ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

29 ΗΥ

Η χρήση του αισθητήρα ανίχνευσης Kinect της Microsoft σε εκπαιδευτικό περιβάλλον: Μια μελέτη περίπτωσης στα εργαστήρια προγραμματισμού.

ΤΕΙ ΠΕΙΡΑΙΑ ΤΜΗΜΑ: ΜΗΧΑΝΙΚΩΝ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

Σπουδαστές : Καραγεώργος Αχιλλέας

Τσουτσουλιανούδης Παναγιώτης

Επιβλέπων: Ψαρομήλιγκος Ιωάννης

Με την ευκαιρία της κατάθεσης της πτυχιακής μας εργασίας θα θέλαμε να ευχαριστήσουμε τα εξής άτομα:

- Όσους έχουν εργαστεί για την δημιουργία του αισθητήρα KinectforWindows και του KinectSDK.

- Τον υπεύθυνο επιβλέποντα καθηγητή κ.Ιωάννη Ψαρομήλιγκο για την σημαντική του καθοδήγηση στην εργασία αυτή, αλλά και για την ανάπτυξη του ενδιαφέροντος μας για την ενασχόληση με τον προγραμματισμό του Kinectforwindowskαι το KinectSDK.

- Τους συμφοιτητές και φίλους μας για την συντροφιά και συνεργασία που μοιραστήκαμε στα χρόνια φοίτησης στο Ίδρυμα.

- Το υπόλοιπο εκπαιδευτικό προσωπικό του εργαστηρίου Συστημάτων Ηλεκτρονικής Μάθησης & Διαδραστικών Πολυμέσων, και ιδιαίτερα τον κ.Χρήστο Κυτάγια, για τις υποδείξεις τους, τη συνεργασία τους και τη φιλική αντιμετώπιση.

Οι σπουδαστές

Καραγεώργος Αχιλλέας

Τσουτσουλιανούδης Παναγιώτης

Περιεχόμενα

Σκοπός Δημιουργίας της Πτυχιακής Πτυχιακής Εργασίας

Οι εφαρμογές αυτής της πτυχιακής εργασίας υλοποιήθηκαν με σκοπό να βοηθήσουν τους εκπαιδευτικούς και τους φοιτητές να γνωρίσουν τον αισθητήρα Kinect.Το Kinect δίνει την δυνατότηταστους φοιτητές να μάθουν καινούριες προγραμματιστικές μεθόδους και τεχνικές, αφούτους οδηγεί σε έναν τελείως διαφορετικό κόσμο και σε μια προγραμματιστική λογική, που ξεφεύγει απο τις συμβατικές μεθόδους χρήσης ενός Η/Υ. Αντικαθιστώντας την πλοήγηση με ποντίκι και πληκτρολόγιο, οι χρήστες περνούν σε ένα διαφορετικό τρόπο προγραμματισμού, χρησιμοποιώντας χειρονομίες ή φωνητικές εντολές. Έτσι χρησιμοποιούν μηχανισμούς που είναι πολύ πιο κοντά στην καθημερινότητα τους και τους βοηθούν να νιώσουν πολύ πιο άνετα αλληλεπιδρώντας με αυτόν τον τρόπο με τον Η/Υ και όλες τις υπόλοιπες ηλεκτρονικές συσκευές που χρησιμοποιούν σε καθημερινή βάση.

Οι δύο εφαρμογές που υλοποιήσαμε έχουν επίσης σκοπό να καλύψουν την ανάγκη των φοιτητών να μαθαίνουν και να ενημερώνονται για τις καινούριες έφαρμογές τις πληροφορικής στην εκπαίδευση. Αυτό θα το καταφέρουν με την βοήθεια και την σωστή καθοδήγησητων καθηγητών τους. Επίσης συμβάλλουν στο να μεταβούν ομαλά από τον ένα τρόπο προγραμματισμού, τον συμβατικό με απλά GUI(GraphicalUserInterface) και την χρήση ποντίκι/πληκτρολόγιου, σε αυτόν του NUI(NaturalUserInterface), με κινήσεις του σώματος και χειρονομίες ή φωνητικές εντολές.Θα ευνοηθούν παράλληλα και οι εκπαιδευτικοί,μιας και θα έχουν την ευκαιρία να παρουσιάσουν και να διδάξουν στους φοιτητές τους κάτι με οποίο θα αποκομίσουν γνώσεις, όχι μόνο πάνω στο μάθημα αλλά θα μπορέσουν να τους δώσουν και μια γεύση για τα σύγχρονα εμπορικά προιόντα τις αγοράς αλλά και για το τι ζητάει η αγορά εργασίας από αυτούς σαν μελλοντικούς απόφοιτους του ιδρύματος.

Στα επόμενα κεφάλαια αναπτύσσεται το υλικό και η λογική που έκαναν δυνατή την εκπόνηση της πτυχιακής εργασίας.

1. Εισαγωγή στο Kinect

Το Kinectείναι μια συσκευή ανίχνευσης κίνησης από την Microsoft, η οποία επιτρέπει στους χρήστες να ελέγχουν και να αλληλεπιδρούν με αυτή, χρησιμοποιώντας χειρονομίες και προφορικές εντολές. Η πρώτη έκδοση ανακοινώθηκε από την Microsoftγια πρώτη φορά την 1η Ιουνίου 2009 με το κωδικό όνομα ProjectNatalγια την βιντεοκονσόλα παιχνιδιών XBOX 360, με μια δεύτερη έκδοση για Windowsνα κυκλοφορεί την 1η Φεβρουαρίου 2012.

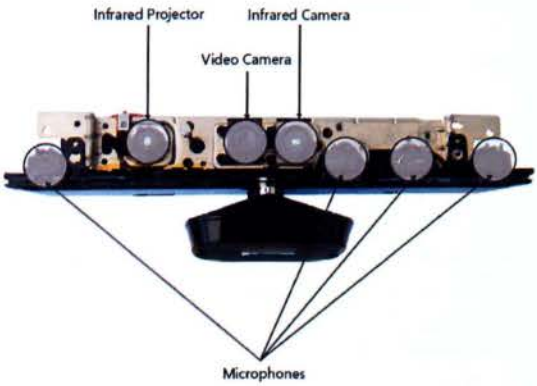
Αν και το Kinectγια Windows με το Kinectγια XBOX 360 εξωτερικά δείχνουν πανομοιότυπα κρύβουν αρκετές διαφορές. Ο αισθητήρας για το XBOX 360 μπορεί να ανιχνεύσει αντικείμενα σε μια απόσταση 4 μέτρων μακριά, αλλά αποτυγχάνει να ανιχνεύσει αντικείμενα σε κοντινή απόσταση (80cm), μιας και ο σκοπός της λειτουργίας του ήταν να αλλάξει την εμπειρία χειρισμού στους παίχτες των βιντεοπαιχνιδιών. Σε αντίθεση ο αισθητήρας Kinectγια Windowsέχει νέο firmware και μπορεί να ανιχνεύσει αντικείμενα σε κοντινή απόσταση έως 40cmχωρίς να χάνει ακρίβεια μιας και ο κύριος σκοπός του είναι η χρήση του στην ανάπτυξη εφαρμογών για το λειτουργικό σύστημα της Microsoft.

MicrosoftKinectγιαXBOX 360

MicrosoftKinectγιαWIndows

2. Ο αισθητήρας κίνησης Kinect

Ο αισθητήρας Kinect αποτελείται από ένα πομπό υπέρυθρων ακτίνων (IREmitter), ένα αισθητήρα βάθους (IRDepthSensor), μια κάμερα χρώματος (VideoCamera), τέσσερα μικρόφωνα καθώς και ένα μοτέρ στην βάση που του επιτρέπει να μετακινείται στον κάθετο άξονα.



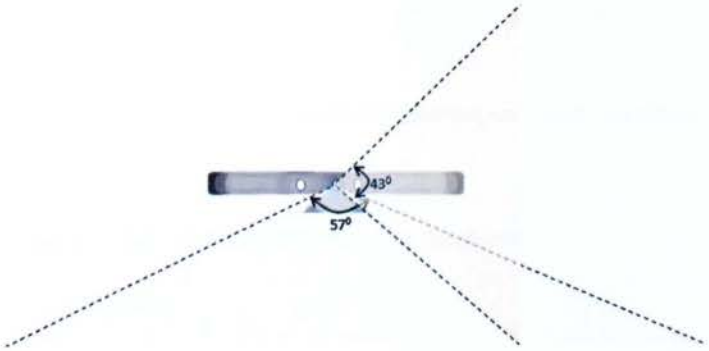
Ο αισθητήρας Kinect χωρίς το εξωτερικό καπάκι

2.1 Κάμερα Χρώματος (ColorCamera)

Ο αισθητήρας Kinect έχει μια ενσωματωμένη κάμερα χρώματος με υποστηριζόμενες αναλύσεις:

- 640x480 pixels στα 60 καρέ ανά δευτερόλεπτο (FPS).
- 1280x960 pixels στα 12 καρέ ανά δευτερόλεπτο (FPS).

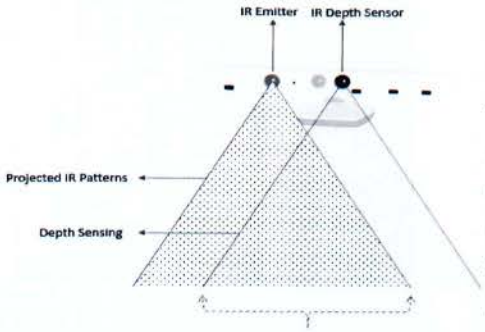
Ενώ η ορατή περιοχή της κάμερας είναι 43° κάθετα με 57° οριζόντια.



Η ορατή περιοχή της κάμερας του Kinect

2.2 Αισθητήρας Βάθους και Πομπός Υπερύθρων

Ο πομπός υπέρυθρων ακτινών εκπέμπει υπέρυθρο φως σε ένα "ψευδο-τυχαίο" μοτίβο σε ό,τι βρίσκεται μπροστά του. Οι κουκκίδες αυτές είναι αόρατες στο ανθρώπινο μάτι, αλλά είναι δυνατόν τις συλλάβει ο αισθητήρας βάθους. Καθώς οι υπέρυθρες ακτίνες ανακλώνται από τα διάφορα αντικείμενα ο αισθητήρας βάθους διαβάζει και μετατρέπει σε πληροφορίες βάθους, μετρώντας την απόσταση μεταξύ του αισθητήρα και του αντικειμένου από όπου η κάθε κουκίδα διαβάστηκε.



Αισθητήρας Βάθους και Πομπός Υπερύθρων

Ο αισθητήρας βάθους υποστηρίζει αναλύσεις:

640x480 pixels

320x240 pixels

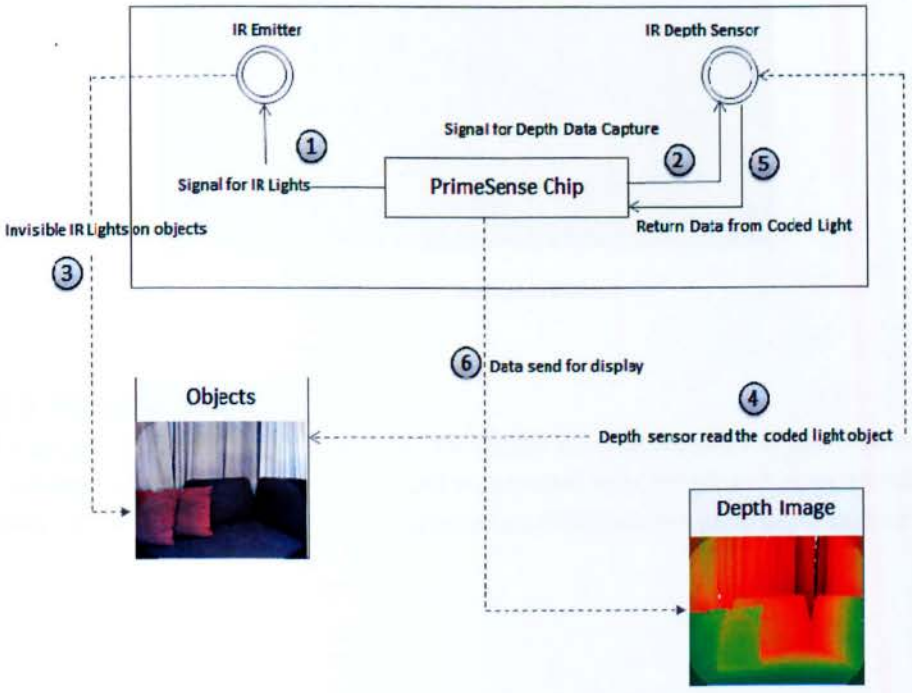
80x60 pixels

και η ορατή περιοχή παραμένει η ίδια όπως και στην κάμερα χρώματος43° κάθετα με 57° οριζόντια.

Οι ακτίνες που εκπέμπει ο Πομπός Υπερύθρων (IREmitter)

2.3 Πώς επεξεργάζονται τα δεδομένα του αισθητήρα βάθους

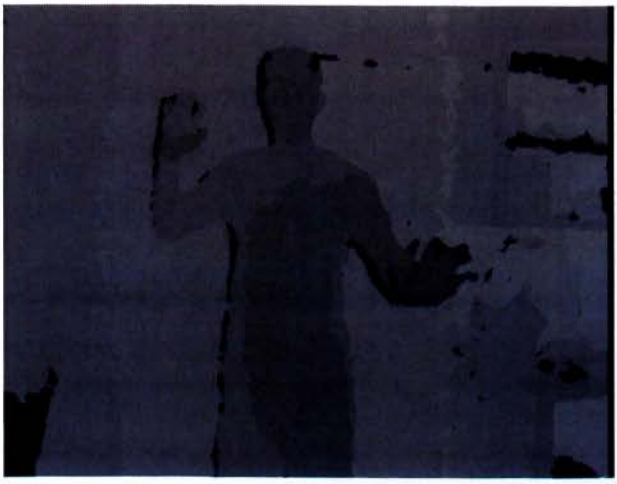
Ο αισθητήρας Kinect έχει την ικανότητα να μπορεί δημιουργεί μια 3D απεικόνιση των αντικειμένων που υπάρχουν μπροστά από αυτόν, ανεξάρτητα από τις συνθήκες φωτισμού που υπάρχουν στο χώρο.Η ραχοκοκαλιά πίσω από αυτή την τεχνολογία δημιουργήθηκε από την Ισραηλινή εταιρία PrimeSense και το ακόλουθο διάγραμμα δείχνει την λειτουργία της.



(1) Το τσιπ της PrimeSense στέλνει ένα μήνυμα προς τον πομπό υπέρυθρων ακτίνων (IR Emitter) για να ενεργοποιήσει το υπέρυθρο φως,(2) καθώς και ένα δεύτερο μήνυμα στον αισθητήρα βάθους (IR Depth Sensor) για την έναρξη της συλλογής των δεδομένωνβάθους από την τρέχουσα ορατή περιοχή του αισθητήρα.

(3) Ο πομπός υπέρυθρων ακτίνων (IR Emitter) εκπέμπει υπέρυθρο φώς στα αντικείμενα που υπάρχουν μπροστά του, (4) και ο αισθητήρας βάθος (IR Depth Sensor) ξεκινά την ανάγνωση των δεδομένων μετατρέποντας σε πληροφορίες βάθους την απόσταση της κάθε ανακλώμενης κουκίδας υπέρυθρου φωτός από τα αντικείμενα.

Το τσιπ της PrimeSense στη συνέχεια, επεξεργάζεται τα καταγεγραμμένα δεδομένα (5), και δημιουργεί μια εικόνα βάθους (6).

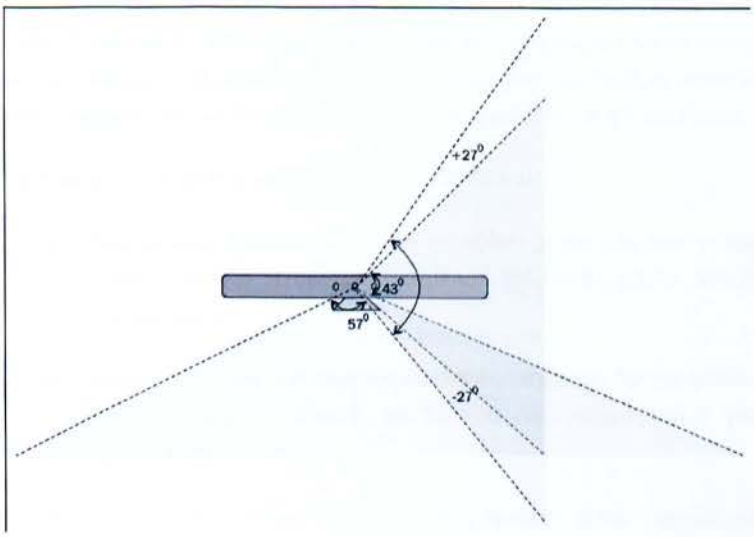


Εικόνα από τον Αισθητήρα Βάθους

2.4 Μοτέρ κλίσης

Το τμήμα της βάσης και το σώμα του αισθητήρα συνδέονται με έναν μικροσκοπικό κινητήρα, κάνοντάς το Kinectva μπορεί να στραφεί προς τα πάνω ή προς τα κάτω κατά 27°. Η μετατόπιση αυτή επιτρέπεται να γίνεται δυναμικά μόνο μέσω κώδικα.

