ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΣΥΣΤΗΜΑΤΩΝ ΜΕΤΑΔΟΣΗΣ ΠΛΗΡΟΦΟΡΙΑΣ
ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ ΥΛΙΚΩΝ

Εφαρμογές εικονικής και επαυξημένης πραγματικότητας στην καταπολέμηση των φοβιών

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Ιάκωβος Γ. Κρητικός

Επιβλέπων: Δημήτριος Κουτσούρης
Καθηγητής Ε.Μ.Π.

Αθήνα, Μάρτιος 2018

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΣΥΣΤΗΜΑΤΩΝ ΜΕΤΑΔΟΣΗΣ ΠΛΗΡΟΦΟΡΙΑΣ
ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ ΥΛΙΚΩΝ

Εφαρμογές εικονικής και επαυξημένης πραγματικότητας στην καταπολέμηση των φοβιών

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Ιάκωβος Γ. Κρητικός

Επιβλέπων: Δημήτριος Κουτσούρης
Καθηγητής Ε.Μ.Π.

.....	.....	.....
Δ. Κουτσούρης	Π. Τσανάκας	Γ. Ματσόπουλος
Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Αν. Καθηγήτρια Ε.Μ.Π.

Αθήνα, Μάρτιος 2018

.....
Ιάκωβος Γ. Κρητικός

Περίληψη

Στην παρούσα εργασία παρουσιάζεται ένα σύστημα, που το ονομάζουμε Action Therapy (AT), το οποίο στοχεύει στην αντιμετώπιση των φοβιών και πιο συγκεκριμένα στην αντιμετώπιση ειδικών φοβιών. Στηρίζεται στη γνωσιακή θεραπεία και τα εργαλεία που χρησιμοποιούνται είναι: γυαλιά εικονικής πραγματικότητας, κάμερα αναγνώρισης κινήσεων και αισθητήρες βιοσημάτων. Στόχος του (AT) είναι ο ασθενής τη γνώση και την εμπειρία, που αποκτά κατά τη διάρκεια της θεραπείας, να τις διοχετεύσει σε δεξιότητες, με τις οποίες ευκολότερα και αβίαστα θα αντιμετωπίσει τη φοβία του στην καθημερινότητα. Μέχρι σήμερα, με τις εφαρμογές της Γνωσιακής Συμπεριφοριστικής Θεραπείας (Cognitive Behavioral Therapy - CBT), ο ασθενής αναμένεται να ξεπεράσει τον φόβο μέσω συζήτησης, με αποτέλεσμα να αποκτά πλήρη κατανόηση της φοβίας του. Στις εφαρμογές της Θεραπείας Έκθεσης (Exposure Therapy - ET), ο ασθενής μαθαίνει πώς να μένει κοντά, να μπορεί να αγγίξει ή να μπορεί να κοιτάξει τη φοβία του μέσω οπτικής ή σωματικής διέγερσης, χωρίς να αντιμετωπίσει κρίση πανικού, με αποτέλεσμα να εξοικειωθεί με τη φοβία του. Παρ' όλα αυτά, καμία εφαρμογή δεν έχει επικεντρωθεί στην εκπαίδευση του ασθενούς, στο «πώς» να ενεργήσει σε μια επίφοβη κατάσταση. Παραδείγματα τέτοιων προπονήσεων είναι: «πώς να πιάσει ένα φοβικό αντικείμενο», «πώς να αποφύγει ένα φοβικό αντικείμενο, όταν κατευθύνεται προς τα πάνω του», κλπ, τα οποία βεβαίως είναι συνήθη σενάρια, που αντιμετωπίζονται από τα μη φοβικά άτομα χωρίς κρίση πανικού. Η (AT), μέσω των γυαλιών εικονικής πραγματικότητας, προσομοιώνει διάφορες επίφοβες καταστάσεις. Παράλληλα, μέσω της κάμερας, το σύστημα παρακολουθεί τις δράσεις του ασθενούς, ελέγχοντας κατά πόσο αυτές εκτελούνται ορθά. Τέλος, μέσω των βιοσημάτων, προσαρμόζει το εικονικό περιβάλλον αναλόγως. Ουσιαστικά, η (AT) είναι μια συστηματική μέθοδος, η οποία στοχεύει στην απόκτηση δεξιοτήτων και κυρίως αυτοπεποίθησης, με τα οποία θα μπορεί ο ασθενής πλέον να δρα πάνω στη φοβία του αυτοβούλως, αβίαστα, όπως και όποτε θέλει, έχοντας τη αίσθηση ότι πλέον μπορεί να τα καταφέρει.

Λέξεις Κλειδιά

Φόβος, Φοβίες, CBT, ET, Εικονική Πραγματικότητα, Κάμερα Βάθους, Βοσκήματα, Εκπαίδευση, Θεραπεία, Λογισμικό, Μάθημα.

Abstract

In this paper we present a system named (AT), which aims at the treatment of phobias and more specifically the treatment of special phobias. It is based on cognitive therapy and the tools it uses are: virtual reality glasses, motion recognition camera and bio-sensors. The aim of the (AT) is the knowledge and experience that the patient acquires during therapy to channel them into skills that will more easily and effortlessly address his phobia in everyday life. To date, with the applications of Cognitive Behavioral Therapy (CBT), the patient is expected to overcome fear through discussion, thus gaining a full understanding of his phobia. In Exposure Therapy (ET) applications, the patient learns how to stay close, touch or be able to look at his phobia through visual or physical excitement without experiencing a panic attack, thereby becoming familiar with his phobia. Nevertheless, no application has focused on patient education, "how" to act in a dreadful situation. Examples of such training are: "how to catch a phobic object," "how to avoid a phobic object when it is directed upwards," etc., which are, of course, usual scenarios that non-fearful individuals face without panic attack. The (AT), through virtual reality glasses, simulates various dreadful situations. At the same time, through the camera, the system monitors the patient's actions, checking whether they are performed correctly. Finally, through biosignals, it accordingly adapts the virtual environment. Essentially, (AT) is a systematic method that aims for the patient to acquire skills and above all self-confidence, with which he/she can now act on his phobia voluntarily, effortlessly, whenever he/she wants, having the feeling that he can now to make it happen.

Keywords

Fear, Phobias, CBT, ET, Virtual Reality, Depth Camera, Beats, Education, Therapy, Software, Lesson.

ΠΕΡΙΕΧΟΜΕΝΑ

1 Εισαγωγή 10
- **1.1 Σκοπός 10**
- **1.2 Εμβέλεια 10**
- **1.3 Τρόπος προσέγγισης 10**

2 Επισκόπηση των εννοιών Φόβου, Φοβίας, Ειδικών Φοβιών και τις Θεραπείες Αυτών. 11
- **2.1 Εισαγωγή: 11**
- **2.2 Φόβος: 11**
- **2.3 Φοβία: 11**
- **2.4 Από τη φοβία στο φόβο: 11**
- **2.5 Ειδικές Φοβίες: 12**
- **2.6 Μέθοδοι θεραπείας: 12**
- **2.7 Θεραπεία γνωστικής συμπεριφοράς (CBT): 12**
- **2.8 Εφαρμογές Θεραπείας Γνωσιακής Συμπεριφοράς (CBT): 13**
- **2.9 Επισκόπηση των Εφαρμογών CBT: 16**
- **2.10 Αποτελέσματα των Εφαρμογών CBT: 16**

3 Ιδέα του Συστήματος (AT) 17
- **3.1 Εισαγωγή 17**
- **3.2 Μια νέα πρόταση: 17**
- **3.3 Νέο βήμα στη μέχρι τώρα θεραπεία: 17**
- **3.4 Γενική ιδέα του (AT) συστήματος: 18**
- **3.5 Υλοποιήσεις παρεμφερών θεραπειών και οι περιορισμοί αυτών: 19**
- **3.6 Υλοποίηση του (AT) και εργαλεία: 19**
- **3.7 Ρύθμιση χώρου: 19**
- **3.8 Αυτοεκτίμηση: 20**

4 Βασικό σύστημα (AT) 22
- **4.1 Εισαγωγή: 22**
- **4.2 Μοντέλο - Βασικό Δέντρο Καταστάσεων και Λύσεων: 22**
- **4.3 Μοντέλο - Βασικό Δέντρο Καταστάσεων, Λύσεων και Δράσεων: 23**
- **4.4 Εφαρμογή του Βασικού Συστήματος (AT): 23**
- **4.5 Παράδειγμα του (AT): 24**

5 Σύνθετο σύστημα (AT) 26
- **5.1 Εισαγωγή: 26**
- **5.2 Μοντέλο - Σύνθετο Δέντρων Καταστάσεων, Λύσεων και Δράσεων: 26**
- **5.3 Περιορισμοί της Βασικού Συστήματος (AT): 27**
- **5.4 Επίλυση περιορισμών: 28**
- **5.5 Σύνθετο Σύστημα (AT): 30**
- **5.6 Εφαρμογή της προηγμένης θεραπείας δράσης: 30**

6 Υλοποίηση του Συστήματος (AT) 33
- **6.1 Εισαγωγή: 33**
- **6.2 Υλικό: 33**
- **6.3 Λογισμικό: 37**
- **6.4 Αρχιτεκτονική Συστήματος: 44**
- **6.5 Σενάριο Υλοποίησης: 45**

7 Συμπεράσματα 57
- **7.1 Σενάριο Υλοποίησης: 57**

8 Κώδικας 60

8.1 Εισαγωγή: 60
8.1.1 Orbbec Persee: 60
8.1.2 Samsung Galaxy S7: 61
8.2 BLE ARM (Bluetooth in Camera): 62
8.2.1 Bluetooth.cc 62
8.2.2 lescan_simple.cc 67
8.2.3 lescan.cc 68
8.2.4 read_device_name.cc 70
8.3 Android Bluetooth 72
8.3.1 BleCommunication.java 72
8.3.2 BluetoothCommunication.java 82
8.3.3 BluetoothService.java 88
8.3.4 Constants.java 98
8.3.5 HalfCutter.java 99
8.3.6 TimeProfile.java 102
8.4 Image Recognition in Camera 105
8.4.1 Main.cpp 105
8.4.2 NuitrackGLSample.cpp 108
8.4.3 NuitrackGLSample.h 118
8.5 Android VR 120
8.5.1 BluetoothAndroid.cs 120
8.5.2 grab_release.cs 121
8.5.3 MirrorReflection.cs 123
8.5.4 SkeletonHead.cs 127
9 Αναφορές 132

1 Εισαγωγή

1.1 Σκοπός

Σκοπός της παρούσας εργασίας είναι η μελέτη των θεραπειών που υφίστανται μέχρι σήμερα για την καταπολέμηση των φοβιών, η παρατήρηση των κενών που υπάρχουν στις μέχρι τώρα εφαρμογές των θεραπειών και η κατασκευή ενός εργαλείου - συστήματος – που καλύπτει αυτά τα κενά και διευκολύνει στη γρηγορότερη, ασφαλέστερη και οικονομικότερη θεραπεία του ασθενούς.

1.2 Εμβέλεια

Η φοβία, ως νόσος, ανήκει στον ευρύτερο κλάδο των ψυχικών ασθενειών. Όλες οι ψυχικές ασθένειες επηρεάζουν τα ίδια μέρη του εγκεφάλου πχ το Λιμβικό Σύστημα και το Νεοφλοιό,, έχει παρατηρηθεί ότι σε όλες οι θεραπείες που εφαρμόζονται για την καταπολέμηση των φοβιών είναι παρεμφερείς. Κατά συνέπεια αν αυτό το σύστημα που προτείνουμε σε αυτή τη διπλωματική εργασία μπορεί να συμβάλει στη θεραπεία των φοβιών τότε θα μπορεί να συμβάλει και στις υπόλοιπες ψυχικές ασθένειες

1.3 Τρόπος προσέγγισης

Στην παρούσα διπλωματική εργασία, ακόμα και αν παρατίθεται ένα ολοκληρωμένο τεχνολογικό σύστημα, δίνεται ιδιαίτερη έμφαση στο θεωρητικό μέρος. Υπάρχει πλήρη ανάλυση των θεραπειών που εφαρμόζονται μέχρι σήμερα, των κενών που έχουν αυτές και στο γιατί το προτεινόμενο σύστημα μπορεί να τις υποκαταστήσει και να παρέχει ένα κατάλληλο τρόπο καταπολέμησης των φοβιών. Κατά συνέπεια στα ακόλουθα κεφάλαια παρατίθεται μια πλήρης ανάλυση της ιδέας πίσω από το σύστημα.

2 Επισκόπηση των εννοιών Φόβου, Φοβίας, Ειδικών Φοβιών και τις Θεραπείες Αυτών.

2.1 Εισαγωγή:

Σε αυτό το κεφάλαιο αναφέρονται οι βασικοί ορισμοί φόβων, φοβίας, οι κατηγορίες φοβιών και οι διάφορες θεραπείες που εφαρμόζονται μέχρι σήμερα. Στη συνέχεια εστιάζουμε σε μια συγκεκριμένη κατηγορία των φοβιών, στις «ειδικές φοβίες». Ακολούθως εστιάζουμε στη γνωσιακή θεραπεία και στις διάφορες μορφές της για την αντιμετώπιση των ειδικών φοβιών. Τέλος αναφέρουμε διάφορες θεραπείες που έχουν εφαρμοστεί και παρουσιάζουμε τα αποτελέσματα τους.

2.2 Φόβος:

Ο φόβος είναι η βιολογική αντίδραση του σώματος να υπερασπιστεί τον εαυτό του ενάντια σε οτιδήποτε μπορεί να του προκαλέσει κακό. Είναι ένα από τα πιο βασικά ανθρώπινα συναισθήματα το οποίο όταν διεγείρεται η κύρια δραστηριότητα, του εγκεφάλου παρατηρείται κυρίως στο λεμφικό σύστημα και δευτερευόντως σε άλλα μέρη του εγκεφάλου όπως η φτέρνα, ο μετωπικός λοβός και άλλοι. Όταν ο φόβος ενεργοποιείται, το σώμα «ξυπνά», ο καρδιακός παλμός αυξάνεται δραστικά και οι μύες διαστέλλονται και συστέλλονται, για να μας προετοιμάσουν - φυσικά και διανοητικά - για αυτό που φαίνεται να είναι μια επιβλαβήςκατάσταση. Ο φόβος είναι ένα χαρακτηριστικό της ανθρώπινης ύπαρξης που συνδέεται άμεσα με την εξέλιξη του ανθρώπου, αφού το κίνητρο για επιβίωση είναι αυτό που διατηρεί την ανθρωπότητα εξελισσόμενη. Ωστόσο, ο χρόνιος και έντονος φόβος μπορεί να προκαλέσει ανεπιθύμητες ενέργειες στο σώμα, όπως αδράνεια ή υπερβολική πίεση [1, σελ. 1-2] [2, σελ. 189-190] [3, σελ. 278-279].

2.3 Φοβία:

Είναι η έντονη, συνεχής και - τις περισσότερες φορές - παράλογη παρουσία του φόβου. Το άτομο αισθάνεται μια πολύ μεγαλύτερη απειλή από αυτή που υπάρχει στην πραγματικότητα. Τα δύο κύρια αποτελέσματα της φοβίας είναι το υπερβολικό άγχος ή η «κατάψυξη». Έτσι, λόγω της παρουσίας άγχους ή σοκ, ο άνθρωπος δεν είναι σε θέση να επιδείξει αυτοέλεγχο, με αποτέλεσμα να υπάρχει πιθανότητα να μην μπορέσει διαχειριστεί την επιβλαβή κατάσταση [1, σελ. 3-4] [2, σελ. 189-190].

2.4 Από τη φοβία στο φόβο:

Με τον ορισμό του φόβου, μπορούμε να συμπεράνουμε ότι ο φόβος δεν είναι κάτι αρνητικό. Αντίθετα, όταν υπάρχει φόβος, το σύστημά μας ενεργοποιείται προκειμένου να είναι έτοιμο να αντιδράσει γρήγορα σε οποιεσδήποτε απειλητικές καταστάσεις. Επομένως, ο στόχος μας δεν είναι να εξαλείψουμε τον φόβο, αλλά να εκμεταλλευτούμε τα θετικά χαρακτηριστικά του. Αντίθετα, η φοβία είναι εξ' ορισμού αρνητική, αφού, όταν το άτομο εκτίθεται στη φοβία, ο εγκέφαλος δεν αντιδρά όπως πρέπει να επιτρέψει στο σώμα να επιβιώσει από την απειλητική

κατάσταση, δηλαδή, από τη μία πλευρά, όταν παρουσιάζεται φόβος, το αμυντικό μας σύστημα ενεργοποιείται αλλά, από την άλλη, το αίσθημα του άγχους ή του σοκ δεν επιτρέπει στο άτομο να αντιμετωπίσει την απειλητική κατάσταση και πρέπει συνεπώς να εξαλειφθεί.

