.2 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.

2.4.3 Color Stream

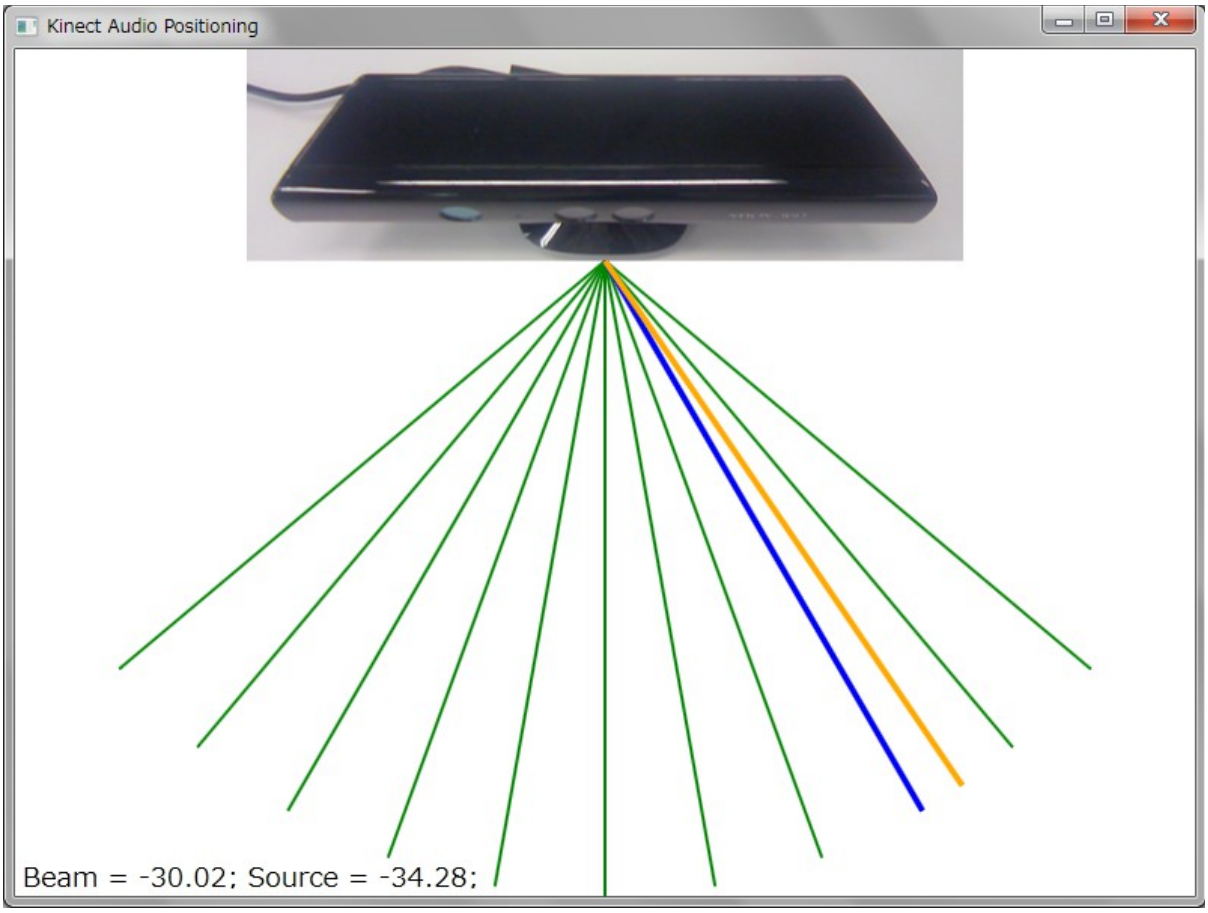
Τα δεδομένα χρώματος είναι διαθέσιμα σε δύο μορφές:

- 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 να αντιπροσωπεύουν την ίδια εικόνα. Διαλέγουμε με πια μορφή θα δουλέψουμε κατά την εκκίνηση του Color Stream.

2.4.4 Audio Stream

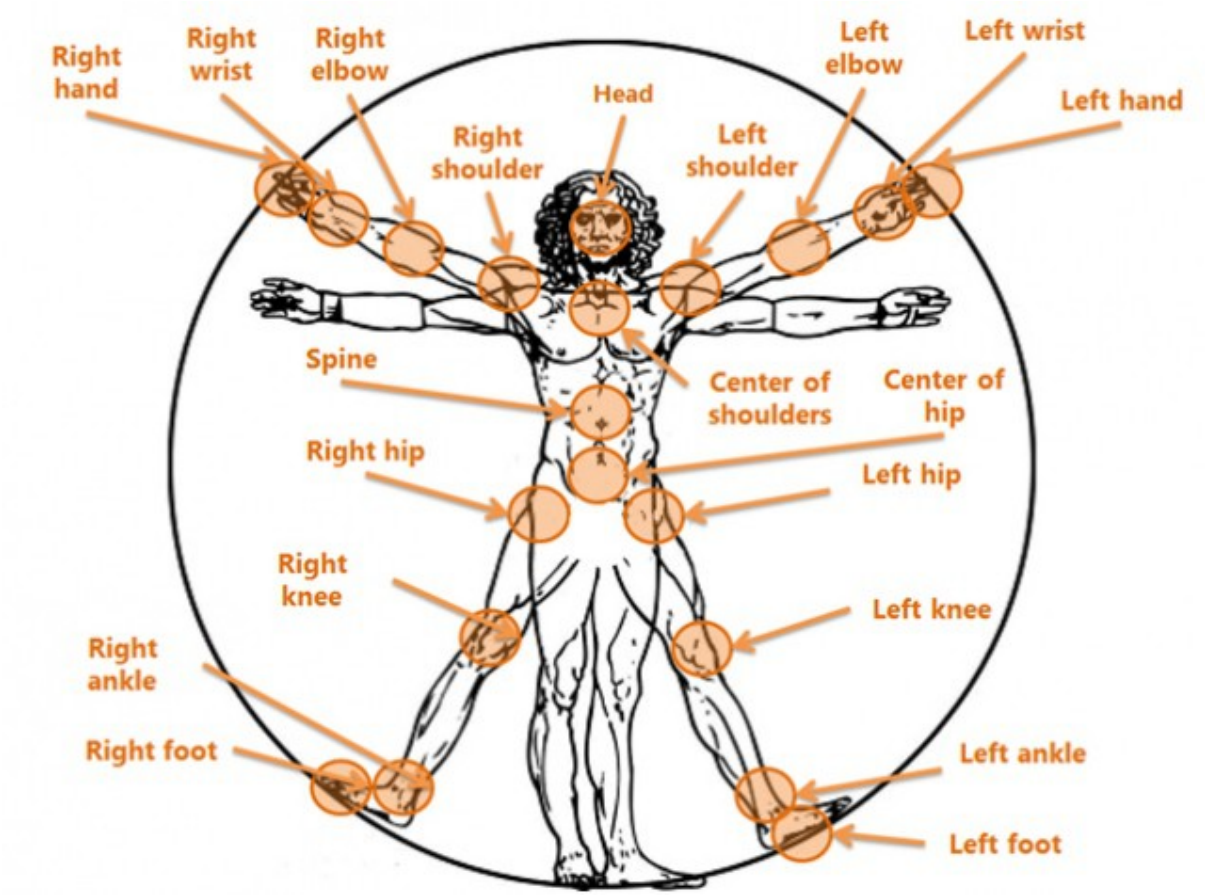
Το Kinect περιέχει, όπως προαναφέρθηκε μια σειρά από τέσσερα μικρόφωνα, η οποία χρησιμοποιεί 24-bit ADC (Analog-to-digital Converter) και παρέχει τοπική επεξεργασία σήματος, συμπεριλαμβανομένων των : Acoustic Echo Cancellation και Noise Suppression. Οι εφαρμογές που αναπτύσσονται με αυτό το SDK μπορούν 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 19: Εφαρμογή εύρεσης θέσης της ηχητικής πηγής

2.4.5 Skeleton Tracking

Το SDK παρέχει επίσης πληροφορίες για την θέση μέχρι και δυο χρηστών οι οποίοι βρίσκονται μπροστά από το Kinect, με λεπτομερή 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 20: Τα Skeleton Positions τα οποία παίρνουμε από το Skeleton Data

2.5 Εφαρμογές που χρησιμοποιούν το Kinect

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).

2.5.1 Εικονική Πραγματικότητα (Virtual Reality)

Πλέον καθημερινά όλο και περισσότερες εφαρμογές δημιουργούνται που έχουν να κάνουν με την εικονική πραγματικότητα. Ένας πρώην υπάλληλος της Microsoft(Razorfish) δημιούργησε ένα ηλεκτρονικό κατάστημα ρούχων και αξεσουάρ το οποίο χρησιμοποιεί το Kinect. Το μεγάλο πλεονέκτημα είναι ότι ο πελάτης πλέον μπορεί να αγοράσει ένα προϊόν και να το δοκιμάσει από το σπίτι του. Έτσι στον δικό του χώρο, χωρίς να χρειάζεται να μεταβεί στο κατάστημα και να περιμένει στο δοκιμαστήριο, έχει ακριβώς τα ίδια αποτελέσματα σαν να ήταν στο κατάστημα. Ο χρήστης βλέπει στην τηλεόρασή ή στον υπολογιστή το ολόγραμμά του καθώς και τα προϊόντα που επιθυμεί να δοκιμάσει και να αγοράσει βλέποντας "πώς στέκουν πάνω του και αν του πάνε". Έτσι, σιγά σιγά, καταλαβαίνουμε ότι κάποια πράγματα τα οποία βλέπαμε σε ταινίες επιστημονικής φαντασίας γίνονται πλέον πραγματικότητα και τα ζούμε.

Εικόνα 21: KinectShop

2.5.2 Αναπηρίες (Disabilities)

Οι δάσκαλοι ειδικής αγωγής βρίσκουν ιδιαίτερα χρήσιμο το Kinect και ανακαλύπτουν συνεχώς νέους τρόπους διδασκαλίας οι οποίοι γίνονται εύκολα αποδεκτοί από παιδιά με πνευματικές και κινητικές αναπηρίες. Σύμφωνα με πρόσφατες έρευνες έχει αποδειχθεί ότι ο ρυθμός με τον οποίο αυτά τα παιδιά αναπτύσουν τις ικανότητες τους είναι αυξημένος όταν χρησιμοποιεί ο δάσκαλος το Kinect. Σίγουρα υπάρχει πολύ μέλλον και σε αυτόν τον τομέα και πρέπει επιτακτικά να προωθήσουν οι εκπαιδευτικοί αυτή τη τεχνολογία στα ειδικά

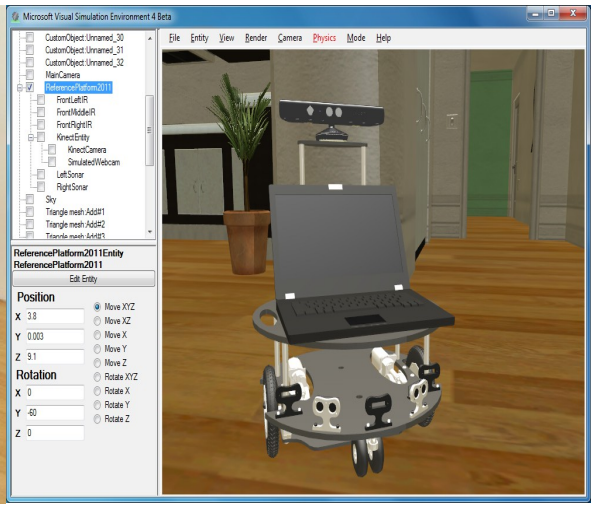
σχολεία.

Εικόνα 22: Χρήση Kinect στην εκπαίδευση ειδική αγωγή

2.5.3 Τεχνητή όραση και Ρομποτική(Computer Vision and Robotics)

Πρόσφατα η Microsoft ανακοίνωσε ότι το Robotics Developer Studio 4(RDS 4) είναι διαθέσιμο στην αγορά για τη δημιουργία εφαρμογών που συνδυάζουν ρομπότ με Kinect. Το RDS 4 είναι ένα εργαλείο εξομοίωσης με το οποίο προγραμματίζεις το Hardware κομμάτι ενός ρομπότ. Για αυτό το λόγο η Microsoft ζήτησε από την εταιρία Parallax να κατασκευάσει ένα ρομπότ(τον Eddie) κατάλληλο για αυτή τη δουλειά. Στον Eddie μπορείς να προσθέσεις αισθητήρες ή ότι άλλο χρειάζεσαι για να πετύχεις το στόχο σου χωρίς περιορισμούς. Η τιμή του είναι κοντά στα 1200 δολλάρια και φυσικά στο πακέτο δεν περιέχεται το Kinect και το laptop.