Ο κινητήρας που υπάρχει στη βάση της συσκευής



Οι μέγιστες γωνίες κατακόρυφης μετατόπισης του Kinect

2.5 Μικρόφωνα

Ο αισθητήρας Kinectέχει τέσσερα διαφορετικά μικρόφωνα που είναι τοποθετημένα σε μία σειρά (τρία από αυτά απλώνονται στη δεξιά πλευρά,ενώτο τέταρτο είναι τοποθετημένο στην αριστερή πλευρά).Τα μικρόφωνα δέχονται 16-bitήχο με συχνότητα δειγματοληψίας ίση με 16KHz.

Τα μικρόφωνα του Kinect

Το κύριο πλεονέκτημα της χρησιμοποίησης 4 μικροφώνων αντί του ενός είναι ότι μπορεί να γίνει πιο αποτελεσματική αναγνώριση της φωνής, συγκρίνοντας την χρονική στιγμή που το κάθε μικρόφωνο λαμβάνει το ίδιο ηχητικό σήμα ο αισθητήρας μπορεί να εξακριβώσει την γωνία της προέλευσης του σήματος.

3. MicrosoftKinectSDKγιαWindows

Στις 21 Μαΐου 2012 η Microsoftκυκλοφόρησε την έκδοση 1.5 του KinectSDKγια Windows, ενώ είχε ήδη προηγηθεί μια μη εμπορική έκδοση στις 16 Ιουνίου 2011, παρέχοντας βιβλιοθήκες με τις οποίες οι προγραμματιστές μπορούσαν να αλληλεπιδράσουν απευθείας με τις κάμερες, τα μικρόφωνα και το μοτέρ του αισθητήρα αναπτύσσοντας εφαρμογές χρησιμοποιώντας τις γλώσσες προγραμματισμού C++,C# ή VisualBasic.

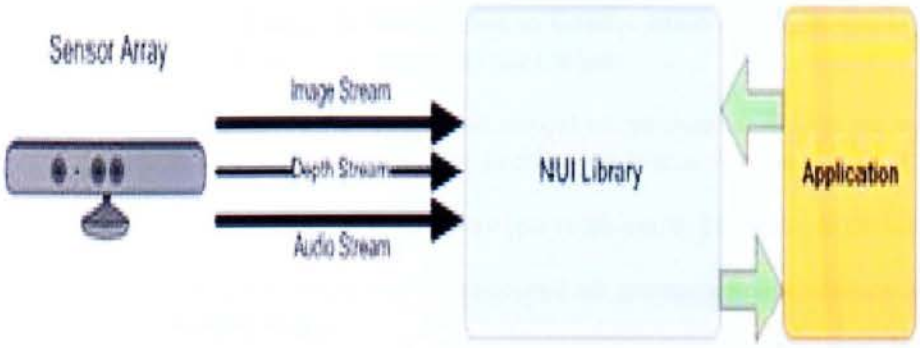
Τα χαρακτηριστικά που παρέχει το KinectSDKείναι:

- Raw sensor streams: Χαμηλού επιπέδου χειρισμός των ρευμάτων του αισθητήρα βάθους (Depth Stream),της κάμερας χρώματος (Color Stream) καθώς και των 4 μικρόφωνων.

- Skeleton Traking: Εντοπισμός σκελετών μέχρι και δυο χρηστών οι οποίοι βρίσκονται μπροστά από το Kinect, με λεπτομερής πληροφορίες για την θέση και τον προσανατολισμό τους.

- Εξελιγμένες δυνατότητες επεξεργασίας ήχου συμπεριλαμβανομένων των AcousticEchoCancellationκαιNoiseSuppression. Επίσης οι εφαρμογές μπορούν να χρησιμοποιήσουν το Kinect ως input device για τη βιβλιοθήκη αναγνώρισης ομιλίας της Microsoft.

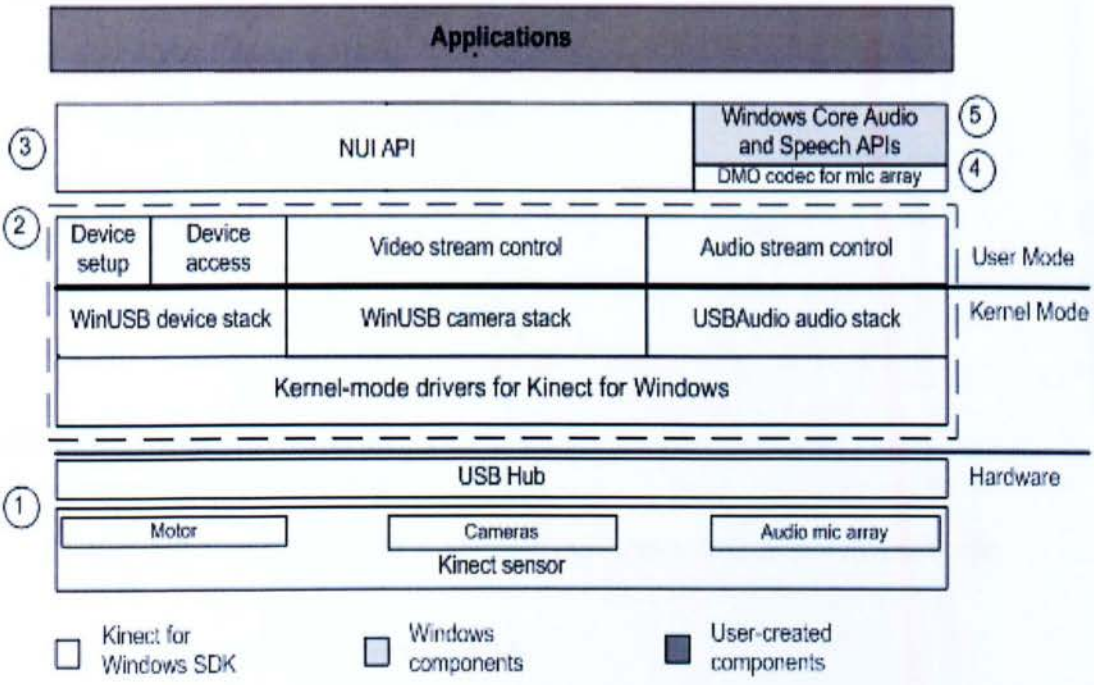
Όπως βλέπουμε στην παρακάτω εικόνα το KinectSDKλειτουργεί ως ενδιάμεσος με τις εφαρμογές, αποκωδικοποιώντας τα ρεύματα εικόνας, βάθους και ήχου που δίνει ο αισθητήρας.



Επικοινωνία των εφαρμογών με τον αισθητήρα μέσω του SDK

3.1 Αρχιτεκτονική του KinectSDK

Όπως φαίνεται στην παρακάτω εικότα η αρχιτεκτονική του KinectSDKπεριλαμβάνει τα ακόλουθα:



Αρχιτεκτονική του KinectSDK

1 KinectHardware: Τον αισθητήρα Kinectκαθώς και το USBhubμέσω του οποίου είναι συνδεδεμένος ο αισθητήρας με τον υπολογιστή.

2 KinectDrivers: Τους οδηγούς τωνWindowsγια το Kinectοι οποίοι εγκαθίστανται ως μέρος της διαδικασίας εγκατάστασης του SDK μέσω των οποίων:

- Η συστοιχία μικροφώνων του Kinect μπορεί να χρησιμοποιηθεί και ως συσκευή ήχου ώστε να υπάρχει πρόσβαση και μέσα από το APIτων Windows.
- Έλεγχος των ρευμάτων ήχου και βίντεο (για το χρώμα, το βάθος και το σκελετό).
- Λειτουργίας ώστε να μπορεί μια εφαρμογή να χρησιμοποιεί περισσότερους από έναν αισθητήρες Kinect.

3 NUIAPI: Φυσική Διεπαφή Χρήστη παρακολουθώντας τα ρεύματα εξόδου για το σκελετό, το χρώμα και το βάθος απεικόνισης.

4 DirectX Media Object (DMO) γιαbeamforming και audio source localization τιςσυστοιχίαςμικροφώνων.

5 WindowAPIsγια τον ήχο και την ομιλία

3.2 Απαιτήσεις Υλικού

Οι ελάχιστες απαιτήσεις υλικού για τον αισθητήρα είναι:

- 32-bit(x86) ή 64-bit(x64) CPU.
- Dual-core, 2.66-GHz or faster processor.
- USB 2.0 Bus dedicated to the Kinect.
- 2 GB RAM
- Graphics card that supports DirectX 9.0c

3.3 Απαιτήσεις Λογισμικού

Οι ελάχιστες απαιτήσεις λογισμικού:

- Microsoft VisualStudio 2010 ή MicrosoftVisualStudio 2012ήMicrosoftVisualStudio 2013
- .NET Framework 4 ή.NET Framework 4.5
- Windows 7
- Windows 8
- Windows 8.1

3.4Δεδομένα Χρώματος (ColorStream)

Οι τύποι των δεδομένων χρώματος που υποστηρίζει ο αισθητήρας Kinectείναιοι:

- RGB
- YUV
- BAYER

3.4.1 RGB Μορφή Χρώματος

Στην RGB (Red – Green - Blue)μορφή χρώματος κάθε pixelέχει μέγεθος 4 Byteόπου τα 3 πρώτα bytexρησιμοποιούνται για το Κόκκινο χρώμα (1 Byte), το Πράσινο χρώμα(1 Byte) και το Μπλε χρώμα (1 Byte), καθώς και ένα τελευταίο Byteπου χρησιμοποιείται για την διαφάνεια (Transparency).

RGBpixels

Οι αναλύσεις που υποστηρίζει η κάμερα χρώματος του αισθητήρα Kinectγια την RGBμορφή είναι:

- 640x480 pixels (30 frames per second)
- 1280x960 pixels (12 frames per second)

3.4.2 YUVΜορφή Χρώματος

Στην YUVμορφή χρώματος του Υαντιπροσωπεύει το κανάλι της φωτεινότητας, το Υαντιπροσωπεύει το κανάλι του Μπλε χρώματος και το Ναντιπροσωπεύει το κανάλι του Κόκκινου χρώματος.

Επειδή η YUV μορφή χρώματος χρησιμοποιεί 2 Bytes (16bits) ανά pixel, δεσμεύει λιγότερη μνήμη για την αποθήκευση των δεδομένων του bitmap και λιγότερη μνήμη στο buffer.

Οι αναλύσεις που υποστηρίζει η κάμερα χρώματος του αισθητήρα Kinectγια την YUV μορφή είναι:

- 640x480 pixels (15 frames per second)

Τόσο η YUVόσο και η RGBμορφή χρώματος αντιπροσωπεύουν την ίδια εικόνα μιας και οι δυο μορφές υπολογίζονται από τα ίδια δεδομένα της κάμερας.

3.4.3 Bayer

Η κάμερα χρώματος του Kinect μπορεί επίσης να επιστρέψει ένα frameεικόνας, χρησιμοποιώντας ένα συνδυασμό των χρωμάτων κόκκινου, πράσινου και μπλε (50% πράσινο, 25% κόκκινο και 25% μπλε). Το φίλτρο που δημιουργεί την εικόνα λέγεται φίλτρο Bayer, και χρησιμοποιείται με αυτόν τον τρόπο επειδή το ανθρώπινο μάτι είναι πιο ευαίσθητο στο πράσινο χρώμα και μπορεί να εντοπίσει περισσότερες αλλαγές στο πράσινο παρά στα άλλα χρώματα.

Οι αναλύσεις που υποστηρίζει η κάμερα χρώματος για την Bayerμορφή είναι:

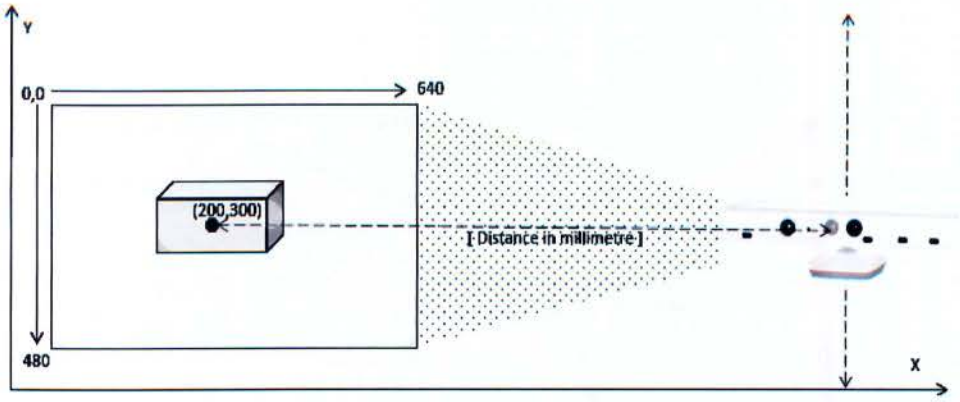
- 640x480 pixels (30 frames per second)
- 1280x960 pixels (12 frames per second)

Green	Red	Green	Red
Blue	Green	Blue	Green
Green	Red	Green	Red
Blue	Green	Blue	Green

Bayer pixel format.

3.5 ΔεδομέναΒάθους (DepthStream)

Ο αισθητήρας Kinectεπιστρέφει τα δεδομένα βάθους ως ένα frameεικόναςφωτεινότητας 16bits (16 Bitgrayscaleframe), όπου η τιμή του κάθε pixelπεριέχει την καρτεσιανή απόσταση σε χιλιοστά μεταξύ του αντικειμένου και τις κάμερας στις συγκεκριμένες x,γσυντεταγμένες μέσα στο οπτικό πεδίο του αισθητήρα.

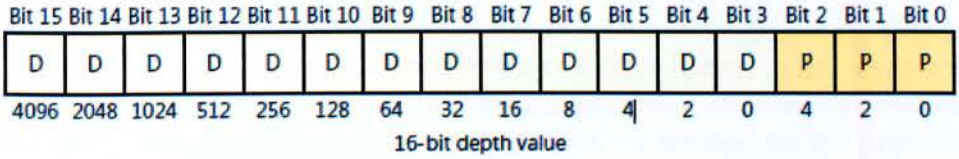


Υπολογισμός της απόστασης μεταξύ του αισθητήρα και του αντικειμένου που βρίσκεται στο ορατό πεδίο του.

Οι αναλύσεις που υποστηρίζονται από το DepthStreamείναι:

- 640x480 pixels (30 frame per second)
- 320x240 pixels (30 frame per second)
- 80x60 pixels (30 frame per second)

ΚάθεpixelστοDepthStreamχρησιμοποιεί 16 bits, από τα οποία τα 13bitsπεριέχουν την απόσταση σε χιλιοστά που μετρήθηκε, και τα υπόλοιπα 3bitsπροσδιορίζουν τον παίκτη που στέκεται μπροστά από τον αισθητήρα.Μια τιμή ίση με 0 στα 3 pixelsπου προσδιορίζουν τον παίκτη σημαίνει ότι κανένας δεν έχει αναγνωριστεί από τον αισθητήρα.

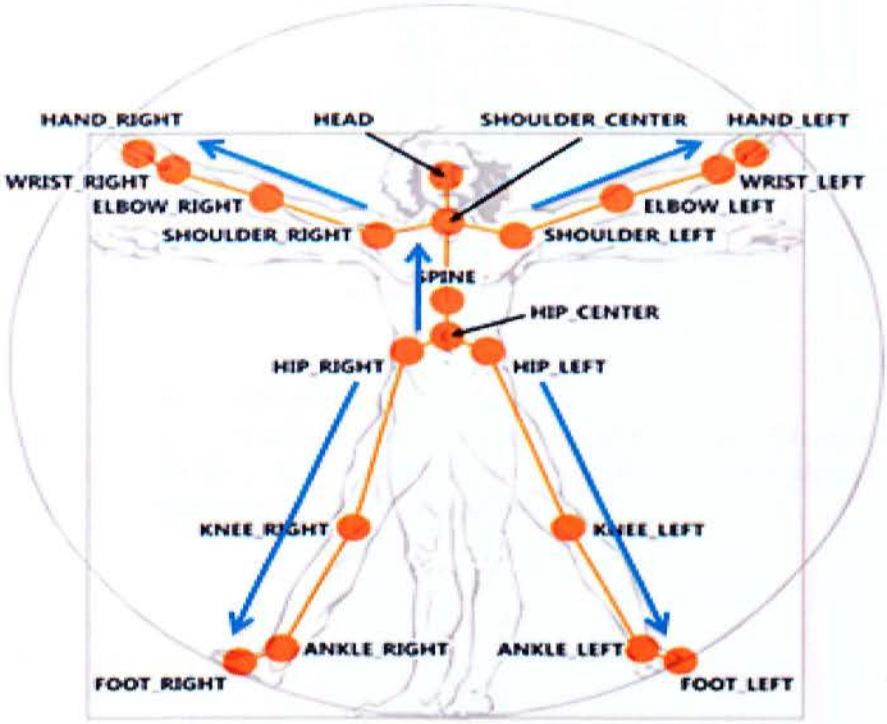


Η χρησιμοποίηση των 16 Bit της του DepthStream (D=Depth,P=Player)

Ο αισθητήρας μπορεί να παρακολουθεί μέχρι και 6 παίκτες ταυτόχρονα, αλλά μπορεί να παρέχει πληροφορίες μόνο για δυο από αυτούς.