2.5 Ειδικές Φοβίες:

Οι φοβίες έχουν ταξινομηθεί σε τρεις κυρίαρχες κατηγορίες, Ειδικές Φοβίες, Αγοραφοβία, Κοινωνικές Φοβίες [2, σελ. 189-233]. Σε αυτή την εργασία θα επικεντρωθούμε στις Ειδικές Φοβίες. Οι ασθενείς που έχουν κάποια τέτοια φοβία φοβούνται αντικείμενα ή καταστάσεις που το ανθρώπινο μυαλό θεωρεί δυνητικά επιβλαβείς, ανεξάρτητα από το αν θα μπορούσαν στην πραγματικότητα να είναι επιβλαβείς. Σύμφωνα με την Αμερικανική Ψυχιατρική Εταιρεία [2, σ. 199], το ποσοστό των ατόμων με κάποια Ειδική Φοβία κατά τη διάρκεια της ζωής τους ανέρχεται στο 7% -9% στις ΗΠΑ και 6% στην Ευρώπη και μπορεί να φτάσει έως και 12,5% από τον Kessler [4]. Υπάρχουν διάφοροι τύποι Ειδικών Φοβιών: **ο φόβος ορισμένων ζώων, η φοβία του αίματος, καθώς και ο φόβος για ύψη, περιορισμένοι χώροι** και άλλοι [2, σελ. 197-198] [5, σελ. 3-4] [6, σελ. 2-3]. Επιπλέον, τα τελευταία χρόνια έχει διεξαχθεί αρκετή έρευνα στον τομέα αυτό, αλλά η μελέτη του Stinson (2007) δείχνει ότι μόνο μια μειοψηφία (8%) ανθρώπων που διαγνώστηκαν με ειδικές φοβίες έλαβαν θεραπεία [7] [5, p . 15].

2.6 Μέθοδοι θεραπείας:

Υπάρχει πληθώρα θεραπειών για τις Ειδικές Φοβίες: Φαρμακευτική Αγωγή/ Φαρμακοθεραπεία, Θεραπεία Γνωσιακής Συμπεριφοράς (CBT), Εφαρμοσμένη Χαλάρωση και πολλά άλλα. Η πιο γνωστή θεραπεία είναι η (CBT), μαζί με τις διάφορες μορφές της, τις οποίες θα καθορίσουμε παρακάτω. Η γνωσιακή συμπεριφοριστική θεραπεία (CBT) δεν έχει ακόμη αποδειχθεί ότι είναι ανώτερη από άλλες μορφές θεραπείας, αλλά η πλειονότητα των ειδικών την προτιμούν, καθώς - αντίθετα με τη φαρμακευτική αγωγή - δεν έχει πιθανές παρενέργειες. είναι μια απλή και συγκεκριμένη θεραπεία που μπορεί να επιφέρει αποτελέσματα, χωρίς την υποστήριξη ενός εξωγενή παράγοντα (όπως φάρμακα) [2, σελ. 189-233] [5, σελ. 32-36] [8]. Στο παρόν κείμενο, θα επικεντρωθούμε αποκλειστικά στις εφαρμογές της (CBT) και των διαφορετικών μορφών της.

2.7 Θεραπεία γνωστικής συμπεριφοράς (CBT):

Γενικά, οι βασικές αρχές αυτής της θεραπείας εστιάζονται στον εντοπισμό, την κατανόηση και την βελτίωση του τρόπου σκέψης και συμπεριφοράς του ασθενή. Ο ασθενής συμμετέχει ενεργά στην αποκατάστασή του και έχει την αίσθηση του ελέγχου κατά τη διάρκεια της θεραπείας. Οι ασθενείς αποκτούν δεξιότητες κατά τη διάρκεια των συνεδριών και πρέπει να εξασκούνται επανειλημμένα σε αυτά που μαθαίνουν, προκειμένου να παρουσιάσουν το επιθυμητό αποτέλεσμα. Τα οφέλη της θεραπείας συνήθως παρατηρούνται μεταξύ των πρώτων 12 έως 16 εβδομάδων, ανάλογα με τον ασθενή. Υπάρχουν διάφορες μορφές CBT, τα όρια μεταξύ αυτών των είναι στενά και κάθε βιβλίο δίνει τους δικούς του ορισμούς. Με βάση την πλειοψηφία, οι διάφορες μορφές CBT είναι οι ακόλουθες [9] [5, σελ. 32-36]:

- Θεραπεία έκθεσης (ET): στην οποία το άτομο εκτίθεται σταδιακά σε μια κατάσταση ή αντικείμενο που φοβάται.

● Θεραπεία αποδοχής και δέσμευσης (ACT): στην οποία το άτομο μαθαίνει πώς να χρησιμοποιεί στρατηγικές αποδοχής και ευαισθησίας (ζώντας τη στιγμή και βιώνοντας τα πράγματα χωρίς κρίση).
● Διαλεκτική Συμπεριφορική Θεραπεία (DBT): στην οποία το άτομο - είτε ατομικά είτε ως μέρος μιας ομάδας - στοχεύει στην επίτευξη πνευματικής ολοκλήρωσης, καθώς και δεξιοτήτων για διαπροσωπική αποτελεσματικότητα, ανεκτική δυσφορία και ρύθμιση των συναισθημάτων (προέρχεται από την Ανατολική κουλτούρα). Στο βιβλίο *"Intensive One-Session Treatment of Specific Phobias"*, αυτή η διαδικασία σχετίζεται με την εφαρμοσμένη χαλάρωση (AR).
● Απευαισθητοποίηση και επανεπεξεργασία κινήσεων οφθαλμών (EMDR): στην οποία το άτομο μειώνει την ένταση των ενοχλητικών σκέψεων. Βοηθά ουσιαστικά τους ασθενείς να παρατηρούν ανησυχητικές εικόνες με λιγότερο δυσάρεστο τρόπο.

Επιπλέον, υπάρχουν αρκετές έρευνες που διαφοροποιούν ή συγκρίνουν τις παραπάνω θεραπείες, όπως η Θεραπεία της Έκθεσης (ET) με τη Θεραπεία της Γνωστικής Συμπεριφοράς (CBT) [10], ή όπως η Θεραπεία Έκθεσης Εικονικής Πραγματικότητας (VET) με την Πραγματική Θεραπεία Έκθεσης (VIVO Exposure) [11]. Συγκεκριμένα, για παράδειγμα, στην δημοσίευση *"CBT vs ET therapy in the treatment of PTSD in refugees"* [10, σελ. 1188-1189], οι επιστήμονες εφαρμόζουν (ET) θεραπεία μέσω εικόνων και βίντεο χωρίς κάποια γνωστική θεραπεία, ενώ η (CBT) εφαρμόζεται μέσω ενός συνδυασμού θεραπείας έκθεσης, γνωστικής θεραπείας και ελεγχόμενης αναπνοής. Όπως λένε: «Η κύρια διαφορά στην (CBT) με την (ET) είναι ο περιορισμένος χρόνος έκθεσης στη φοβία, εξαιτίας των γνωστικών παρεμβάσεων και της ελεγχόμενης αναπνοής που συμπεριλαμβάνονται στη θεραπεία». Σε αυτό το κείμενο, θα θεωρήσουμε την (ET) ως υποκατηγορία (CBT). Γενικά, η (CBT) είναι η κύρια θεραπεία και η (VET), (VIVO), (AR) κλπ. θεωρούνται παραλλαγές της (CBT).

2.8 Εφαρμογές Θεραπείας Γνωσιακής Συμπεριφοράς (CBT):

Τα τελευταία χρόνια, εκτός από το θεωρητικό πλαίσιο, έχουν αναπτυχθεί πολλές εφαρμογές. Σε εργαστηριακό επίπεδο η μέθοδος που ακολουθείται αποτελείται από τα εξής βήματα:

●Συμμετέχοντες: Το Κέντρο Ερευνών βρίσκει ασθενείς βάσει ειδικών κριτηρίων.
●Μετρήσεις - Αξιολόγηση: Το Κέντρο αποκτά πληροφορίες για τους ασθενείς και καθορίζει ποιες μετρήσεις πρέπει να αξιολογηθούν.
●Διαδικασία: Η κατάλληλη θεραπεία εφαρμόζεται στους ασθενείς από τους επόπτες του Κέντρου Ερευνών.
●Ανάλυση αποτελεσμάτων: Το Κέντρο Ερευνών αναλύει τα αποτελέσματα της διαδικασίας με βάση τις μετρήσεις.

Στο παρόν κείμενο θα επικεντρωθούμε αποκλειστικά στη «Διαδικασία» και σε όλες τις διαφορετικές εφαρμογές που υπάρχουν μέχρι τώρα:

1. Σύμφωνα με τη δημοσίευση *"Effect of combined multiple contexts and multiple stimuli exposure in spider phobia"* [12], ο στόχος είναι να αντιμετωπιστεί η αραχνοφοβία μέσω γυαλιών εικονικής πραγματικότητας (VR). Η διαδικασία είναι η ακόλουθη: πέντε σαφώς διακριτά δωμάτια εικονικής πραγματικότητας. πέντε διαφορετικές αράχνες. Στις εικόνες, οι αράχνες φαίνεται να είναι ταραντούλες, αντί για κοινές αράχνες του σπιτιού. Οι αράχνες εμφανίστηκαν στο κέντρο του εκάστοτε εικονικού δωματίου και προγραμματίστηκαν να μεγαλώνει το μέγεθος τους αργά χωρίς να αλλάξουν θέση.

Υπήρχαν δύο συνεδρίες. Οι συνεδρίες χωρίστηκαν σε δύο μέρη: το τμήμα έκθεσης στον κόσμο (VR) και τη «Δοκιμή Αποφυγής» (BAT). Κατά τη διάρκεια της έκθεσης σε (VR), οι συμμετέχοντες καθίσαν σε μια καρέκλα, φόρεσαν τα γυαλιά εικονικής πραγματικότητας και τους δόθηκε να κρατήσουν ένα joystick. Πριν από τη συνεδρία, οι ερευνητές έδωσαν οδηγίες στους ασθενείς να παρακολουθήσουν προσεκτικά την αράχνη και να μην αποφύγουν την οπτική επαφή με αυτήν, καθώς και να αναπνεύσουν κανονικά και να χαλαρώσουν. Όταν η συνεδρία ξεκίνησε, τους δόθηκε η εντολή να μετακινηθούν από το ένα δωμάτιο στο άλλο χρησιμοποιώντας το joystick. Κατά την είσοδο σε ένα νέο δωμάτιο VR, οι ασθενείς έπρεπε να παραμείνουν σε σταθερό σημείο και να παρατηρούν την εμφανιζόμενη αράχνη για μερικά λεπτά. Κατά τη διάρκεια της «Δοκιμή Αποφυγής» (BAT), τοποθετήθηκε μια αράχνη σε ένα διαφανές πλαστικό κουτί και το κουτί τοποθετήθηκε σε απόσταση 3 μέτρα μακριά από την καρέκλα του συμμετέχοντα. Ο στόχος του συμμετέχοντα ήταν να γυρίσει μια λαβή, ώστε να τραβήξει αργά το κιβώτιο που περιέχει την αράχνη όσο το δυνατόν πλησιέστερα προς τον εαυτό του. Αυτά τα βήματα αντιπροσωπεύουν την ακριβή διαδικασία που ακολουθήθηκε σε αυτή την έρευνα.

2. Επιπλέον, σε άλλη μελέτη – *"An experimental test of the role of control in spider fear"* [13] - οι ειδικοί μελέτησαν την αραχνοφοβία χρησιμοποιώντας ένα joystick σε οθόνη χωρίς VR. Διαχώρισαν τους ασθενής σε δύο ομάδες, στην πρώτη, ζητήθηκε από τους ασθενείς να χρησιμοποιήσουν ένα joystick μπροστά από μια οθόνη, προκειμένου να ελέγξουν την απόσταση τους από μια στατική εικονική αράχνη. Στη δεύτερη ομάδα, εκτός από τη μετακίνηση του χειριστηρίου για τον έλεγχο της απόστασης μεταξύ τους και της αράχνης, δημιουργήθηκε μια δεύτερη παράμετρος: η εικόνα της αράχνης κινούνταν από μόνη της βάσει συγκεκριμένων κανόνων. Συνολικά, οι ερευνητές επεδίωξαν να εντοπίσουν τις αντιδράσεις των ασθενών.
3. Επιπρόσθετα, στο *"Effects of an expanding-spaced VS massed exposure schedule on fear reduction and return of fear"* [14] εκτός από την παροχή πληροφοριών σχετικά με την αραχνοφοβία, η κύρια θεραπεία που χρησιμοποιήθηκε ήταν η θεραπεία έκθεσης. Ουσιαστικά, κάθε δοκιμή έκθεσης διαιρέθηκε σε επτά βήματα 1ος λεπτού. Τα επτά βήματα ήταν: στέκονται 12ft μακριά από την αράχνη, στη συνέχεια 5,2ft και τέλος στέκονται δίπλα στο αντικείμενο στην αράχνη, τοποθετώντας τα πρόσωπά τους στην άκρη του δοχείου από γυαλί και καθοδηγώντας την ταραντούλα γύρω.
4. Επιπλέον, όπως αναφέρεται στην *"A controlled study of virtual reality exposure therapy for fear of flying"* [15], ο στόχος είναι να εκτίθεται ο ασθενής σε προσομοιωτή πτήσης με αεροπλάνο σε (VR). Από την περίοδο 1 έως 4, οι ασθενείς ενημερώνονται για τη φοβία τους και οι ειδικοί μειώνουν τις παράλογες σκέψεις των ασθενών τους, όπως «αυτό το αεροπλάνο θα καταρρεύσει» κλπ. Συνεπώς, από την περίοδο 5 έως 8, η έκθεση λαμβάνει χώρα. Οι ασθενείς περνάνε από έναν προσομοιωτή πτήσης VR, βιώνοντας απογειώσεις και προσγειώσεις, ενώ πετούν τόσο με ήρεμο και θυελλώδη καιρό.
5. Μία άλλη μελέτη δίνεται στο έγγραφο *"Avoiding Treatment Failures in Specific Phobias"* [16]. Σχετικά με την αραχνοφοβία, η θεραπεία που χρησιμοποιήθηκε ήταν η εξής: πρώτον, οι ασθενείς ενθαρρύνονται να ανοίξουν ένα βιβλίο και να αγγίξουν την εικόνα μιας αράχνης. Στη συνέχεια, καλούνται να επιθεωρήσουν μια πραγματική αράχνη σε ένα άδειο ποτήρι. Μετά από πολυάριθμες εβδομαδιαίες συνεδρίες 10-50 λεπτών και αρκετές ώρες πρακτικής στο σπίτι, ένας ασθενής κατάφερε να έρθει σε επαφή με μια μεγάλη αράχνη και να την αφήσει να σέρνει στην παλάμη της.
6. Οι ερευνητές στη *"One session treatment for specific phobias in children: Co-morbid anxiety disorders and treatment outcome"* [17] εκθέσαν παιδιά σε φοβικό ερέθισμα πραγματικής ζωής (σκύλοι, χαρακτήρες φορεμάτων, σκοτεινό). Στη συνέχεια, οι

θεραπευτές βοήθησαν τα παιδιά να προσπαθήσουν να διορθώσουν τις αρνητικές αντιλήψεις που σχετίζονταν με το φοβικό ερέθισμα. Επίσης, επικεντρώθηκαν στη «Δοκιμής Αποφυγής» έναντι του φοβικού όντος. Τέλος, δημιούργησαν ένα σύστημα ανταμοιβής για όλα τα παιδιά που έλαβαν θεραπεία.

7. Ένα ακόμα παράδειγμα *"Endogenous cortisol levels influence exposure therapy in spider phobia"* [18]. Στη μελέτη αυτή, οι ειδικοί επικεντρώθηκαν στη θεραπεία της αραχνοφοβίας. Το πρώτο βήμα ήταν να πιάσουν οι ασθενείς τις μικρότερες αράχνες με ένα ποτήρι και μια κάρτα. Μετά το πέρας αυτού του βήματος - και διατηρώντας ένα χαμηλό επίπεδο ανησυχίας - ζητήθηκε από τον ασθενή να αγγίξει την αράχνη με το δείκτη του. Το τρίτο βήμα ήταν να περπατήσει η αράχνη στο χέρι του ασθενούς. Μετά από αυτό, οι θεραπευτές έδωσαν οδηγίες στον ασθενή να αφήσει τον αράχνη να περπατήσει στο σώμα του. Τα τέταρτο βήματα ήταν η επανάληψη με άλλες 2 αράχνες των περιηγούμενων βημάτων. Κατά τη διάρκεια ολόκληρης της διαδικασίας, παρατηρήθηκε μείωση διέγερσης της φοβίας. Τελικά, μετά από τη συνεδρία, οι ασθενείς έλαβαν μια παρακολούθηση αξιολόγησης μέσω της «Δοκιμής Αποφυγής» (BAT), η οποία αφορούσε στο να σταθεί ο ασθενής μπροστά σε κλειδωμένο κουτί που περιείχε αράχνη και να τη προσεγγίσει. Τέλος, εάν ήταν δυνατόν, ο ασθενής έπρεπε να αφαιρέσει το καπάκι, να βάλει το χέρι του και να προσπαθήσει να κρατήσει την αράχνη για τουλάχιστον 20 δευτερόλεπτα.
8. Σύμφωνα με την *"Fear reactivation prior to exposure therapy Does it facilitate the effects of VR exposure in a randomized clinical sample"* [19], στην πρώτη συνεδρίαση οι ασθενείς έλαβαν γενικές γνώσεις σχετικά με την αρανοφοβία και τους ανατέθηκε να παρακολουθήσουν μια αράχνη. Κατά τη διάρκεια των συνεδριών 2 και 3, οι συμμετέχοντες τοποθετήθηκαν σε δωμάτιο VR για 2 λεπτά, χωρίς αράχνες, προκειμένου να οικειοποιηθούν τον εξοπλισμού VR. Στη συνέχεια, οι ασθενείς χωρίστηκαν σε 2 ομάδες. η ομάδα 1 εκτέθηκε σε μια εικονική αράχνη, ενώ η ομάδα 2 σε μια πραγματική αράχνη. Στην πρώτη ομάδα, η θεραπεία ξεκίνησε με μια σκηνή VR μόνο μιας αράχνης, οδηγώντας σε μια σκηνή με τέσσερις αράχνες που κινούνται. Στη συνεδρία 4, όλοι οι ασθενής πάλι – και από τις δυο ομάδες εκτέθηκαν σε πραγματική ταραντούλα (Vivo Θεραπεία Έκθεσης). ο στόχος ήταν να αγγίξουν την αράχνη χωρίς να βιώνουν μεγάλο φόβο.
9. Άλλες δύο μελέτες – *"Augmenting one-session treatment of children's specific phobias with attention training to positive stimuli"* [20] και *"Harm beliefs and coping expectancies in youth with specific phobias"* [21] – βασισμένες στη θεωρία του Ost [22], η οποία αφορά τη θεραπεία έκθεσης VIVO, τα βήματα που ακολουθήθηκαν κατά τη διάρκεια της Θεραπείας Έκθεσης VIVO είναι:
    a. Ο ασθενής εγγυάται να παραμείνει στην κατάσταση έκθεσης μέχρι να εξασθενήσει το άγχος.
    b. Ο ασθενής ενθαρρύνεται να προσεγγίσει το ερέθισμα όσο το δυνατόν περισσότερο και να συνεχίσει με την έκθεση VIVO.
    c. Όταν το επίπεδο άγχους έχει μειωθεί, ο ασθενής είναι εκπαιδευμένος να προσεγγίσει ακόμη περισσότερο το φοβικό ερέθισμα.
    d. Η θεραπεία ολοκληρώνεται μόνο όταν τα επίπεδα άγχους του ασθενούς έχουν μειωθεί κατά 50%. Ο σκοπός της έκθεσης είναι να μάθει πώς να μένεις ήρεμος κοντά στο φοβικό αντικείμενο και να την αγγίξεις τελικά.