Εικόνα 23: Το Robot ο Eddie μαζί με το Kinect



Εικόνα 24: Το Robotics Developer Studio 4

3 Τεχνολογίες Υλοποίησης

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

3.1 .NET Framework

Το .NET Framework είναι μία βιβλιοθήκη για υπολογιστές με λειτουργικό σύστημα Windows. Δίνει τη δυνατότητα στους προγραμματιστές να χρησιμοποιήσουν διάφορες εξελιγμένες λειτουργίες και η εφαρμογή του να είναι συμβατή με όλα τα συστήματα που υποστηρίζουν και έχουν εγκατεστημένο το .NET Framework.Η αρχιτεκτονική του .NET Framework χωρίζεται σε 4 τμήματα ! το Common Language Runtime (CLR), ένα σύνολο από βιβλιοθήκες, ένα σύνολο από γλώσσες προγραμματισμού και από την ASP.NET. Το .NET Framework σχεδιάστηκε ώστε να ικανοποιεί τρεις στόχους. Αρχικά έπρεπε να κάνει τις εφαρμογές των Windows πιο αξιόπιστες, βελτιώνοντας το βαθμό της ασφάλειάς τους. Δεύτερο, προοριζόταν για την απλούστευση της ανάπτυξης Web εφαρμογών και υπηρεσιών (Web services), οι οποίες θα έτρεχαν και σε φορητές συσκευές. Τρίτο, το Framework σχεδιάστηκε για να παρέχει ένα σύνολο βιβλιοθηκών που μπορούσαν να λειτουργήσουν με πολλές γλώσσες.

- ***Common Language Runtime***

Οι γλώσσες προγραμματισμού συνήθως αποτελούνται από έναν compiler και ένα runtime περιβάλλον. Ο compiler μεταφράζει τον κώδικα σε εκτελέσιμο αρχείο που μπορεί να εκτελεστεί από τους χρήστες. Το runtime περιβάλλον παρέχει ένα σύνολο υπηρεσιών του λειτουργικού συστήματος, στον εκτελέσιμο κώδικα. Οι υπηρεσίες αυτές είναι ενσωματωμένες σε ένα επίπεδο runtime (Runtime Layer) που επιτρέπει στον κώδικα να μην ασχολείται με λεπτομέρειες χαμηλού επιπέδου του λειτουργικού συστήματος. Τέτοιες λειτουργίες μπορεί να είναι η διαχείριση μνήμης, εγγραφή και ανάγνωση αρχείων κλπ. Πριν το .NET Framework, κάθε γλώσσα είχε και το δικό της runtime περιβάλλον. Η Visual Basic ερχόταν με το MSVBVM60.DLL, ενώ η Visual C++ με το MSVCRT.DLL. Το περιβάλλον ενσωματωνόταν με τον εκτελέσιμο κώδικα και έπρεπε να εγκατασταθεί στο μηχάνημα του χρήστη. Το βασικό πρόβλημα με τα περιβάλλοντα αυτά, βρίσκεται στο ότι ήταν σχεδιασμένα για χρήση με μόνο μία γλώσσα. Δεν μπορούσαν να χρησιμοποιηθούν λειτουργίες από το περιβάλλον μιας γλώσσας, σε μία άλλη. Έτσι, ένας από τους βασικούς στόχους του .NET Framework ήταν να ενοποιήσει τα runtime περιβάλλοντα ώστε οι προγραμματιστές να μπορούν να χρησιμοποιούν μόνο ένα περιβάλλον. Έτσι η λύση που δόθηκε ήταν η Common Language Runtime (CLR). Το CLR παρέχει δυνατότητες όπως διαχείριση μνήμης, ασφάλεια, διαχείριση λαθών κλπ, και όλα αυτά για κάθε γλώσσα που δουλεύει με το .NET Framework. Το CLR επίσης επιτρέπει στις γλώσσες να συνεργάζονται μεταξύ τους. Μπορεί για παράδειγμα να δεσμευτεί ένα κομμάτι μνήμης με κώδικα γραμμένο στην Visual Basic .NET και το ίδιο κομμάτι να ελευθερωθεί με κώδικα γραμμένο σε άλλη γλώσσα όπως η C#.

- ***.NET Class Libraries***

Οι προγραμματιστές αρέσκονται στο να δουλεύουν με κώδικα που ήδη έχει δοκιμαστεί και φαίνεται να λειτουργεί, όπως για παράδειγμα το Win32 API και οι βιβλιοθήκες MFC. Η επαναχρησιμοποίηση κώδικα ήταν στόχος της προγραμματιστικής κοινότητας, από πολύ παλιά. Πολλές γλώσσες είχαν πρόσβαση σε κομμάτια κώδικα δοκιμασμένα, έτοιμα για εκτέλεση. Οι προγραμματιστές που χρησιμοποιούσαν την Visual C++ είχαν επωφεληθεί από βιβλιοθήκες όπως η Microsoft Foundation Classes (MFC) που τους επέτρεπαν να δημιουργήσουν εφαρμογές Windows εύκολα και γρήγορα. Ωστόσο, το ότι οι βιβλιοθήκες αυτές ήταν προορισμένες για μία μόνο γλώσσα σήμαινε ότι δεν μπορούσαν να χρησιμοποιηθούν με καμία άλλη γλώσσα. Το .NET Framework παρέχει πολλές κλάσεις για να βοηθήσει τους προγραμματιστές στην επαναχρησιμοποίηση κώδικα. Οι βιβλιοθήκες .NET Class Libraries περιέχουν κώδικα για προγραμματιστικά θέματα όπως νήματα, εγγραφή/ανάγνωση αρχείων, υποστήριξη βάσεων δεδομένων, μετατροπή σε XML, δομές δεδομένων όπως στοίβες και ουρές κλπ. Το καλύτερο σημείο βέβαια είναι το ότι η βιβλιοθήκη είναι διαθέσιμη σε κάθε γλώσσα που λειτουργεί με το .NET Framework.

- ***.NET Γλώσσες Προγραμματισμού***

Το .NET Framework παρέχει ένα σύνολο εργαλείων για να βοηθήσει στην κατασκευή κώδικα που λειτουργεί με αυτό. Η Microsoft παρέχει ένα σύνολο γλωσσών που είναι ήδη συμβατές με το .NET. Η C# είναι μία από αυτές. Επίσης δημιουργήθηκαν νέες εκδόσεις της Visual Basic και της Visual C++ όπως και μία νέα έκδοση της Jscript.NET. Ένα πολύ σημαντικό στοιχείο είναι ότι οι συμβατές γλώσσες με το .NET δεν είναι αποκλειστικά της Microsoft, αφού η εταιρία έχει δημοσιεύσει πλήρης τεκμηρίωση που δείχνει το πώς οι κατασκευαστές γλωσσών μπορούν να κάνουν τις γλώσσες τους συμβατές με το .NET, και διάφοροι κατασκευαστές το επιχείρησαν όπως η COBOL και η Perl. Υπάρχουν αυτή τη στιγμή πάνω από 20 γλώσσες τρίτων κατασκευαστών που μπορούν και λειτουργούν στο περιβάλλον .NET Framework.

- ***ASP.NET***

Το Internet αρχικών δημιουργήθηκε για την παροχή στατικού περιεχομένου στους Web browsers. Το Active Server Pages (ASP) δημιουργήθηκε από τη Microsoft για να προσθέσει δυναμικές ιδιότητες στις web σελίδες. Αυτό το πέτυχε με προγραμματισμό πίσω από την web σελίδα, κυρίως σε VB Script. Όταν οι χρήστες επισκέπτονταν ένα site τους ζητιόταν να επιβεβαιώσουν κάποιες πληροφορίες (είτε manual είτε με χρήση cookies) και στη συνέχεια ένα script δημιουργούσε τη σελίδα ανάλογα με τα στοιχεία που εισήγαγε ο χρήστης. Η ASP.NET βελτιώνει κατά πολύ την αρχική ASP. Με την ASP το HTML και το script βρίσκονταν στο ίδιο έγγραφο. Με την ASP.NET το script και το HTML βρίσκονται σε διαφορετικά έγγραφα. Η ASP.NET υποστηρίζει το Web Forms. Το Web Forms επιτρέπει στον προγραμματιστή να σύρει και να αφήσει (drag and drop) controls των φορμών, και να δημιουργηθεί αυτόματα ο κώδικας από πίσω, όπως ακριβώς σε μία τυπική εφαρμογή Windows.

3.2 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.

3.3 ActiveX

Αρκετός κόσμος έχει συνδέσει την τεχνολογία ActiveX με κάτι κακό, με κάτι που έχει σχέση με ιούς, με κάτι που έχει σχέση με τον Internet Explorer και γενικά με ότι αρνητικό υπάρχει στο διαδίκτυο. Το πρόβλημα ξεκινάει από τις εποχές των παλαιών λειτουργικών συστημάτων, δηλαδή του DOS και ίσως και πιο πίσω. Κάθε εφαρμογή κάνει κάτι από μόνη της (π.χ. ένα calculator κάνει πράξεις), ορισμένες όμως φορές μία εφαρμογή κάνει κάτι ίδιο με μία άλλη κοινή εφαρμογή. Για παράδειγμα αν γυρίσουμε πίσω στην εποχή του DOS βρίσκουμε προγράμματα τα οποία εκείνη την εποχή χρησιμοποιούσαν όλα την «εξτρα» μνήμη που είχε ο υπολογιστής. Αντί όμως κάθε εφαρμογή να υλοποιεί τον δικό της τρόπο για να παίρνει την μνήμη, κάτι που θα μας πήγαινε σε ασυμβατότητες και κρασαρίσματα του υπολογιστή, χρησιμοποιούσαν όλοι μία κοινή βιβλιοθήκη, το HIMEM.SYS. Με αυτόν τον τρόπο η Ms και η κάθε Ms της εποχης ασχολείτο η ίδια με το πρόβλημα ενώ το κάθε πρόγραμμα είχε πλέον ένα σίγουρο και ασφαλή τρόπο να βλέπει την έξτρα μνήμη που χρειάζεται.

Στα Windows καθώς και στο Linux ή στο OSX κάθε εφαρμογή κάνει το δικό της πράγμα για το οποίο είναι γραμμένη, πολλές εφαρμογές όμως κάνουν κάτι κοινό, π.χ. να κάνουν update από το Internet, να εκτυπώνουν στον ίδιο εκτυπωτή, να προβάλουν 3D δεδομένα στο ίδιο framework (DirectX/ OpenGL) κλπ. Για να μην κουβαλάει η κάθε εφαρμογή τον ίδιο κώδικα, στα Windows υπάρχει ένα set από διαθέσιμο κώδικα, γνωστο σαν Application Programming Interface στο οποίο έχουν πρόσβαση όλα τα προγράμματα. Αυτά είναι τα γνωστά μας αρχεία DLL.

Η τεχνολογία ActiveX κάνει διαθέσιμη, όπως και με τα DLL, κώδικα στο σύστημα. Αντίθετα όμως με τα DLL τα αρχεία του ActiveX μπορούν να τοποθετηθούν οπουδήποτε στο σύστημα και στη συνέχεια καταγράφεται ο προορισμός τους μαζί με το τί κάνουν στο HKEY_CLASSES_ROOT. Με μία διαδικασία σχετικά απλή για τους προγραμματιστές, όταν κάποιος θέλει να χρησιμοποιήσει ένα διαθέσιμο τμήμα κώδικα, απλά καλεί το ActiveX που θέλει (το οποίο ξεχωρίζει μοναδικά από κάποιο άλλο με ένα 128-bit αριθμό που λέγεται CLSID).

Το αποτέλεσμα φαίνεται όταν για παράδειγμα ένας printer driver εγκαθίσταται στο λειτουργικό. Αντί κάθε πρόγραμμα να περιέχει τον κώδικα που χρειάζεται για να εκτυπώσεις (και άρα να ξαναγραφτεί για να υποστηρίζει τον συγκεκριμένο εκτυπωτή όπως γινόταν στο DOS) , καλεί απλά το ActiveX του εκτυπωτή (ορισμένο από τα Windows) το οποίο επιτρέπει να ρυθμίσει και να εκτυπώσει ο χρήστης με μια διαδικασία transparent και

για το πρόγραμμα και για τον εκτυπωτή.

Άλλο παράδειγμα, όταν κάποιος έχει εγκατεστημένο στο σύστημά του το Excel, μπορεί από μια τρίτη εφαρμογή να το χρησιμοποιήσει για ένα λογισμικό φύλλο. Καλεί το ActiveX του Excel (το οποίο και επιστρέφεται μόνο όταν υπάρχει το Excel , αλλιώς φυσικά δεν γίνεται) και χρησιμοποιεί τις δυνατότητες του Excel.

Κάποιος θα πει για παράδειγμα ότι για κάτι απλό όπως την εμφάνιση ενός μηνύματος δεν χρησιμοποιείται ActiveX. Αυτό δεν ισχύει, και για τα ποιο μικρά θέματα όπως την δημιουργία συντόμευσης, τη σύνδεση στο Internet από οποιονδήποτε browser χρησιμοποιείται ActiveX με τον ένα ή τον άλλο τρόπο.