3.6 SkeletonTracking

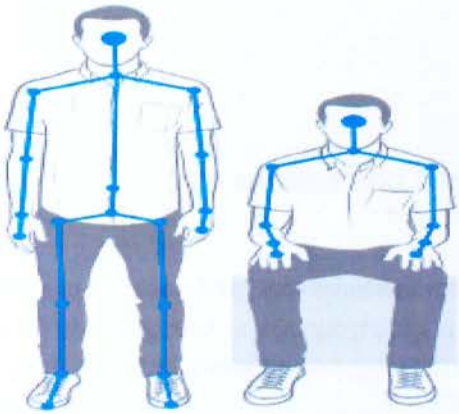
Ένα σημαντικό χαρακτηριστικό του KinectSDKείναι η δυνατότητα ανίχνευσης της κίνησης ενός ανθρώπινου σκελετού που βρίσκεται μπροστά από τον αισθητήρα. Το SDKμπορεί να ανιχνεύσει μέχρι και 20 αρθρώσεις σε ένα σκελετό δίνοντας πληροφορίες για την θέση της κάθε άρθρωσης και τον προσανατολισμό της.



Τα σημεία του σκελετού που μπορεί να ανιχνεύσει ο αισθητήρας

Τα δεδομένα δίνονται στην εφαρμογή σαν ένα σύνολο σημείων (SkeletonPositions), τα οποία συγκροτούν τον σκελετό, ο οποίος αντιπροσωπεύει την θέση και την στάση του χρήστη. Αν και ο αισθητήρας μπορεί να ανιχνεύσει μέχρι και 6 σκελετούς που στέκονται μπροστά του, μπορεί να επιστρέψει τις λεπτομέρειες των αρθρώσεων μόνο για τους 2 από αυτούς.

Άλλο ένα χαρακτηριστικό είναι η δυνατότητα να μπορεί να χρησιμοποιεί ο χρήστης μόνο το πάνω μέρος του σώματος του(SeatedTrackingMode). Έτσι μπορεί να κάθεται, με το Kinect να εντοπίζει και να χρησιμοποιεί μονάχα τις αρθρώσεις του κεφαλιού και των χεριών (σε αυτή την περίπτωση ο αισθητήρας ανιχνεύει μόνο τις 10 αρθρώσεις του πάνω μέρους του σώματος).



3.7 AudioStream

Ο αισθητήρας Kinectόπως προαναφέρθηκε, έχει τέσσερα διαφορετικά μικρόφωνα που είναι τοποθετημένα σε μία σειρά. Η ενσωματωμένη δυνατότητα επεξεργασίας σήματος που υπάρχει στο υλικό του αισθητήρα χρησιμοποιεί 24-bitADC(Analog-to-Digital Converter) με δυνατότητες όπως:Acoustic Echo Cancellation και Noise Suppression. Έτσι μας παρέχει την δυνατότητα της εύρεσης της ηχητικής πηγής (Source Localization) ενός ήχου ή της επιλογής λήψης του ήχου από μια συγκεκριμένη κατεύθυνση (Beamforming).

3.7.1 Αναγνώριση Ομιλίας

Χρησιμοποιώντας τις δυνατότητες Audiotou αισθητήρα αλλά και το SpeechRecognitionAPI των Windowseίναιδυνατή η ανάπτυξη εφαρμογών στις οποίες η αλληλεπίδραση με το χρήστη θα μπορεί να γίνεται μέσω φωνητικών εντολών.

Αλληλεπίδραση μέσω φωνητικών εντολών

4. Προγραμματισμός με τον αισθητήρα Kinect

Στα επόμενα τρία υποκεφάλαια παρουσιάζεται τοολοκληρωμένοπεριβάλλον ανάπτυξης που χρησιμοποιήθηκε για να υλοποιήσουμε τις εφαρμογές μας καθώς και τα απαραίτητα αρχεία και βιβλιοθήκες του KinectSDK(Software development kit) που θα χρησιμοποιήσουμε στην εφαρμογή.

4.1Εγκατάσταση του απαραίτητου λογισμικού

Ακολουθούμε τα εξής βήματα:

1) Το πρώτο βήμα ειναι η εγκατασταση Visual Studio της Microsoft.

2) Μετα κάνουμε εγκατασταση του Kinest SDK που έχει τις απαραίτητες βιβλιοθήκες για τον προγραμματισμό του αισθητήρα αλλά και τους drivers.

3) Μόλις ολοκληρωθεί και η εγκατάσταση του Kinect SDK, τοτε συνδεουμε τον αισθητηρα Kinect στον η/υ.

4.1.1Η διεπαφή χρήστη του VisoualStoudio

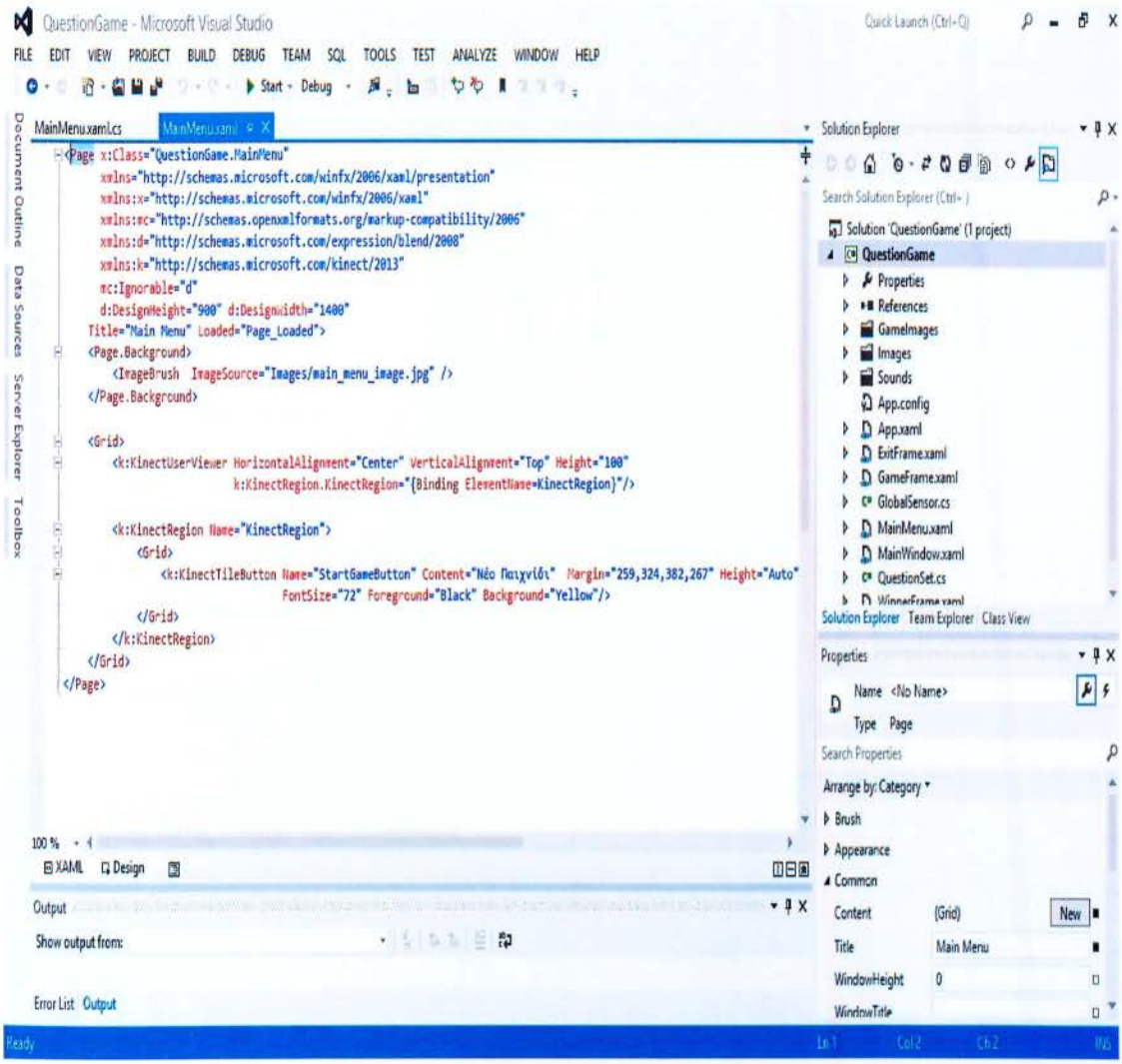
Το ολοκληρωμένοπεριβάλλον ανάπτυξης(IDE Integrated development environment)που χρησιμοποιήθηκε ειναι το VISUALSTUDIO 2012.Η εφαρμογή αποτελείται από δύο μέρη:

- Το xamlκομμάτι στο οποίο σχεδιάζεται το GUI(GraphicalUserInterface) της εφαρμογής

- Το προγραμματιστικό κομμάτι που αποτελείτε από τις κλάσεις που εχουμε γράψει σε C#

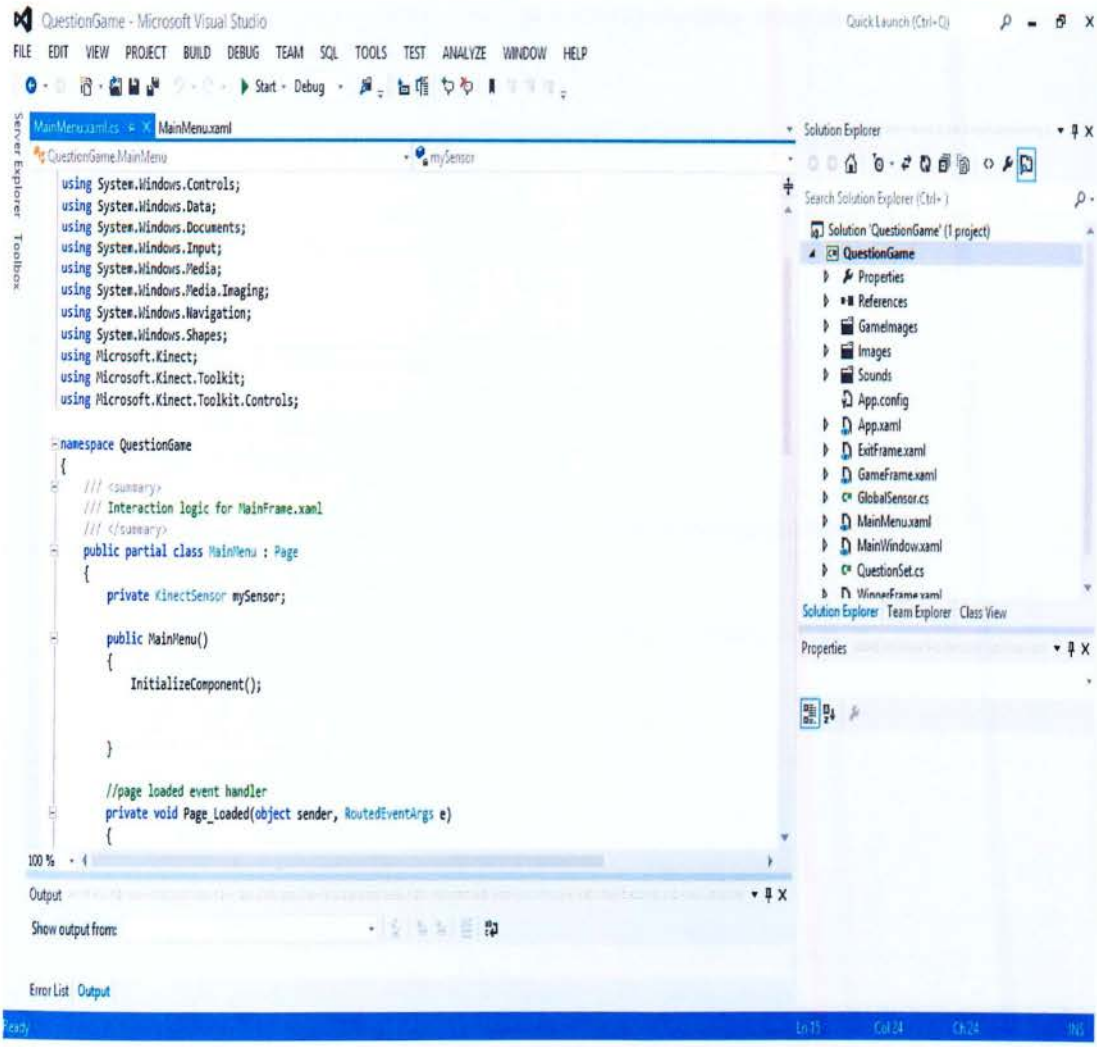
Ακολουθούν 2 στιγμιότυπα για το κάθε προγραμματιστικό κομμάτι.

Η εικόνα για τον xaml κώδικα :



Εικόνα XAMLκώδικας για την σχεδίαση των γραφικών της εφαρμογής

Η εικόνα για το C# κώδικα :



Εικόνα C#κώδικας

4.1.2 Προγραμματίζοντας στο VisualStoudio

Ακολουθούμε τα εξής βήματα:

1)Δημιουργουμε ενα WPF (WindowsPresantationFoundation) project και κάνουμε import τις βιβλιοθηκες που ειναι απαραιτητες για τον προγραμματισμο του αισθητηρα (Kinect.dll).Το WPFprojectαποτελείτε απο το Xamlαρχείο στο οποίο προσθέτουμε τα controlστης εφαρμογής όπως τα Buttons,Labels και TextBoxes και σχεδιάζουνε το γραφικό περιβάλλον της εφαρμογής όπως αναφέραμε στην παραπάνω παράγραφο .Αποτελείτε

επίσης απο το C# αρχείο στο οποίο γράφουμε τους χειριστές(handlers)των controlsπου έχουμε χρησιμοποιήσει στο Xamlερχείο.

2)Σε ένα pc μας δίνεται η δυνατότητα να συνδέσουμε περισσότερους από έναν αισθητήρες,
πρώτα ελέγχουμε αν έχει συνδεθεί τουλάχιστον ένας και πέρνουμε μια αναφορά προς αυτον, η οποία αποθηκέυεται στην μεταβλητή mySensor.

Κώδικας:

```
KinectSensor mySensor;

if(KinectSensor.KinectSensors.Count>0)
  mySensor=KinectSensor.KinectSensors[0];
```

(ο πινακας KinectSensors έχει μέγεθος όσο το πληθος των αισθητήρων kinect που έχουμε συνδέσει)

στη συνέχεια αναλόγως την εφαρμογή που θέλουμε να υλοποιήσουμε ενεργοποιούμε και τα αντίστοιχα ρεύματα:

- Colorstream
- Depthstream
- skeletonstream

και τέλος ενεργοποιούμε τον αισθητήρα με την μέθοδο mySensor.start()

5.KINECTSDK

Για την υλοποίηση των εφαρμογών χρησιμοποιήθηκε το KinectSDK 1.7 και στην συνέχεια το KinectSDK 1.8 .

Το κύριο εργαλείο ή παλέτα στην οποία αναπτύχθηκεη εφαρμογή είναι το KinectRegion.Το KinectRegionείναι η περιοχή στην οθόνη όπου οι αλληεπιδράσεις με τον αισθητήρα Kinectλαμβάνουν χώρο,είναι επίσης η περιοχή στην οποία εμφανίζονται ο WindowsCursorκαι τα υπόλοιπα στοιχεία με τα οποία αλληλεπιδρούν οι χρήστες της εφαρμογής.

Παρακάτω θα αναλύσουμε τα εργαλεία του SDKπου χρησιμοποιήσαμε στην εφαρμογή μας.

5.1 KINECT FOR WINDOWS CURSOR(ΚέρσοραςτουKINECT γιαWINDOWS)

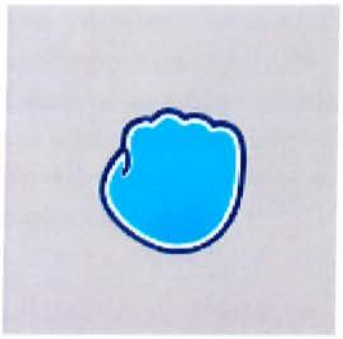
Είναι ο κέρσορας που αντικαθιστά τον κέρσορα του ποντικιού στο PC.