2.9 Επισκόπηση των Εφαρμογών CBT:

Εάν δώσουμε μια γενική εικόνα των προαναφερθεισών ερευνών και φτιάξουμε μια γενική μορφή θεραπείας, αυτό που παρατηρούμε είναι η επανάληψη ενός μοτίβου, το οποίο είναι το ακόλουθο:

- **Βήμα 1:** Ο ασθενής ενημερώνεται για τη φοβία του και για τον παράλογο τρόπο σκέψης του. Στη συνέχεια, ο ασθενής συζητά με τον ιατρό οποιεσδήποτε δυσάρεστες σκέψεις και πιθανά σενάρια που τον φοβίζουν. Αυτό το βήμα είναι γνωστό και ως γνωστική θεραπεία (CBT).
- **Βήμα 2:** Ο ασθενής εκτίθεται στην επίφοβη κατάσταση σε πραγματικό περιβάλλον (Έκθεση VIVO), ή σε εικονικό περιβάλλον μέσω εικονικής πραγματικότητας (έκθεση VR) ή μέσω ηλεκτρονικής οθόνης. Αυτό το βήμα είναι γνωστό και θεραπεία έκθεσης (ET).
- **Βήμα 3:** Η θεραπεία έκθεσης από το βήμα 2 επαναλαμβάνεται, μέχρι ο ασθενής να μην παρουσιάζει έντονα σημάδια φόβου.
- **Βήμα 4:** Μετά από μια ένα χρονικό διάστημα διορίζεται μια θεραπεία παρακολούθησης, στην οποία οι ειδικοί αξιολογούν κατά πόσο έχουν παραμείνει τα οφέλη από την θεραπεία που προηγήθηκε.

Ορισμένα ερευνητικά κέντρα επικεντρώνονται μόνο σε ένα συγκεκριμένο βήμα, και άλλα επικεντρώνονται σε περισσότερα από ένα βήματα, ενώ άλλα συγκρίνουν τα τέσσερα διαφορετικά βήματα και μερικά άλλα συγκρίνουν διαφορετικές μεθόδους σε ένα συγκεκριμένο βήμα - βασισμένο στον στόχο του εργαστηρίου. Ωστόσο, γενικά, τα βήματα που ακολουθούνται συνήθως είναι αυτά που αναφέρθηκαν παραπάνω.

Επίσης πρέπει να επισημάνουμε ότι στο Βήμα 2 – στην θεραπεία έκθεσης - η εκπαίδευση του ασθενή εστιάζεται γύρω από τις ακόλουθες άσκησης , π.χ. «Παραμείνετε κοντά στην φοβική κατάσταση», «κοιτάξτε το φοβικό αντικείμενο», «περπατήστε προς το φοβικό αντικείμενο», «κρατήστε το φοβικό αντικείμενο».

2.10 Αποτελέσματα των Εφαρμογών CBT:

Οι εφαρμογές της CBT μέχρι σήμερα έχουν αξιόλογα αποτελέσματα. Βάσει των συμπερασμάτων που παρουσιάζουν οι προαναφερθείσες μελέτες, από την παράγραφο 2.7, αν προσπαθήσουμε να βγάλουμε ένα μέσο όρο, τότε το 40% με 50% των ασθενών κατάφερε να μειώσει το άγχος που προκαλείται από τη φοβία. Εν τούτης, στηριζόμενοι σε πληθώρα ερευνών που έχουν εστιάσει στη μετααναλυτική αποτελεσμάτων - για παράδειγμα στη δημοσίευση *"A meta-analytic review of neuroimaging studies of specific phobia to small animals"* [23] και στη δημοσίευση "*A review and meta-analysis of the heritability of specific phobia subtypes and corresponding fears"* [33] ιδιαίτερο ενδιαφέρον έχει η μεγάλη διακύμανση των αποτελεσμάτων. Δηλαδή υπάρχουν έρευνες που παρουσιάζουν αποτελέσματα μέχρι και 70% μείωσης της φοβίας και άλλες το πολύ 30%. Ταυτόχρονα, ένα ακόμα κριτήριο αξιολόγησης είναι ο χρόνος επιστροφής της φοβίας, δηλαδή πόσο διαρκούν τα οφέλη της θεραπείας μετά την ολοκλήρωση της. Ακόμα και σε αυτό το κριτήριο υπάρχει ασάφεια, παραδείγματος χάριν, οι ερευνητές της *"Harm beliefs and coping expectancies in youth with specific phobias*" [21] ισχυρίζονται ότι η φοβία δεν επιστρέφει σε λιγότερο από 6 μήνες. Αλλά στη μελέτη, *"Effect of combined multiple contexts and multiple stimuli exposure in spider phobia: A randomized clinical trial in virtual reality"* [12], οι επιστήμονες αποκαλύπτουν ότι η CBT δεν διαρκεί παραπάνω από 20 μέρες.

3 Ιδέα του Συστήματος (AT)

3.1 Εισαγωγή

Σε αυτό το κεφάλαιο αναγνωρίζονται τα κενά των εφαρμογών που παρουσιάστηκαν στο κεφάλαιο 1 και αναφέρονται θεωρητικές προτάσεις και ιδέες από επιστήμονες που δεν έχουν ακόμα εφαρμοστεί. Έπειτα δομείται μια νέα εφαρμογή που εν δυνάμει μπορεί να καλύψει τα κενό των διαφόρων μέχρι τώρα εφαρμογών. Τέλος επιδιώκεται η επαλήθευση της νέας αυτής εφαρμογής και από άλλες οπτικές γωνίες.

3.2 Μια νέα πρόταση:

Από τις προηγούμενες παραγράφους, μπορούμε να συμπεράνουμε ότι οι εφαρμογές της (CBT) χρειάζονται περαιτέρω διερεύνηση. Το πρώτο βήμα για την εξεύρεση νέων εργαλείων και βελτιωμένων θεραπειών, είναι η μελέτη περισσότερων ιδεών που ακόμα δεν έχουν εφαρμοστεί στην πράξη λόγο των φυσικών-τεχνολογικών περιορισμών που υπάρχουν όταν αυτές δημοσιεύονταν. Οι συγγραφείς του βιβλίου *"Intensive One-Session Treatment of Specific Phobias"* [5, σελ. 68-72] αναλύουν πληθώρα θεωριών σχετικά με τη θεραπεία CBT. Ιδιαίτερο ενδιαφέρον παρουσιάζει στις σελίδες 68-72 μια θεωρητική εφαρμογή της αραχνοφοβικής θεραπείας. Σε αυτό το παράδειγμα οι συγγραφείς σημειώνουν συγκεκριμένα: «*Το πρώτο βήμα είναι να διδάξουμε στον ασθενή πώς να πιάσει την αράχνη με ένα ποτήρι και ένα κομμάτι χαρτί, και να το πετάξει έξω από το σπίτι, το οποίο είναι κάνει αβίαστα ένας μη φοβικός άνθρωπος [....]. Ο θεραπευτής στη συνέχεια βοηθά τον ασθενή να εκτελέσει αυτή την άσκηση. Συνήθως, αυτή η διαδικασία επαναλαμβάνεται τρεις ή τέσσερις φορές [...]*». Αναλύοντας αυτή την ιδέα παρατηρούμε ότι στόχος είναι ο ασθενής να μπορεί ευκολά και αβίαστα να εκτελεί μια δράση (πιάσε την αράχνη και το ποτήρι και βγάλε την έξω) και έτσι η φοβία θα περιοριστεί. Επαναλαμβάνουμε, ότι, μέχρι σήμερα -όπως αναφέραμε και στην παράγραφο 2.8 - σχεδόν όλες οι έρευνες έχουν εστιάσει στο να εκπαιδευθεί ο ασθενείς στο «πώς» να συνυπάρχει με τη φοβία του, πως να οικειοποιηθεί τη φοβία του, όχι πως να συμπεριφερθεί και να δράσει όταν βρεθεί αντιμέτωπος με τη φοβία του, το οποίο προτείνει το βιβλίο *"Intensive One-Session Treatment of Specific Phobias"*.

3.3 Νέο βήμα στη μέχρι τώρα θεραπεία:

Σύμφωνα με τα προηγούμενα, μπορούμε να παρατηρήσουμε ότι ίσως υπάρχει ένα βήμα που λείπει στα 4ρα Βήματα Εφαρμογών της CBT, παράγραφο 2.8. Το βήμα που λείπει είναι να εκπαιδευτεί ο ασθενής σε βασικές ενέργειες που πρέπει να κάνει όταν βρεθεί αντιμέτωπος με τη φοβία του, προκειμένου να ελέγξει το άγχος του, να γίνει ο «κύριος» της φοβίας του και όχι απλώς να συνυπάρξει μαζί με αυτή. Ο σκοπός αυτής της πρόσθετης εκπαίδευσης είναι να είναι σε θέση ο ασθενής να σκέφτεται και να ενεργεί αβίαστα, αυτόβουλος και χωρίς άγχος. Γενικά, η πρότασή μας για να ολοκληρωθεί η Εφαρμογή του CBT είναι:

- **Βήμα 1 (ίδιο):** Ο ασθενής ενημερώνεται για τη φοβία του και για το παράλογο τρόπο σκέψης του. Στη συνέχεια, ο ασθενής συζητά με τον ιατρό οποιεσδήποτε δυσάρεστες σκέψεις και πιθανά ενοχλημένα σενάρια των φοβίζουν. Αυτό το βήμα είναι γνωστό και ως γνωστική θεραπεία (CBT).

●**Βήμα 2 (ίδιο):** Ο ασθενής εκτίθεται στην επίφοβη κατάσταση μέσω της έκθεσης σε πραγματικό περιβάλλον (Έκθεση VIVO), ή έκθεσης εικονικό περιβάλλον μέσω εικονικής πραγματικότητας (έκθεση VR) ή μέσω ηλεκτρονικής οθόνης. Αυτό το βήμα είναι γνωστό και θεραπεία έκθεσης (ET).
●**Βήμα 3 (νέο):** Ο ασθενής μαθαίνει πώς να διαχειρίζεται τη φοβία του μέσω εκπαίδευσης σε δράσεις.
●**Βήμα 4 (ίδιο):** Η θεραπεία έκθεσης από το βήμα 2 και η εκπαίδευση δράσεων επαναλαμβάνονται από το βήμα 3, μέχρι ο ασθενής να μην παρουσιάζει έντονα σημάδια φόβου.
●**Βήμα 5 (ίδιο):** Μετά από μια ένα χρονικό διάστημα διορίζεται μια θεραπεία παρακολούθησης, στην οποία οι ειδικοί αξιολογούν κατά πόσο έχουν παραμείνει τα οφέλη από την θεραπεία που προηγήθηκε.

Κάνοντας μια περεταίρω ανάλυση των προαναφερθέντων βημάτων είναι σημαντικό να τονίσουμε τα ακόλουθα.

- Από το (βήμα 1), οι επιστήμονες προσπαθούν να μειώσουν τη σοβαρότητα της επίφοβης κατάστασης στον εγκέφαλο του ασθενούς με λογικά συμπεράσματα και να συζητήσουν κάποιες από τις σκέψεις του για πιθανές φοβικές καταστάσεις. Κυρίως, το βήμα 1 έχει στόχο τον εξορθολογισμό των φοβιών του.
- Κατά συνέπεια, κατά τη διάρκεια του τμήματος Έκθεσης (βήμα 2), ο ασθενής διδάσκεται πώς να είναι σε θέση να συνυπάρχει με τη φοβία του. Ενθαρρύνετε ο ασθενής να μην φοβάται τη φυσική επαφή με το εν λόγω φοβικό αντικείμενο / κατάσταση.
- Στο (βήμα 3) στόχος είναι ο ασθενής να εκπαιδευτεί στο πώς να ενεργήσει σε μια επίφοβη κατάσταση, πώς να φτάσει σε ασφαλή κατάσταση, πώς να ελέγξει τη φοβία του και να φέρει τον εαυτό του σε ασφαλή θέση, σε οποιαδήποτε δεδομένη στιγμή. Για παράδειγμα: «πώς να πιάσει ένα φοβικό αντικείμενο», «πώς να αποφύγεις ένα φοβικό αντικείμενο όταν πρόκειται για τον ασθενή», «πού να θέσει το φοβικό αντικείμενο έτσι ώστε να αισθάνεται ασφαλές», «ο ασθενής πρέπει να παραμείνει για να αισθανθεί ασφαλής» κλπ. Ναι μεν αυτές οι τεχνικές είναι θέμα συζήτησης ασθενούς με θεραπευτή στο (βήμα 1) [5, σελ. 62, δηλαδή στο (CBT) μέρος. Για παράδειγμα, ο θεραπευτής μπορεί να ζητήσει από τον ασθενή ερωτήσεις όπως: «Φανταστείτε ότι περπατάτε σε ένα δωμάτιο και βλέπετε μια αράχνη, τι κάνεις;», «Φανταστείτε ότι η αράχνη είναι πάνω σας, τι θα κάνατε». Αλλά το (βήμα 3) εκπαιδεύει τον ασθενή στην εκτέλεση αυτών των σκέψεων. Ο σκοπός του (AT) είναι να εκπαιδεύσει τον ασθενή για το πώς να αντιδράσει σε παρόμοιες καταστάσεις, χρησιμοποιώντας τόσο το μυαλό όσο και το σώμα του, πώς ο ασθενής μπορεί να καταλάβει και να ελέγξει τις κινήσεις του σώματος και να ενεργήσει με σιγουριά.

3.4 Γενική ιδέα του (AT) συστήματος:

Η γενική ιδέα του νέου εργαλείου είναι να ανιχνεύσει όλα τα πιθανά σενάρια γύρω από τη φοβία του ασθενούς που θα μπορούσαν να συμβούν στην πραγματική ζωή και να τους βοηθήσει να αποκτήσουν τις απαιτούμενες δεξιότητες για να ενεργήσουν ήρεμα και να είναι σε θέση να διατηρήσουν τον εαυτό τους ασφαλή. Για παράδειγμα, ένα πιθανό σενάριο που θα μπορούσε να αντιμετωπίσει ο ασθενής είναι να δει μια αράχνη στο σπίτι μπροστά του. Ο στόχος για τον ασθενή να βρει γρήγορα κάτι που θα μπορούσε να χρησιμοποιηθεί ως ένα ποτήρι και κάτι που θα μπορούσε να χρησιμοποιηθεί ως κάρτα, θα κινηθεί προς την αράχνη, στη συνέχεια να την αρπάξει και να την τοποθετήσει κάπου μακριά - έξω από το σπίτι του -

έτσι ώστε η αράχνη να μην μπορεί να επιστρέψει. Μια τέτοια εκπαίδευση θα πρέπει να δίνει στον ασθενή τη δυνατότητα να παραμείνει ήρεμος και να αισθάνεται ασφαλής, ακόμη και αν αντιμετωπίζει ένα φοβικό αντικείμενο ή κατάσταση.