4 Υλοποίηση του Project

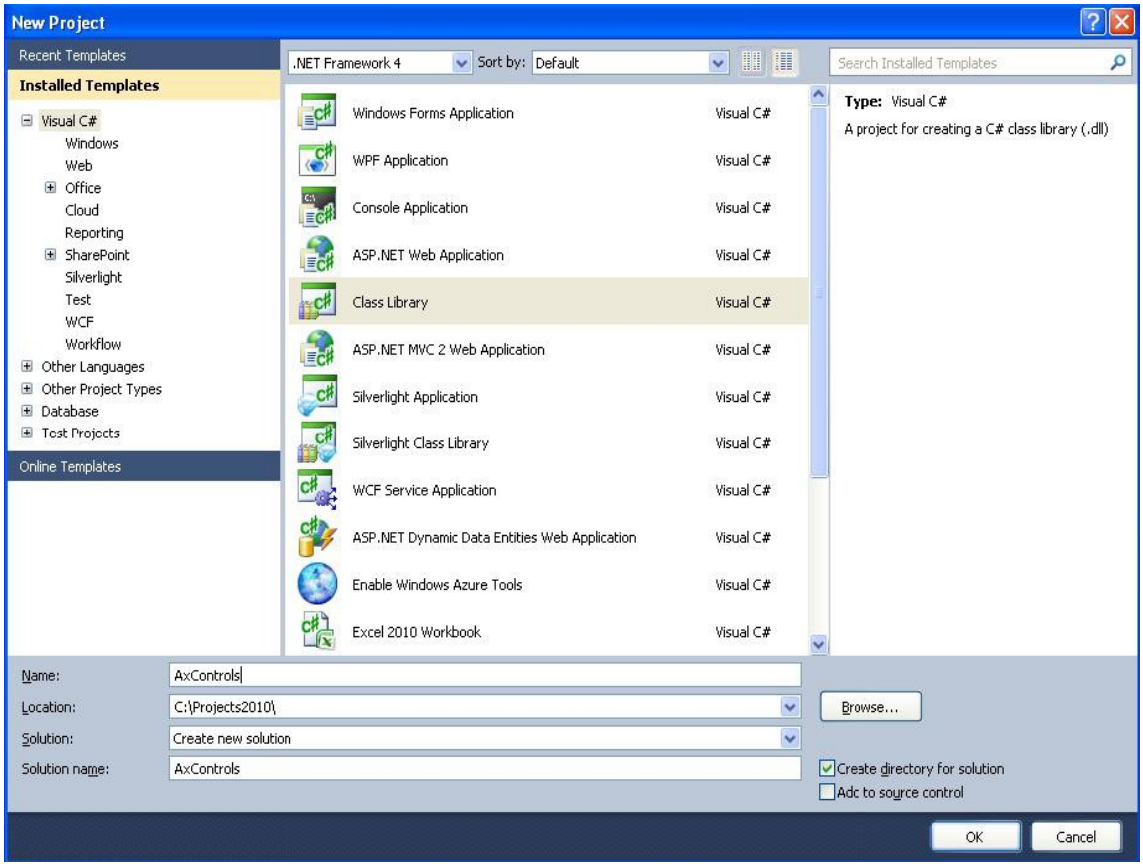
Εδώ τώρα θα δώσουμε σαφής οδηγίες για την υλοποίηση του κώδικά μας καθώς και θα τονίσουμε συγκεκριμένα σημεία που έχουν ιδιαίτερο ενδιαφέρον. Στο πρώτο κομμάτι θα αναλύσουμε τη δημιουργία ενός ActiveX αντικειμένου και στη συνέχεια, περνώντας στο δεύτερο κομμάτι, θα αναλύσουμε το κώδικα που σχετίζεται με την επικοινωνία του Kinect.

4.1 Δημιουργία ActiveX αντικειμένου

Στον παρακάτω οδηγό δημιουργίας ενός ActiveX Control, παρουσιάζω ένα απλό παράδειγμα δημιουργίας ActiveX Control πάνω στο οποίο δούλεψα μετέπειτα την επικοινωνία των drivers του Kinect με το ActiveX αντικείμενο.

4.1.1 Δημιουργία ενός Class Library στο Visual Studio 2010.

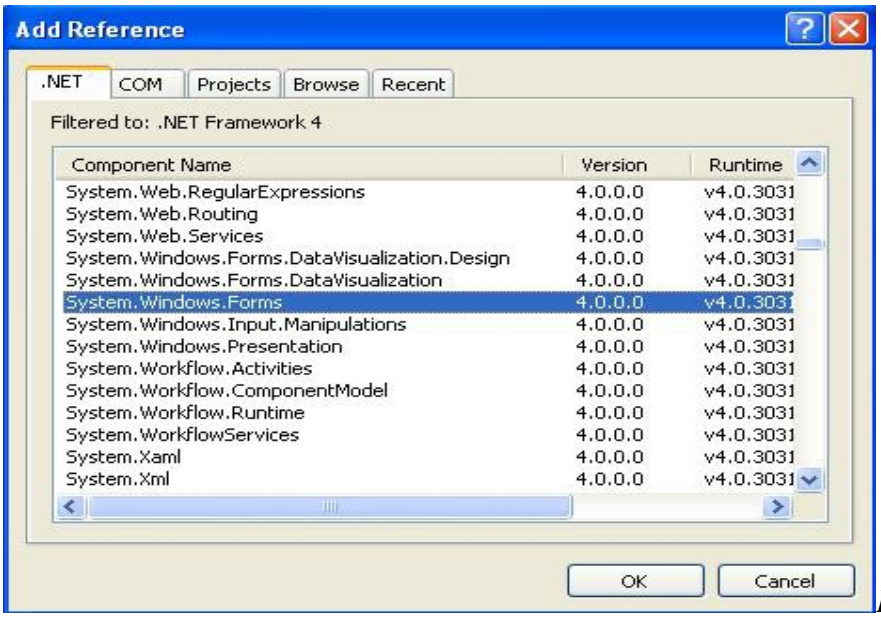
- Μόλις ανοίξουμε το Visual Studio 2010 επιλέγουμε: File → New → Project και στη συνέχεια διαλέγουμε την επιλογή Class Library η οποία βρίσκεται στη καρτέλα C#.
- Ονομάζουμε το Project 'AxControls' και πατάμε το OK.



Εικόνα 25: ActiveX - Class Library

4.1.2 Δημιούργησε μια νέα κλάση η οποία κληρονομεί απο το User Control

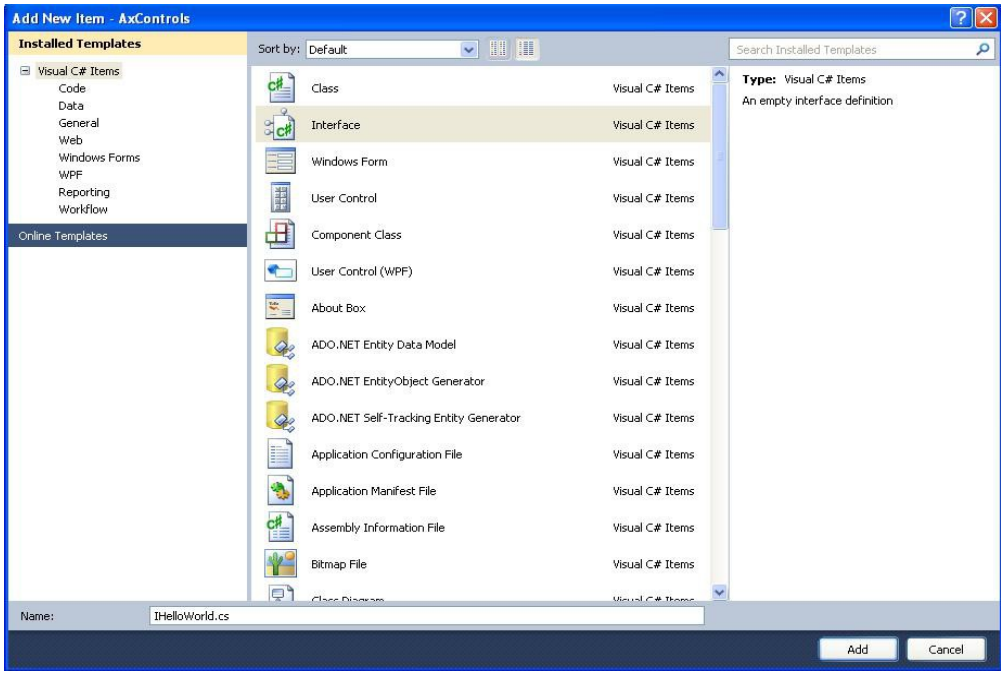
- Μετονόμασε τη κλάση 'Class1.cs' σε 'HelloWorld.cs'
- Βάλε την αναφορά στο πρότζεκτ(project reference) με όνομα 'System.Windows.Form'



Εικόνα 26: ActiveX - Reference

4.1.3 Δημιούργησε μια νέα διεπαφή η οποία θα εκθέτει τις μεθόδους ελέγχου(control methods) και τις ιδιότητες(properties) της στη COM

- Δεξί κλικ στο πρότζεκτ και επιλέγουμε Add → New Item.
- Επιλέγουμε Interface(διεπαφή) από τις διαθέσιμες επιλογές και της δίνουμε όνομα 'IHelloWorld' και κάνουμε κλικ στο Add.



Εικόνα 27: ActiveX - Interface

4.1.4 Διαμορφώνουμε το Interface 'IHelloWorld' σύμφωνα με τον παρακάτω κώδικα

```
using System;

using System.Collections.Generic;

using System.Text;

using System.Windows.Forms;

using System.Runtime.InteropServices;



namespace AxControls

{

    [ComVisible(true)]

    [InterfaceType(ComInterfaceType.InterfaceIsDual)]

    [Guid("E66C39CB-BB8B-4738-AA0E-5E0D1F2DB230")]

    public interface IHelloWorld

    {

        string GetText();

    }

}

[ComVisible(true)] makes the interface visible to COM.

[Guid("E66C39CB-BB8B-4738-AA0E-5E0D1F2DB230")] let's us manually assign a
GUID to the interface. Use guidgen.exe to generate your own.
```

4.1.5 Κάνε την κλάση ελέγχου(control class) να κληρονομήσει τη νέα μας διεπαφή(Interface).

Εφόσον σύμφωνα με τον παρακάτω κώδικα η κλάση μας κληρονομήσει τη νέα μας διεπαφή πρέπει να υλοποιήσουμε την μέθοδο της διεπαφής μας στη κλάση σύμφωνα με τις δικές μας ανάγκες.

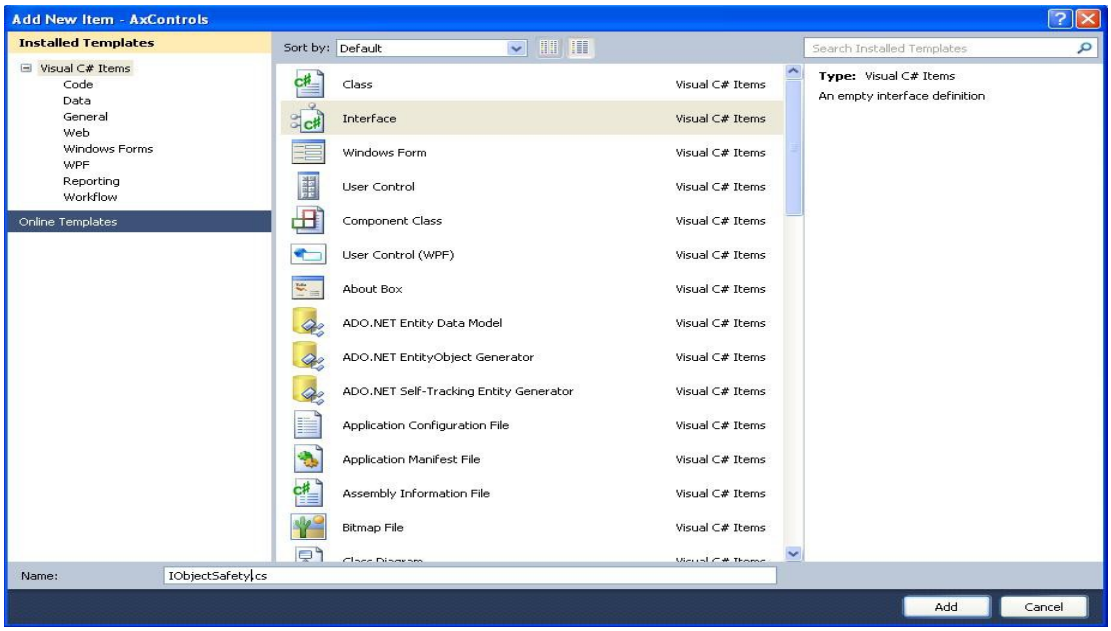
```
Using System;

using System.Collections.Generic;

using System.Text;

using System.Windows.Forms;

using System.Runtime.InteropServices;



namespace AxControls

{

    [ComVisible(true)]

    [ClassInterface(ClassInterfaceType.None)]

    [Guid("D100C392-030A-411C-92B6-4DBE9AC7AA5A")]

    [ProgId("AxControls.HelloWorld")]

    [ComDefaultInterface(typeof(IHelloWorld))]

    public class HelloWorld : UserControl, IHelloWorld

    {

        #region IHelloWorld Members

        public string GetText()

        {
```

```
            return "Hello ActiveX World!";

        }

        #endregion

    }

}
```

4.1.6 Κάνε τον έλεγχο ασφαλή για scripts που θα χρησιμοποιήσουμε και για αρχικοποίηση στον φυλομετρητή μας(browser).