Επισής παρέχει στον χρήστη γραφική ανάδραση ανάλογα με το πως χρησιμοποιείται στην εφαρμογή.Για παράδειγμα μπορεί να θέλουμε να στοχεύσουμε σε κάποιο σημείο στην οθόνη, οπότε και θα εμφανιστεί η εικόνα :

Μπορει να θέλουμε να πατήσουμε καποιο κουμπί και να επιλέξουμε κάτι...και θα μας εμφανιστεί η παρακάτω εικόνα :

Όταν πάει να πατηθεί το κουμπί και δίπλα ακριβώς όταν το κούμπί είναι πατημένο.

Όταν το Kinectανιχνεύσει το χέρι μας σε κατάσταση γροθιάς ή λαβής τότε βλέπουμε αυτό το χέρι στην οθόνη:



Επίσης το Kinectαλληλεπιδρά και με τα δύο χέρια του χρήστη :

5.2 Το USERVIEWER

Σχεδόν σε όλες τις αλληλεπιδράσεις μεταξύ ανθρώπου και υπολογιστή είναι εύκολο να καταλάβουμε αν γίνεται ηθελημένα ή όχι αφού κουνάμε το ποντίκι ή πατάμε πλήκτρα στο πληκτρολόγιο ή αγγίζουμε την οθόνη αν πρόκειται για οθόνη αφής.Με το Kinectείναι αρκετά πιο δύσκολο να καταλάβουμε αν πρόκειται για ηθελημένη κίνηση ή για μια απλά φυσική κίνηση. Εδώ έρχεται να βοηθήσει το UserViewer, το οποίο παρέχει απλά μια εικόνα της σιλουέτας του χρήστη που έχει ανιχνευθεί απο τον αισθητήρα και ξεχωρίζει με χρώματα τον κύριο χρήστη απο τους υπόλοιπους.

Για παράδειγμα :

Οκύριος χρήστης

Άλλοι χρήστες

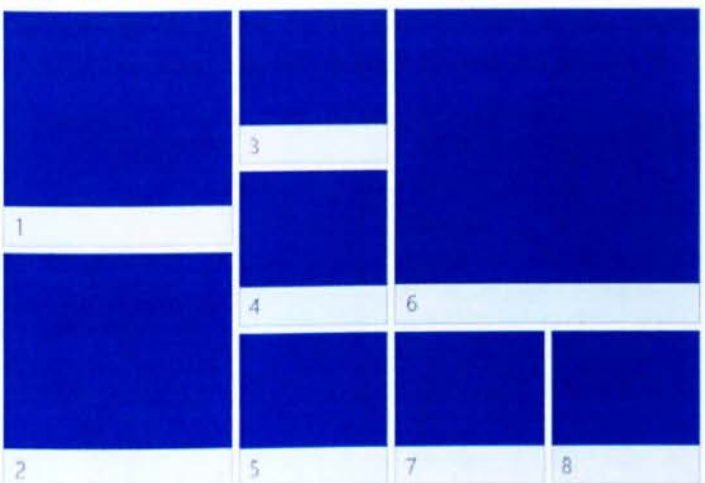
Έτσι βάζοντας το UserViewerστην εφαρμογή μας βοηθά σημαντικά τους χρήστες να καταλάβουν που θα πρέπει να τοποθετηθούν και σε ποια απόσταση από τον αισθητήρα έτσι ώστε να μπορεί να τους ανιχνεύσει με μεγαλύτερη ακρίβεια.Επίσης μειώνει αισθητά την αίσθηση που μπορεί να έχει ο χρήστης κατα την αλληλεπίδραση του με την εφαρμογή ότι δεν δουλεύει σωστά ή ότι το Kinect έχει κολλήσει για κάποιο λόγο είτε ότι είναι υπερευαίσθητο.

5.2.1Πως «επίλεγουμε» με την βοήθεια κουμπιών

Με την βοήθεια του KinectButtonκαι του KinectCursorείσαγουμε εναν καινούριο τρόπο αλληλεπίδρασης «την πίεση του χερίου προς τα μέσα!».Αυτό συνέβη πολύ απλά γιατί ο χρήστης όταν αλληλεπιδρά με το Kinectκαι αν υπάρχουν κουμπιά στην εφαρμογή η πρώτη αντίδραση που θα έχει είναι να πιέσει το χέρι του σαν να πίεζε ένα κουμπί στον πραγματικό κόσμο ή να έδειχνε με το χέρι του ένα αντικείμενο.

Το KinectToolkitπαρέχειΔύο είδη κουμπιών στον προγραμματιστή:

5.2.2 Τα TileButtons :



- ✓ Τα οποία μεγαλώνουν και μικραίνουν

- ✓ Μπορούν να αλλάξουν χρώμα

- ✓ Μπορούμε να προσθέσουμε κείμενο πάνω σε αυτά

5.2.3 Τα CircleButtons

- ✓ Τα οποία μεγαλώνουν και μικραίνουν

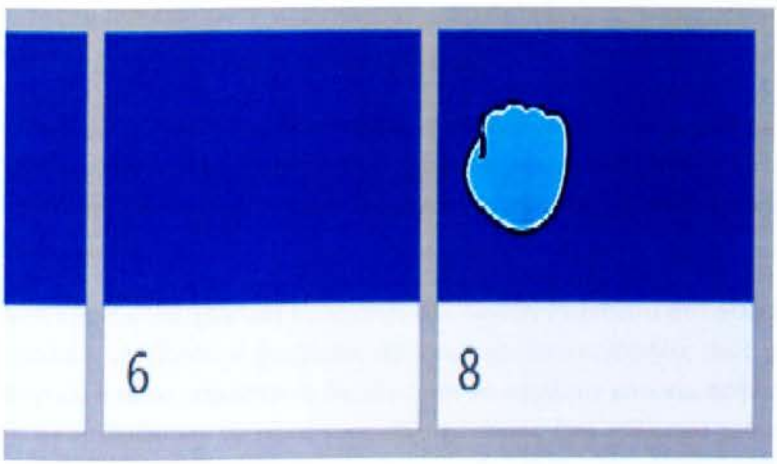
- ✓ Μπορούν να αλλάξουν χρώμα

- ✓ Μπορούμε να προσθέσουμε κείμενο πάνω σε αυτά

Συνήθως τα CircleButtonsχρισημοποιούνται για εντολές όπως: Back,Home,Exit.

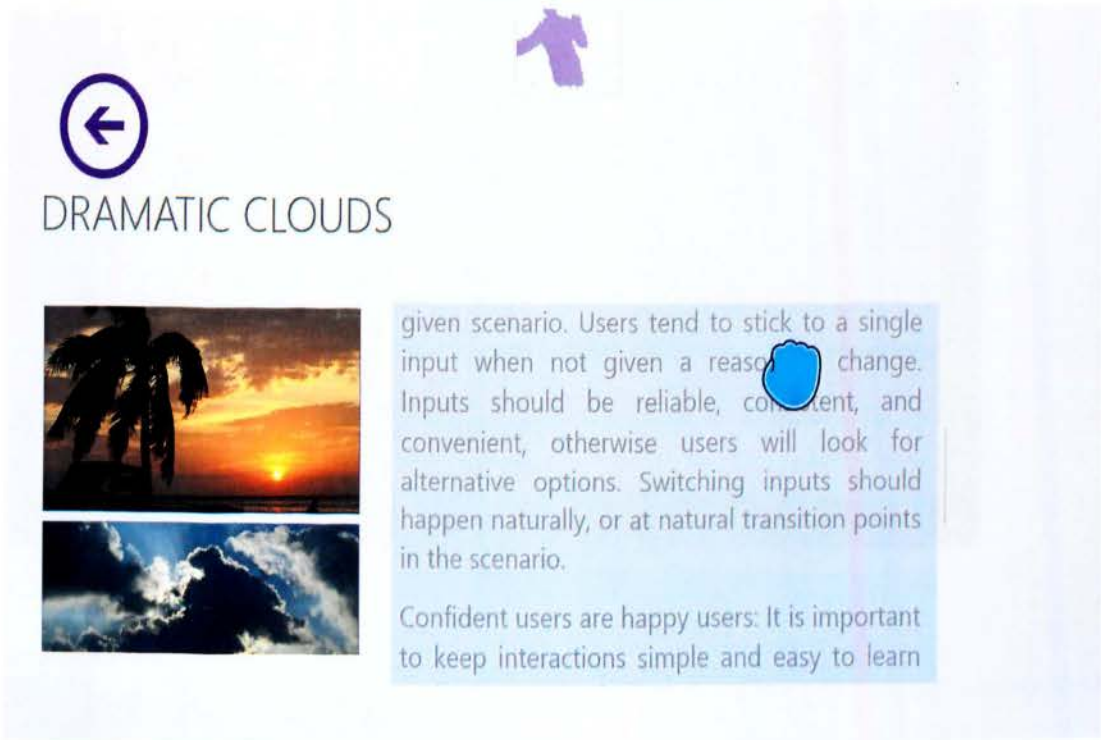
5.3 Scrolling(Κύλιση)

Με το KinectScrollεισάγετε ένας καινούριος άμεσος χειρισμός με το «πιάσε – κίνησε».Το Kinectμπορεί να ανιχνέυσει το χέρι ενος χρήστη που γίνεται γροθιά και να κυλίσει με ευκολία το περιεχόμενο της εφαρμογής μέσα στο χώρο.



Το Scrollingμπορει να επιτευχθεί είτε σε οριζόντια κίνηση είτε κάθετη.

Η χρησιμότητα του Scrollingγια διάβασμα κειμένου :



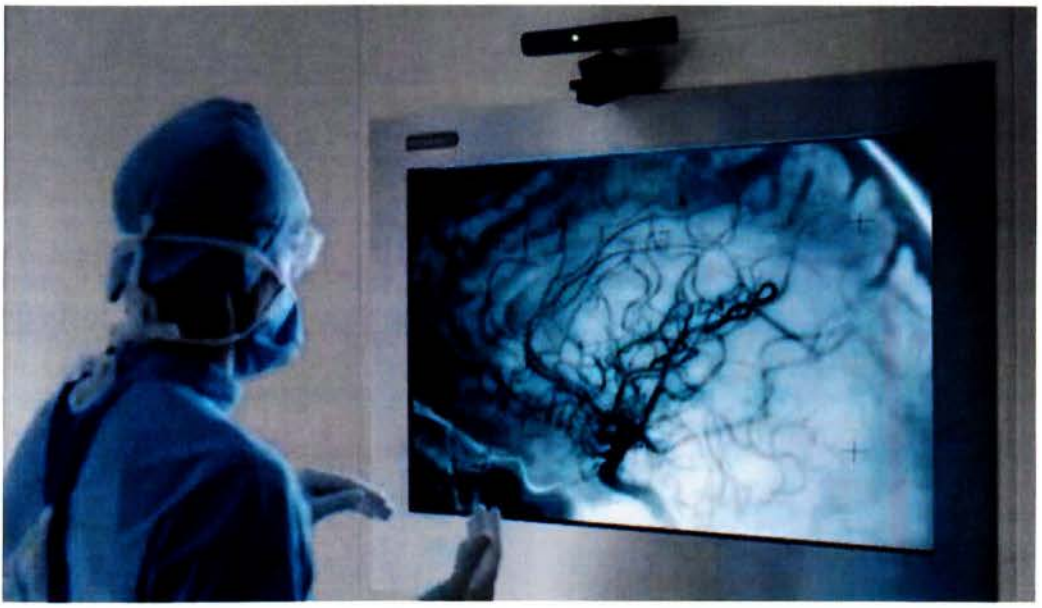
6.1Παραδείγματα από υπάρχουσες εφαρμογές που βρίσκουν εφαρμογή στον πραγματικό κόσμο

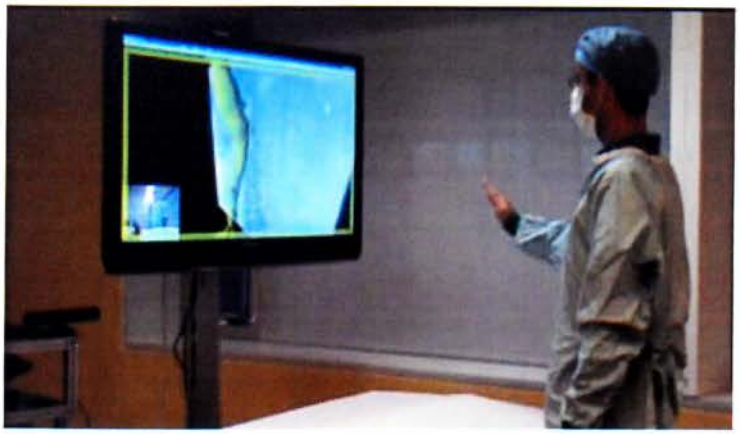
6.1.1 Εφαρμογή στην ιατρική

Μπορούν να δημιουργηθούν διάφορες εφαρμογές για την ιατρική οπως η παρακολούθηση κινήσεων ασθενή και βοήθεια στην διάγνωση νοσήματος όπως ο αυτισμός, η διαταραχή προσοχής ή και ακόμη η ιδεοψυχαναγκαστική διαταραχή.Μετρήσεις άσκησης για τέστ κοπώσεως.

Μια πάρα πολύ χρήσιμη εφαρμογή του Kinectσυναντάται στα χειρουργία όπου οι γιατροί μπορούν να βλέπουν φακέλους ασθενών και ακτινογραφίες τους χωρίς να χρειάζεται να φύγουν απο το χειρουργικό δωμάτιο και να αγγίξουν κάτι και στην συνέχεια να χρειάζεται να αποστηρώσουν τα χέρια τους πάλι χάνοντας έτσι πολύτιμο χρόνο απο την χειρουργική επέμβαση την οποία πραγματοποιούν την δεδομένη στιγμή.

Φωτογραφίες κατα την διάρκεια χειρουργίων όπου γιατροί βλέπουν ακτινογραφίες ασθενών χωρίς να αγγίξουν τίποτα και χωρίς να βγούν άσκοπα απο το χειρουργείο τους.





Αναφορές:

http://www.digitaltrends.com/computing/kinect/

6.1.2 Εφαρμογή στην ρομποτική

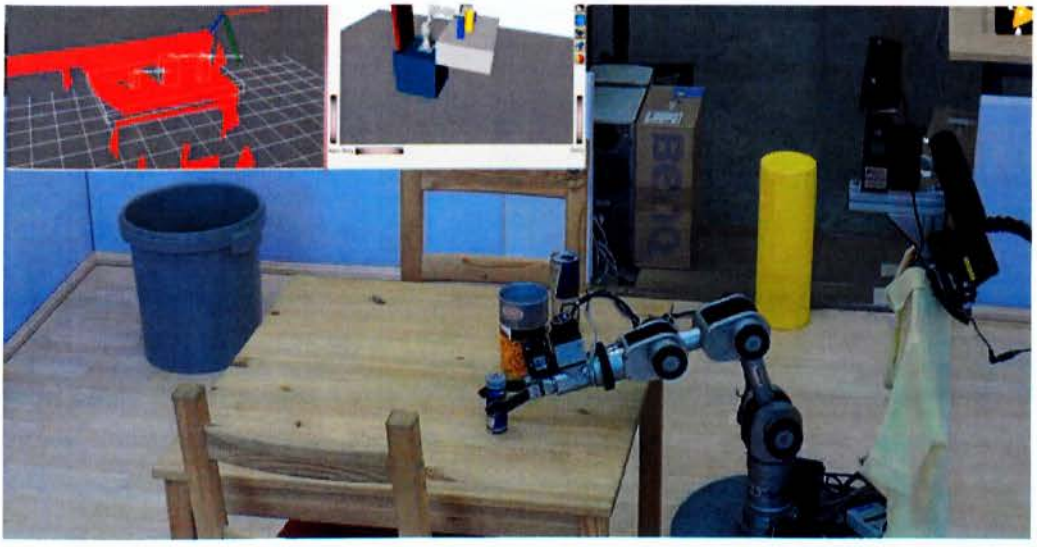
Εφαρμογή πλοήγησης για ρομπότ με την βοήθειαχειρονομιών φωνητικών εντολών ή κινήσεις σώματος.