3.5 Υλοποιήσεις παρεμφερών θεραπειών και οι περιορισμοί αυτών:

Όπως βλέπουμε στο *"Intensive One-Session Treatment of Specific Phobias"* [5, σελ. 68-72], η έννοια αυτής της τεχνικής υπάρχει θεωρητικά. Ωστόσο, μέχρι σήμερα, η υπάρχουσα τεχνολογία δεν μπόρεσε να προσομοιώσει πλήρως μια τέτοια μορφή θεραπείας. Στη μελέτη *"Endogenous cortisol levels influence exposure therapy in spider phobia"* [18] οι επιστήμονες, προσπάθησαν να διδάξουν στον ασθενή πώς να βάλει μια πραγματική αράχνη σε ένα πραγματικό ποτήρι και να το σφραγίσει με μια πραγματική κάρτα. Ωστόσο, υπήρχαν κάποια προβλήματα όπως

- η προσομοίωση σε πραγματικό περιβάλλον με αληθινές αράχνες και αντικείμενα έχει μεγάλο κόστος [24, σ. 6].
- η προσομοίωση σε πραγματικό περιβάλλον με αληθινές αράχνες και αντικείμενα έχει μεγάλο ρίσκο και απρόβλεπτους παράγοντες όπως η αράχνη να αναπηδήσει [24, σ. 6].

Επιπλέον, αν και υπάρχει εξοπλισμός εικονικής πραγματικότητας από το 2015, τα ακουστικά VR μπορούν να παράγουν μόνο μια έξοδο, που σημαίνει ότι ο ασθενής μπορεί να λάβει εικόνες VR αλλά δεν μπορεί να αντιδράσει σε αυτά, λόγω έλλειψης εξοπλισμού εισόδου VR. Επομένως, δεδομένου ότι το σύστημα δεν μπορεί να προσαρμοστεί στον ασθενή, δεν μπορεί ακόμη να εφαρμοστεί θεραπεία. Ομοίως, εμπειρογνώμονες στην *"Virtual reality in the assessment, understanding, and treatment of mental health disorders"* υπογραμμίζουν την έλλειψη ορθής τεχνολογίας στη θεραπεία ψυχικών διαταραχών [24, σ. 5].

3.6 Υλοποίηση του (AT) και εργαλεία:

Η λύση των προαναφερθέντων περιορισμών επιτυγχάνεται με τη χρήση μιας συσκευής που θα μπορούσε να παρακολουθήσει τις κινήσεις του ασθενούς και να δώσει τα συλλεχθέντα δεδομένα πίσω στο σύστημα ώστε να κρίνει αν η άσκηση εκτελέστηκε ορθά. Για να το επιτύχουμε, πρέπει να χρησιμοποιήσουμε κάμερες βάθους μαζί με τα ακουστικά VR.

Στην εφαρμογή, θα χρησιμοποιηθούν:
Αξεσουάρ Samsung που συνδέεται με τηλέφωνο Samsung S7 και
Κάμερα βάθους (Orbbec Persee), .
Η κάμερα βάθους είναι ένα εργαλείο που παρακολουθεί τις κινήσεις και δίνει ανά πάσα στιγμή πληροφορίες για το σώμα του ατόμου, τις κινήσεις, τη θέση του κλπ. Βάσει του προηγούμενου παραδείγματος, θα δημιουργήσουμε ένα εικονικό σπίτι με τη βοήθεια γυαλιών VR. Στη συνέχεια, θα παρακολουθήσουμε τις κινήσεις του ασθενούς μέσω της Κάμερας Βάθους και θα χρησιμοποιήσουμε αυτό το υλικό για να εκπαιδεύσουμε τον ασθενή ώστε να εκτελέσει με επιτυχία συγκεκριμένες ασκήσεις.

3.7 Ρύθμιση χώρου:

Για υλοποιηθεί το (AT) σύστημα ορθά, πρέπει να είμαστε σε θέση να προσομοιώσουμε όσο το δυνατόν με πιο ρεαλιστικό τρόπο την φοβική κατάσταση. Για να επιτευχθεί αυτό, θα πρέπει

γενικά να προσομοιώσουμε τις αισθήσεις του ασθενούς - γεύση, όραση, αφή, οσμή και ακοή. Σε αυτές όμως που πρέπει να δώσουμε τη μεγαλύτερη βάση είναι στην όραση, την αφή και την ακοή.

● **Προσομοίωση της αφής:** Η προσομοίωση της αίσθησης της αφής είναι απλή, χρειάζεται μόνο όσα αντικείμενα έρχονται σε επαφή με τον ασθενή. Μπορεί στο εικονικό περιβάλλον να βάλουμε όσα αντικείμενα θέλουμε, αλλά δεν χρειάζεται όλα αυτά τα αντικείμενα να είναι στον πραγματικό κόσμο. Πρώτα ξεκινάμε με την εκκένωση μιας περιοχής περίπου 30 m2, όπου θα είναι ο χώρος όπου ο ασθενής θα εκπαιδευτεί. Και στη συνέχεια τοποθετούμε όσα αντικείμενα θα έρθει σε επαφή ο ασθενής και απαιτούνται για να ολοκληρώσει τη θεραπεία δράσης.

● **Προσομοίωση οραματισμού και ακρόασης:** Χρησιμοποιώντας ένα ακουστικό VR (Samsung Headset), προσομοιώνουμε το εικονικό περιβάλλον του χώρου που αναφέρθηκε παραπάνω. Το εικονικό περιβάλλον περιέχει τα διακοσμητικά αντικείμενα στο δωμάτιο, καθώς και την εικονική το φοβικό αντικείμενο ή κατάσταση. Το εικονικό περιβάλλον θα πρέπει να είναι όσο το δυνατόν πιο ρεαλιστικό ώστε να δίνει στον ασθενή την αίσθηση ότι αναβιώνει στη πραγματικότητα της φοβίας του. Όσον αφορά την προσομοίωση ακρόασης, τα ακουστικά VR παράγουν ήχο, οι ήχοι συνιστάται να είναι οικείοι στον ασθενή όταν αντιμετωπίζει τη φοβία του.

Αυτό το πείραμα διεξάγεται σε ένα δωμάτιο που ο ασθενής δεν θα πρέπει να μπορεί να δει. Πριν ξεκινήσει η θεραπεία ο επιβλέπων τοποθετεί το ακουστικό VR στο κεφάλι του ασθενούς σε διαφορετικό δωμάτιο από αυτό που θα γίνει η εκπαίδευση. Εν συνεχεία ο επιβλέπων οδηγεί τον ασθενή στο δωμάτιο προσομοίωσης και τον τοποθετεί στο κέντρο του δωματίου. Όταν ο ασθενής είναι τελικά έτοιμος, το ακουστικό VR είναι ενεργοποιημένο και το άτομο βρίσκεται στο εικονικό περιβάλλον.

3.8 Αυτοεκτίμηση:

Πριν προχωρήσουμε στη πρακτική ανάλυση του (ΑΤ), είναι σημαντικό να παρουσιάσουμε τη θεωρητική άποψη που υπογραμμίζει τα οφέλη του (ΑΤ) συστήματος. Ουσιαστικά το (ΑΤ) **βασίζεται στη γνώμη του ασθενούς για τον εαυτό του, στην αυτοεκτίμηση και στην ικανότητά του να αντέχει σε μια κατάσταση φόβου**. Ο στόχος του (ΑΤ) είναι να καλλιεργήσει την πεποίθηση ότι ο ασθενής μπορεί να χειριστεί οποιαδήποτε φοβική κατάσταση και να εκπαιδευτεί με τρόπο που θα μπορεί να του δώσει την απαραίτητη εμπιστοσύνη για να ενεργήσει ορθά σε μια τρομακτική κατάσταση. Ουσιαστικά, το (ΑΤ) σύστημα στοχεύει στον ασθενή να αποκτήσει υψηλότερη αυτοεκτίμηση όσον αφορά στη φοβία του.

Στην κοινωνιολογία και την ψυχολογία, η αυτοεκτίμηση αντανακλά τη συνολική υποκειμενική εκτίμηση του ατόμου για την αξία του. Είναι μια μορφή αυτοκριτικής, καθώς και μια έννοια του εαυτού. Η υψηλή αυτοεκτίμηση περιλαμβάνει την εμπιστοσύνη στον εαυτό του («είμαι πλήρως ικανός», «είμαι άξιος»), καθώς και αισθήματα όπως ο θρίαμβος, η απελπισία, η υπερηφάνεια και η ντροπή [25, σελ. 217-244 · 25, σελ. 217-244]. Ο Smith και ο Mackie σημείωσαν συγκεκριμένα: «Η αυτοεκτίμηση είναι ουσιαστικά αυτό που σκεφτόμαστε για τον εαυτό μας. Η αυτοεκτίμηση είναι οι θετικές ή αρνητικές αξιολογήσεις των εαυτών μας και το πώς αισθανόμαστε για εμάς» [26]. Ως εκ τούτου, μπορούμε να πούμε ότι η χαμηλή αυτοεκτίμηση είναι η κακή εκτίμηση του εαυτού μας, γενικά, ή σχετικά, με μια συγκεκριμένη

κατάσταση. Δηλαδή, το άτομο θεωρεί ότι δεν είναι σε θέση να αντιμετωπίσει μία ή περισσότερες φοβικές καταστάσεις [27, 28].

Η χαμηλή αυτοεκτίμηση θεωρείται μια από τις κύριες πηγές φοβιών. Το άτομο σκέφτεται ότι δεν ξέρει πώς να αντιμετωπίσει ή δεν μπορεί να αντιμετωπίσει μια συγκεκριμένη κατάσταση, άρα αισθάνεται άγχος εξαιτίας αυτού και ως αποτέλεσμα, δημιουργείται η φοβία [29, 30, 31]. Πιο συγκεκριμένα, το άτομο είτε είχε ή είδε κάποιον άλλον να έχει κακή εμπειρία στην εν λόγω κατάσταση, ή ο ίδιος δεν κατάφερε να φέρει το επιθυμητό αποτέλεσμα σε μια κατάσταση ή αν το έκανε, ήταν με μεγάλη δυσκολία. Έτσι, το άτομο αποφεύγει αυτή την κατάσταση ή αποκτά πολύ άγχος όταν επίκειται το βίωμά της.

Κατά συνέπεια, ανάλογα με την αυτοεκτίμηση του ατόμου σχετικά με τη δεδομένη κατάσταση, θα προκύψουν αντίστοιχα συναισθήματα. Εάν το άτομο έχει υψηλή αυτοεκτίμηση, τότε θα έχει συναισθήματα όπως η χαρά της νίκης. Αν το άτομο έχει χαμηλή αυτοεκτίμηση, τότε θα εμφανιστούν συμπτώματα όπως η απογοήτευση, ο φόβος και το άγχος. Έτσι, προκειμένου να βοηθηθεί το άτομο να αποκτήσει υψηλότερα επίπεδα αυτοπεποίθησης, πρέπει να αποκτήσει τις κατάλληλες δεξιότητες [32, σελ. 632-633].

Ο στόχος του (ΑΤ) είναι να εκπαιδεύσει τον ασθενή στη φοβία του, να αποκτήσει υψηλότερη αυτοεκτίμηση, να ενημερωθεί για κάθε πιθανή τρομακτική κατάσταση που μπορεί να προκύψει, να διδαχθεί πώς αντιδρά σε τέτοιες περιπτώσεις και να είναι σίγουρος ότι μπορεί να βάλει ένα τέλος σε αυτό όποτε το θελήσει.

4 Βασικό σύστημα (ΑΤ)

4.1 Εισαγωγή:

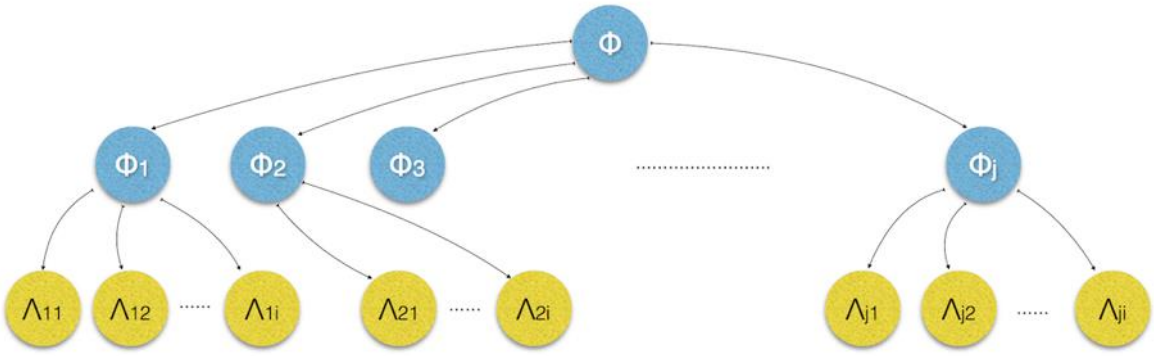
Στο κεφάλαιο 2 δομήσαμε ένα πρωταρχικό σύστημα και παρουσιάσαμε την τεχνολογία που χρησιμοποιούμε. Εν τούτης ακόμα η διαδικασία της εκπαίδευσης και του πώς το σύστημα λειτουργεί δεν είναι ακόμα σαφές. Στο ακόλουθο κεφάλαιο παρουσιάζουμε τη διαδικασία της εκπαίδευσης και ένα παράδειγμα εφαρμογής αυτής της εκπαίδευσης στην αραχνοφοβία.

4.2 Μοντέλο - Βασικό Δέντρο Καταστάσεων και Λύσεων:

Το πρώτο πράγμα που πρέπει να κάνουμε είναι να οικοδομήσουμε το μοντέλο που θα μας βοηθήσει να κατασκευάσουμε το σύστημα (ΑΤ). Όταν ο ασθενής βρίσκεται σε απειλητική κατάσταση, ένα από τα πρώτα πράγματα που κάνει είναι να σκεφτεί τα πιθανά σενάρια και τα αποτελέσματα, καθώς και τι πρέπει να κάνει για να επιβιώσει από την επίφοβη κατάσταση. Αυτό μπορεί να αναπαρασταθεί στο παρακάτω μοντέλο.

Αντιπροσωπεύουμε τη φοβία με το γράμμα «Φ» και πιθανές φοβικές καταστάσεις με «Φj» - για παράδειγμα, Φ1: αποτελείται από μια αράχνη που βρίσκεται πάνω σε ένα τραπέζι μπροστά στον ασθενή, Φ4: μια αράχνη που βρίσκεται στην οροφή του σπιτιού κλπ. Στη συνέχεια, ορίζουμε για κάθε «Φj» την κατάσταση στην οποία ο ασθενής αισθάνεται ασφαλής και θέλει να βρεθεί την οποία ονομάζουμε «Λύση» - για παράδειγμα, στο Φ1 η λύση Λ11 είναι: ο ασθενής να πάρει ένα κομμάτι χαρτί για να πιάσει την αράχνη και να την πετάξει έξω. Γενικά, το Φj περιλαμβάνει όλες τις δυνητικά επιβλαβείς ή επίφοβες καταστάσεις στις οποίες μπορεί να βρεθεί ο ασθενής ενώ οι Λij είναι οι ασφαλείς καταστάσεις στις οποίες επιθυμεί να βρεθεί, για να νιώθει ασφαλής.

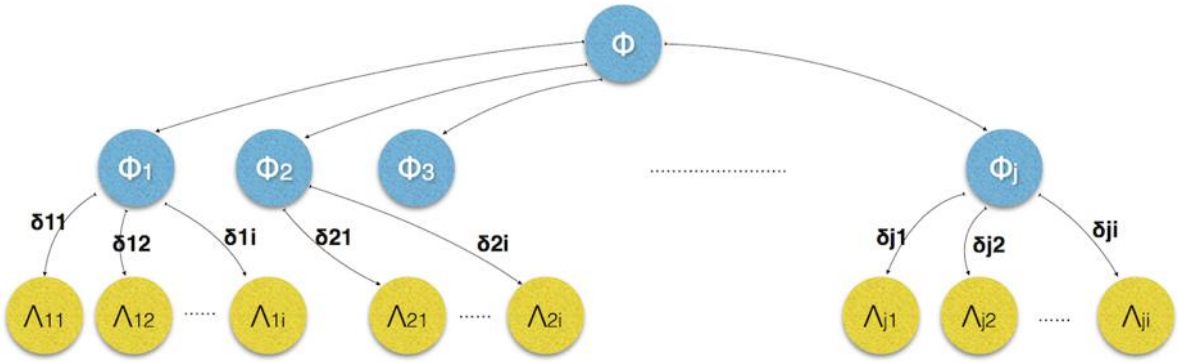
Ονομάζουμε Φj «Καταστάσεις» και Λij «Λύσεις». Η φοβία μπορεί να προέρχεται από πολλές διαφορετικές περιστάσεις και κάθε μία από αυτές μπορεί να έχει περισσότερες από μία πιθανές λύσεις. Η παρακάτω διάταξη ονομάζεται Βασικό Δέντρο Καταστάσεων και Λύσεων (Εικόνα 1).



Εικόνα 1

4.3 Μοντέλο - Βασικό Δέντρο Καταστάσεων, Λύσεων και Δράσεων:

Το πρώτο βήμα για να ξεπεράσει ο ασθενής τη φοβία του, είναι να γνωρίσει τη κάθε πιθανή φοβική κατάσταση (Φj) και τις λύσεις (Λιij) της. Το επόμενο βήμα είναι ο ασθενής να μάθει πώς να ενεργεί ώστε να επιτύχει την εκάστοτε λύση, προκειμένου να επιτύχει το επιθυμητό αποτέλεσμα. Αυτή τη διαδικασία που ο ασθενής από τη Κατάσταση πάει στη Λύση την ορίζουμε ως Δράση (δij). Δηλαδή ποιες κινήσεις πρέπει να κάνει ο ασθενής ώστε να φέρει τον εαυτό του σε ασφαλή κατάσταση. Δηλαδή προσθέσουμε μια νέα παράμετρο - τις Δράσεις (δij), στο «Δέντρο των Βασικών Καταστάσεων και Λύσεων», θα έχουμε ένα νέο διάγραμμα: «Δέντρο Βασικών Καταστάσεων, Λύσεων και Δράσεων». Προσοχή οι Δράσεις δεν είναι οι Λύσεις. Λύση ορίζουμε το αποτέλεσμα, το στόχο που έχει ο ασθενής να βρεθεί σε ασφαλή θέση,ενώ Δράση είναι η διαδικασία που θα οδηγήσει στη Λύση, δηλαδή το «πώς» θα πετύχει τη λύση, τι σκέψεις θα κάνει, τι πράξεις θα κάνει, πώς θα κινηθεί ο ασθενής για να βρεθεί στη λύση (Εικόνα 2).