Ο Internet Explorer(by Default) δεν επιτρέπει την αρχικοποίηση και το 'Scripting' σε ένα ActiveX Control εκτός και αν έχει σημαδευτεί ως ασφαλή. Αυτό σημαίνει ότι δε θα μπορούμε να δημιουργήσουμε στιγμιότυπα(instances) της ActiveX κλάσης μας με JavaScript by Default. Μπορούμε να προσπεράσουμε αυτό το πρόβλημά μας αν ρυθμίσουμε σωστά τις ρυθμίσεις ασφαλείας του browser μας, αλλά ο πιο σωστός τρόπος είναι να σημαδέψουμε το ActiveX Control ως ασφαλή. Πριν το κάνει αυτό όμως κάποιος σε ένα 'πραγματικό' ActiveX Control πρέπει να γνωρίζει καλά τους κινδύνους και τις πιθανές συνέπειες. Για να ξεπεράσουμε το πρόβλημά μας όμως τώρα, θα σημαδέψουμε τον έλεγχο(control) ως ασφαλή κληρονομώντας την διεπαφή 'IObjectSafety'.

- Δεξί κλικ στο πρότζεκτ μας και επιλέγουμε: Add → New Item
- Επέλεξε Interface(διεπαφή) από τις διαθέσιμες επιλογές και δώσε της όνομα 'IObjectSafety' και κάνε κλικ στο Add.



Εικόνα 28: ActiveX - Interface for Safety

- Στη συνέχεια διαμόρφωσε το κώδικα της 'IObjectSafety' σύμφωνα με τα παρακάτω:

```
using System;

using System.Collections.Generic;

using System.Text;

using System.Runtime.InteropServices;



namespace AxControls

{

    [ComImport()]

    [Guid("51105418-2E5C-4667-BFD6-50C71C5FD15C")]

    [InterfaceType(ComInterfaceType.InterfaceIsIUnknown)]

    interface IObjectSafety

    {

        [PreserveSig()]

        int GetInterfaceSafetyOptions(ref Guid riid, out int
pdwSupportedOptions, out int pdwEnabledOptions);

        [PreserveSig()]

        int SetInterfaceSafetyOptions(ref Guid riid, int dwOptionSetMask,
int dwEnabledOptions);

    }

}
```

- Στη συνέχεια κάνε την κλάση 'HelloWorld' να κληρονομεί τη διεπαφή(interface) 'IObjectSafety'. Το τελικό αποτέλεσμα φαίνεται παρακάτω:

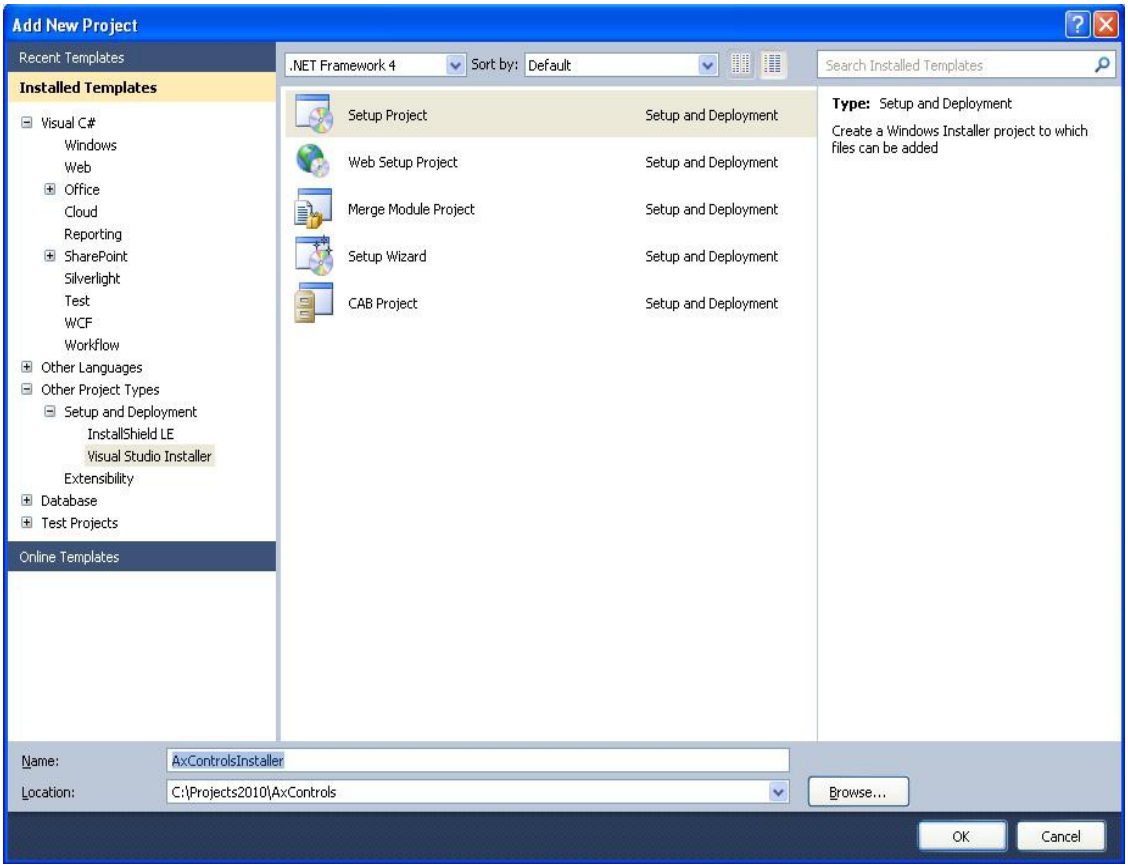
```
using System;

using System.Collections.Generic;

using System.Text;

using System.Windows.Forms;

using System.Runtime.InteropServices;



namespace AxControls

{

    [ComVisible(true)]

    [ClassInterface(ClassInterfaceType.None)]

    [Guid("D100C392-030A-411C-92B6-4DBE9AC7AA5A")]

    [ProgId("AxControls.HelloWorld")]

    [ComDefaultInterface(typeof(IHelloWorld))]

    public class HelloWorld : UserControl, IHelloWorld, IObjectSafety

    {

        #region IHelloWorld Members

        public string GetText()

        {

            return "Hello ActiveX World!";

        }

        #endregion

        #region IObjectSafety Members
```

```
        public enum ObjectSafetyOptions

        {

            INTERFACESAFE_FOR_UNTRUSTED_CALLER = 0x00000001,

            INTERFACESAFE_FOR_UNTRUSTED_DATA = 0x00000002,

            INTERFACE_USES_DISPEX = 0x00000004,

            INTERFACE_USES_SECURITY_MANAGER = 0x00000008

        };

        public int GetInterfaceSafetyOptions(ref Guid riid, out int
pdwSupportedOptions, out int pdwEnabledOptions)

        {

            ObjectSafetyOptions m_options =
ObjectSafetyOptions.INTERFACESAFE_FOR_UNTRUSTED_CALLER |
ObjectSafetyOptions.INTERFACESAFE_FOR_UNTRUSTED_DATA;

            pdwSupportedOptions = (int) m_options;

            pdwEnabledOptions = (int) m_options;

            return 0;

        }

        public int SetInterfaceSafetyOptions(ref Guid riid, int
dwOptionSetMask, int dwEnabledOptions)

        {

            return 0;

        }

        #endregion

    }

}
```

4.1.7 Δημιούργησε έναν .msi installer για το ActiveX Control

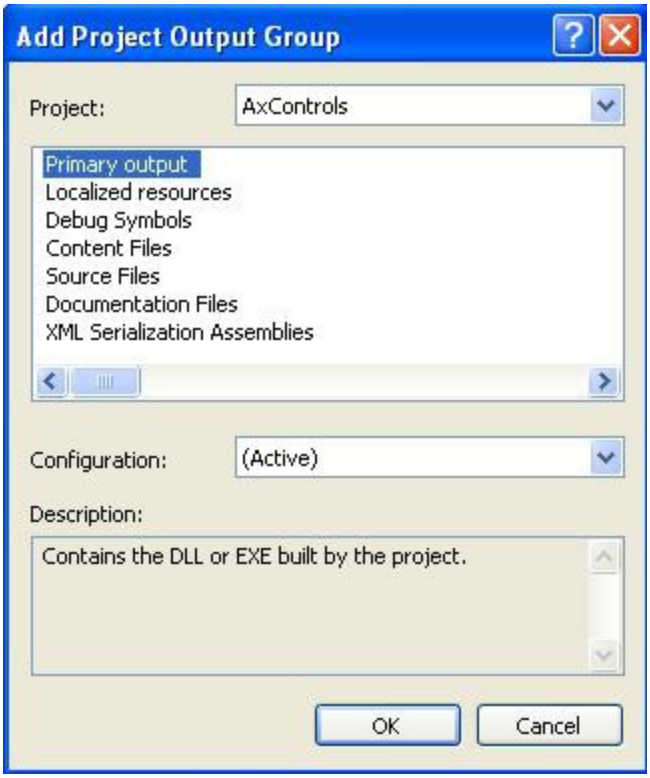
Πριν χρησιμοποιηθεί το ActiveX Control πρέπει πρώτα να εγκατασταθεί και να καταχωρηθεί στον Client. Αυτό μπορεί να γίνει με πολλούς τρόπους, όπως χειροκίνητα να τροποποιήσουμε τη registry για να χρησιμοποιήσουμε το regasm.exe αλλά εμείς θέλουμε κάτι πολύ πιο απλό. Για αυτό και θέλουμε έναν εύχρηστο msi installer για να μην μπλέκεται ο χρήστης με περιττές πληροφορίες και κόπο.

- Δεξί κλικ στο Visual Studio Solution, επιλέγουμε Add → New Project και επιλέγουμε Setup Project από τη καρτέλα Other Project Types.
- Ονομάζουμε το πρότζεκτ 'AxControlsInstaller' και πατάμε OK.



Εικόνα 29: ActiveX - msi installer

- Μετά πατάμε δεξί κλικ στο πρότζεκτ 'AxControlsInstaller', Add → Project Output και επιλέγουμε 'Primary Output' από τη κατηγορία 'AxControls' και πατάμε OK.



Εικόνα 30: ActiveX - msi installer output

- Στη συνέχεια κάνουμε δεξί κλικ στο 'Primary output from AxControls (Active)' και επιλέγουμε ιδιότητες(properties).
- Αλλάζουμε την ιδιότητα 'Register' που μόλις εμφανίστηκε από 'vsdrpDoNotRegister' σε 'vsdrpCOM'.
- Τέλος κάνουμε δεξί κλικ στο πρότζεκτ 'AxControlsInstaller' και επιλέγουμε την εντολή 'Build'

Ο Installer τώρα πλέον βρίσκεται στον output φάκελο(bin\Debug ή bin\Release) του AxControlsInstaller. Ο msi installer μπορεί να τρέξει χειροκίνητα από τον χρήστη ή αυτόματα με Group policy.

4.1.8 Πακετάρισμα του installer σε ένα .cab αρχείο για να είναι διαθέσιμος στο διαδίκτυο.

Στους ιστοχώρους(Web Sites) δεν μπορούμε να χρησιμοποιήσουμε το ActiveX Control στον Client με Group Policy. Σε αυτή τη περίπτωση είμαστε υποχρεωμένοι να χρησιμοποιήσουμε τη δυνατότητα του Internet Explorer να 'κατεβάζει' και να εγκαθιστεί ActiveX Controls τα οποία βρίσκονται πακεταρισμένα σε .cab αρχεία.

- Κατεβάζουμε το Microsoft Cabinet Software Development Kit.
- Ξεπακετάρουμε το kit σε ένα φάκελο και αντιγράφουμε το αρχείο Cabarc.exe στο φάκελο του 'AxControlsInstaller'.
- Δημιουργούμε ένα νέο αρχείο στο φάκελο του 'AxControlsInstaller' το οποίο το ονομάζουμε 'AxControls.inf' και του προσθέτουμε το παρακάτω περιεχόμενο.

```
[version]

signature="$CHICAGO$"

AdvancedINF=2.0



[Add.Code]

AxControlsInstaller.msi=AxControlsInstaller.msi



[AxControlsInstaller.msi]

file-win32-x86=thiscab

clsid={1FC0D50A-4803-4f97-94FB-2F41717F558D}

FileVersion=1,0,0,0



[Setup Hooks]

RunSetup=RunSetup
```

```
[RunSetup]

run="""msiexec.exe""" /i """%EXTRACT_DIR%\AxControlsInstaller.msi""" /qn
```

- Κλικάρουμε το πρότζεκτ 'AxControlsInstaller' και πατάμε F4 ή αλλίως View → Properties Windows εάν δεν εμφανίζεται
- Κλικάρουμε το κουμπί '...' το οποίο βρίσκεται δίπλα από το 'PostBuildEvent' και προσθέτουμε τον ακόλουθο κώδικα

```
"$(ProjectDir)\CABARC.EXE" N "$(ProjectDir)AxControls.cab"

"$(ProjectDir)AxControls.inf" "$(ProjectDir)$(Configuration)\AxControlsInstaller.msi"
```

- Μετά δεξί κλικ στο πρότζεκτ 'AxControlsInstaller' και επιλέγουμε Build.
- Τώρα δημιουργήθηκε το αρχείο 'AxControls.cab' στο φάκελο του 'AxControlsInstaller'.