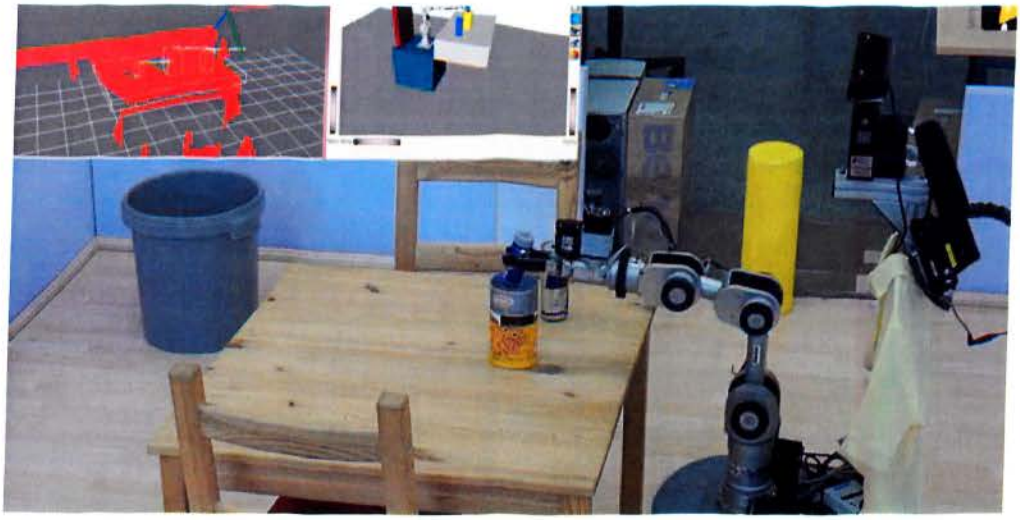
Επίσης μπορεί να χρησιμοποιηθεί για την αναγνώριση αντικειμένων και την εκτίμηση απόστασης απο το αντικείμενο.Βραχίονες μπορούν να μεταχειριστούν τα αντικείμενα αφού αναγνωρισθούν από τον αισθητήρα.Αυτό μπορεί να επιτευχθεί σε τρία στάδια. Αρχικά, το Kinectθα χαρτογραφήσει το περιβάλλον μέσα στο οποίο βρίσκεταικαι θα αναγνωρίσει το αντικείμενο και τυχόν εμπόδια που θα υπάρχουν μπροστά του. Στη συνέχεια,θα πιάσει το αντίκειμενο κι έπειτα θα μπορέσει να χειριστεί το αντικείμενο στο χαρτογραφημένο περιβάλλον.

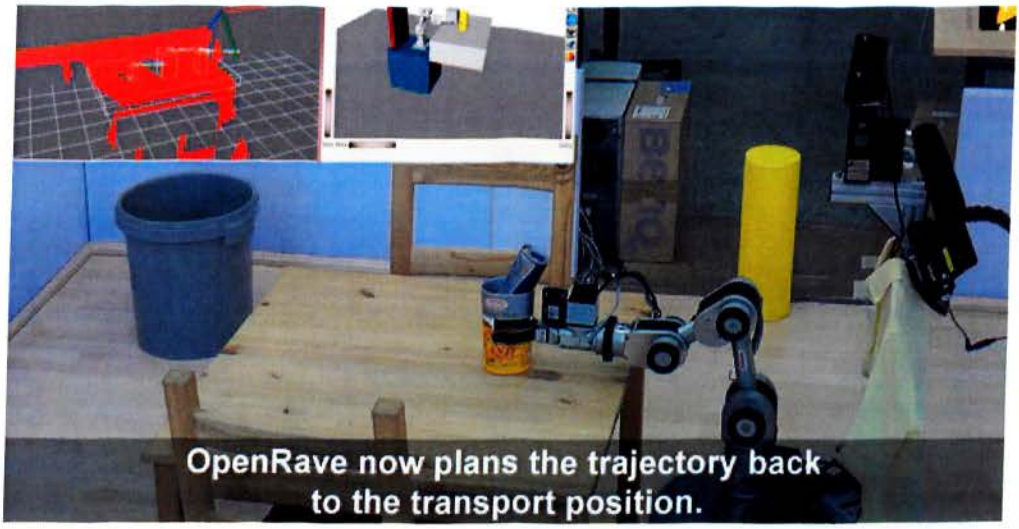
Φωτογραφίες απο μια τέτοια εφαρμογή(βήμα προς βήμα)

1° βημα χαρτογράφηση περιβάλλοντος και αναγνώριση αντικειμένου

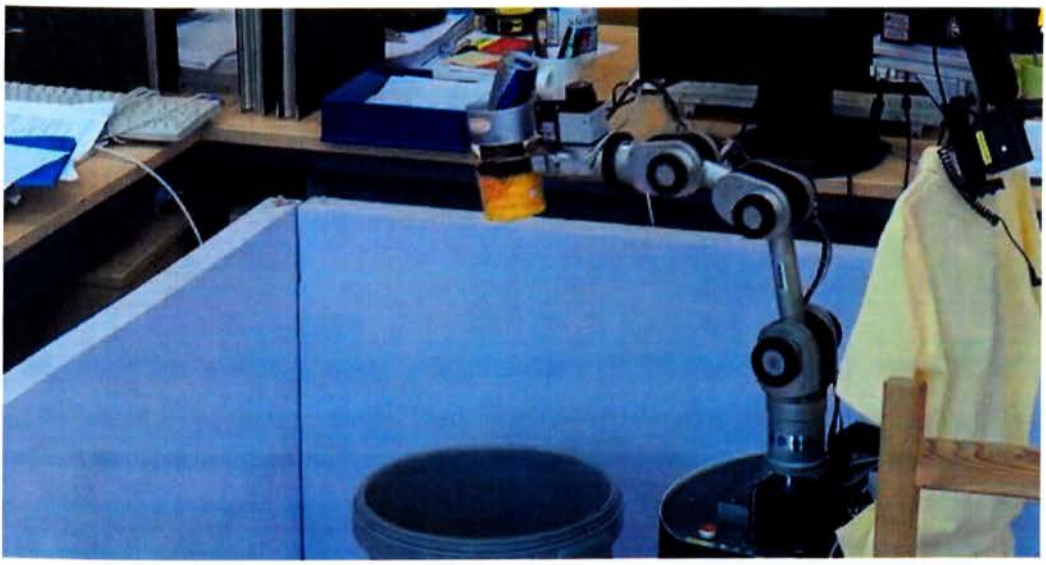
2° βήμα πιάσιμο αντικειμένου

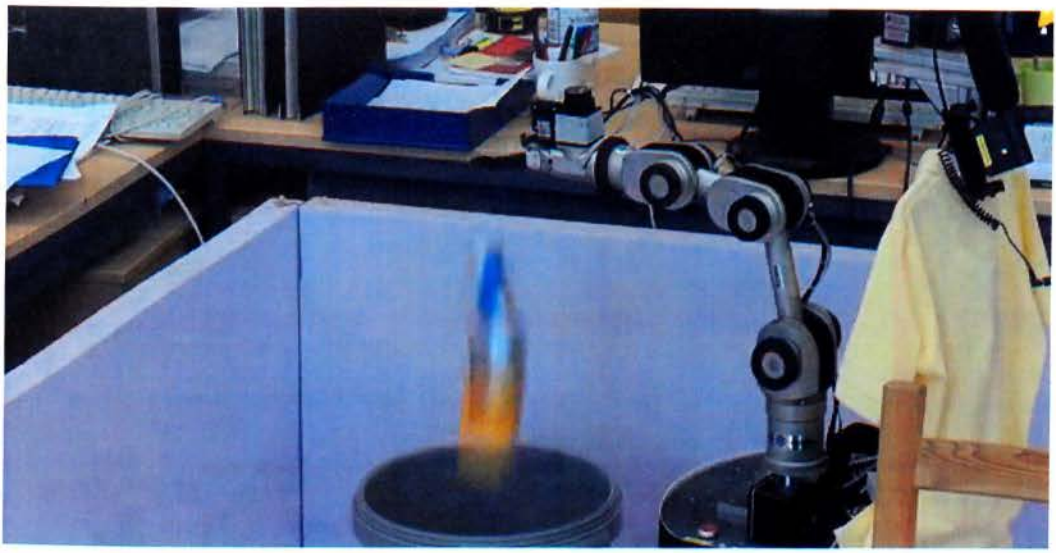






3^{ο} βήμα χειρισμός αντικειμένου στο χαρτογραφημένο περιβάλλον





Αναφορές:

http://blog.robotiq.com/bid/43616/Kinect-in-Industrial-Robotic-Application

6.1.3 Εφαρμογή στα συστήματα ασφαλείας

Το Kinectμπορεί να χρισημοποιηθεί ώστε να ανιχνέυει κινήσεις σώματος ή πρόσωπα και να στέλνει το υλικό στο χρήστη.

Επίσης, μπορεί να χρησιμοποιηθεί για βιομετρική πιστοποίηση ανιχνεύοντας τους σκελετούς και συγκρίνοντάς τους με αυτούς που υπάρχουν στην βάση δεδομένων μας.

Εικόνες από την εφαρμογή

Αναφορές:

http://www.kinecthacks.com/the-securty-camera-kinectonitor/

http://abhijitjana.net/2012/04/16/home-security-system-using-kinect-azure-windows-phone-and-windows-8/

6.1.4 Εφαρμογή στην εικονική πραγματικότητα

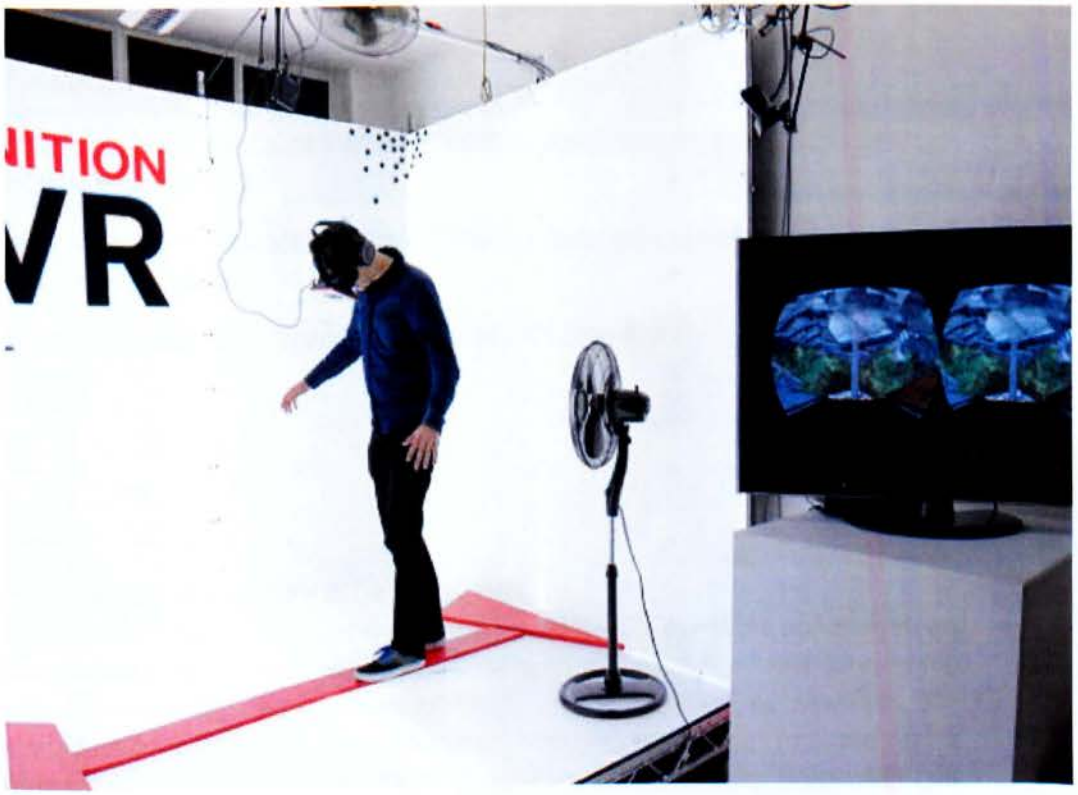
Με την βοήθεια της 3Dτεχνολογίας του Kinect και την δυνατότητα ανίχνευσης ενός ανθρώπινου σκελετού μπορούν να δημιουργηθούν πολλές εφαρμογές με την χρήση του αισθητήρα.

Εικόνες εφαρμογών εικονικής πραγματικότητας με την χρήση του Kinect.

Δημιουργία τρισδιάστατου χώρου με το Kinect.

Δημιουργία τρισδιάστατου τοπίου με την βοήθεια του Kinectκαι κίνηση με την δυνατότητα ανίχνευσης ενός ανθρώπινου σκελετού.


ON
R


ITION
VR

Δημιουργία στολής υπερήρωαεπάνω στο σώμα ενός ανθρώπου με την βοήθεια τού Kinectκαι την δυνατότητα ανίχνευσης ενός ανθρώπινου σκελετού.

Αναφορές:

http://www.worldviz.com/press-release/consumer-priced-immersive-virtual-reality-with-kinect-and-sony-3d-goggles

http://www.theverge.com/2013/7/31/4575758/a-virtual-reality-paperboy-with-help-from-kinect-and-the-oculus-rift

http://channel9.msdn.com/coding4fun/kinect/Kinect--virtual-reality--telepresence--Virtual-traveller

Η Βάση της πτυχιακής εργασίας

Όπως φαίνεται από τον τίτλο της πτυχιακής δεν χωράει καμια αμφιβολία ότι πρόκειται για εκπαιδευτικού περιεχομένου εφαρμογή που έχει ως σκοπό να βοηθήσει τους φοιτητές που παρακολουθούν εργαστήρια προγραμματισμού να κατανοησούν τις καινούριες και καινοτόμες δυνατότητες που παρέχει ο αισθητήρας Kinectκαι να μεταβούν ομαλά απο το GUIστο νέο NUI (naturaluserinterface) περιβάλλον με την χρήση χειρονομιών και φωνητικών εντολών.Με τις δυνατότητες αυτές απλες εφαρμογές και παιχνίδια που

έφτιαχναν πριν οι φοιτητές οπως μια τρίλιζα η κρεμάλα μπορούν πλέον να αλλάξουν επίπεδο μπαίνοντας σιγά σιγά στην λογική του NUIξεφεύγοντας απο το ποντίκι και το πληκτρολόγιο.

7 Οι εφαρμογές που αναπτύχθηκαν

Η εφαρμογή μας είναι βασισμένη σε 2 ξεχωριστές εφαρμογές!

1η εφαρμογή το QUIZGAMEτο οποίο είναι ενα κλασσικό παιχνίδι γνώσεων

2η εφαρμογή το BUGSPLUTτο οποίο είναι ένα παιχνίδι στο οποίο ο παίκτης προσπαθεί να πετύχει έντομα με μια μυγοσκοτώστρα!

7.1 ToQUIZGAME!

Σκοπός ανάπτυξης του παιχνιδιού

Το QUIZGAME αναπτύχθηκε με το σκεπτικό οτι θα μπορέσει να γίνει ένα χρήσιμο εργαλείο στα χέρια των εκπαιδευτικών αλλά και των φοιτητών.Θα μπορεί να χρησιμοποιηθεί ώστε ο καθηγητής να παίζει το παιχνίδι με τους φοιτητές τους στο τέλος κάθε μαθήματος για να καταλάβει τι γνώσεις αποκομίσαν από το κάθε μάθημα ή θα μπορούσε να βάλει ερωτήσεις από παλιά θέματα του εργαστηρίου ώστε να δώσει μια γεύση στους φοιτητές για το τι θα συναντήσουν στην τελική εξέταση του εργαστηρίου.

7.1.1 Περιγραφή παιχνιδιού

ToQUIZGAMEείναι βασισμένο στον αισθητήρα KINECTκαι πιο συγκεκριμένα στον αισθητήρα βάθους και στην δυνατότητα που παρέχει το KINECTανίχνευσης κίνησηςενός ανθρώπινου σκελετού.Το παιχνίδι ξεκινάει κι αμέσως εμφανίζεται στην οθόνη αρχική σελίδα του παιχνιδιού και στη συνέχεια η πρώτη ερώτηση προς απάντηση και ακολουθούν αλλες 31. Το παιχνίδι δίνει τη δυνατότητα ο χρήστης να κάνει μέχρι 3 λάθη καθώς μετράει και τους πόντους του χρήστη κατα την διάρκεια του παιχνιδιού. Με κάθε σωστή απάντηση ο χρήστης κερδίζει 150 πόντους.Επίσης, έχουν προστεθεί και ήχοι κατα την διάρκεια του παιχνιδιού όταν ο χρήστης επιλέγει σωστή απάντηση και όταν ο χρήστης επιλέγει λάθος απάντηση. Ένας άλλος ήχος ακούγεται στο τέλος του παιχνιδιού εφόσον ο χρήστης έχει απαντήσει σωστά και στις 32 ερωτήσεις της εφαρμογής.

7.1.2 Στιγμιότυπα απο την πορεία του παιχνιδιού

Με την εκίνηση της εφαρμογής ο αισθητήρας ανιχεύει τα χέρια του χρήστη και εμφανίζεται η αρχική σελίδα του παιχνιδιού.

Αρχική σελίδα παιχνιδιού

Όπως βλέπουμε το χεράκι εμφανίζεται αφού ο αισθητήρας ανιχνεύσει το αριστερό ή το δεξί χέρι του χρήστη.

Αρχική σελίδα παιχνιδιού(με το χέρι του παίκτη να αλληλεπιδρά με την εφαρμογή!)

7.1.3 Κώδικας του xamlαρχείου για την δημιουργία των κουμπιών και του φόντου και του UserViewer(επάνω στο κέντρο).