Εικόνα 2

Έστω παράδειγμα, Φj: «Ο ασθενής βλέπει την αράχνη μπροστά του» και Λji: «Ο ασθενής πιάνει με το ποτήρι και το χαρτί την αράχνη και τη βγάζει έξω» τότε η δράσει δij είναι: «Ο ασθενής να βρει ένα ποτήρι ή ένα υποκατάστατο ενός ποτηριού, να σηκωθεί ήρεμα χωρίς να χάσει την αράχνη, να βρει ένα κομμάτι χαρτί ή ένα υποκατάστατο χαρτιού, να κατευθυνθεί προς την αράχνη και να την τοποθέτηση με ασφάλεια στο ποτήρι, να κλίσει το ποτήρι με το χαρτί, τέλος να βγει έξω και να τοποθετήσει την αράχνη σε κάποιο μέρος που δεν θα είναι σε θέση να επιστρέψει.» Ουσιαστικά, το (ΑΤ) σύστημα που σκοπό έχει να εκπαιδεύσει τον ασθενή στο πώς να εκτελεί ενέργειες που θα τον κάνουν να αισθάνονται ασφαλείς – δηλαδή το (ΑΤ) σύστημα προσπαθεί να ενισχύσει την αυτοπεποίθηση του ασθενούς έτσι ώστε, όταν βρεθεί σε μια φοβική κατάσταση, να αισθάνεται ικανός να τη διαχειριστεί με τον κατάλληλο τρόπο.

4.4 Εφαρμογή του Βασικού Συστήματος (ΑΤ):

Σύμφωνα με τις παραγράφους 4.2 και 4.3 για κάθε φοβία που θέλουμε θεραπεύσουμε με τη χρήση του συστήματος (ΑΤ), το πρώτο πράγμα που πρέπει να κάνουμε είναι να αναγράψουμε

το Δέντρο Καταστάσεων Λύσεων και Δράσεων, δηλαδή να αναγράψουμε τις φοβικές καταστάσεις του ασθενούς, τι πρέπει να κάνει για να βρεθεί σε ασφαλή θέση και πως θα φέρει τον εαυτό σου σε αυτή τη θέση για κάθε κατάσταση. Στη συνέχεια προσομοιώνουμε αυτές τις φοβικές καταστάσεις στο εικονικό περιβάλλον. Έπειτα τοποθετούμε τον ασθενή στο εικονικό περιβάλλον και για κάθε φοβική κατάσταση και τον προτρέπουμε να εκτελέσει τις δράσεις για να πετύχει την εκάστοτε λύση. Τέλος επαναλαμβάνουμε την εκπαίδευση έως ότου ο ασθενής παρατηρηθεί ότι έχει μείωση του αισθήματος φοβίας.

4.5 Παράδειγμα του (AT):

Στην ακόλουθη παράγραφο θα παρουσιάσουμε μια εφαρμογή του (AT) συστήματος με τη χρήση του Απλού Δέντρου Καταστάσεων Λύσεων και Δράσεων για την αραχνοφοβία.

(Σημαντική σημείωση: Οι διάφορες καταστάσεις, λύσεις και δράσεις που χρησιμοποιούμε σε αυτό το παράδειγμα είναι ενδεικτικές, οι οποίες δεν είναι απαραίτητο να υιοθετηθούν. Ο κάθε ψυχολόγος, ψυχίατρος μπορεί να εφαρμόσει τις δικές του).

βήμα 1:Ο επιβλέπων δημιουργεί τα πιθανά φοβικά σενάρια της αραχνοφοβίας, δηλαδή τις διάφορες Καταστάσεις, όπως:

- Σενάριο Φ1: Βρίσκετε μια μεγάλη αράχνη στο πάτωμα, ακριβώς μπροστά σας.
- Σενάριο Φ2: Βρίσκετε μια αράχνη μεσαίου μεγέθους στο πάτωμα, ακριβώς μπροστά σας.
- Σενάριο Φ3: Βρίσκετε μια μικρή αράχνη στο πάτωμα, ακριβώς μπροστά σας.
- Σενάριο Φ4: Βλέπετε μια μεγάλη αράχνη κάτω-χαμηλό σε ένα αντικείμενο.
- Σενάριο Φ5: Βλέπετε μια μεσαίου μεγέθους αράχνη σε ένα αντικείμενο.
- Σενάριο Φ6: Βλέπετε μια μικρή αράχνη χαμηλά κάτω σε ένα αντικείμενο.
- Σενάριο Φ7: Βρίσκετε μια μεγάλη αράχνη στο ύψος της μέσης.
- Σενάριο Φ8: Βρίσκετε μια μεσαία αράχνη στο ύψος της μέσης.
- Σενάριο Φ9: Βρίσκετε μια μικρή αράχνη στο ύψος της μέσης.
- Σενάριο Φ10: Βρίσκετε μια μεγάλη αράχνη στον τοίχο.
- Σενάριο Φ11: Βρίσκετε μια μεσαίου μεγέθους αράχνη στον τοίχο.
- Σενάριο Φ12: Βρίσκετε μια μικρή αράχνη στον τοίχο.

βήμα 2: Για κάθε κατάσταση, ο επιβλέπων καταγράφει όλες τις πιθανές λύσεις που μπορεί να επιλέξει ο ασθενής, ώστε να φέρει τον εαυτό του σε ασφαλή κατάσταση.

- Λύση Λ1,1: Πάρτε ένα κομμάτι χαρτί, αρπάξτε την αράχνη και ρίξτε το έξω.
- Λύση Λ1,2: Σηκώστε και αφήστε το.
- Λύση Λ1,3: Σκοτώστε το με το παπούτσι σας.
- Λύση Λ2,1: Πάρτε ένα κομμάτι χαρτί, αρπάξτε την αράχνη και ρίξτε το έξω.
- Λύση Λ2,2: Σηκώστε και αφήστε το
- Λύση Λ2,3: Σκοτώστε το με το παπούτσι σας.
- .....
- Λύση Λ12,1: Πάρτε ένα κομμάτι χαρτί, πιάστε την αράχνη και ρίξτε το έξω.
- Λύση Λ12,2: Σηκώστε και αφήστε το.

βήμα 3: Ο επιβλέπων καταγράφει τις δράσεις που πρέπει να ακολουθήσει ο ασθενής για κάθε Λύση, έτσι ώστε να επιφέρει το επιθυμητό αποτέλεσμα.

- Δράση δ1,1: Κοιτάξτε γύρω σας και βρείτε ένα κομμάτι χαρτί. Πάρτε το και κατευθυνθείτε προς την αράχνη. Αρπάξτε το και βγείτε από το σπίτι.
- Δράση δ1,2: Σηκωθείτε και κατευθυνθείτε προς την αντίθετη κατεύθυνση από εκεί που βρίσκεται η αράχνη.
- .....
- Δράση δ12,1: Κοιτάξτε γύρω σας και βρείτε ένα κομμάτι χαρτί. Πάρτε το και κατευθυνθείτε προς την αράχνη. Αρπάξτε το και βγείτε από το σπίτι.
- Δράση δ12,2: Σηκωθείτε και κατευθυνθείτε προς την αντίθετη κατεύθυνση από εκεί που βρίσκεται η αράχνη.

βήμα 4: Ο επιβλέπων προσομοιώνει καθεμία από τις παραπάνω καταστάσεις σε ένα σύστημα εικονικής πραγματικότητας. Ο στόχος είναι ο ασθενής να εκτελέσει τις διάφορες δράσεις, προκειμένου να επιτύχει την αντίστοιχη λύση. Η κάμερα βάθους χρησιμοποιείται για την παρακολούθηση των κινήσεων του ασθενούς και την επαλήθευση της ορθής εκτέλεσης των ενεργειών.

βήμα 5: Αυτή η μέθοδος θεραπείας ολοκληρώνεται όταν ο ασθενής εκτελεί με επιτυχία την προσομοίωση σε κάθε σενάριο και καταφέρνει να επιφέρει τις αντίστοιχες λύσεις χωρίς να αισθάνεται άγχος.

5 Σύνθετο σύστημα (ΑΤ)

5.1 Εισαγωγή:

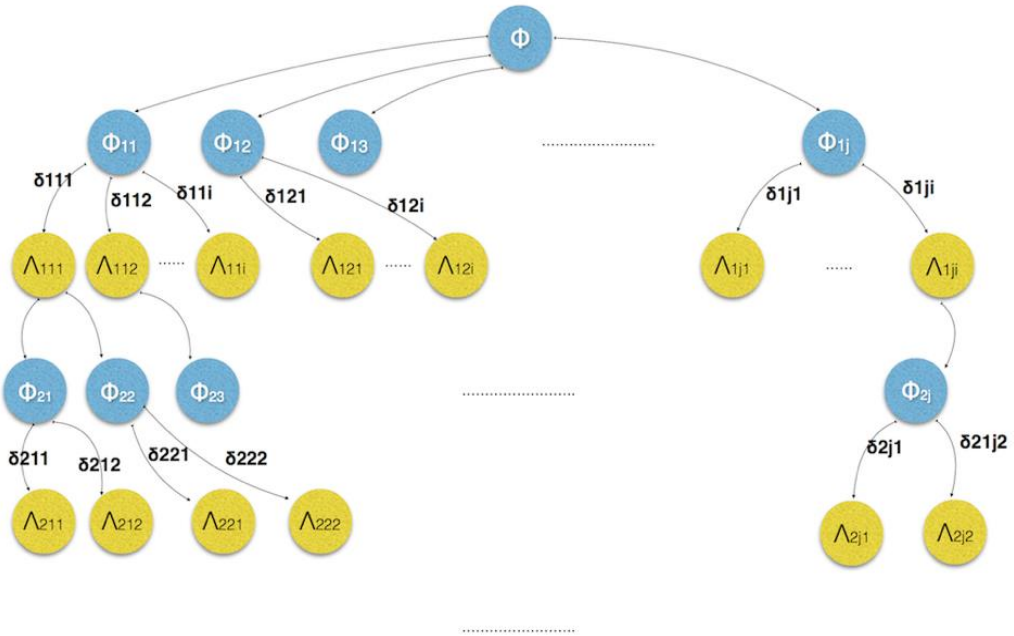
Σε συνέχεια του κεφάλαιού 4, σε αυτό το κεφάλαιο ασχολούμαστε με τον τρόπο που θα γίνει η εκπαίδευση με το σύστημα (ΑΤ). Παρατηρούμε κενά που έχει το Απλό Δέντρο Καταστάσεων και Λύσεων και κατασκευάζουμε ένα πιο ολοκληρωμένο μοντέλο για την αντιμετώπιση των φοβιών με τη χρήση του συστήματος (ΑΤ).

5.2 Μοντέλο - Σύνθετο Δέντρων Καταστάσεων, Λύσεων και Δράσεων:

Στο βασικό δέντρο των καταστάσεων, των λύσεων και των δράσεων, έχουμε κάνει την υπόθεση ότι οι φοβίες που αντιμετωπίζουμε δεν είναι περίπλοκες. Δηλαδή από μια κατάσταση (Φj) ο ασθενής μόλις βρεθεί στη λύση (Λji) η θεραπεία ολοκληρώνεται. Ωστόσο, στην πραγματικότητα, αυτό το σενάριο δεν είναι πολύ λογικό, αφού υπάρχουν περιπτώσεις όπου το άτομο πρέπει να περάσει από μια σειρά φοβικών καταστάσεων και κατ' επέκταση λύσεων, ώστε να ανταπεξέλθει στη φοβία του. Επομένως, αυτή η ακολουθία πλέον προσομοιώνεται από το Σύνθετο Δέντρο Καταστάσεων Λύσεων και Δράσεων.

Ας πάρουμε την φοβία των σκυλιών ως παράδειγμα. Για κάθε μία από αυτές τις πιθανές συμπεριφορές του σκυλιού, θα πρέπει το άτομο να σκεφτεί σειριακά τις αντίστοιχες πιθανές λύσεις. Για παράδειγμα αρχικά εάν ο σκύλος γρυλίσει, το άτομο πρέπει να πάει προς τα πίσω ή αν ο σκύλος κουνάει την ουρά του δεν χρειάζεται να κάνει κάτι. Έπειτα ο ασθενής πρέπει να περιμένει και να δει πώς θα αντιδράσει ο σκύλος, ανάλογα με την αντίδραση του σκυλιού το άτομο πρέπει να πάρει την επόμενη απόφαση. Με τον ίδιο τρόπο ο επιβλέπων πρέπει να προσομοιώνει σειριακά τις καταστάσεις και λύσεις μέχρι να αισθανθεί ότι ο ασθενής έχει ξεπεράσει την φοβία του.

Αναπαράγοντας αυτή τη διαδικασία, ορίζουμε ως «Φij» την εκάστοτε φοβική κατάσταση και «Λkij» ως τις διάφορες «k» λύσεις της «Φij» φοβίας. Για παράδειγμα, το (Φ1,1) είναι «ο σκύλος γαβγίζει», ενώ (Φ1,2) είναι «ο σκύλος κουνάει την ουρά του». Αντίστοιχα, για το (Φ1,1) έχουμε τις λύσεις (Λ1,1,1), (Λ1,1,2) κλπ - που είναι για παράδειγμα «ο ασθενής πρέπει να πάει προς τα πίσω» και «ο ασθενής πρέπει να υπερασπιστεί τον εαυτό του». Ωστόσο, από το (Λ1,1, 1), για παράδειγμα, προκύπτει νέα κατάσταση λόγω του ότι ο σκύλος αντέδρασε π.χ. (Φ2,1) είναι «το σκυλί έρχεται προς τον ασθενή». Αυτή η ακολουθιακή συμπεριφορά συνεχίζεται μέχρις ότου βρεθούμε σε μια Λύση «Λkij» όπου δεν προκύπτει επόμενη φοβική κατάσταση. Τέλος, ορίσαμε «δkij» όλες τις πιθανές ενέργειες που πρέπει να εκτελέσει ο ασθενής, για να προχωρήσει από την εν λόγω κατάσταση στην επιθυμητή λύση. Το ακόλουθο διάγραμμα ονομάζεται Σύνθετο Δέντρο Καταστάσεων, Λύσεων και Δράσεων (Εικόνα 3)



Εικόνα 3

5.3 Περιορισμοί της Βασικού Συστήματος (ΑΤ):

Στο Βασικό Σύστημα (ΑΤ), η χρήση της λέξης «Βασικό» δεν είναι τυχαία, καθώς έχουμε ορισμένους περιορισμούς. Στην πραγματικότητα, ο ανθρώπινος εγκέφαλος δεν λειτουργεί όπως απεικονίζεται στο Βασικό Δέντρο, αλλά όπως φαίνεται στο Σύνθετο Δέντρο. Αυτό σημαίνει ότι πρέπει να κατασκευάσουμε ένα πολύπλοκο σύστημα πιθανών σειριακών γεγονότων. Σε γενικές γραμμές, οι περιορισμοί που έχουμε ορίσει είναι οι εξής:

Α. Διαπιστώσαμε ότι, για κάθε φοβία, πρέπει να βρούμε όλα τα σενάρια - τα οποία, φυσικά, είναι πολύ δύσκολο να επιτύχουμε, αφού μπορεί να είναι αμέτρητα.

Β. Έχουμε διαπιστώσει ότι για κάθε κατάσταση, πρέπει να βρούμε όλες τις πιθανές λύσεις - οι οποίες, όπως και οι καταστάσεις, μπορούν να έρθουν σε πάρα πολλούς συνδυασμούς.

Γ. Το βασικό δέντρο αφορά απλές φοβίες, όπου το άτομο μπορεί να επιτύχει την επιθυμητή λύση σε ένα βήμα, δηλαδή με μια δράση. Αντ' αυτού, στο σύνθετο δένδρο, το άτομο πρέπει να επιτύχει μια σειρά ενεργειών για να φτάσει σε ένα ασφαλές σημείο - το οποίο μπορεί αντιστοιχεί σε ένα πολύ μεγάλο αριθμό πιθανών δράσεων.

Δ. Πολλές από τις δράσεις που καλείται να ολοκληρώσει ο ασθενής είναι όμοιες για διαφορετικές καταστάσεις και λύσεις (π.χ. ο ασθενής μπορεί να επιφέρει την επιθυμητή λύση, σε δύο ή περισσότερα διαφορετικά σενάρια, μέσω της ίδιας δράσης).