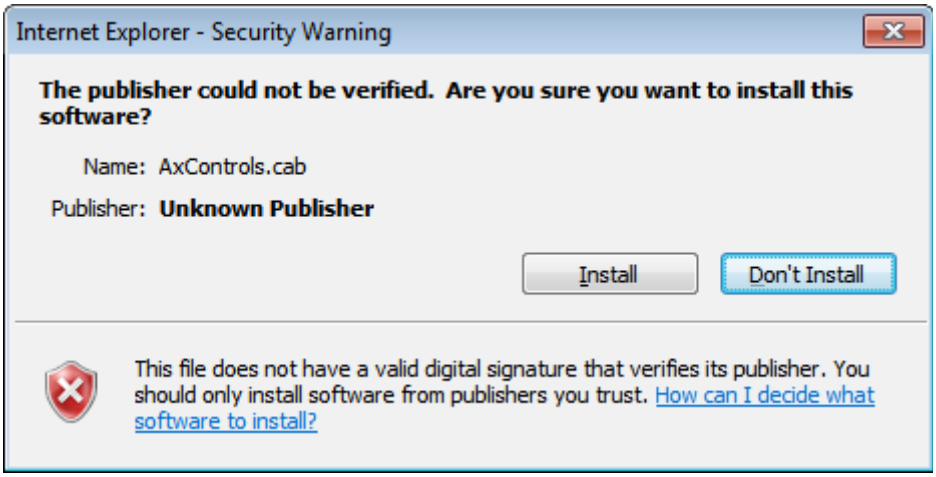
4.1.9 Αρχικοποιούμε και τεστάρουμε με JavaScript

- Δεξί κλικ στο 'AxControls' solution, Add → New Project και επιλέγουμε 'ASP .Net Web Application' το οποίο βρίσκεται στη καρτέλα 'Web'.
- Ονομάζουμε το πρότζεκτ 'WebAppTest' και πατάμε OK.
- Δεξί κλικ στο πρότζεκτ 'WebAppTest', Add → New Item και επιλέγουμε 'HTML Page'.
- Το ονομάζουμε 'index.html' και πατάμε OK.
- Τέλος προσθέτουμε το παρακάτω κείμενο στο 'index.html'.

```
<html>

    <head>



        <objectname="axHello" style='display:none' id='axHello' classid='CLSID:D100C392-030A-411C-92B6-4DBE9AC7AA5A'

 codebase='AxControls.cab#version=1,0,0,0'></object>
```

```
        <script language="javascript">



            <!-- Load the ActiveX object  -->

            var x = new ActiveXObject("AxControls.HelloWorld");



            <!-- Display the String in a messagebox -->

            alert(x.GetText());



        </script>

    </head>

    <body>

    </body>

</html>
```

- Δεξί κλικ στο 'index.html' και επιλέγουμε 'Set as start page'.
- Δεξί κλικ στο πρότζεκτ 'WebAppTest' και επιλέγουμε 'Set as startup project'.
- Αντιγράφουμε το 'AxControls.cab' από το φάκελο 'AxControlsInstaller' στον ίδιο φάκελο που βρίσκεται το 'index.html'.
- Κάνουμε απεγκατάσταση το ActiveX Control από τον Client πηγαίνοντας Control Panel → Add or Remove και επιλέγουμε το 'AxControlsInstaller' από τη λίστα επιλέγοντας Unistall. Αυτή η ενέργεια που κάναμε μόλις εξαναγκάζει τον Internet Explorer να 'κατεβάσει' και να εγκαταστήσει το αρχείο cab.
- Τρέχουμε την εφαρμογή μας πατώντας F5. Αυτό θα μας ανοίξει την 'index.html' στον Internet Explorer.
- Ο Internet Explorer θα μας εμφανίσει μία ειδοποίηση ασφαλείας, ρωτώντας μας αν θέλουμε να εγκαταστήσουμε το 'AxControls.cab' και φυσικά επιλέγουμε Install.



Εικόνα 31: ActiveX - Security Warning

4.2 Επικοινωνία Kinect με ActiveX Object

Εφόσον έχουμε δημιουργήσει το ActiveX Object στη προηγούμενη ενότητα τώρα καλούμαστε να το παραμετροποιήσουμε έτσι ώστε να λειτουργεί σύμφωνα με τις δικές μας ανάγκες για την ορθή επικοινωνία του Kinect με το ActiveX Object.

4.2.1 Προσθέτουμε όσες μεθόδους χρειαζόμαστε στο Interface 'IHelloWorld'

Πηγαίνουμε στο AxControls το οποίο είχαμε δημιουργήσει στη προηγούμενη ενότητα και βρίσκουμε το Interface 'IHelloWorld' στο οποίο δηλώσαμε τη μέθοδο που εκθέσαμε στο COM. Εκεί θα προσθέσουμε τις υπόλοιπες μεθόδους τις οποίες θα χρειαστούμε, όπως φαίνεται στο παρακάτω κώδικα.

```
1  using System;
2  using System.Collections.Generic;
3  using System.Text;
4  using System.Runtime.InteropServices;
5  using System.Drawing;
6
7  namespace AxControls
8  {
9      [ComVisible(true)]
10     [InterfaceType(ComInterfaceType.InterfaceIsDual)]
11     [Guid("E66C39CB-BB8B-4738-AA0E-5E0D1F2DB230")]
12     public interface IHelloWorld
13     {
14         String getText();
15         String getBase64ImageString();
16         String getBase64DepthString();
17         String getStringValue();
18         Boolean startKinect();
19         Boolean kinectIsRunning();
20         String getJsonObject();
21         void stopKinect();
22         Bitmap getBitmap();
23         int getRightHandX();
24         int getRightHandY();
25         int getLeftHandX();
26         int getLeftHandY();
27         //Image getImage();
28     }
29 }
```

Εικόνα 32: Kinect Connection - Interface

4.2.2 Επικοινωνία ActiveX Control με το API του Kinect

Πηγαίνουμε στη κλάση 'HelloWorld' του AxControls στην οποία υπάρχει το μεγαλύτερο κομμάτι κώδικα για την επικοινωνία του Kinect με το AxControls. Όπως βλέπουμε στον παρακάτω κώδικα υπάρχουν κάποιες εντολές οι οποίες ενεργοποιούν συγκεκριμένες λειτουργίες του Kinect μέσα από το API του. Επίσης μπορούμε να παρατηρήσουμε την μέθοδο 'startKinect()' την οποία δηλώσαμε στο Interface και τώρα καλούμαστε να την υλοποιήσουμε στη κλάση

```
42      public bool startKinect()
43      {
44          kinect = KinectSensor.KinectSensors[0];
45          kinect.ColorStream.Enable();
46          kinect.ColorFrameReady += new EventHandler<ColorImageFrameReadyEventArgs>(Nui_ColorImageFrameReady);
47          kinect.SkeletonStream.Enable();
48          kinect.SkeletonFrameReady += new EventHandler<SkeletonFrameReadyEventArgs>(Nui_SkeletonFrameReady);
49          //kinect.DepthStream.Enable(DepthImageFormat.Resolution640x480Fps30);
50          //kinect.DepthFrameReady += new EventHandler<DepthImageFrameReadyEventArgs>(Nui_DepthImageFrameReady);
51          kinect.Start();
52          kIsRunning = true;
53
54          return true;
55      }
```

Εικόνα 33: Kinect Connection - API

- Στη γραμμή 44 εκχωρούμε στη μεταβλητή 'kinect' την πρώτη συσκευή kinect η οποία είναι συνδεμένη στον υπολογιστή μας. Υπάρχει το ενδεχόμενο να συνδέσουμε παραπάνω από μία συσκευή kinect στον υπολογιστή μας για αυτό εμείς εδώ διαλέγουμε να χρησιμοποιήσουμε την πρώτη που συνδέθηκε στον υπολογιστή μας.
- Στη γραμμή 45 χρησιμοποιούμε την μεταβλητή 'kinect' για να ενεργοποιήσουμε την έγχρωμη ροή εικόνας προς εμάς. Μπορούμε να ορίσουμε αν θέλουμε το format της εικόνας το οποίο θέλουμε να παίρνουμε μέσα από τις διαθέσιμες επιλογές τις οποίες μας δίνει το API του kinect. Η προεπιλεγμένη τιμή του format είναι : ColorImageFormat.RgbResolution640x480Fps30. Δηλαδή κωδικοποίηση:RGB, ανάλυση εικόνας:640x480 και καρέ ανά δευτερόλεπτο:30.
- Στη γραμμή 46, όπως βλέπουμε, ενεργοποιούμε έναν Event Handler έτσι ώστε σε κάθε καρέ(frame) που θα στέλνει το kinect να εκτελείται η μέθοδος 'Nui_ColorImageFrameReady' έτσι ώστε να μπορούμε εμείς να το επεξεργαστούμε ή να το στείλουμε κατευθείαν στον browser.
- Στη γραμμή 47 γίνεται ότι και στη γραμμή 45 αλλά με τη διαφορά ότι η ροή δεδομένων μας έχει να κάνει με τα δεδομένα του σκελετού και όχι της εικόνας.
- Στη γραμμή 48 γίνεται ότι και στη γραμμή 46 αλλά για τη ροή δεδομένων σκελετού.

- Στις γραμμές 49 και 50 γίνονται τα ίδια αλλά για τα δεδομένα τα οποία παίρνουμε από τον αισθητήρα βάθους (Depth Sensor).
- Στη γραμμή 51 ενεργοποιούμε όλα τα παραπάνω που αναφέραμε και από εκείνη τη στιγμή και μετά ενεργοποιείται το kinect του οποίου μπορούμε πλέον να χρησιμοποιήσουμε τα δεδομένα που εξάγει.

4.2.3 Πως χρησιμοποιούμε τα δεδομένα τα οποία παίρνουμε από το kinect

Στο παρακάτω κώδικα βλέπουμε την λειτουργία του Event Handler που διαχειρίζεται Events σχετικά με την εικόνα που στέλνει σε frames η κάμερα του kinect. Στο προηγούμενο κώδικα που αναλύσαμε εκτενώς αναφέραμε σε κάποιο σημείο ότι ο Event Handler που διαχειρίζεται τα καρέ εικόνας που έρχονται από το kinect υποχρεώνει τη ροή του προγράμματος μας να συμπεριλάβει ένα ακόμα κομμάτι κώδικα.. Αυτό το κομμάτι κώδικα θα το αναλύσουμε παρακάτω.

```
201     static void Nui_ColorImageFrameReady(object sender, ColorImageFrameReadyEventArgs e)
202     {
203
204
205         ColorImageFrame imageFrame = e.OpenColorImageFrame();
206         if (imageFrame != null)
207         {
208             byte[] pixeldata = new byte[imageFrame.PixelDataLength];
209             imageFrame.CopyPixelDataTo(pixeldata);
210
211             Bitmap bmp = new Bitmap(640, 480, PixelFormat.Format32bppArgb);
212             BitmapData bmpData = bmp.LockBits(new Rectangle(0, 0, bmp.Width, bmp.Height), ImageLockMode.WriteOnly, bmp.PixelFormat);
213
214             Marshal.Copy(pixeldata, 0, bmpData.Scan0, pixeldata.Length);
215
216             byte[] BMPBytes;
217             using (MemoryStream ms = new MemoryStream())
218             {
219                 bmp.Save(ms, System.Drawing.Imaging.ImageFormat.Jpeg);
220                 BMPBytes = ms.ToArray();
221             }
```

Εικόνα 34: Kinect Connection - ImageFrameReady

- Στη γραμμή 205 βλέπουμε ότι δημιουργείται ένα αντικείμενο(imageFrame) τύπου 'ColorImageFrame' το οποίο παίρνει από το αντίστοιχο Event κάποια δεδομένα. Αυτά τα δεδομένα είναι οι διαστάσεις της εικόνας, το format το οποίο καθορίσαμε να έχει και όλες τις πληροφορίες τις οποίες μπορούμε να εξάγουμε από τα pixels της

εικόνας(έχουμε αναφέρει σε προηγούμενο κεφάλαιο πληροφορίες σχετικά με τα pixels)

- Στη γραμμή 206 κάνουμε έναν έλεγχο για να δούμε αν μας επέστρεψε τίποτα το Event.
- Στη γραμμή 208 δημιουργούμε ένα πίνακα από bytes. Το μέγεθος που του ορίζουμε να έχει το παίρνουμε από τις πληροφορίες που μας παρέχει το imageFrame όπως προανέφερα.
- Στη γραμμή 209 αντιγράφουμε τα δεδομένα των pixels στον πίνακα που δημιουργήσαμε στη γραμμή 208.
- Στις γραμμές 211 με 214 δημιουργούμε ένα αντικείμενο τύπου Bitmap στο οποίο βάζουμε τα δεδομένα του πίνακα pixeldata.
- Στις γραμμές 216 με 220, εφόσον έχουμε βάλει την εικόνα που παίρνουμε από το kinect στο αντικείμενο bmp, βάζουμε τώρα τα bytes του bmp σε έναν άλλο πίνακα, τον BMPBytes. Αυτά τα bytes όμως τώρα δεν περιγράφουν τα pixels της εικόνας που πήραμε από το kinect αλλά τα bytes από τα οποία αποτελείται οποιοδήποτε αρχείο στο λειτουργικό μας σύστημα και στον σκληρό μας δίσκο.

```
260             string base64ImageString = Convert.ToBase64String(BMPBytes);
261             srcBase64ImageString = "data:;base64," + base64ImageString;
262
263             imageFrame.Dispose();
264             //JPGImage.Dispose();
265             bmp.Dispose();
266             GC.Collect();
```

Εικόνα 35: Kinect Connection - Convert to Base64 String

- Στη γραμμή 260 μετατρέπουμε τα bytes του bmp(BMPBytes) σε string της μορφής base64. Αυτή η μετατροπή γίνεται διότι είναι ο μόνος τρόπος να περάσουμε τη πληροφορία της εικόνας σε ένα canvas του browser.
- Στη γραμμή 261 βάζουμε στο τελικό μας πλέον String όλα τα απαραίτητα δεδομένα για να μπορεί να διαβαστεί από ένα canvas στο browset.
- Στη γραμμή 263 αποδεσμεύουμε από τη μνήμη το αντικείμενο imageFrame, το οποίο δε μας χρειάζεται πλέον, διότι αλλιώς μέσα σε λίγα δευτερόλεπτα θα είχε γεμίσει όλη η μνήμη του υπολογιστή μας.
- Στη γραμμή 265 αποδεσμεύουμε το bmp για τους ίδιους λόγους με πριν.
- Στη γραμμή 266 εξαναγκάζουμε το πρόγραμμά μας να καλέσει τον Garbage Collector και να αδειάσει ότι αχρείαστο κομμάτι υπάρχει στη μνήμη το οποίο δε θα ξαναχρησιμοποιήσει το πρόγραμμά μας.