Σημείωση για τον xaml κώδικα :με το μπλέ χρώμα είναι γραμμένα τα controlστης φόρμας.πχ τα buttons,labels, textblocks, KinectUserViewer,backgrounds,images.

```
<Page x:Class="QuestionGame.MainMenu"
      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
      xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
      xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
      xmlns:k="http://schemas.microsoft.com/kinect/2013"
mc:Ignorable="d"
d:DesignHeight="768" d:DesignWidth="1366"
       Title="Main Menu" Loaded="Page_Loaded">

<Page.Background>
<ImageBrush  ImageSource="Images/main_menu_image.jpg" />
</Page.Background>

<Grid>
<k:KinectUserViewer HorizontalAlignment="Center" VerticalAlignment="Top"
Height="100"
k:KinectRegion.KinectRegion="{Binding ElementName=KinectRegion}"/>

<k:KinectRegion Name="KinectRegion" Margin="98,28,-98,-28">
<Grid>
<k:KinectTileButton Name="StartGameButton" Content="Νέοπαιχνίδι"
Margin="263,184,378,392" Height="Auto" Width="Auto"
Click="StartGameButton_Click"
                                    FontSize="72" Foreground="Black"
Background="Yellow"/>

<k:KinectTileButton Name="ExitGameButton" Content="ΕΞΟΔΟΣ"
Margin="792,518,162,98" Width="Auto" Height="Auto" FontSize="72"
                                    Background="Yellow"  Foreground="Black"
Click="ExitGameButton_Click" />

</Grid>
</k:KinectRegion>
</Grid>

</Page>
```

7.1.4 Ο C# κώδικας της φόρμας MainMenu

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
```

```
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using Microsoft.Kinect;
using Microsoft.Kinect.Toolkit;
using Microsoft.Kinect.Toolkit.Controls;

namespace QuestionGame
{
///<summary>
/// Interaction logic for MainFrame.xaml
///</summary>
publicpartialclassMainMenu : Page
    {
privateKinectSensor mySensor;

public MainMenu()
        {
InitializeComponent();


        }

//page loaded event handler
privatevoid Page_Loaded(object sender, RoutedEventArgs e)
        {
mySensor = GlobalSensor.globalSensor;

//bind sensor to kinect region
            KinectRegion.KinectSensor = mySensor;

//if (mySensor != null)
//{
//    //bind the sensor to the kinect region
//    var regionSensorBinding = new Binding("KinectRegion")
//    {
//        Source = mySensor
//    };

//    BindingOperations.SetBinding(this.KinectRegion,
KinectRegion.KinectRegionProperty, regionSensorBinding);
//}
        }

//event handler for start game button
privatevoid StartGameButton_Click(object sender, RoutedEventArgs e)
        {
//load GameFrame page
            (Application.Current.MainWindow.FindName("MainFrame")
asFrame).Source = newUri("GameFrame.xaml", UriKind.Relative);

mySensor = null;
            KinectRegion.KinectSensor = null;
        }

privatevoid ExitGameButton_Click(object sender, RoutedEventArgs e)
        {
//stop sensor
GlobalSensor.stopKinectSensor(mySensor);

//close application
```

```
Application.Current.MainWindow.Close();
}
    }
}
```

Στη συνέχεια ακολουθούν κάποια Screenshots από την εφαρμογή, καθώς οχρήστης παίζει το παιχνίδι.

Γραφικό περιβάλλον της εφαρμογής

7.1.5 Κώδικας του xamlαρχείου για την δημιουργία των κουμπιών και του φόντου,του UserViewer(επάνω αριστερά),των labelsγια τα κουμπιά,το Labelγια την ερώτηση και τα labelsγια τους πόντους και τις ζωές.

```
<Page x:Class="QuestionGame.GameFrame"
      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
      xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
      xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
      xmlns:k="http://schemas.microsoft.com/kinect/2013"
mc:Ignorable="d"
d:DesignHeight="768" d:DesignWidth="1366"
       Title="GameFrame" Loaded="Page_Loaded">

<Page.Background>
<ImageBrush ImageSource="Images/background_image.jpg" />
</Page.Background>

<Grid>
<k:KinectUserViewer HorizontalAlignment="Left" VerticalAlignment="Top"
Height="100"
                           k:KinectRegion.KinectRegion="{Binding
ElementName=KinectRegion}" />

<k:KinectRegion Name="KinectRegion">
<Canvas Name="GameCanvas">
<Image Name="QuestionImageBox" Height="285" Width="480" Canvas.Left="375"
Canvas.Top="10" />

<Image Source="Images/lifes.png" Height="82" Width="76" Canvas.Left="931"
Canvas.Top="184" />

<Label Name="PlayerTriesLabel" Content="3" Canvas.Left="1023" Canvas.Top="194"
FontSize="48" Height="72" Width="46" Foreground="#FFE6FD00"
HorizontalContentAlignment="Stretch" Padding="1,1,1,3"
VerticalContentAlignment="Stretch" />

<Label Name="QuestionNumberLabel" Content="Ερώτηση: 1/32" Height="82"
Width="451" Canvas.Right="10" Canvas.Left="931" Canvas.Top="10"
                      FontSize="48" Foreground="#FFE6FD00"/>

<Image Source="Images/money_image.png" Height="82" Width="87" Canvas.Right="10"
Canvas.Left="931" Canvas.Top="97" />

<Label Name="PointsLabel" Content="0 $" Height="82" Width="359"
Canvas.Right="10" Canvas.Top="97" Canvas.Left="1023"
                      FontSize="48" Foreground="#FFE6FD00"/>

<Label Name="QuestionsLabel" Width="1158" Height="124" FontSize="36"
                      FontWeight="Bold" Canvas.Left="14" Canvas.Top="300"
BorderThickness="2" BorderBrush="Bisque" Foreground="Black"
HorizontalContentAlignment="Center" Background="#FFF7FF52"/>

<k:KinectTileButton Name="AnswerButton_A" Height="122" Width="581"
Background="Yellow"  FontWeight="Bold" Foreground="Black"
Click="AnswerButton_A_Click" Canvas.Left="14" Canvas.Top="444"/>

<k:KinectTileButton Name="AnswerButton_B" Height="122" Width="581"
Canvas.Left="609" Canvas.Top="444" Background="Yellow"  FontWeight="Bold"
Foreground="Black" Click="AnswerButton_B_Click"/>
```

```
<k:KinectTileButton Name="AnswerButton_C" Height="122" Width="581"
Background="Yellow"  FontWeight="Bold" Foreground="Black"
Click="AnswerButton_C_Click" Canvas.Left="10" Canvas.Top="586"/>

<k:KinectTileButton Name="AnswerButton_D" Height="122" Width="576"
Background="Yellow" Canvas.Left="609" Canvas.Top="586"  FontWeight="Bold"
Foreground="Black" Click="AnswerButton_D_Click"/>

<k:KinectCircleButton Name="CloseButton"  Content="Exit" Canvas.Right="10"
Canvas.Bottom="10" Height="146" Canvas.Left="1223" Canvas.Top="703"
Width="159" Click="CloseButton_Click" />
</Canvas>
</k:KinectRegion>
```

7.1.6 OC# κώδικαςτιςφόρμαςGameFrame

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using Microsoft.Kinect;
using Microsoft.Kinect.Toolkit;
using System.Media;

namespace QuestionGame
{
///<summary>
/// Interaction logic for GameFrame.xaml
///</summary>
publicpartialclassGameFrame : Page
    {
privateKinectSensor mySensor;

privateQuestionSet questionSet = newQuestionSet();
privateString[][] questions; //2d array with random questions
privateString[] answers;   //array with the answers
privateImageSource[] images; //array with question images
privateString correctAnswer;

privatestaticint index;
privatestaticint questionNumber;
publicstaticint playerPoints;
privateint playerTries;

//sounds
privateSoundPlayer playCorrectSound =
newSoundPlayer(QuestionGame.Properties.Resources.correctSound);
```

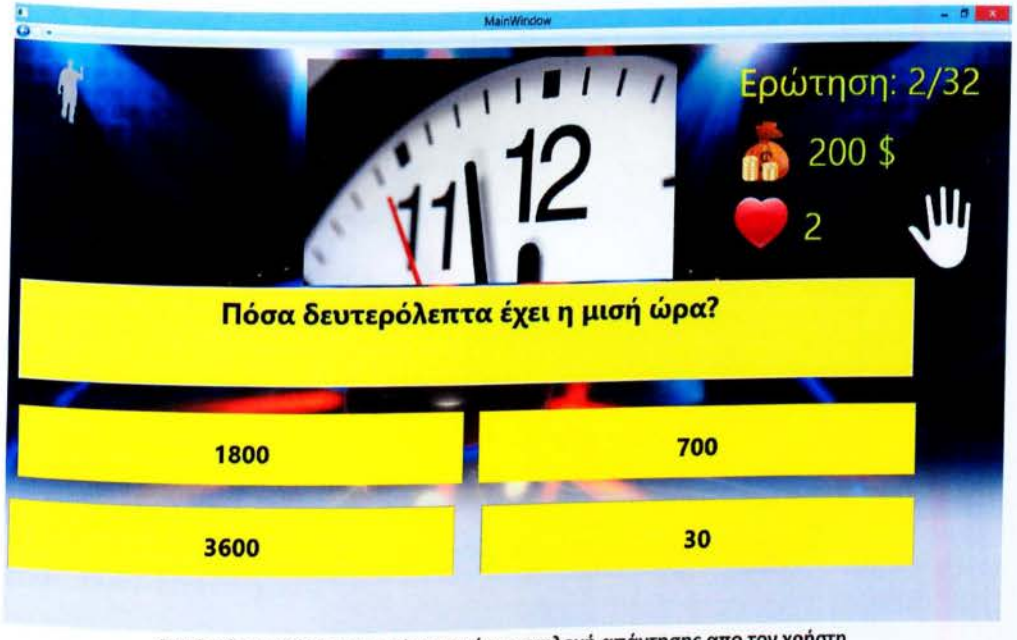
```
privateSoundPlayer playWrongSound =
newSoundPlayer(QuestionGame.Properties.Resources.wrongSound);

public GameFrame()
        {
InitializeComponent();
        }

privatevoid Page_Loaded(object sender, RoutedEventArgs e)
        {
if (GlobalSensor.globalSensor != null)
            {
mySensor = GlobalSensor.globalSensor;

//bind sensor to kinect region
                KinectRegion.KinectSensor = mySensor;

//load the questions
loadQuestions();

//add question to label
nextQuestion();
            }
        }

//load questions
privatevoid loadQuestions()
        {
questions = questionSet.loadQuestions();
answers = questionSet.getQuestionsAnswers();
images = questionSet.getQuestionsImages();

index=0;
questionNumber=1;
playerPoints=0;
playerTries=3;
        }

//show the next question and next image
privatevoid nextQuestion()
        {
//add question to question label
            QuestionsLabel.Content = questions[index][0];

//add 4 posible ansers to tile buttons
            AnswerButton_A.Content = questions[index][1];
            AnswerButton_B.Content = questions[index][2];
            AnswerButton_C.Content = questions[index][3];
            AnswerButton_D.Content = questions[index][4];

//add image to imagebox
            QuestionImageBox.Source = images[index];

//get questions correct answer
correctAnswer = answers[index];

index++;
        }

//calc points
privateint getPointScale(int questionNum)
        {
```

```
int points = 0;

if (questionNum <= 5)
points=100;
elseif (questionNum <= 10)
points=150;
elseif (questionNum <= 15)
points=250;
elseif (questionNum <= 20)
points=350;
elseif (questionNum <= 25)
points=500;

return points;
        }

//check player answer
privatevoid checkUserAnswer(object sender, String answer)
        {
//1 sec delay
System.Threading.Thread.Sleep(1000);

if (answer == correctAnswer)
            {
//play sound
playCorrectSound.Play();

//load new question
nextQuestion();

                QuestionNumberLabel.Content = "Ερώτηση: " +
(questionNumber++)+"/32";

playerPoints += getPointScale(questionNumber);
                PointsLabel.Content = playerPoints+" $";

//load winner frame
if (questionNumber == 32)
                {
                    (Application.Current.MainWindow.FindName("MainFrame")
asFrame).Source = newUri("WinnerFrame.xaml", UriKind.Relative);

mySensor = null;
                    KinectRegion.KinectSensor = null;
                }
            }
else
            {
playerTries--;
playWrongSound.Play();

if (playerTries >= 0)
                {
                    PlayerTriesLabel.Content = "" + playerTries;
nextQuestion();
                }
else
                {
//game ends, load ExitFrame page
                    (Application.Current.MainWindow.FindName("MainFrame")
asFrame).Source = newUri("ExitFrame.xaml", UriKind.Relative);
```

```
mySensor = null;
                    KinectRegion.KinectSensor = null;
                }
            }

        }

//AnswerButton_A event handler
privatevoid AnswerButton_A_Click(object sender, RoutedEventArgs e)
        {
checkUserAnswer(sender, AnswerButton_A.Content.ToString());
}

//AnswerButton_B event handler
privatevoid AnswerButton_B_Click(object sender, RoutedEventArgs e)
        {
checkUserAnswer(sender, AnswerButton_B.Content.ToString());
}

//AnswerButton_C event handler
privatevoid AnswerButton_C_Click(object sender, RoutedEventArgs e)
        {
checkUserAnswer(sender, AnswerButton_C.Content.ToString());
}

//AnswerButton_D event handler
privatevoid AnswerButton_D_Click(object sender, RoutedEventArgs e)
        {
checkUserAnswer(sender, AnswerButton_D.Content.ToString());
        }

//close button event handler
privatevoid CloseButton_Click(object sender, RoutedEventArgs e)
        {
//stop sensor
GlobalSensor.stopKinectSensor(mySensor);

//close application
Application.Current.MainWindow.Close();

}

    }
}
```

Καθώς το παιχνίδι ξεκινάει ο χρήστης βλέπει την πρώτη ερώτηση του παιχνιδιού με 4 επιλογές για να απαντήσει. Επάνω δεξιά υπάρχει μετρητής για το πόσες λάθος απαντήσεις μπορεί να κάνει ο χρήστης και ακριβώς από κάτω οι πόντοι που κερδίζει με κάθε σωστή απάντηση. Επίσης ακούγεται ήχος αν η απάντηση είναι σώστη ή λάθος αντίστοιχα.

Γραφικό περιβάλλον της εφαρμογής με επιλογή απάντησης από τον χρήστη



Γραφικό περιβάλλον της εφαρμογής με επιλογή απάντησης απο τον χρήστη

Σε αυτή την εικόνα φαίνεται η χρήση του μετρητή λάθους απαντήσεων επάνω δεξιά.