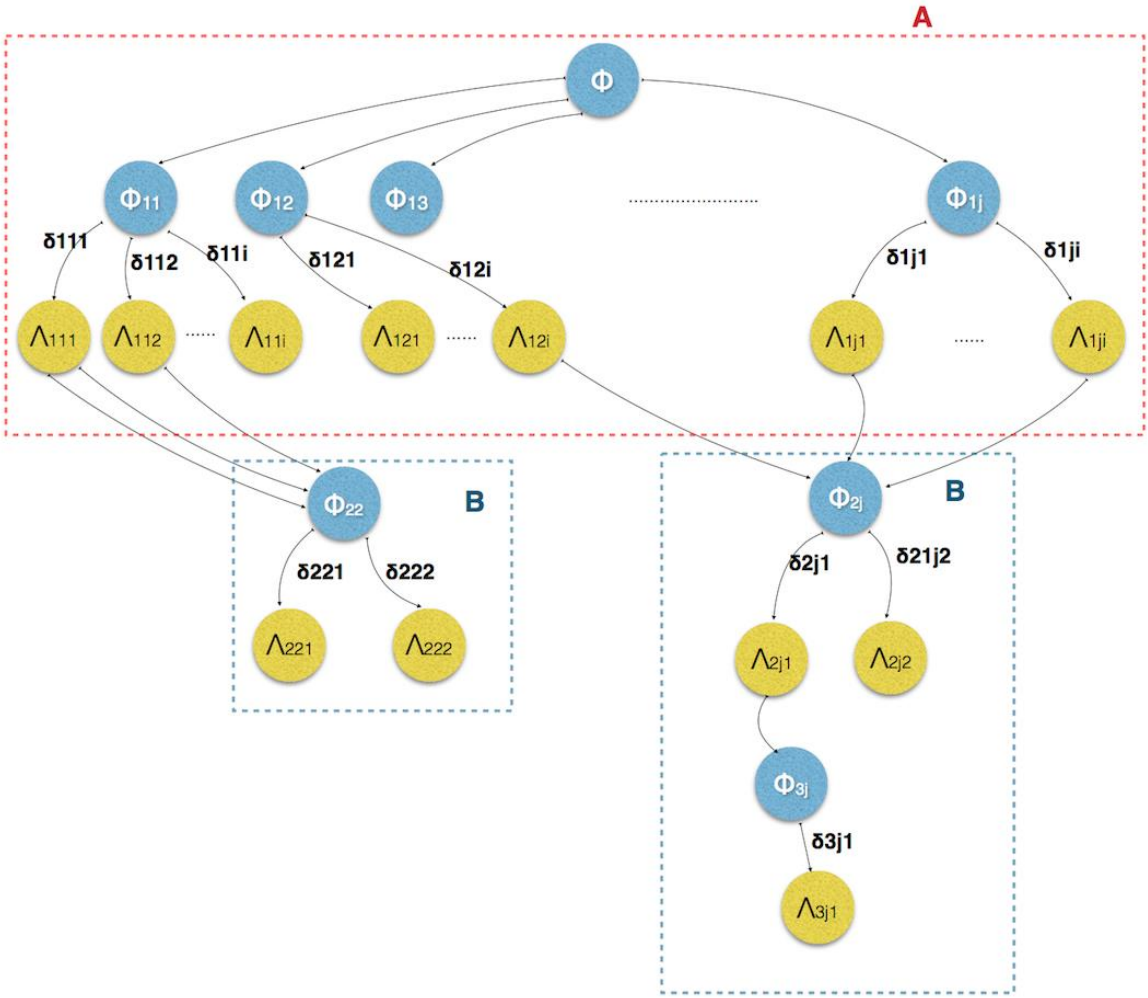
Ε. Τέλος, υποθέσαμε ότι όταν το άτομο ολοκληρώσει μια συγκεκριμένη δράσει, με βεβαιότητα, θα φέρει την επιθυμητή λύση. Αλλά σε ένα ρεαλιστικό περιβάλλον, δεν

μπορεί να εξασφαλιστεί ότι θα επιτευχθεί η επιθυμητή λύση, δηλαδή μπορεί να οδηγηθεί σε αποτυχία.

5.4 Επίλυση περιορισμών:

Παρατηρούμε ότι, θεωρητικά, η Βασικό Σύστημα (ΑΤ) είναι ναι μεν λογικό, αλλά πρακτικά δεν είναι εφαρμόσιμο λόγω της μεγάλης πολυπλοκότητας των περιορισμών (Α), (Β) και (Γ). Ωστόσο, αυτούς τους περιορισμούς μπορούμε να τους άρουμε στις ειδικές φοβίες με τη χρήση της ακόλουθης ιδέας. Οι ειδικές φοβίες από τον ίδιο τον ορισμό τους αφορούν κυρίως σωματικό φόβο, δηλαδή αντικείμενα και καταστάσεις που είναι πιθανόν να προκαλέσουν σωματικό «κακό» στον ασθενή, δηλαδή ανεξάρτητα από το πόσο περίπλοκη και αν είναι η ειδική φοβία στο μυαλό του ασθενούς, όλα τα σενάρια που σκέπτεται θα καταλήξουν σε κάποιες συγκεκριμένες καταστάσεις που θα προκαλέσουν σωματικό κακό. Για παράδειγμα:

- Στην αραχνοφοβία: Οτιδήποτε και αν σκεφτεί το άτομο για τις αράχνες (από το πόσο αηδιαστικές είναι μέχρι το μικροσκοπικό μέγεθος τους), όλα θα καταλήξουν σε κάποιες συγκεκριμένες καταστάσεις που αφορούν σωματικό φόβο όπως να πεταχτεί η αράχνη πάνω στο άτομο.
- Στην αλεξίπτωτοφοβία: Ανεξάρτητα από το τι ισχυρίζεται το άτομο– όπως αν δεν εμπιστεύονται τον πιλότο ή τον εκπαιδευτή του ή το ίδιο το αλεξίπτωτο όλα θα καταλήξουν στο πως θα εξασφαλίσει ότι η πτώση του θα είναι ασφαλής.
- Φόβος για ύψη: Οτιδήποτε και αν σκεφτεί το άτομο για τα ύψη, όλα θα καταλήξουν σε σωματικό φόβο και τι θα κάνει όταν βρεθεί σε μεγάλο υψόμετρο και πως θα αποφύγει την πτώση.



Εικόνα 4

Με βάση τα προηγούμενα παραδείγματα και το παραπάνω διάγραμμα, διαιρούμε το δέντρο σε δύο μέρη (Εικόνα 4). Το Μέρος Α περιέχει όλες τις διαφορετικές καταστάσεις που φοβάται το άτομο, αλλά δεν είναι επιβλαβείς (όπως η υφή των αραχνών στην αραχνοφοβία κλπ). Το Μέρος Β περιέχει τις καταστάσεις που είναι δυνητικά επιβλαβείς για το άτομο (όπως η αράχνη που πέφτει πάνω στο άτομο κ.λπ.). Στην πραγματικότητα, παρατηρούμε ότι, ανεξάρτητα από την πορεία που θα μπορούσε να ακολουθήσει το άτομο στο Μέρος Α, όλα θα οδηγηθούν στο μέρος Β, δηλ. **σε ένα από τα υπό-δέντρα των καταστάσεων, λύσεων και δράσεων.**

Έτσι, ανεξάρτητα από το πόσο μεγάλο και περίπλοκο είναι το Δέντρο των Καταστάσεων, των Λύσεων και των Δράσεων, το Μέρος Α θα οδηγήσει τελικά πάντα στο Μέρος Β - το οποίο φυσικά είναι πολύ μικρότερο. Είναι ενδιαφέρον, ωστόσο, ότι το μέρος Β δεν είναι απλά ένα τυχαίο μέρος, είναι ένα κρίσιμο μέρος της θεραπείας των φοβιών, καθώς σχετίζεται με την ίδια την επιβίωση. Επομένως, εκπαιδεύοντας το άτομο μόνο στο Μέρος Β και αφήνοντας στην άκρη το Μέρος Α, μπορούμε ακόμα να έχουμε τα ίδια αποτελέσματα με την εκπαίδευση του ατόμου σε όλο το Δέντρο.

Από τον περιορισμό (Δ) παρατηρούμε ότι - ως επί το πλείστον - οι ασκήσεις που πρέπει να ολοκληρώσει ο ασθενής είναι όμοιες, έτσι μπορούμε να χωρίσουμε τις καταστάσεις σε ομάδες και να επιλέξουμε μια κατάσταση από κάθε ομάδα ώστε να εκπαιδεύσουμε τον ασθενή.

Επιπλέον, τα υπό-δέντρα διαφέρουν μεταξύ τους ανάλογα με το επίπεδο δυσκολίας τους. Αυτό σημαίνει ότι, παρά το γεγονός ότι οι δράσεις είναι ακριβώς οι ίδιες, μπορεί να απαιτούνται διαφορετικές δεξιότητες για την αποτελεσματική εκτέλεση τους. Με βάση αυτό, θα ξεκινήσουμε την εκπαίδευση του ασθενούς με δράσεις που απαιτούν λιγότερες δεξιότητες και θα προχωρήσουμε σε πιο δύσκολες.

Αυτό που απαιτείται να παρατηρήσουμε στον περιορισμό (Ε) είναι ότι το μεγάλο ζήτημα που τίθεται είναι ο τρόπος με τον οποίο ο ασθενής θα διαχειριστεί την αποτυχία. Προκειμένου να επιλυθεί αυτό το πρόβλημα, πρέπει να καταλάβουμε ότι, ακόμη και αν το άτομο αποτύχει να εκτελέσει την δράση ορθά, χρειάζεται γρήγορα να επανέλθει και να επαναλάβει την εκτέλεση της δράσης ξανά. Αυτό μπορούμε να προσομοιώσουμε ένα σενάριο αποτυχίας μέσω του VR, έτσι ώστε να εκπαιδεύσουμε κατάλληλα τον ασθενή να παραμείνει ήρεμος και να επιχειρήσει εκ νέου την ίδια δράση. Για παράδειγμα, στην αραχνοφοβία, στο τελευταίο επίπεδο δυσκολίας, ξεκινά η επόμενη σειρά προσομοιώσεων - κατά την οποία ο ασθενής εκπαιδεύεται στο πώς να χειριστεί μια αποτυχημένη προσπάθεια.

5.5 Σύνθετο Σύστημα (ΑΤ):

Ανεξάρτητα από το πόσο περίπλοκη είναι η φοβία του ατόμου, με άλλα λόγια, ανεξάρτητα από το πόσο μεγάλο είναι το Δέντρο των Καταστάσεων, Λύσεων και Δράσεων, θα προσομοιώσουμε μόνο αυτά τα υπό-δέντρα που αντιστοιχούν στο κρίσιμο μέρος της επιβίωσης. Αφού ο ασθενής λάβει την κατάλληλη εκπαίδευση και ξεπεράσει με άνεση όλες τις επίφοβες καταστάσεις συνεχίζουμε με τη δεύτερη ομάδα προσομοιώσεων, σχετικά με το πώς το άτομο μπορεί να χειριστεί πιθανή αποτυχία. Εάν το άτομο πετύχει και το δεύτερο στάδιο της θεραπείας, τότε μπορούμε με ασφάλεια να πούμε ότι έχει αποκτήσει τις απαραίτητες δεξιότητες για να διαχειριστεί τη φοβία του.

5.6 Εφαρμογή της προηγμένης θεραπείας δράσης:

Εάν προεκτείνουμε το παράδειγμα της παραγράφου 4.5 ώστε να ικανοποιεί το Σύνθετο Σύστημα (ΑΤ) τότε έχουμε την ακόλουθη διαδικασία:

βήμα 1: Ο επιβλέπων καταγράφει τα πιθανά σενάρια αραχνοφοβίας, καθώς και την πιθανή ακολουθία καταστάσεων και λύσεων:

Σενάρια πρώτου επιπέδου:

- Σενάριο Φ1,1: Βλέπετε μια μεγάλη αράχνη μπροστά σας στο πάτωμα.
- Σενάριο Φ1,2: Βλέπετε μια αράχνη μεσαίου μεγέθους μπροστά σας στο πάτωμα.
- Σενάριο Φ1,3: Βλέπετε μια μικρή αράχνη μπροστά σας στο πάτωμα.
- Σενάριο Φ1,4: Βλέπετε μια μεγάλη αράχνη χαμηλή σε ένα.
- Σενάριο Φ1,5: Βλέπετε αράχνη μεσαίου μεγέθους σε ένα χαμηλό αντικείμενο.
- Σενάριο Φ1,6: Βλέπετε μια μικρή αράχνη σε ένα χαμηλό αντικείμενο.
- Σενάριο Φ1,7: Παρατηρείτε μια μεγάλη αράχνη στο ύψος της μέσης.
- Σενάριο Φ1,8: Παρατηρείτε μια αράχνη μεσαίου μεγέθους στο ύψος της μέσης.
- Σενάριο Φ1,9: Παρατηρείτε μια μικρή αράχνη στο ύψος της μέσης.
- Σενάριο Φ1,10: Παρατηρείτε μια μεγάλη αράχνη στον τοίχο.

● Σενάριο Φ1,11: Παρατηρείτε μια αράχνη μεσαίου μεγέθους στον τοίχο.
● Σενάριο Φ1,12: Παρατηρείτε μια μικρή αράχνη στον τοίχο.

Λύσεις πρώτου επιπέδου:
● Λύση Λ1,1,1: Πάρτε ένα κομμάτι χαρτί, αρπάξτε την αράχνη και ρίξτε το έξω.
● Λύση Λ1,1,2: Σηκώστε και αφήστε την.
● Λύση Λ1,1,3: Σκοτώστε την με το παπούτσι σας.
● Λύση Λ1,2,1: Πάρτε ένα κομμάτι χαρτί, αρπάξτε την αράχνη και ρίξτε την έξω.
● Λύση Λ1,2,2: Σηκώστε και αφήστε
● Λύση Λ1,2,3: Σκοτώστε την με το παπούτσι σας.
● .....
● Λύση Λ1,12,1: Πάρτε ένα κομμάτι χαρτί, αρπάξτε την αράχνη και ρίξτε την έξω.
● Λύση Λ1,12,2: Σηκώστε και αφήστε.

Σενάρια δευτέρου επιπέδου:
● Σενάριο Φ2,1: Μια μεγάλη αράχνη κατευθύνεται προς τον ασθενή με μεγάλη ταχύτητα.
● Σενάριο Φ2,2: Μια μεσαίου μεγέθους αράχνη κατευθύνεται προς τον ασθενή.
● Σενάριο Φ2,3: Μια μικρή αράχνη κατευθύνεται προς τον ασθενή με μεγάλη ταχύτητα.
● Σενάριο Φ2,4: Μια μεγάλη αράχνη κατευθύνεται προς τον ασθενή.
● Σενάριο Φ2,5: Μια μεσαίου μεγέθους αράχνη κατευθύνεται προς την αντίθετη κατεύθυνση του ασθενούς
● Σενάριο Φ2,6: Μια μικρή αράχνη κατευθύνεται προς την αντίθετη κατεύθυνση του ασθενούς.
● Σενάριο Φ2,7: Μια μεγάλη αράχνη προσπαθεί να αναρριχηθεί πάνω στον ασθενή.
● Σενάριο Φ2,8: Μια μεσαίου μεγέθους αράχνη προσπαθεί να αναρριχηθεί πάνω από τον ασθενή.
● Σενάριο Φ2,9: Μια μικρή αράχνη προσπαθεί να αναρριχηθεί πάνω στον ασθενή.
● Σενάριο Φ2,10: Μια μεγάλη αράχνη αναπηδά στο πάτωμα.
● Σενάριο Φ2,11: Μια αράχνη μεσαίου μεγέθους αναπηδά στο πάτωμα.
● Σενάριο Φ2,12: Μια μικρή αράχνη αναπηδά στο πάτωμα.

Λύσεις δευτέρου επιπέδου:
● Λύση Λ2,1,1: Κάντε μερικά βήματα πίσω και περιμένετε μέχρι να μπορέσετε να τραβήξετε την αράχνη με το κομμάτι χαρτί.
● ... ..
● Λύση Λ2,4,1: Σηκώστε το κομμάτι χαρτί, μετακινηθείτε προς την αράχνη και τραβήξτε το.
● ....
● Λύση Λ2,12,1: Μείνετε στάσιμοι και περιμένετε αν θα σταματήσει ή όχι.
Η διαδικασία αυτή συνεχίζεται, έως ότου ο επιβλέπων είναι σίγουρος ότι ο ασθενής έχει λάβει την κατάλληλη εκπαίδευση.

βήμα 2: Από όλα τα παραπάνω σενάρια, θα διατηρήσουμε μόνο αυτά που είναι πιθανόν επιβλαβή για τους ασθενείς. Παρόλο που πολλοί ασθενείς φοβούνται πολλά διαφορετικά σενάρια, αρκετά από αυτά είναι μη ρεαλιστικά και έτσι θα αφαιρεθούν από το πείραμα.

βήμα 3: Ο επιβλέπων παρουσιάζει τις διαφορετικές δράσεις για κάθε κατάσταση και λύση.

- Δράση δ1,1,1: Κοιτάξτε γύρω και βρείτε ένα κομμάτι χαρτί. Πάρτε το και κατευθυνθείτε προς την αράχνη. αρπάξτε την και βγείτε από το σπίτι.
- Δράση δ1,1,2: Σηκώστε το κεφάλι προς την αντίθετη κατεύθυνση από εκεί που βρίσκεται η αράχνη.
- .....
- Δράση δ1,12,1: Κοιτάξτε γύρω και βρείτε ένα κομμάτι χαρτί. Πάρτε το και κατευθυνθείτε προς την αράχνη. αρπάξτε την και βγείτε από το σπίτι.
- Δράση δ1,12,2: Σηκώστε και κατευθυνθείτε προς την αντίθετη κατεύθυνση από εκεί που βρίσκεται η αράχνη.