Με παρόμοιο τρόπο χρησιμοποιούμε τα άλλα δύο streams που μας παρέχει το kinect(Depth Stream και Skeleton Stream) για να εξάγουμε τη πληροφορία που χρειαζόμαστε

κάθε φορά από το kinect. Δεν είναι υποχρεωτικό να ενεργοποιήσουμε και να χρησιμοποιήσουμε όλα τα streams που μας δίνουν οι αισθητήρες του kinect παρά μόνο όσα χρειαζόμαστε. Η χρήση όλων των αισθητήρων μπορεί να προκαλέσει μεγάλη καθυστέρηση στη μεταφορά των δεδομένων από το kinect στο browser αν δε γράψουμε ποιοτικό και αποδοτικό κώδικα.

```
303         static void Nui_SkeletonFrameReady(object sender, SkeletonFrameReadyEventArgs e)
304         {
305             //if (!_initialized) return;
306 
307             Skeleton[] skeletonData = null;
308 
309             SkeletonFrame skeletonFrame = e.OpenSkeletonFrame();
310             if (skeletonFrame != null)
311             {
312                 // DISPLAY
313                 //OR PROCESS IMAGE DATA IN pixelData HERE
314                 //List<Skeleton> users = new List<Skeleton>();
315 
316                 //if (skeletonData == null || skeletonData.Length != kinect.SkeletonStream.FrameSkeletonArrayLength)
317                 //{
318                 //    skeletonData = new Skeleton[kinect.SkeletonStream.FrameSkeletonArrayLength];
319                 //}
320                 skeletonData = new Skeleton[kinect.SkeletonStream.FrameSkeletonArrayLength];
321 
322                 skeletonFrame.CopySkeletonDataTo(skeletonData);
323 
324 
325                 foreach (var skeleton in skeletonData)
326                 {
327                     if (skeleton.TrackingState == SkeletonTrackingState.Tracked)
328                     {
329                         foreach (Joint joint in skeleton.Joints)
330                         {
331                             if (JointType.HandRight == joint.JointType)
332                             {
333                                 //float posX = joint.Position.X;
334                                 //float posY = joint.Position.Y;
335                                 Joint scaledJoint = joint.ScaleTo(640, 480);
336                                 rightHandX = (int)scaledJoint.Position.X;
337                                 rightHandY = (int)scaledJoint.Position.Y;
338                             }
339                             else if (JointType.HandLeft == joint.JointType)
340                             {
341                                 //float posX = joint.Position.X;
342                                 //float posY = joint.Position.Y;
343                                 Joint scaledJoint = joint.ScaleTo(640, 480);
344                                 leftHandX = (int)scaledJoint.Position.X;
345                                 leftHandY = (int)scaledJoint.Position.Y;
346                             }
347                         }
```

Εικόνα 36: Kinect Connection - SkeletonFrameReady

Στο παραπάνω κομμάτι κώδικα βλέπουμε τη ροή δεδομένων που σχετίζεται με το σκελετό που αναγνωρίζει το kinect. Όταν χρησιμοποιούμε αυτή τη ροή τότε αυτόματα ενεργοποιείται και η ροή του βάθους. Αυτό είναι αναμενόμενο διότι για να μπορέσει το kinect να σου στείλει τα σημεία του σκελετού πρέπει να πάρει πρώτα τα δεδομένα από τη ροή βάθους(Depth Stream) και στη συνέχεια, εφόσον τα επεξεργαστεί, να είναι πλέον έτοιμο για να σου τα στείλει. Το αντίθετο δεν συμβαίνει, δηλαδή όταν ενεργοποιούμε τη ροή βάθους(Depth Stream) δεν ενεργοποιείται αυτόματα και η ροή που σχετίζεται με τα σημεία του σκελετού(Skeleton Stream).

- Στη γραμμή 309 βλέπουμε ότι δημιουργείται ένα αντικείμενο(skeletonFrame) τύπου 'SkeletonFrame' το οποίο παίρνει από το αντίστοιχο Event κάποια δεδομένα. Αυτά τα δεδομένα είναι τα 20 σημεία του σκελετού που αναγνωρίζει και επιστρέφει σε εμάς το kinect(έχουν αναφερθεί σε προηγούμενο κεφάλαιο).
- Στις γραμμές 320 και 322 δημιουργούμε έναν πίνακα, τον skeletonData, και στη συνέχεια τον γεμίζουμε με τα δεδομένα τα οποία παίρνουμε από το skeletonFrame.
- Στις γραμμές 325 με 329 παίρνουμε ένα προς ένα τα σημεία που μας επιστρέφονται από το kinect και τα χρησιμοποιούμε σύμφωνα με τις ανάγκες μας.
- Στη γραμμή 331 γίνεται ο έλεγχος για να δούμε αν το σημείο το οποίο διαχειριζόμαστε εκείνη τη στιγμή είναι το δεξί χέρι του σκελετού του οποίου αναγνώρισε το kinect.
- Στη γραμμή 335 το σημείο το οποίο διαχειριζόμαστε το μετατρέπουμε έτσι ώστε να μπαίνει σε παράθυρο διαστάσεων 640x480.
- Στις γραμμές 336 με 337 παίρνουμε τη x συντεταγμένη του σημείου που μετατρέψαμε προηγουμένως και τη y συντεταγμένη του ίδιου σημείου αντίστοιχα.
- Στις γραμμές 339 με 345 συμβαίνει ακριβώς το ίδιο αλλά για τις συντεταγμένες του αριστερού χεριού.

Έτσι καταφέραμε και αναγνωρίσαμε από το kinect την ακριβή θέση στην οποία βρίσκεται το δεξί και το αριστερό μας χέρι. Από όλα τα σημεία του σκελετού που μας επιστράφηκαν, εμείς επιλέξαμε δύο μόνο, διότι τόσα χρειαζόντουσαν οι ανάγκες μας. Αν θέλαμε για παράδειγμα μαζί με την εικόνα να φαινόντουσαν και τα σημεία του σκελετού τότε θα διαμορφώναμε τον κώδικά μας σύμφωνα με τις νέες ανάγκες μας.

Εικόνα 37: Kinect Connection - Skeleton Points

4.3 Χρήση ActiveX αντικειμένου στον Internet Explorer

Σε αυτό το σημείο έχουμε καταφέρει να φτιάξουμε ότι είναι απαραίτητο για να διαχειριστούμε και να επεξεργαστούμε τα δεδομένα του kinect στον Internet Explorer. Η διαχείριση και επεξεργασία των δεδομένων εύκολα μπορεί να γίνει με τη χρήση της JavaScript μέσα στην HTML σελίδα όπως θα δούμε παρακάτω.

4.3.1 Δημιουργία ActiveX αντικειμένου και ενεργοποίηση του kinect

Τώρα θα δούμε με ποιο τρόπο μπορούμε να δημιουργήσουμε και να αρχικοποιήσουμε το ActiveX αντικείμενο του οποίου σε προηγούμενα κεφάλαια διαμορφώσαμε τη δομή του και τη λειτουργία του.

```
63          function startKinectDevice() {
64              activeX = new ActiveXObject("AxControls.HelloWorld");
65              activeX.startKinect();
66          }
```

Εικόνα 38: Browser Scripting - Enable Kinect

- Στις γραμμές 63 με 66 τοποθετούμε τον κώδικά μας μέσα σε μία συνάρτηση, όπως και όλα τα άλλα κομμάτια κώδικα που χρησιμοποιούμε, για να είναι ευανάγνωστος και εύκολα διαχειρίσιμος.
- Στη γραμμή 64 μπορούμε να δούμε πως δημιουργείται επιτέλους το ActiveX αντικείμενο μας. Βλέπουμε ότι από το Class Library 'AxControls' χρησιμοποιούμε τη κλάση 'HelloWorld' για να έχουμε διαθέσιμες στην JavaScript όλες τις μεθόδους που φορτώσαμε στο ActiveX αντικείμενο.
- Στη γραμμή 65 βλέπουμε πως χρησιμοποιούμε τη μέθοδο 'startKinect()' την οποία είχαμε φορτώσει στο ActiveX αντικείμενό μας. Αυτή η μέθοδος, όπως είδαμε και σε προηγούμενη ενότητα, έχει μέσα της τον απαραίτητο κώδικα για να ενεργοποιήσουμε το kinect σύμφωνα την παραμετροποίηση την οποία είχαμε κάνει όταν γράφαμε αυτή τη μέθοδο.

4.3.2 Απενεργοποίηση του kinect

Τώρα θα δούμε με ποιο τρόπο μπορούμε να απενεργοποιήσουμε το kinect μέσα από το ActiveX αντικείμενο.

```
68          function stopKinectDevice() {
69              activeX.stopKinect();
70              kinectStopped = true;
71          }
```

Εικόνα 39: Browser Scripting - Disable Kinect

- Στη γραμμή 69 βλέπουμε τη μέθοδο 'stopKinect' η οποία όπως προαναφέραμε σταματάει τη λειτουργία του kinect.

4.3.3 Λήψη συντεταγμένων από το ActiveX αντικείμενο

Στις περισσότερες εφαρμογές που δημιούργησα έκανα χρήση των χεριών μου για αλληλεπίδραση με την εφαρμογή. Το kinect μέσα από το skeletonStream αναγνωρίζει τις συντεταγμένες των χεριών όπως και των άλλων σημείων του σκελετού μας και με την κατάλληλη διαχείριση τους στο ActiveX Αντικείμενο μπορούμε να τις διαχειριστούμε και να τις χρησιμοποιήσουμε.

```
174         function updateKinectPositionVariables() {
175
176             rightHandX = activeX.getRightHandX();
177             rightHandY = activeX.getRightHandY();
178             leftHandX = activeX.getLeftHandX();
179             leftHandY = activeX.getLeftHandY();
180         }
```

Εικόνα 40: Browser Scripting - Get Coordinates

- Η συνάρτηση που βλέπουμε πρέπει να καλείται πολύ συχνά έτσι ώστε οι συντεταγμένες των χεριών μας τις οποίες χρησιμοποιούν διάφορες εφαρμογές να είναι ενημερωμένες. Στις εφαρμογές τις οποίες έχω φτιάξει η συνάρτηση αυτή καλείται κάθε ένα χιλιοστό του δευτερολέπτου(1ms).
- Στις γραμμές 176 με 179 εκχωρούμε τις τιμές, τις οποίες παίρνουμε από το

ActiveX αντικείμενο, στις μεταβλητές τις οποίες χρησιμοποιούν οι εφαρμογές μας.

4.3.4 Διαχείριση σημείων σκελετού που λάβαμε από το ActiveX αντικείμενο

Εδώ θα δούμε τώρα την διαχείριση των δεδομένων που λαμβάνουμε από το kinect μέσω του ActiveX αντικειμένου.

```
88          function drawSkeleton() {
89              //          var jsonObject = eval('(' + evt.data + ')');
90              var jsonObject = activeX.getJsonObject();
91
92              if (false) {//jsonObject != ""
93                  context.clearRect(0, 0, canvas.width, canvas.height);
94                  context.fillStyle = "#FF5555";
95                  context.beginPath();
96
97                  var colSel = true;
98                  var bool = true;
99                  // Display the skeleton joints.
100                 for (var i = 0; i < jsonObject.skeletons.length; i++) {
101                     for (var j = 0; j < jsonObject.skeletons[i].joints.length; j++) {
102                         var joint = jsonObject.skeletons[i].joints[j];
103                         // Draw!!!
104                         context.arc(parseFloat(joint.x), parseFloat(joint.y), 15, 0, Math.PI * 2, true);
```

Εικόνα 41: Browser Scripting - Get Skeleton Points

35 17 Στη γραμμή 90 παίρνουμε από το ActiveX αντικείμενο τα σημεία του σκελετού σε μορφή json.

35 17 Στις γραμμές 93 με 95 ρυθμίζουμε τον Canvas ο οποίος υπάρχει στην HTML5 έτσι ώστε να ζωγραφίσει τα σημεία σύμφωνα με τις δικές μας ρυθμίσεις. Στη γραμμή 93 καθαρίζει την περιοχή στην οποία θα ζωγραφιστούν τα σημεία. Στη γραμμή 94 δίνουμε το χρώμα το οποίο θα χρησιμοποιήσουμε και στη γραμμή 95 δίνουμε εντολή ότι από τότε και μετά είναι επιτρεπτό να ζωγραφίσουμε.