Εικόνα που βλέπει ο χρήστης μετα το πέρας του παιχνιδιού με επιλογή απάντησης απο τον χρήστη

7.1.7 Κώδικας του xamlαρχείου για την δημιουργία των circleButtonsκαι του φόντου,του UserViewer(επάνω στο κέντρο)και των labelsπου εμφανίζει τα συνολικά κέρδη και το μήνυμα ότι ο χρήστης έχασε.

```
<Page x:Class="QuestionGame.ExitFrame"
      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
      xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
      xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
      xmlns:k="http://schemas.microsoft.com/kinect/2013"
mc:Ignorable="d"
d:DesignHeight="900" d:DesignWidth="1400"
       Title="ExitFrame" Loaded="Page_Loaded">

<Page.Background>
<ImageBrush ImageSource="Images/exit_menu_image.jpg" />
</Page.Background>

<Grid>
<k:KinectUserViewer HorizontalAlignment="Center" VerticalAlignment="Top"
Height="100"
                          k:KinectRegion.KinectRegion="{Binding
ElementName=KinectRegion}" />

<k:KinectRegion Name="KinectRegion">
<Canvas>
```

```
<Image Source="Images/money_image.png" Height="120" Width="115"
Canvas.Left="298" Canvas.Top="212" />

<Label Name="PlayerTotalPoints" Content="ΣυνολικάΚέρδη " FontWeight="Bold"
Height="120" FontSize="48" Canvas.Left="418" Width="812" Canvas.Top="212" />

<Label Name="InfoLabel" Content="Έχασες..! Θέλειςναξαναπαίξεις?"
FontWeight="Bold"   Height="120" FontSize="72" Canvas.Left="146"
Canvas.Top="357" Background="{x:Null}" />


<k:KinectCircleButton Name="YesCircleButton" Content="NAI" Canvas.Left="366"
Canvas.Top="482" Height="222" Width="249" Click="YesCircleButton_Click" />

<k:KinectCircleButton Name="NoCircleButton" Content="OXI" Canvas.Left="775"
Canvas.Top="482" Height="222" Width="249" Click="NoCircleButton_Click" />

</Canvas>
</k:KinectRegion>
</Grid>
</Page>
```

7.1.8 OC# κώδικαςτιςφόρμαςExitFrame

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using Microsoft.Kinect;
using Microsoft.Kinect.Toolkit;
using Microsoft.Kinect.Toolkit.Controls;

namespace QuestionGame
{
///<summary>
/// Interaction logic for ExitFrame.xaml
///</summary>
publicpartialclassExitFrame : Page
    {
privateKinectSensor mySensor;

public ExitFrame()
        {
InitializeComponent();
        }

//page loaded event handler
privatevoid Page_Loaded(object sender, RoutedEventArgs e)
        {
if (GlobalSensor.globalSensor != null)
            {
```

```
mySensor = GlobalSensor.globalSensor;

//bind sensor to kinect region
                KinectRegion.KinectSensor = mySensor;

                PlayerTotalPoints.Content = "Συνολικάκέρδη"
+GameFrame.playerPoints+" $";
            }
        }

//YesCircleButton event handler
privatevoid YesCircleButton_Click(object sender, RoutedEventArgs e)
        {
//load GameFrame page to restart game
            (Application.Current.MainWindow.FindName("MainFrame")
asFrame).Source = newUri("GameFrame.xaml", UriKind.Relative);

mySensor = null;
            KinectRegion.KinectSensor = null;
        }

//NoCircleButon event handler
privatevoid NoCircleButton_Click(object sender, RoutedEventArgs e)
        {
//stop sensor
GlobalSensor.stopKinectSensor(mySensor);

//close application
Application.Current.MainWindow.Close();
}

////YesTileButton event handler
//private void YesTileButton_Click(object sender, RoutedEventArgs e)
//{
//    //load GameFrame page to restart game
//    (Application.Current.MainWindow.FindName("MainFrame") as Frame).Source =
new Uri("GameFrame.xaml", UriKind.Relative);

//    mySensor = null;
//    KinectRegion.KinectSensor = null;
//}

////NoTileButon event handler
//private void NoTileButton_Click(object sender, RoutedEventArgs e)
//{
//    //stop sensor
//    GlobalSensor.stopKinectSensor(mySensor);
//}
}
}
```

Εικόνα που βλέπει ο χρήστης μετα το πέρας του παιχνιδιού εφόσον έχει νικήσεικαι ακούγετε ήχος εφόσον φτάσαμε σε αυτό το σημείο.

7.1.9 Κώδικας του xamlαρχείου για την δημιουργία του φόντου,των labelsπου εμφανίζει τα συνολικά κέρδη και το μήνυμα ότι ο χρήστης κέρδισε και των κουμπιών που μας επιστρέφουν στην αρχική οθώνη η κάνει έξοδο απο το παιχνίδι.

```
<Page x:Class="QuestionGame.WinnerFrame"
      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
      xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
      xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
      xmlns:k="http://schemas.microsoft.com/kinect/2013"
mc:Ignorable="d"
d:DesignHeight="900" d:DesignWidth="1400"
       Title="WinnerFrame" Loaded="Page_Loaded">

<Page.Background>
<ImageBrush ImageSource="Images/win_frame_image.jpg" />
</Page.Background>

<Grid>
<k:KinectUserViewer HorizontalAlignment="Center" VerticalAlignment="Top"
Height="100"
                            k:KinectRegion.KinectRegion="{Binding
ElementName=KinectRegion}" />

<k:KinectRegion Name="KinectRegion">
<Canvas>
```

```
<Label Name="PlayerTotalPoints"  Content="ΣΥΓΧΑΡΗΤΗΡΙΑ!" FontWeight="Bold"
Height="110" FontSize="72" Canvas.Left="421" Canvas.Top="147" Width="573"
Foreground="Black" Background="#FF1867F0" />

<Image Source="Images/money_image.png" Canvas.Left="73" Canvas.Top="398"
Height="84" Width="123" />

<Label Name="TotalPointsLabel" Content="ΣΥΝΟΛΙΚΑΚΕΡΔΗ:" Canvas.Left="201"
Canvas.Top="338" FontSize="56" FontWeight="Bold" Width="921"
Background="#FF1867F0" />

<k:KinectTileButton Name="StartGameTileButton" Content="ΑΡΧΙΚΗΟΘΩΝΗ"
Canvas.Left="467" Canvas.Top="560" Width="350" Height="117" Foreground="Black"
FontWeight="Bold" Click="StartGameTileButton_Click" Background="Yellow" />

<k:KinectTileButton Name="ExitGameTileButton" Content="ΕΞΟΔΟΣ"
Canvas.Left="876" Canvas.Top="560" Height="117" Width="350" Foreground="Black"
Background="Yellow" FontWeight="Bold" Click="ExitGameTileButton_Click" />

</Canvas>
</k:KinectRegion>
</Grid>
</Page>
```

7.1.10 OC# κώδικαςτιςφόρμαςWinnerFrame

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using Microsoft.Kinect;
using Microsoft.Kinect.Toolkit;
using Microsoft.Kinect.Toolkit.Interaction;
using System.Media;

namespace QuestionGame
{
///<summary>
/// Interaction logic for WinnerFrame.xaml
///</summary>
publicpartialclassWinnerFrame : Page
    {
privateKinectSensor mySensor;

//winner music
privateSoundPlayer winning_song =
newSoundPlayer(QuestionGame.Properties.Resources.winner_sound);

public WinnerFrame()
        {
```

```
InitializeComponent();
        }

//page loaded event handler
privatevoid Page_Loaded(object sender, RoutedEventArgs e)
        {
if (GlobalSensor.globalSensor != null)
            {
mySensor = GlobalSensor.globalSensor;

//bind sensor to kinect region
                KinectRegion.KinectSensor = mySensor;

//show player total points
showPlayerPoints();

//play winning song
                winning_song.Play();
            }

        }

//StartGameTileButton event handler
privatevoid StartGameTileButton_Click(object sender, RoutedEventArgs e)
        {
//load GameFrame page to restart game
            (Application.Current.MainWindow.FindName("MainFrame")
asFrame).Source = newUri("MainMenu.xaml", UriKind.Relative);

mySensor = null;
            KinectRegion.KinectSensor = null;

//stop music
            winning_song.Stop();
}

privatevoid showPlayerPoints()
        {
            TotalPointsLabel.Content = "ΣΥΝΟΛΙΚΑΚΕΡΔΗ: " +
GameFrame.playerPoints + " $";
        }

privatevoid ExitGameTileButton_Click(object sender, RoutedEventArgs e)
        {
//stop sensor
GlobalSensor.stopKinectSensor(mySensor);

//stop music
            winning_song.Stop();

//close application
Application.Current.MainWindow.Close();
}
}
}
```

7.2 Το παιχνίδι BUGSPLAT

Σκοπός ανάπτυξης του παιχνιδιού

Το BUGSPLAT αναπτύχθηκε με το σκεπτικό οτι θα δώσει την δυνατότητα στους φοιτητές να εμπλουτίσουν τις προγραμματιστικές τους γνώσεις καθώς θα έχουν την ευκαιρία να πειραματισθούν επάνω στην υπάρχουσα εφαρμογή κάνοντας μετατροπές στον κώδικα μαθαίνοντας έτσι όλες τις λειτουργίες του αισθητήρα Kinect και παίζοντας με τις δυνατότητες του SDKνα αναπτύξουν ακόμη πιο πολύ την εφαρμογή ανάλογα με την φαντασία και την όρεξη για μάθηση του κάθε φοιτητή. Ο καθηγητής από την πλευρά του θα μπορούσε να αξιολογεί τους φοιτητές του με μια εργασία εξαμήνου, έχοντας σαν κριτήριο τις αλλαγές που πραγματοποίησαν οι φοιτητές πάνω στην εφαρμογή, για παράδειγμα να πρόσθεταν μια επιπλέον βέργα και στο άλλο χέρι ή να απαιτείται να πετύχει ο χρήστης δύο φορές τα έντομα για να πάρει πόντους ή αφού πετύχει πάνω από 15 έντομα στην σειρά να κερδίζει μια επιπλέον ζωή.

7.2.1 Περιγραφή παιχνιδιού

Το BUGSPLATείναι ένα παιχνίδι στο οποίο χρησιμοποιούνται οι αισθητήρες χρώματος και βάθους καθώς και η δυνατότητα ανίχνευσης της κίνησης ενός ανθρώπινου σκελετού.Το παιχνίδι αρχίζει και μικρά έντομα κατεβαίνουν κατα μήκος της οθόνης, όπου ο χρήστης με την πέτρα η οποία είναι προέκταση του δεξιού χεριού του (το παιχνίδι παίζεται μόνο με το δεξί χέρι) προσπαθεί να χτυπήσει τα έντομα με την άκρη της για να κερδίσει πόντους. Όσο ο χρήστης πετυχαίνει τα έντομα,τόσοη ταχύτητα με την οποία κινούνται αυξάνεται σταδιακά.Αρχικά η ταχύτητα αυξάνεται όταν ο χρήστης πετύχει τουλάχιστον 5 έντομα, στην συνέχεια όταν έχει πετύχει 10 και μετά 20 και κερδίζει πόντους αντίστοιχα μέχρι τα 10 έντομα κερδίζει 10 πόντους,μέχρι τα 20 έντομα κερδίζει 25 πόντους και πάνω απο 20 έντομα κερδίζει 75 πόντους.Επίσης έχουν προστεθεί ήχοι όταν ο χρήστης χτυπάει επιτυχώς ένα έντομο και αντίστοιχαστην περίπτωση που αστοχήσει να το χτυπήσει.Το παιχνίδι τελιώνει όταν οι ζωές του χρήστη μηδενιστούν.

7.2.2 Στιγμιότυπα απο την πορεία του παιχνιδιού

Με την εκίνηση της εφαρμογής ο αισθητήρας ανιχεύει τα χέρια του χρήστη και εμφανίζεται η αρχική σελίδα του παιχνιδιού.

Αρχική σελίδα παιχνιδιού

Αρχική σελίδα παιχνιδιού (με το χέρι του παίκτη να αλληλεπιδρά με την εφαρμογή!)

Όπως βλέπουμε το χεράκι εμφανίζεται αφού ο αισθητήρας ανιχνεύσει το αριστερό ή το δεξί χέρι του χρήστη.

7.2.3 Κώδικας του xamlαρχείουMainMenu για την δημιουργία του κουμπιού,του φόντου και του UserViewer(επάνω στο κέντρο).

```
<Page x:Class="SpiderAttack.MainMenu"
      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
      xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
      xmlns:k="http://schemas.microsoft.com/kinect/2013"
mc:Ignorable="d"
d:DesignHeight="900" d:DesignWidth="1400"
       Title="MainMenu" Loaded="Page_Loaded">
<Page.Background>
<ImageBrush  ImageSource="Images/spiders_background.jpg" />
</Page.Background>

<Grid>
<k:KinectUserViewer HorizontalAlignment="Center" VerticalAlignment="Top"
Height="100"
                            k:KinectRegion.KinectRegion="{Binding
ElementName=KinectRegion}" />

<k:KinectRegion Name="KinectRegion">
<Grid>
<k:KinectTileButton Name="StartGameButton" Content="Εναρξη"
Margin="339,320,349,248" Width="Auto" Background="#FFCD6702" Foreground="Black"
FontSize="72" Click="StartGameButton_Click" Height="Auto" />

</Grid>
</k:KinectRegion>
</Grid>
</Page>
```

7.2.4 Ο C# κώδικας της φόρμας MainMenu

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
```

```
using Microsoft.Kinect;

namespace SpiderAttack
{
///<summary>
/// Interaction logic for MainMenu.xaml
///</summary>
publicpartialclassMainMenu : Page
    {
privateKinectSensor mySensor;

public MainMenu()
        {
InitializeComponent();
        }

//page loaded event handler
privatevoid Page_Loaded(object sender, RoutedEventArgs e)
        {
if (GlobalSensor.globalSensor != null)
            {
mySensor = GlobalSensor.globalSensor;

//bing kinect sensor to KinectRegion
                KinectRegion.KinectSensor = mySensor;
            }
        }

//start game event handler
privatevoid StartGameButton_Click(object sender, RoutedEventArgs e)
        {

//load game frame
            (Application.Current.MainWindow.FindName("MainFrame")
asFrame).Source = newUri("GameFrame.xaml", UriKind.Relative);

mySensor = null;
            KinectRegion.KinectSensor = null;
}
    }
}
```

Εικόνα – Εικόνα που βλέπει ο χρήστης μετην έναρξη της εφαρμογής

Εικόνα – Εικόνα που βλέπει ο χρήστης αφού προχωρίσει λίγο το παιχνίδι

Βλέπουμε σε αυτό το στιγμιότυπο ότι το σκορ έχει αλλάξει κάτω δεξιά και οι ζώες του χρήστη μειώθηκανστην αριστερή πλευρά.

7.2.5 Κώδικας του xamlαρχείου για την δημιουργία του φόντου,την τοποθέτηση της εικόνας της αράχνης και της πέτρας και των labelsπου εμφανίζει τα συνολικά κέρδη και τις ζωές(προσπάθειες) που απομένουν στον χρήστη.

```
<Page x:Class="SpiderAttack.GameFrame"
      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
      xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
      xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
      xmlns:k="http://schemas.microsoft.com/kinect/2013"
mc:Ignorable="d"
d:DesignHeight="900" d:DesignWidth="1400"
       Title="GameFrame" Loaded="Page_Loaded">

<Page.Background>
<ImageBrush ImageSource="Images/game_frame_background.jpg" />
</Page.Background>

<Grid>
<Canvas Name="GameFrameCanvas" Width="640" Height="480" Canvas.Left="100"
Canvas.Top="100">
<Image Name="GameVideoImage" Width="640" Height="480" Stretch="Fill"/>
<Canvas Name="MalletRegion" Width="640" Height="480" />
<Canvas Name="SpiderWebRegion" Width="640" Height="480" />
<Image Name="SpiderImage" Source="Images/spider.png" Width="48" Height="48"/>
<Image Name="HandRockImage" Source="Images/rock.png" Width="87" Height="87" />