βήμα 4: Μπορούμε να παρατηρήσουμε από τα παραπάνω παραδείγματα ότι πολλές από τις δράσεις που πρέπει να εκτελέσει ο ασθενής είναι παρόμοιες, και ταυτόχρονα με διαφορετικό επίπεδο δυσκολίας. Για παράδειγμα, στο Φ1,3 η αράχνη είναι μικρότερη, γεγονός που το καθιστά πιο δύσκολο επίπεδο από του Φ1,1. Έτσι τις διάφορες καταστάσεις θα τις διαχωρίσουμε με κριτήριο το επίπεδο δυσκολίας και από κάθε επίπεδο δυσκολίας θα προσομοιώσουμε μόνο μια αντιπροσωπευτική των άλλων. Για παράδειγμα, οι ενέργειες δ1,1,1 και δ1,12,1 θα ανήκουν στην ίδια κατηγορία, ενώ οι δράσεις δ1,1,1 και δ1,1,2 σε διαφορετική. Επίσης, θα αλλάξουμε κάποιες παραμέτρους σε κάθε κατηγορία, προκειμένου να αλλάξουμε τη δυσκολία της δράσης (π.χ. θα ξεκινήσουμε με αργές και μικρές αράχνες και θα προχωρήσουμε σε μεγαλύτερες και ταχύτερες).

βήμα 5: Τέλος, πρέπει να βρούμε και σενάρια αποτυχίας, δηλαδή καταστάσεις στις οποίες, ακόμη και αν το πρόσωπο ακολουθεί τις οδηγίες που παρέχονται για την ολοκλήρωση της δράσης, δεν θα οδηγηθεί στο επιθυμητό αποτέλεσμα - π.χ. η αράχνη διαφεύγει. Όπως έχουμε ήδη αναφέρει, αυτό είναι ένα πολύ σημαντικό βήμα στην εκπαίδευση του ασθενούς, καθώς βοηθά στη διαχείριση της αποτυχίας και στη γρήγορα ανάκαμψη.
βήμα 6: Μόλις συγκεντρώσουμε όλα τα διάφορα φοβικά σενάρια που θέλουμε να προσομοιωθούν, τις διάφορες λύσεις και τα σενάρια αποτυχίας τότε είμαστε έτοιμοι να τα προσομοιώσουμε στο σύστημα VR και να αρχίσουμε την εκπαίδευση του ασθενούς.

βήμα 7: Αυτή η μέθοδος θεραπείας ολοκληρώνεται, όταν ο ασθενής ολοκληρώνει με επιτυχία κάθε προσομοίωση και καταφέρνει να βρεθεί στις εκάστοτε λύσεις χωρίς να αισθάνεται άγχος.

6 Υλοποίηση του Συστήματος (AT)

6.1 Εισαγωγή:

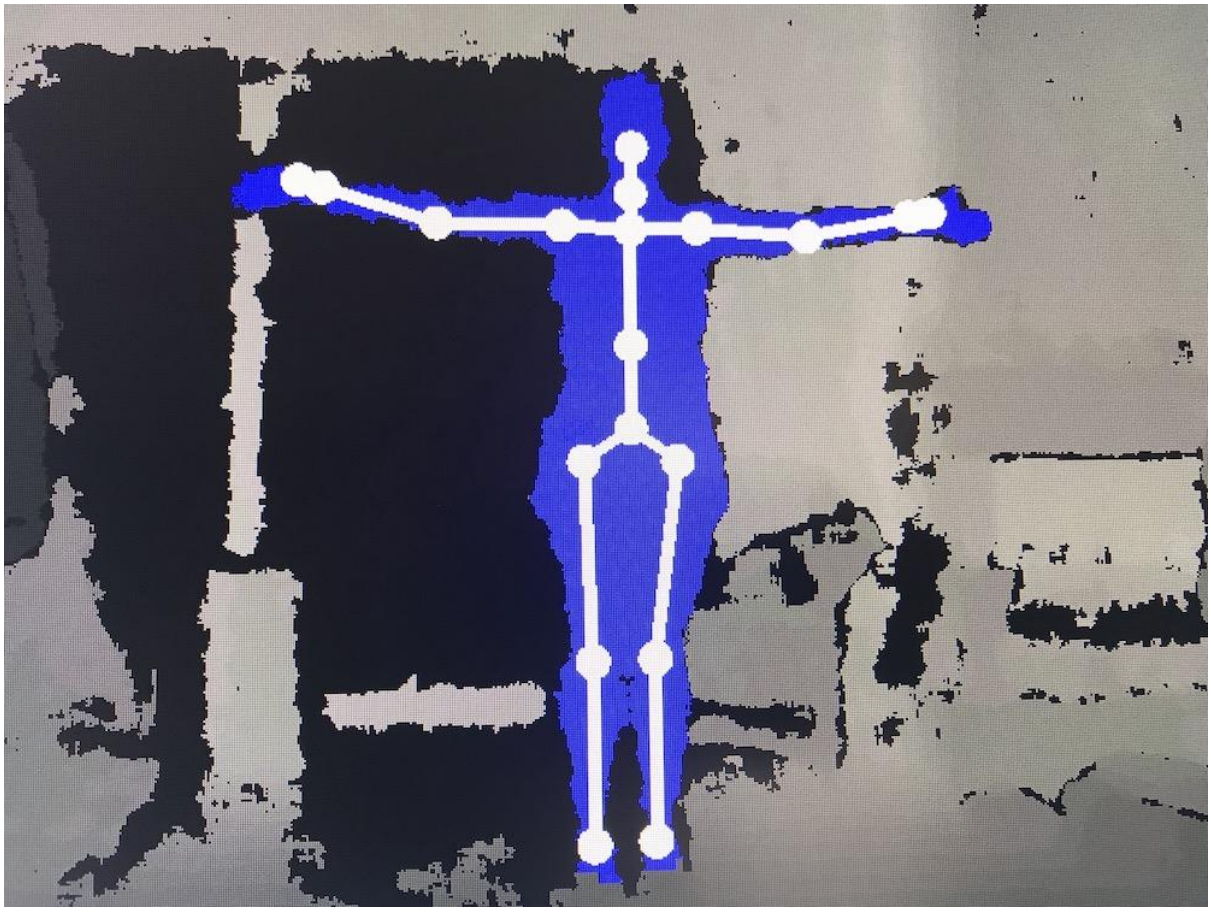
Σε αυτό το μέρος θα παρουσιάσουμε τη ακριβή δομή ενός πρωτότυπου συστήματος που υλοποιεί το (AT) σύστημα. Στα κεφάλαια 3,4,5 αναφέραμε το θεωρητικό μέρος και τις προεκτάσεις που μπορεί να αποκτήσει αυτό το σύστημα. Λόγο του περιορισμού της διπλωματικής εργασίας, σε αυτό το μέρος θα υλοποιήσουμε ένα πολύ συγκεκριμένο σενάριο εκπαίδευσης μια συγκεκριμένης φοβίας σε ένα συγκεκριμένο περιβάλλον.

6.2 Υλικό:

Το υλικό που χρησιμοποιούμε είναι μια (1) κάμερα βάθους, ένα (2) headset VR και ένα (3) κινητό.

(1) Κάμερα βάθους: έχει κατασκευαστεί από την εταιρία «ORBBEC», η ιστοσελίδα της εταιρείας βρίσκεται εδώ (https://orbbec3d.com). Το μοντέλο κάμερας που χρησιμοποιούμε ονομάζετε «Persee», η ιστοσελίδα του μοντέλου βρίσκεται εδώ (https://orbbec3d.com/product-persee/). Το μοντέλο «Persee», αποτελείται από 2 κάμερες, ένα laser, και έναν επεξεργαστή ARM (εικόνα 5). Η κύρια εργασία της κάμερας είναι να αναγνωρίζει τις κινήσεις του ατόμου σε ένα δωμάτιο (εικόνα 6).

Εικόνα 5



Εικόνα 6

(2) Headset VR: είναι από την εταιρεία Samsung η ιστοσελίδα της εταιρείας βρίσκεται εδώ (http://www.samsung.com/global/galaxy/). Το μοντέλο είναι το «Gear VR», η ιστοσελίδα του μοντέλου βρίσκεται (http://www.samsung.com/global/galaxy/gear-vr/) (εικόνα 7).

Εικόνα 7

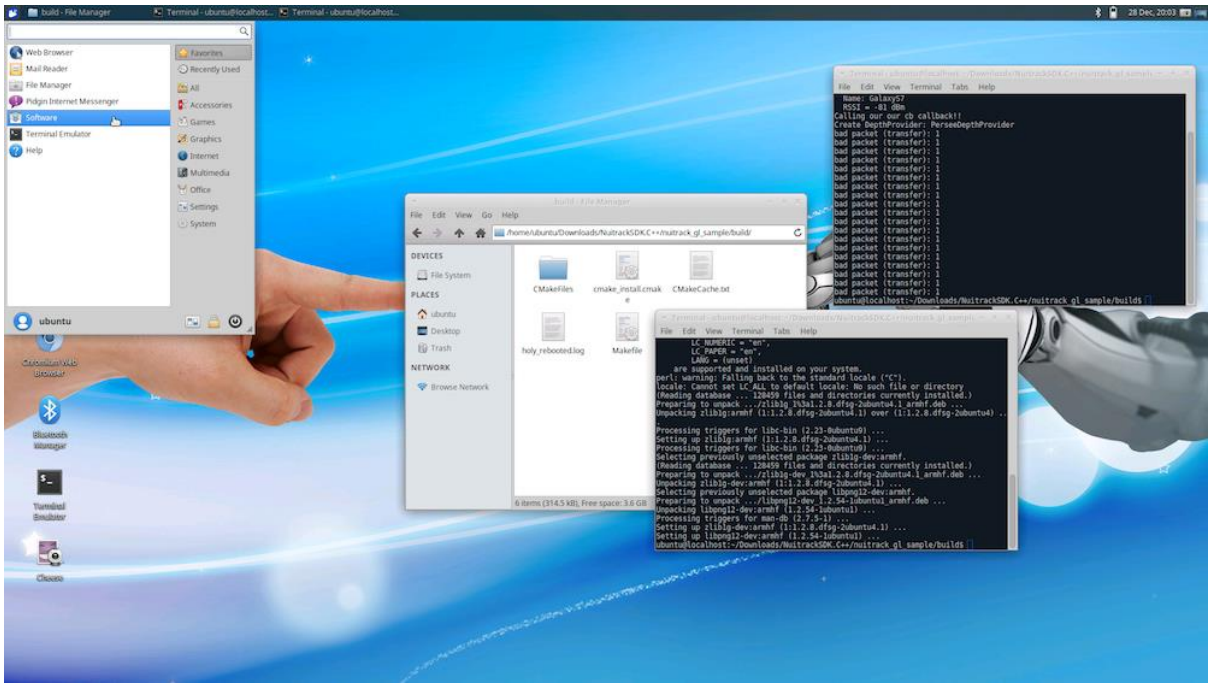
(3) Κινητό: είναι από την εταιρεία Samsung η ιστοσελίδα της εταιρείας βρίσκεται εδώ (http://www.samsung.com/global/galaxy/). Το μοντέλο είναι το «Galaxy S7 Gold» », η ιστοσελίδα του μοντέλου βρίσκεται (Samsung Galaxy S7) (Εικόνα 8).

Εικόνα 8

6.3 Λογισμικό:

Το λογισμικό που χρησιμοποιούμε είναι το
(1) λειτουργικό σύστημα Linux,
(2) λογισμικό Nuitrack,
(3) C++ compiler,
(4) Classic Bluetooth / Low Energy Bluetooth
(5) Android Studio,
(6) Unity,
(7) Blender.

(1) Λειτουργικό Σύστημα Linux: στον επεξεργαστή ARM που έχει η κάμερα τοποθετήσαμε το λογισμικό με κάποιους συγκεκριμένους Drivers για να αναγνωρίζει τους αισθητήρες (Εικόνα 9).

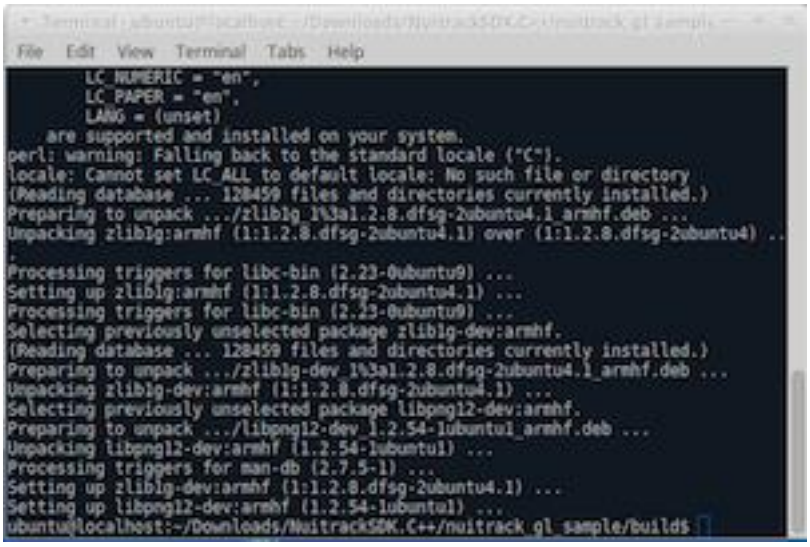


Εικόνα 9

(2) Nuitrack: για την αναγνώριση εικόνας έχουμε στηριχθεί σε κάποιες βασικές συναρτήσεις του λογισμικού Nuitrack, η ιστοσελίδα του λογισμικού βρίσκεται εδώ (https://nuitrack.com) (Εικόνα 10).

Εικόνα 10

(3) C++ compiler: στον ARM επεξεργαστή, το πρόγραμμα μας το έχουμε γράψει σε C++. Το πρόγραμμα αυτό διαχωρίζετε τα δεδομένα από την κάμερα και τα αποστέλλει στο VR (Εικόνα 11).



Εικόνα 11

(4) Bluetooth: για την αποστολή δεδομένων από την κάμερα στο VR χρησιμοποιούμε bluetooth. Στην πρώτη έκδοση του συστήματος χρησιμοποιήσαμε classic Bluetooth. Στην δεύτερη έκδοση του συστήματος χρησιμοποιούμε low energy Bluetooth. Ο λόγος που δεν χρησιμοποιούμε WIFI – και από Classic Bluetooth πήγαμε σε low energy Bluetooth είναι διότι θέλουμε την ελάχιστη καθυστέρηση στα δεδομένα μας. Στο δικό μας σύστημα αυτή τη στιγμή έχουμε καθυστέρηση 7,5 mlsecs το οποίο είναι το καλλίτερο δυνατό που γίνεται (εικόνα 12).

Εικόνα 12

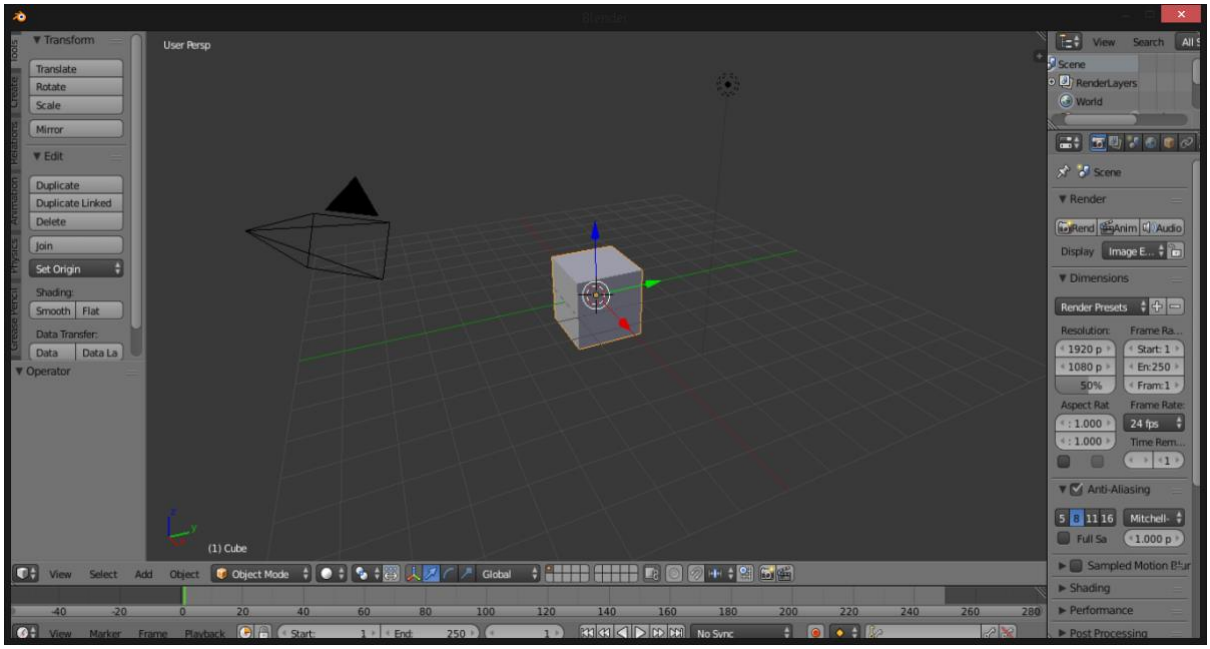
(5) Android: Το εικονικό περιβάλλον του VR και την λήψη δεδομένων από την κάμερα το διαχειρίζεται το Android περιβάλλον (εικόνα 13).