35 17 Στις γραμμές 100 με 102 παίρνουμε ένα προς ένα τα σημεία που μας επιστράφηκαν από το ActiveX αντικείμενο.

35 17 Στη γραμμή 104 ζωγραφίζουμε έναν κύκλο σύμφωνα με το συντακτικό της HTML5 που αφορά το Canvas.

4.3.5 Επικοινωνία kinect με X3DOM

Εδώ είναι μια σύντομη δοκιμή(χρησιμοποιώντας το SkeletonFrame) που έκανα για να αποδείξω ότι το kinect μπορεί να επικοινωνήσει και με X3D αντικείμενα στο browser με JacaScript. Εδώ μας ανοίγονται νέοι ορίζοντες στην έρευνα καθώς και οι δύο τεχνολογίες είναι ότι πιο καινούριο υπάρχει στο χώρο και μπορούν να συνδυαστούν για να βγάλουν μαγικά αποτελέσματα τα οποία θα χρησιμοποιηθούν σε καινοτόμες εφαρμογές στο άμεσο μέλλον.

```
105                 //              if (joint.name == "hipcenter") {
106
107
108                 //                  colSel = !colSel;
109
110                 //                  var mat = document.getElementsByTagName("Material");
111                 //                  var n = mat.length;
112                 //                  var aMat = mat[0];
113                 //                  aMat.setAttribute("diffuseColor", (!colSel ? "0 0 1" : "1 0 0"));
114
115                 //                  var colorValue = aMat.getAttribute("diffuseColor");
116                 //                  type.innerHTML = " -   ColorValue: " + colorValue;
117
118                 //              if (bool) {
119                 //                  //                  var tra = document.getElementsByTagName("Box");
120                 //                  //                  var num = tra.length;
121                 //                  //                  var aTra = tra[0];
122                 //                  //                  aTra.setAttribute("size", "5 5 5);
123
124                 //                  bool = false;
125                 //              }
126
127                 //              }
```

Εικόνα 42: Browser Scripting - X3DOM

- Στη γραμμή 105 ελέγχουμε αν το τρέχων σημείο είναι το 'HipCenter'(Ισχίο).
- Στη γραμμή 110 εκχωρούμε στη μεταβλητή mat το X3D το οποίο θέλουμε να κινήσουμε.
- Στη γραμμή 113 μπορούμε να του αλλάξουμε το χρώμα. Η αλλαγή χρώματος όμως έγινε με βάση τις τιμές που μας επιστρέφει το kinect.

Έτσι μπορούμε να αναπτύξουμε αρκετά τον κώδικα ώστε μέσω του kinect να κινούμε X3D αντικείμενα και να δώσουμε ακόμη περισσότερη ζωντάνια και μαγεία στη δημιουργική μας φαντασία.

5 Εφαρμογές Kinect στον Browser

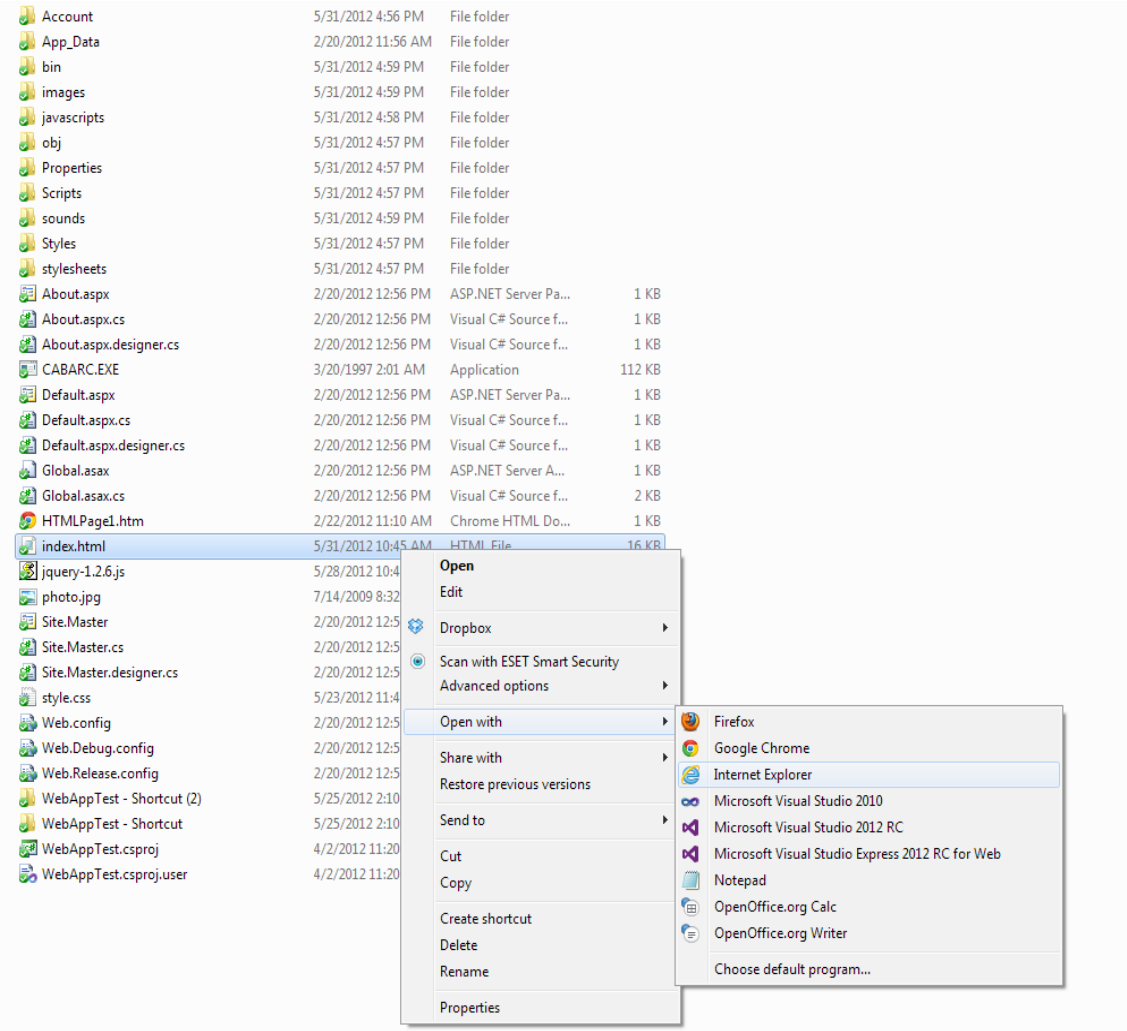
Όπως έχω αναφέρει σε προηγούμενα κεφάλαια το Kinect είναι αυτή τη στιγμή ίσως η μόνη νέα τεχνολογία(όσον αφορά τις συσκευές εισόδου του υπολογιστή) στην οποία δουλεύουν τόσες πολλές εταιρίες αλλά και αυτόνομοι προγραμματιστές. Παρακάτω θα δούμε μερικά παιχνίδια τα οποία παραμετροποίησα έτσι ώστε να λαμβάνουν κίνηση από το kinect με βάση τις κινήσεις του παίχτη. Ο browser ο οποίος χρησιμοποιώ είναι ο Internet Explorer διότι είναι ο μόνος browser που αναγνωρίζει ActiveX αντικείμενα τα οποία οι άλλοι browser απαγορεύουν τη χρήση τους για λόγους ασφαλείας. Αυτό δε σημαίνει όμως ότι δε μπορούμε να τρέξουμε και σε άλλους browsers αυτές τις εφαρμογές, απλά χρειάζεται να τους δώσουμε πρόσβαση σε ActiveX αντικείμενα εγκαθιστώντας τα αντίστοιχα plugin. Επίσης οι ιστοσελίδες που έχω φτιάξει είναι έτοιμες για να ανέβουν σε κάποιο Server, έτσι ώστε οποιοσδήποτε χρήστης από οπουδήποτε και αν βρίσκεται να μπορεί να παίξει τα παιχνίδια έχοντας συνδεμένο το kinect στον υπολογιστή του και χρησιμοποιώντας για browser τον Internet Explorer.

5.1 Οδηγίες χρήσης για τα παιχνίδια

Τα παιχνίδια αυτά τρέχουν κατευθείαν αν έχουν φορτωθεί σε κάποιο Server 'χτυπώντας' απλά τη διεύθυνση. Στο ενδεχόμενο που θέλουμε να τα τρέξουμε μέσα από το cd ή από τον υπολογιστή μας τότε θα ακολουθήσουμε πιστά τα επόμενα βήματα.

5.1.1 Βημα 1

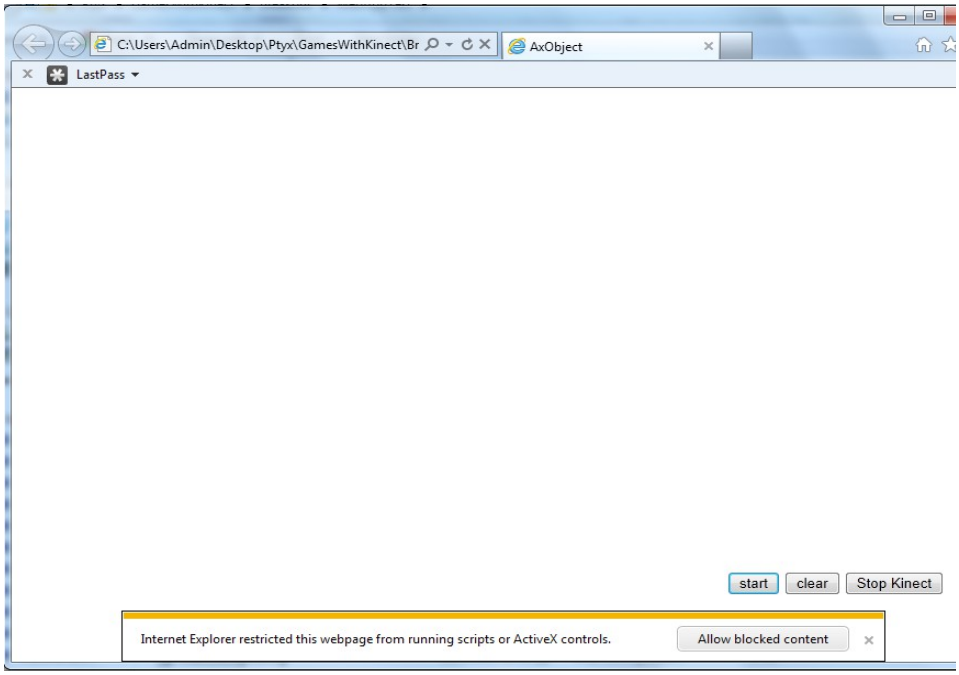
Πηγαίνουμε στο φάκελο στον οποίο βρίσκεται η σελίδα που κατασκευάσαμε, στη προκειμένη περίπτωση είναι ο 'WebAppTest' και κάνουμε δεξί κλικ και άνοιγμα με τον Internet Explorer, όπως βλέπουμε στη παρακάτω εικόνα.



Εικόνα 43: Παιχνίδι - Βημα 1

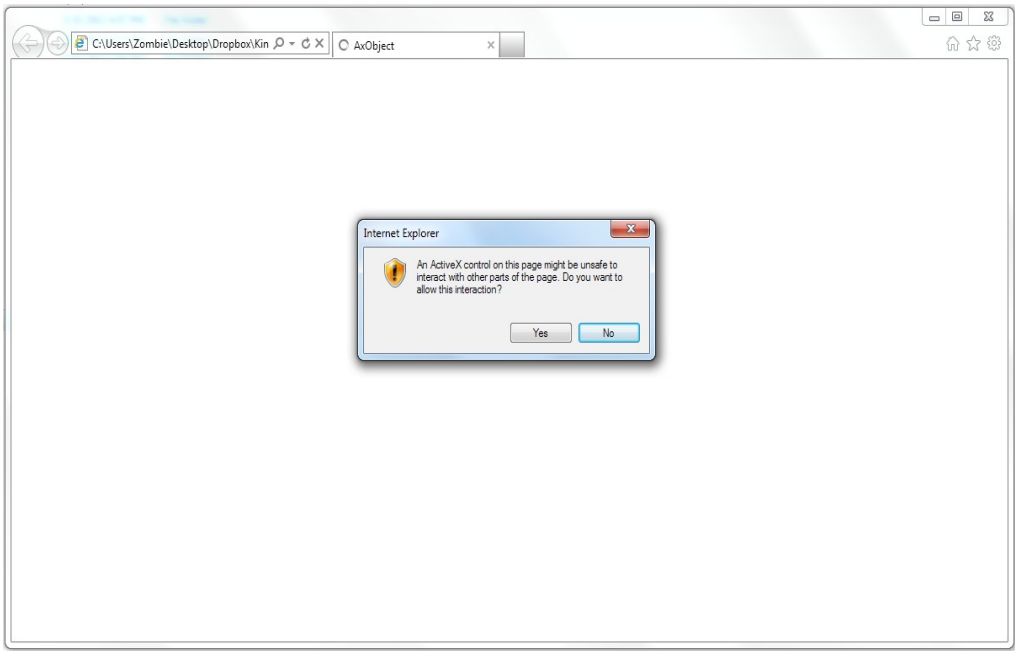
5.1.2 Βήμα 2

Όταν ανοίξει ο Internet Explorer θα μας βγάλει μια πρώτη επιλογή στο κάτω μέρος του παραθύρου που θα μας ρωτάει αν θέλουμε να τρέξουμε ActiveX αντικείμενα και εμείς επιλέγουμε 'allow blocked context'.