<Label Name="LivesLabel" Content="ΖΩΕΣ: 3" Background="#FFF3AA0D" FontSize="36"
                    FontWeight="Bold" Canvas.Top="485" Width="164"/>
<Label Name="PointsLabel" Content="ΒΑΘΜΟΙ: 0" Background="#FFF3AA0D"
FontSize="36"
                    FontWeight="Bold" Canvas.Left="295" Canvas.Top="485"
Width="345" />
</Canvas>
</Grid>
</Page>
```

7.2.6 OC# κώδικαςτηςφόρμαςGameFrame

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
```

```
using System.Windows.Shapes;
using Microsoft.Kinect;
using Microsoft.Kinect.Toolkit.BackgroundRemoval;
using System.Media;
using System.Threading;

namespace SpiderAttack
{
///<summary>
/// Interaction logic for GameFrame.xaml
///</summary>
publicpartialclassGameFrame : Page
    {
privateKinectSensor mySensor;

Skeleton[] skeletons = null;

publicstaticint playerScore = 0;
int spiderHits = 0;
int points = 10;
int playerLives;

double spiderImageY = 0;
double spiderImageX = 0;
double spiderSpeed = 2;
Random rand = newRandom(1);

//spider sounds
privateSoundPlayer escapeSpiderSound =
newSoundPlayer(SpiderAttack.Properties.Resources.laugh);
privateSoundPlayer hitSpiderSound =
newSoundPlayer(SpiderAttack.Properties.Resources.ouch);

Brush skeletonBrush = newSolidColorBrush(Colors.Red);
Brush malletHeadBrush = newSolidColorBrush(Colors.Black);
Brush webColorLine = newSolidColorBrush(Colors.White);

float malletHandleLength = 100;
float malletHeadLength = 50;

        System.Windows.Vector malletPosition;
float malletHitRadius = 45;
bool malletValid = false;

byte[] colorData = null;
WriteableBitmap imageBitmap;

public GameFrame()
        {
InitializeComponent();
        }

//page loaded event handler
privatevoid Page_Loaded(object sender, RoutedEventArgs e)
        {
initializeGame();
        }

//initialize game
privatevoid initializeGame()
        {
if (GlobalSensor.globalSensor != null)
```

```
            {
mySensor = GlobalSensor.globalSensor;

//check streams
if (!mySensor.ColorStream.IsEnabled)
mySensor.ColorStream.Enable();

if (!mySensor.SkeletonStream.IsEnabled)
mySensor.SkeletonStream.Enable();

if (!mySensor.DepthStream.IsEnabled)
mySensor.DepthStream.Enable();

//add event handlers to streams
                mySensor.AllFramesReady += mySensor_AllFramesReady;

//start game
startGameThread();

playerLives = 3;
            }
        }

//stop sensor
privatevoid stopSensor(KinectSensor sensor)
        {
if (sensor != null)
            {
GlobalSensor.stopKinectSensor(sensor);
            }
        }

privatevoid startGameThread()
        {
Thread game = newThread(gameLoop);
            game.IsBackground = true;
game.Start();
        }

privatevoid gameLoop()
        {
while (true)
            {
Dispatcher.Invoke(newAction(() => updateGame()));
Thread.Sleep(34);
            }
        }

//update bug on screen
privatevoid updateGame()
        {
//set bug to image
Canvas.SetTop(SpiderImage, spiderImageY);
spiderImageY = spiderImageY + spiderSpeed;

//draw spider web
drawSpiderWeb((int)(spiderImageX + (SpiderImage.Width / 2)), 0,
(int)(spiderImageX + (SpiderImage.Width / 2)), (int)spiderImageY);

//check for collisions
if (malletValid)
            {
```

```
double bugCenterX = spiderImageX + (SpiderImage.Width / 2);
double bugCenterY = spiderImageY + (SpiderImage.Height / 2);

                System.Windows.Vector hitVector =
newSystem.Windows.Vector(malletPosition.X - bugCenterX, malletPosition.Y -
bugCenterY);

if (hitVector.Length < malletHitRadius)
                {
//increase spider hits
spiderHits++;
//increase score
playerScore+=points;
//show player score
                    PointsLabel.Content = "ΒΑΘΜΟΙ: " + playerScore.ToString();

spiderImageX = rand.Next(0, (int)GameFrameCanvas.Width -
(int)SpiderImage.Width);
Canvas.SetLeft(SpiderImage, spiderImageX);
spiderImageY = -SpiderImage.Height;
//play spider hit sound
hitSpiderSound.Play();
//clear SpiderWebRegion
SpiderWebRegion.Children.Clear();

if (spiderHits == 5)
                    {
spiderSpeed *= 2;
                    }
elseif (spiderHits == 10)
                    {
spiderSpeed *= 2;
points = 25;
// GameBackgroundRegion.Source = image;
                    }
elseif (spiderHits >= 20)
                    {
spiderSpeed = 12;
points = 75;
                    }
                }
            }

//if spider escape
if (spiderImageY > GameFrameCanvas.Height)
            {
//new spider x-position
spiderImageX = rand.Next(0, (int)GameFrameCanvas.Width -
(int)SpiderImage.Width);
//add spider on canvas
Canvas.SetLeft(SpiderImage, spiderImageX);
spiderImageY = -SpiderImage.Height;

//play spider escape sound
escapeSpiderSound.Play();
//clear SpiderWebRegion
SpiderWebRegion.Children.Clear();
//player loose 1 life
playerLives--;

if (playerLives >= 0)
                    LivesLabel.Content = "ΖΩΕΣ: " + playerLives.ToString();
```

```
 else
                {
                    mySensor.AllFramesReady -= mySensor_AllFramesReady;

 //load exit menu frame
                    (Application.Current.MainWindow.FindName("MainFrame")
 asFrame).Source = newUri("ExitMenuFrame.xaml", UriKind.Relative);

 mySensor = null;
                }
            }

        }

 //add line on canvas
 privatevoid addLine(Joint joint1, Joint joint2)
        {
 Line boneLine = newLine();
            boneLine.Stroke = skeletonBrush;
            boneLine.StrokeThickness = 5;

 ColorImagePoint joint1Point =
 mySensor.CoordinateMapper.MapSkeletonPointToColorPoint(joint1.Position,
 ColorImageFormat.RgbResolution640x480Fps30);
            boneLine.X1 = joint1Point.X;
            boneLine.Y1 = joint1Point.Y;

 ColorImagePoint joint2Point =
 mySensor.CoordinateMapper.MapSkeletonPointToColorPoint(joint2.Position,
 ColorImageFormat.RgbResolution640x480Fps30);
            boneLine.X2 = joint2Point.X;
            boneLine.Y2 = joint2Point.Y;

 MalletRegion.Children.Add(boneLine);
        }

 //draw spider web
 privatevoid drawSpiderWeb(int startWebLineX, int startWebLineY, int
 stopWebLineX, int stopWebLineY)
        {
 Line webLine = newLine();
            webLine.Stroke = webColorLine;
            webLine.StrokeThickness = 1;

 //starting point of line
            webLine.X1 = startWebLineX;
            webLine.Y1 = startWebLineY;

 //end point of line
            webLine.X2 = stopWebLineX;
            webLine.Y2 = stopWebLineY;

 SpiderWebRegion.Children.Add(webLine);
        }

 //draw mallet on right hand
 privatevoid drawHandWeapon(Joint handJoint)
        {
 if (handJoint.TrackingState != JointTrackingState.Tracked)
 return;
```

```
ColorImagePoint handPoint =
mySensor.CoordinateMapper.MapSkeletonPointToColorPoint(handJoint.Position,
ColorImageFormat.RgbResolution640x480Fps30);

//lines and brush for mallet head
Line headLine = newLine();
            headLine.Stroke = malletHeadBrush;
            headLine.StrokeThickness = 50;

            headLine.X1 = handPoint.X;
            headLine.Y1 = handPoint.Y;

            headLine.X2 = handPoint.X + 30;
            headLine.Y2 = handPoint.Y + 30;

MalletRegion.Children.Add(headLine);

//add rock image on player's right hand
Canvas.SetLeft(HandRockImage, handPoint.X-30);
Canvas.SetTop(HandRockImage, handPoint.Y-30);

malletPosition = new System.Windows.Vector(handPoint.X, handPoint.Y);
malletValid = true;
        }

//streams event handlers
void mySensor_AllFramesReady(object sender, AllFramesReadyEventArgs e)
        {
//clear Mallet Region
MalletRegion.Children.Clear();

//color frame stream event handler
using (ColorImageFrame colorFrame = e.OpenColorImageFrame())
            {

if (colorFrame == null)
return;

if (colorData == null)
colorData = newbyte[colorFrame.PixelDataLength];

colorFrame.CopyPixelDataTo(colorData);

if (imageBitmap == null)
                {
this.imageBitmap = newWriteableBitmap(
                                                colorFrame.Width,
                                                colorFrame.Height,
                                                96,
                                                96,

PixelFormats.Bgr32,
null);
}

this.imageBitmap.WritePixels(
newInt32Rect(0, 0, colorFrame.Width, colorFrame.Height),
colorData,
                                        colorFrame.Width *

colorFrame.BytesPerPixel,
                                        0);

                GameVideoImage.Source = imageBitmap;
```

```
            }

//skeleton frame stream event handler
using (SkeletonFrame skeletonFrame = e.OpenSkeletonFrame())
            {
if (skeletonFrame == null)
return;

if (skeletonFrame != null)
                {
skeletons = newSkeleton[skeletonFrame.SkeletonArrayLength];
skeletonFrame.CopySkeletonDataTo(skeletons);
                }

if (skeletons[0].TrackingState == SkeletonTrackingState.Tracked)
                {
//draw weapon on player's right hand
drawHandWeapon(skeletons[0].Joints[JointType.WristRight]);
}

            }

        }

    }
}
```

Εικόνα – Εικόνα που βλέπει ο χρήστης με το πέρας του παιχνιδιού

7.2.7 Κώδικας του xamlαρχείου για την δημιουργία του κουμπιού και του φόντου,του UserViewer(επάνωστο κέντρο), τουLabelγια την ενημέρωση του χρήστη για το σκόρ

```
<Pagex:Class="SpiderAttack.ExitMenuFrame"
```

```
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
      xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
      xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
      xmlns:k="http://schemas.microsoft.com/kinect/2013"
mc:Ignorable="d"
d:DesignHeight="900" d:DesignWidth="1400"
       Title="ExitMenuFrame" Loaded="Page_Loaded">

<Page.Background>
<ImageBrush ImageSource="Images/exit_frame_background.jpg" />
</Page.Background>

<Grid>
<k:KinectUserViewer HorizontalAlignment="Center" VerticalAlignment="Top"
Height="100"
                           k:KinectRegion.KinectRegion="{Binding
ElementName=KinectRegion}" />

<k:KinectRegion Name="KinectRegion">
<Canvas>
<Label  Name="PlayerScoreLabel" Content="ΣυνολικόΣκόρ: " Background="#FFE8AC43"
Foreground="Black" FontWeight="Bold" FontSize="72" Canvas.Left="250"
Canvas.Top="217" Width="872" />

<k:KinectTileButton Name="ExitGameButton" Content="Έξοδος" Canvas.Left="472"
Canvas.Top="346" Width="402"
                                Foreground="Black" FontWeight="Bold"
FontSize="72" Background="#FFE8AC43" Click="ExitGameButton_Click"/>
</Canvas>
</k:KinectRegion>
</Grid>
</Page>
```

7.2.8 Ο C# κώδικας της φόρμας ExitMenuFrame

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using Microsoft.Kinect;

namespace SpiderAttack
{
///<summary>
/// Interaction logic for ExitMenuFrame.xaml
///</summary>
publicpartialclassExitMenuFrame : Page
    {
```

```
privateKinectSensor mySensor;

public ExitMenuFrame()
        {
InitializeComponent();
        }

//initialize sensor
privatevoid initKinectSensor()
        {
if (GlobalSensor.globalSensor != null)
            {
mySensor = GlobalSensor.globalSensor;

//check streams
if (!mySensor.ColorStream.IsEnabled)
mySensor.ColorStream.Enable();

if (!mySensor.SkeletonStream.IsEnabled)
mySensor.SkeletonStream.Enable();

if (!mySensor.DepthStream.IsEnabled)
mySensor.DepthStream.Enable();

//bind sensor to KinectRegion
                KinectRegion.KinectSensor = mySensor;

                PlayerScoreLabel.Content="ΣυνολικόΣκόρ:
"+GameFrame.playerScore.ToString();

            }
        }

//page loaded event handler
privatevoid Page_Loaded(object sender, RoutedEventArgs e)
        {
initKinectSensor();
        }

//exit game button event handler
privatevoid ExitGameButton_Click(object sender, RoutedEventArgs e)
        {
//stop sensor
GlobalSensor.stopKinectSensor(mySensor);

//close application
Application.Current.MainWindow.Close();
}


    }
}
```

8 Περίληψη

Στα προηγούμενα κεφάλαια έγινε εισαγωγή στον αισθητηρα Kinectκαι στα τεχνικά χαρακτηριστικά του, όπως το ότι αποτελείται από δυο αισθητήρες, έναν αισθητήρα βάθους

ένα πομπό υπερύθρων ακτίνων, καθώς και μια κάμερα χρώματος και 4 μικρόφωνα,στη συνέχεια έγινε ανάλυση αυτών.Ακολούθησε μια αναφορά στο SDKτου Kinect κυρίως τα τεχνικά χαρακτηριστικά του όπως οι αναλυτικές τεχνολογικές λεπτομέριες των αισθητήρων και στις απαιτήσεις υλικού και λογισμικού.Η παρουσίαση επίσης περιελάμβανε πληροφορίες για το IDEπου χρησιμοποιήθηκε για την ανάπτυξη των εφαρμογών καθώς κι επεξήγηση των εργαλείων του SDK που μας βοήθησαν στο να υλοποιήσουμε τις εφαρμογές μας.Αναφέρθηκαν παραδείγματα χρήσης του αισθητήρα που βρίσκουν εφαρμογή στον πραγματικό κόσμο κι εικόνες αυτών. Στο τέλος της πτυχιακής εργασίας έγινε ανάλυση των εφαρμογών που αναπτύχθηκαν απο εμάς, κι ο σκοπός της κάθε εφαρμογής, μαζί με εικόνες που δείχνουν βήμα προς βήμα την λειτουργία τους.

9 Συμπεράσματα

Σαν συμπέρασμα σημειώνεται ότι η χρήση του αισθητήρα Kinectγια ανάπτυξη εκπαιδευτικών εφαρμογών μπορεί να περιοριστεί μόνο απο την φαντασία των ανθρώπων καθώς η τεχνολογία στο συγκεκριμένο κομμάτι προχωράει με ραγδαίους ρυθμούς εξέλιξης. Ήδη η εταιρία κατασκευής του Kinectέχει προωθήσει στην αγορά τον ανανεωμένο αισθητήρα KinectOne,με δυνατότητες πολύ πιο ανεπτυγμένες σε σχέση με τον προκάτοχο του. Για παράδειγμα αναγνωρίζει μέχρι και 30 αρθρώσεις του ανθρώπινου σώματος καθώς παρέχει κι αναγνώριση πάχους μυών και καρδιακών παλμών.Όσον αφορά στην πτυχιακή εργασία μας και τις εφαρμογές που αναπτύξαμε δημιουργήθηκαν με σκοπό να βοηθήσουν και τους φοιτητές που παρακολουθούν εργαστήρια προγραμματισμού αλλά και τους εκπαιδευτικούς που διδάσκουν προγραμματισμό.Οι φοιτητές θα αποκτήσουν καινούριες γνώσεις ή θα εμπλουτίσουν τις ήδη υπάρχουσες με την καθοδήγηση των εκπαιδευτικών τους. Οι εκπαιδευτικοί, δε θα παρέχουν απλά γνώση επάνω στο αντικείμενο που διδάσκουν τους φοιτητές, αλλά θα τους εισάγουν και στο εμπορικό κομμάτι, προετοιμάζοντας τους κατά μια έννοια για το πως είναι διαμορφωμένη η αγορά εργασίας και τι ζητάει. Ταυτόχρονα ενημερώνονται και μαθαίνουν τα εμπορικά εργαλεία και εφαρμογές που υπάρχουν και πολύ πιθανόν να χρησιμοποιήσουν αφού αποφοιτήσουν.Στο μέλλον περιμένουμε καινούριες εκδόσεις του KinectSDKμε καινούριες βιβλιοθήκες ώστε να μας δοθεί η δυνατότητα επέκτασης των εφαρμογών μας,για παράδειγμα να αναγνωρίζονται πάνω από 2 ολόκληροι σκελετοί και να μπορούν να παίζουν τα παιχνίδια πολλοί χρήστες ταυτόχρονα όπως επίσης και καινοτόμα εργαλεία όσον αφορά το GUIτης εφαρμογής, κάνοντας πιο ευχάριστη και διασκεδαστική την αλληλεπίδραση με τον χρήστη.

10 Βιβλιογραφία

Ξένη βιβλιογραφία

- Andrew Troelsen . Pro C# 2010 and the .NET 4 Platform fifth edition

- Andrew Stellman & Jennifer Grinne. Head first C# second.second edition 2010
- John Sharp. Microsoft Visoual C# step by step 2012
- Andrew Troelsen.Pro C# 5.0 and the .NET 4.5 Framework fifth edition
- Rob Miles.C# programming.Edidtion 4.0 August 2012 University of Hull
- Sam A. Abolrous.Learn C# .2008
- Jarrett Webb ,James Ashley.Beginning Kinect Programming with the Microsoft Kinect SDK.2012
- Abhijit Jana. Kinect for Windows SDK Programming Guide.2012
- Rob Miles.Start here! Learn the Kinect API.2012
- David Catuhe .Microsoft Programming with the Kinect for Windows SDK.2012

Αναφορές

- http://books.google.gr/books?id=MupB_VAmtdEC&pg=PA96&lpg=PA96&dq=kinect+sdk+CopySkeletonDataTo&source=bl&ots=Egjehwcytb&sig=bqZnw0iZLHeCGNnFNEll55rfmUI&hl=el&sa=X&ei=8iJnT5LhBYGbOpbM1IAI&ved=0CCAQ6AEwAA#v=onepage&q=kinect%20sdk%20CopySkeletonDataTo&f=false
- http://www.i-programmer.info/programming/hardware/3503-getting-started-with-microsoft-kinect-sdk-skeletons.html?start=1
- http://stackoverflow.com/questions/9249133/what-causes-kinect-error-warning-a-skeleton-frame-was-not-disposed-and-how-do
- http://channel9.msdn.com/coding4fun/kinect/Of-course-our-first-Kinect-for-Windows-SDK-Project-has-to-involve-a-Light-Saber
- http://www.i-programmer.info/programming/hardware/2623-getting-started-with-microsoft-kinect-sdk.html?start=2
- http://blogs.microsoft.co.il/blogs/shair/archive/2011/07/03/kinect-calculator-adjust-skeleton-movements-to-mouse.aspx
- http://www.i-programmer.info/ebooks/practical-windows-kinect-in-c.html
- http://msdn.microsoft.com/en-us/library/jj131023.aspx -> Kinect SDK Architecture
- http://msdn.microsoft.com/en-us/library/jj663790.aspx -> Accelerometer
- http://en.wikipedia.org/wiki/YUV -> YUV
- http://msdn.microsoft.com/de-de/library/windows/desktop/dd206750(v=vs.85).aspx - YUV
- http://elbruno.com/2013/09/20/kinect-howto-remove-the-background-while-preserving-a-body-with-the-new-kinectsdkenter-a-post-title/ -> backgroundremoval

- http://www.tuicool.com/articles/3QBbMv ->background removal
- http://msdn.microsoft.com/en-us/library/ms752059%28v=vs.110%29.aspx ->xaml
- http://en.wikipedia.org/wiki/Extensible_Application_Markup_Language -> xaml
- http://studentguru.gr/w/tutorials/164.aspx -> xaml
- http://en.wikipedia.org/wiki/C_Sharp_%28programming_language%29 ->c#