Εικόνα 44: Παιχνίδι - Βήμα 2α

Στη συνέχεια θα μας βγει μια δεύτερη επιλογή όπως βλέπουμε στο παρακάτω παράθυρο και εμείς προφανώς θα επιλέξουμε 'YES' για να δώσουμε πλήρη πρόσβαση της εφαρμογής στον υπολογιστή μας μέσω του ActiveX αντικειμένου.



Εικόνα 45: Παιχνίδι - Βήμα 2β

5.1.3 Βήμα 3

Πλέον μπορούμε να παίξουμε το παιχνίδι κουνώντας μόνο τα χέρια μας, διότι στο συγκεκριμένο παιχνίδι βάλαμε το kinect να αναγνωρίζει μόνο τα χέρια μας. Αν θέλαμε πιο σύνθετη αλληλεπίδραση του παιχνιδιού με το χρήστη τότε θα μπορούσαμε να χρησιμοποιήσουμε το κεφάλι, τα πόδια ή οποιοδήποτε άλλο σημείο του σκελετού καλύπτει τις ανάγκες μας.

Εικόνα 46: Παιχνίδι - Βήμα 3

5.2 Παραδείγματα

5.2.1 AirForce

Εικόνα 47: Παιχνίδι - AirForce

Στη παραπάνω εικόνα βλέπουμε το χρήστη να χειρίζεται το μικρό αεροπλανάκι, το οποίο φαίνεται αμυδρά, κινώντας τα χέρια του. Με το δεξί του χέρι κινεί το αεροπλανάκι δεξιά και αριστερά ενώ με το αριστερό του χέρι πυροβολάει τους εχθρούς που εμφανίζονται στην οθόνη.

5.2.2 Brick Breaker

Εικόνα 48: Παιχνίδι - Brick Breaker

Εδώ ο χρήστης εξτασιασμένος παίζει το κλασικό παιχνίδι Brick Breaker στο οποίο κουνώντας τη μπάρα στο κάτω μέρος της οθόνης χτυπάει το μπαλάκι το οποίο με τη σειρά του σπάει τα τουβλάκια.

5.2.3 Super Mario Bros

Εικόνα 49: Παιχνίδι - Super Mario Bros

Εδώ ο χρήστης παίζει ένα από τα πιο ιστορικά παιχνίδια, το Super Mario Bros, με το οποίο έχουν ασχοληθεί μικροί και μεγάλοι ατέλειωτες ώρες. Τη στιγμή που τραβήχτηκε η φωτογραφία ο χρήστης φαίνεται να σηκώνει το αριστερό του χέρι το οποίο σηματοδοτεί ότι ο 'Mario' πρέπει να πηδήξει, ενώ παράλληλα με το δεξί του χέρι κινεί ταυτόχρονα τον 'Mario' προς τα δεξιά έτσι ώστε να μπορέσει να ξεπεράσει τα εμπόδια που θα του παρουσιαστούν.

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**

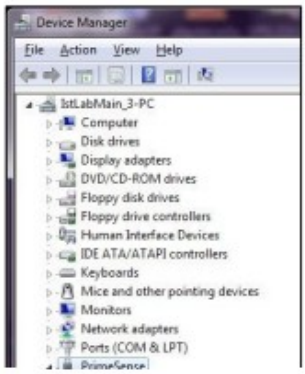
  Κατεβάζουμε και εγκαθιστούμε τις νεότερες Stable ή Unstable εκδόσεις των:

  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)

- **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.

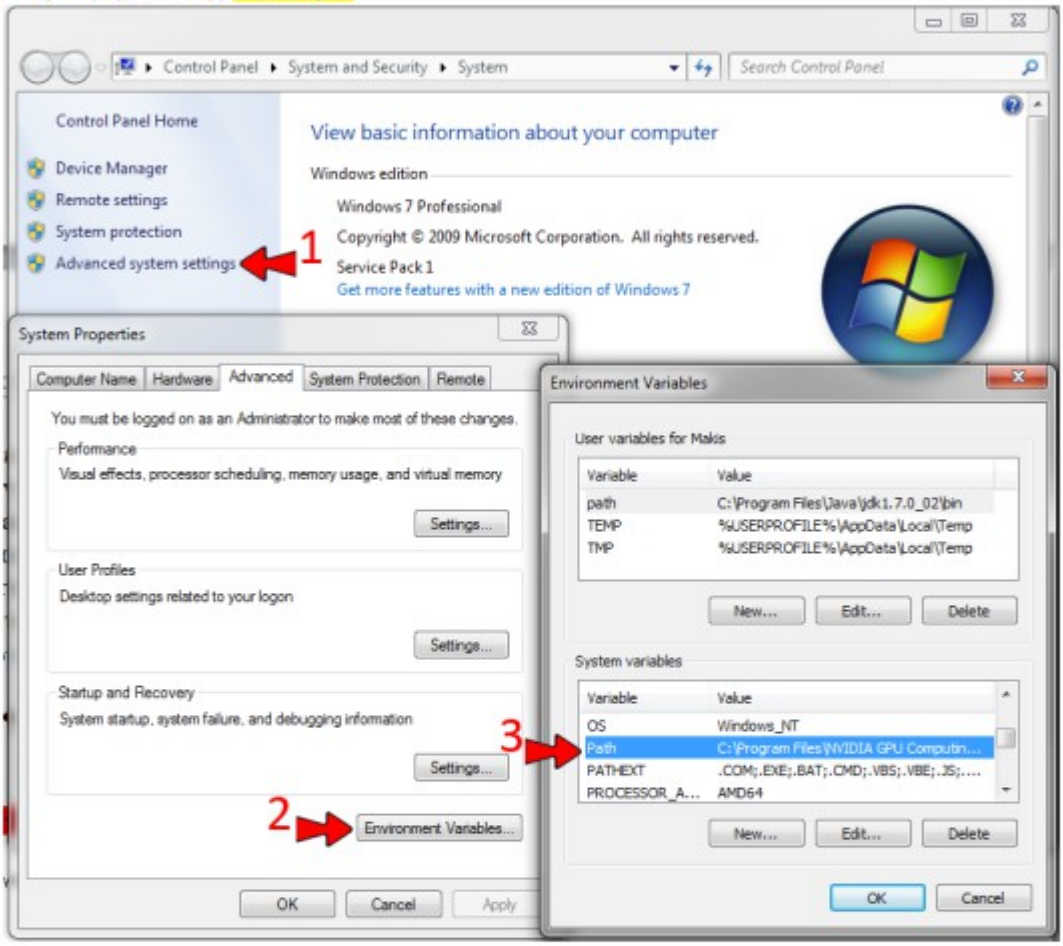
  Master και Unstable branch
  Εάν επιθυμείτε να αλλάξετε το Libfreenect, να συντάξετε wrappers, ή θέλετε να δοκιμάσετε πιο πρόσφατες δυνατότητες, τότε είναι πιθανό να θέλετε το unstable branch. Το Unstable έχει τις πιο πρόσφατες αλλαγές που δεν έχουν ελεγχθεί ακόμη. Η άλλη επιλογή είναι το Master branch 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.



- **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.

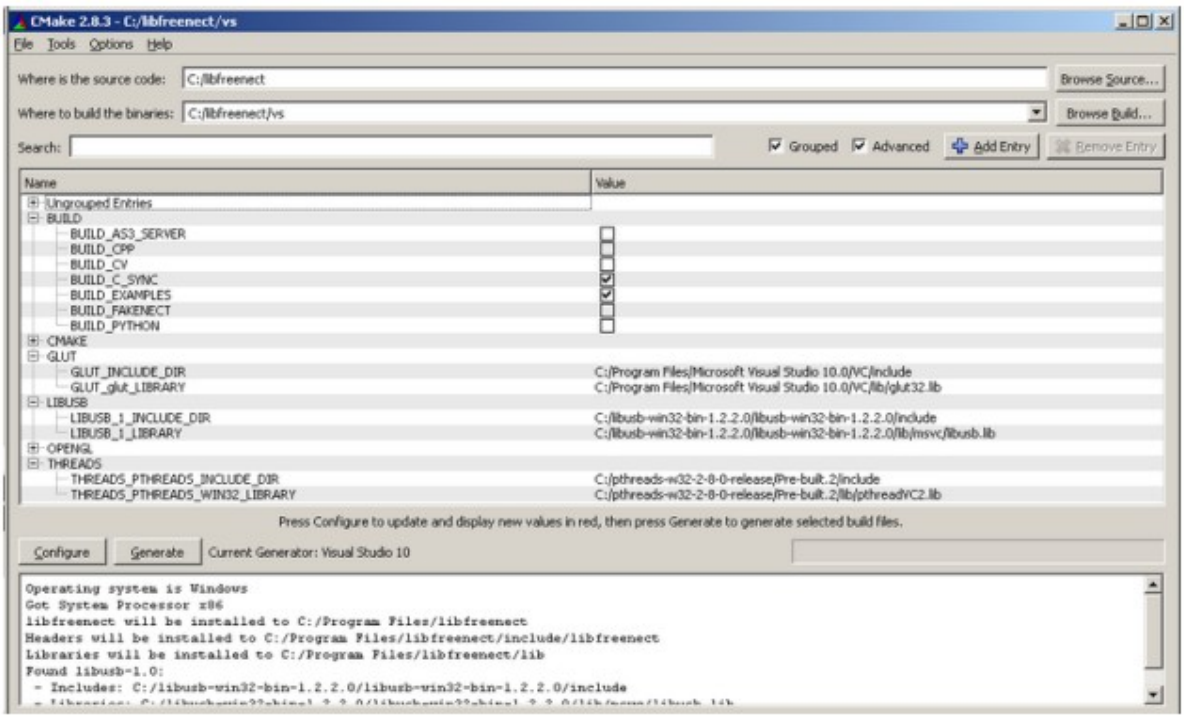
1) Συνδέστε το Kinect στον υπολογιστή. Τα Windows θα σας ειδοποιήσουν ότι βρέθηκε μια καινούρια συσκευή αλλά δεν υπάρχουν διαθέσιμα προγράμματα οδήγησης (το LED της συσκευής θα παραμείνει κλειστό). Εάν τα Windows εμφανίσουν ένα μήνυμα το οποίο θα ζητάει άδεια να ψάξει για προγράμματα οδήγησης, απλά πατήστε 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**

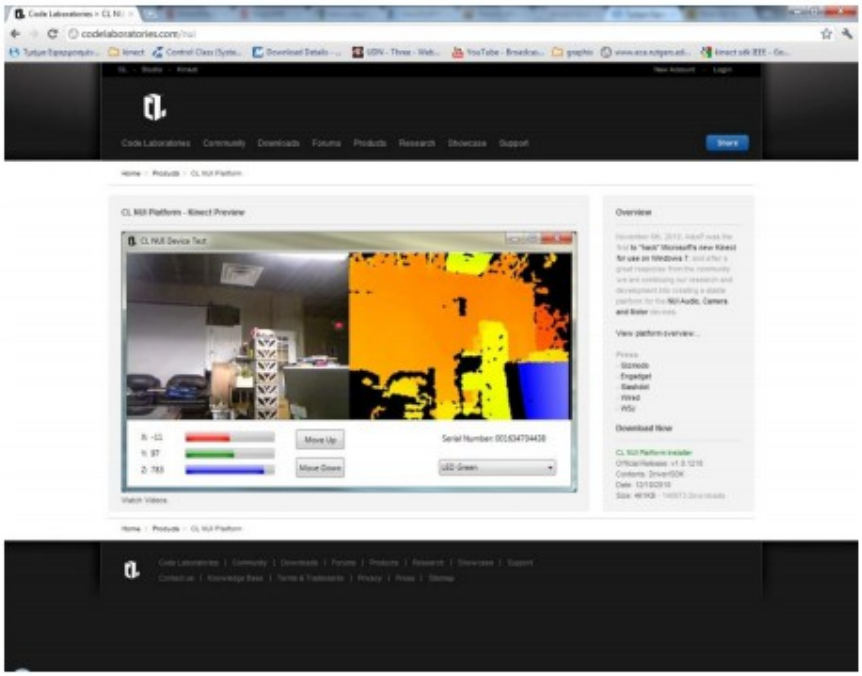
Αφού τελειώσουμε το Βήμα 4, προχωράμε στο build της βιβλιοθήκης μας.

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.



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.



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 Evoluce 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.

Βιβλιογραφία

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

[2] 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 , pp.349-350, 19-20 March 2011

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

[4] PrimeSense Ltd, "The PrimeSensor(TM) Reference Design 1.08."

[5] OPENKINECT OpenKinect Main Page. http://openkinect.org/

[6] OPENNI OpenNI. http://openni.org/

[7] LABORATORIES, C. About: CL NUI Platform. Code

http://patentscope.wipo.int/search/en/detail.jsf?docId=WO2007043036&recNum=1&maxRec=&office=&prevFilter=&sortOption=&queryString=&tab=PCT+Biblio

Laboratories, http://codelaboratories.com/kb/nui

[8] Jarrett Webb, James Ashley, "Beginning Kinect Programming with the Microsoft Kinect SDK", publication date: February 23 2012, edition:1

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

[10] 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