Εικόνα 13

(6) Unity: Το εικονικό περιβάλλον στο οποίο ο ασθενής εκπαιδεύεται το έχουμε υλοποιήσει σε Unity (εικόνα 14).

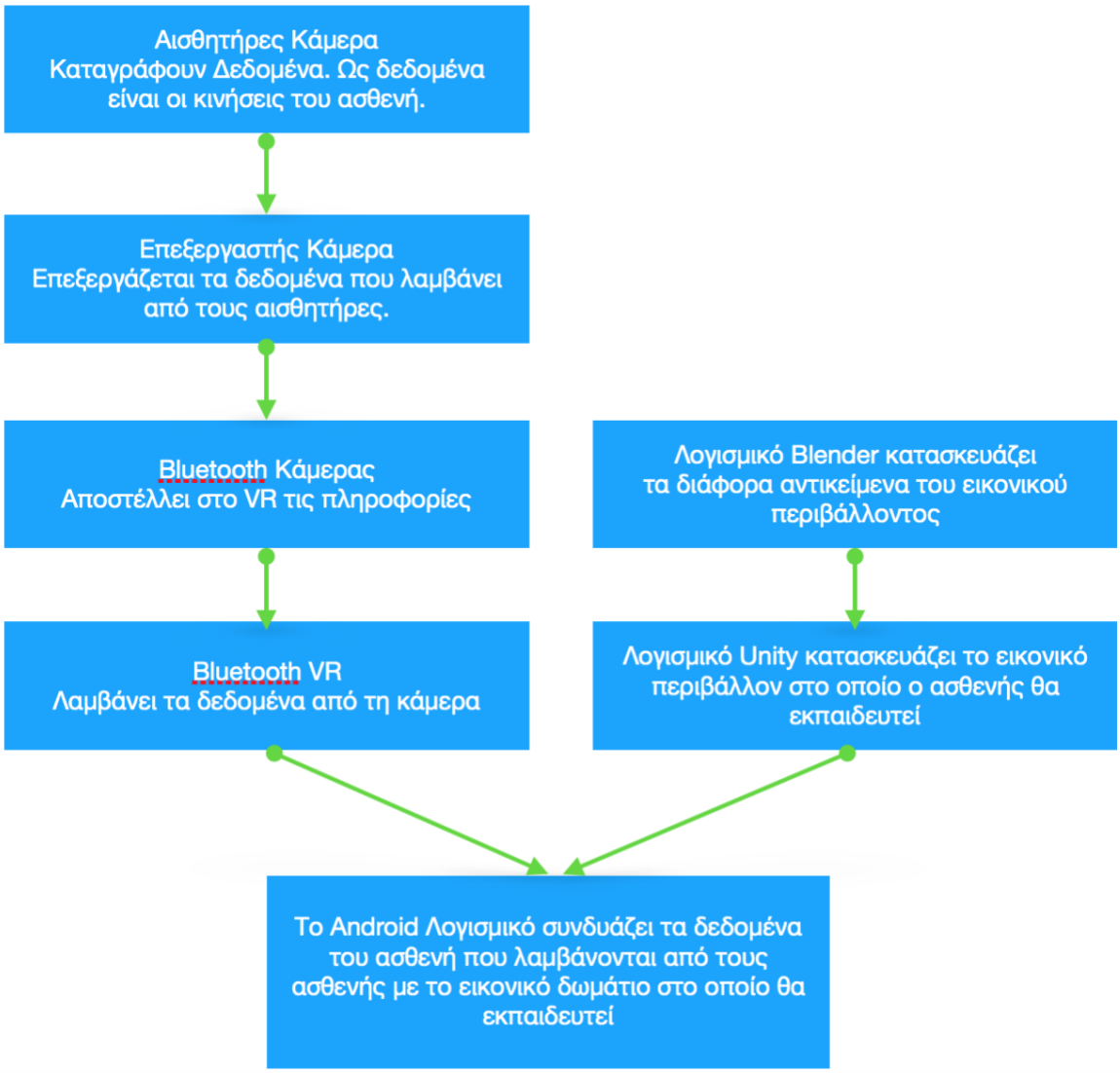
Εικόνα 14

(7) Blender: Τα γραφικά του εικονικού δωματίου, όπως αντικείμενα καναπέδες, φώτα κλπ τα έχουμε υλοποιήσει στο Blender (εικόνα 15).



Εικόνα 15

6.4 Αρχιτεκτονική Συστήματος:



6.5 Σενάριο Υλοποίησης:

Όπως αναφέραμε στην παράγραφο 6.1 το (ΑΤ) σύστημα μπορεί να πάρει μεγάλες διαστάσεις για να υλοποιηθεί στην πλήρη έκταση του. Για τους σκοπούς αυτής της διπλωματικής θα παρουσιάσουμε ένα πολύ συγκεκριμένο σενάριο εκπαίδευσης του ασθενούς. Το σενάριο είναι το ακόλουθο:

1. Σε ένα κενό χωρίο 15 τετραγωνικών μέτρων με διαγραμμίσεις τοποθετούμε την κάμερα βάθους σε απόσταση 1^{ος} μέτρου (Εικόνα 16) (Εικόνα 17) .

Εικόνα 16

Εικόνα 17

2. Ο ασθενής τοποθετείται στο κέντρο του χώρου στην ίδια ευθεία με την κάμερα βάθους (Εικόνα 18).

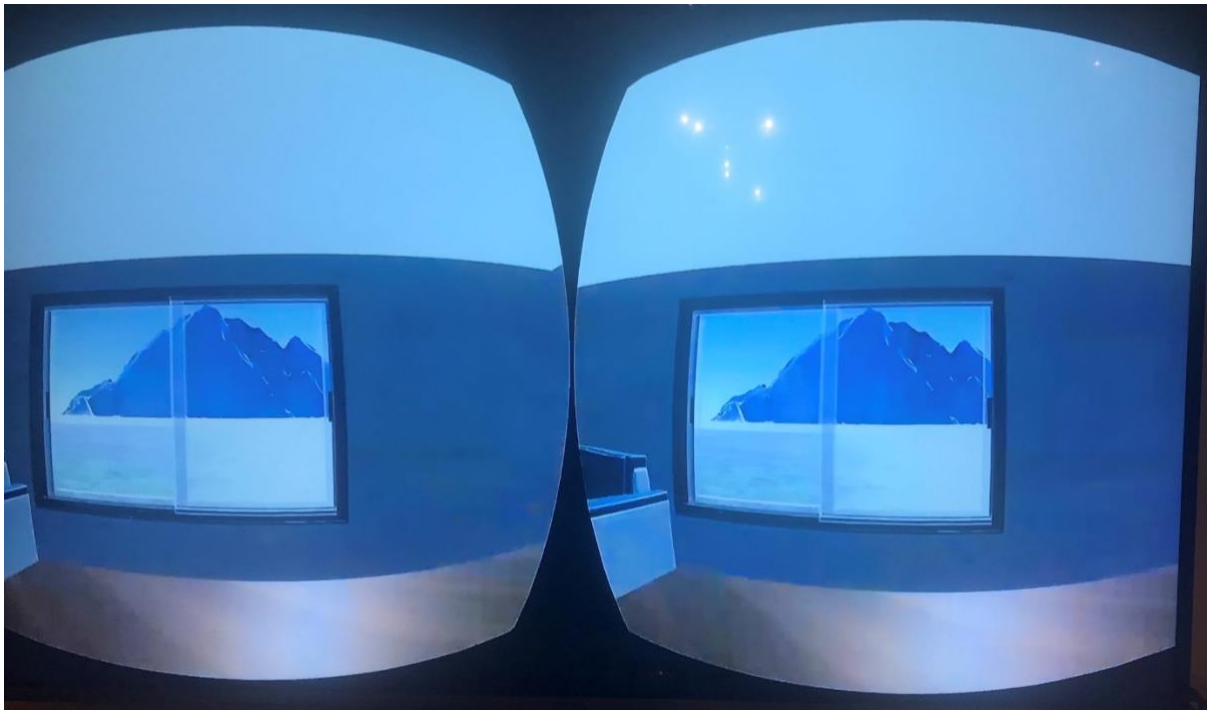
Εικόνα 18

3. Βάζει τα γυαλιά εικονικής πραγματικότητας (Εικόνα 19).

Εικόνα 19

4. Ο ασθενής πλέον βρίσκεται στο εικονικό περιβάλλον, σε ένα δωμάτιο (Εικόνα 20) (Εικόνα 21) (Εικόνα 22).

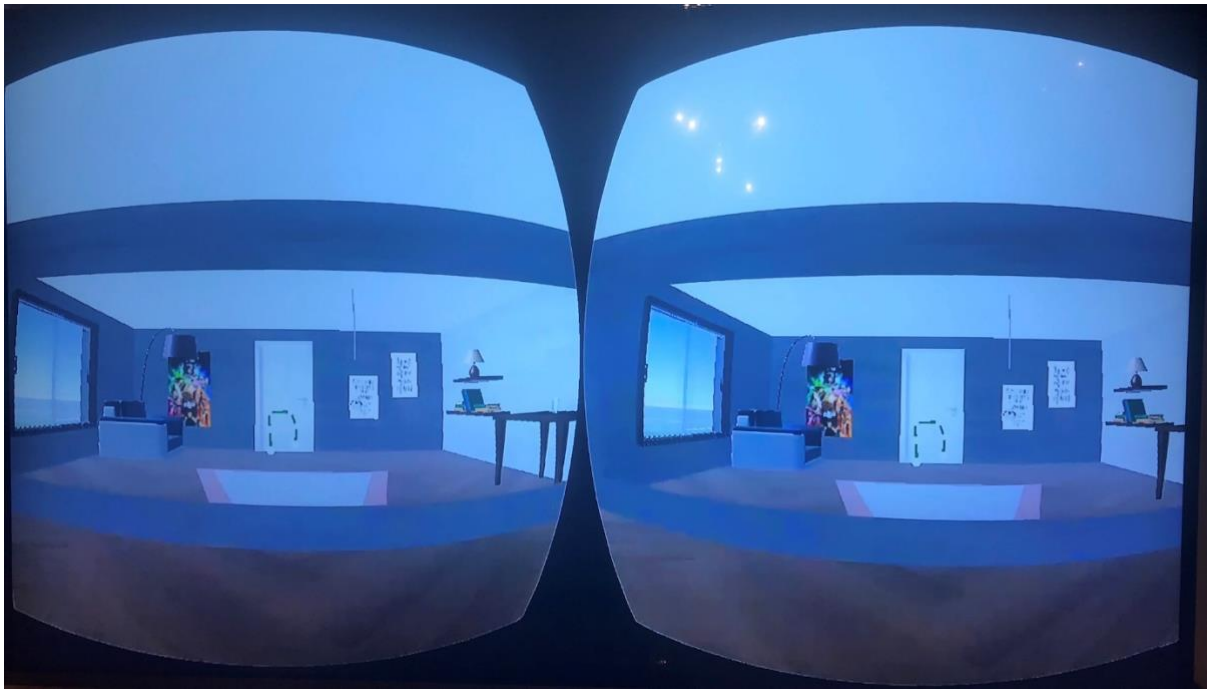
Εικόνα 20



Εικόνα 21

Εικόνα 22

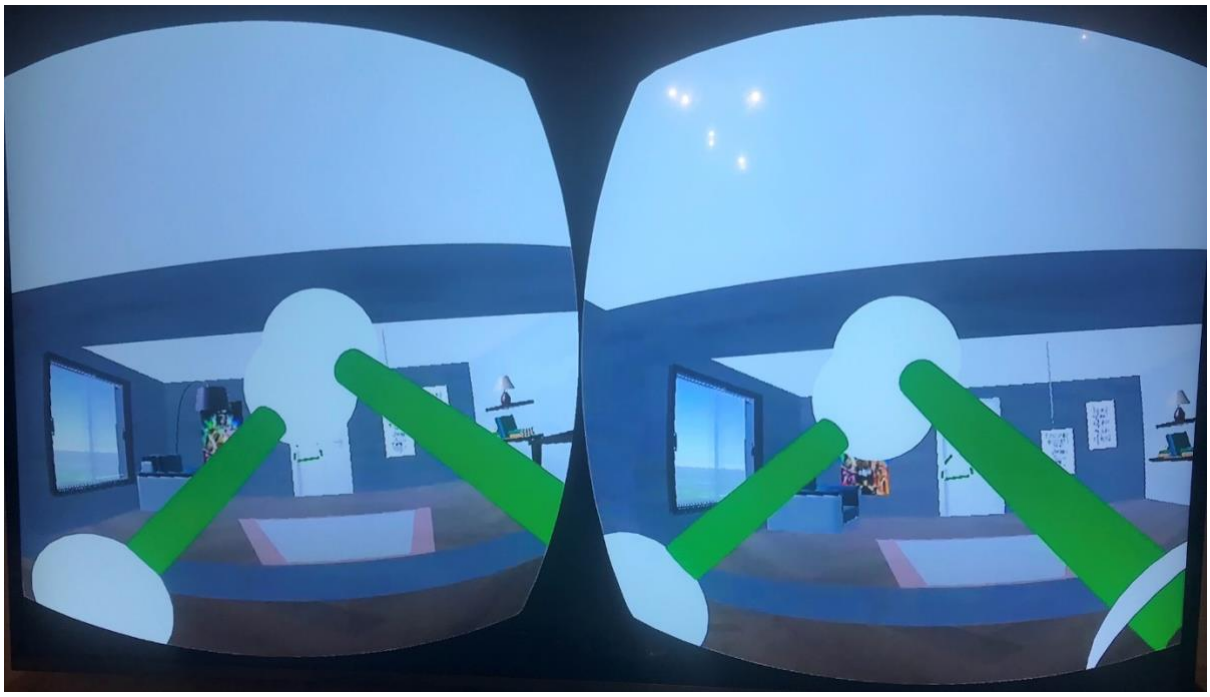
5. Ο ασθενής βλέπει μπροστά του έναν καθρέφτη. Στον καθρέφτη βλέπει επίσης το σκελετό του (Εικόνα 23) (Εικόνα 24).



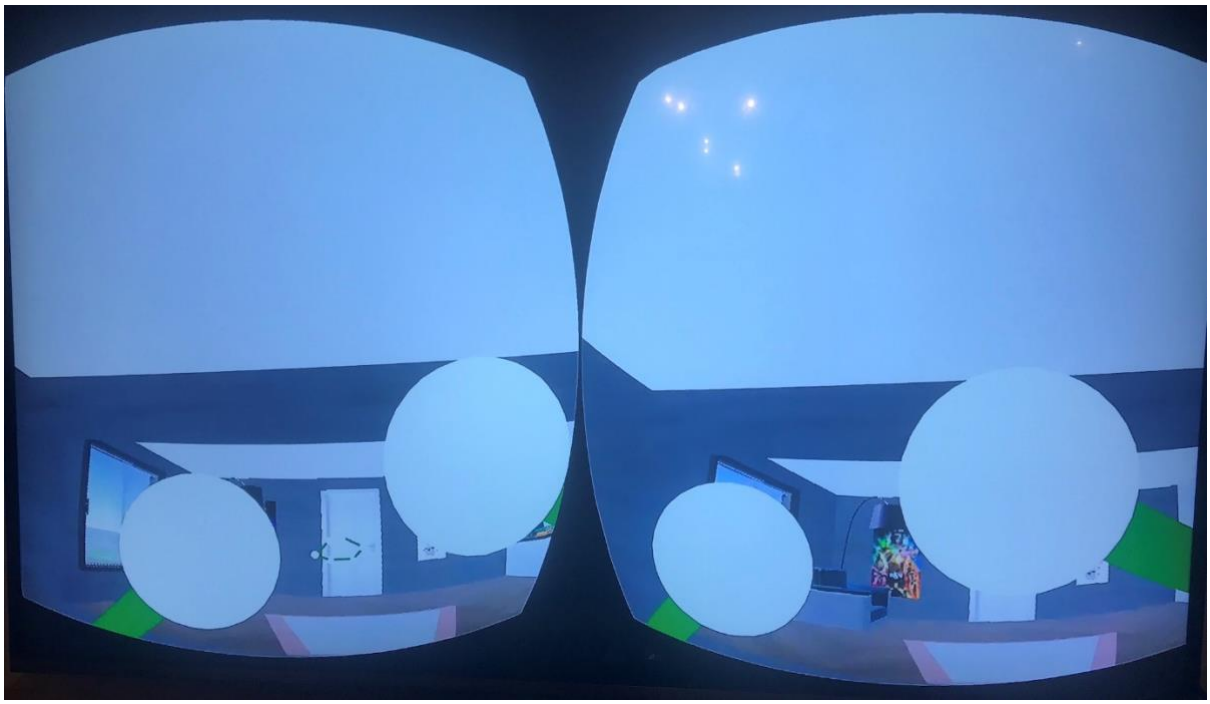
Εικόνα 23

Εικόνα 24

6. Ο ασθενής μπορεί να περπατάει στο εικονικό περιβάλλον κανονικά και να αγγίζει όσα αντικείμενα μπορεί (Εικόνα 25) (Εικόνα 26).

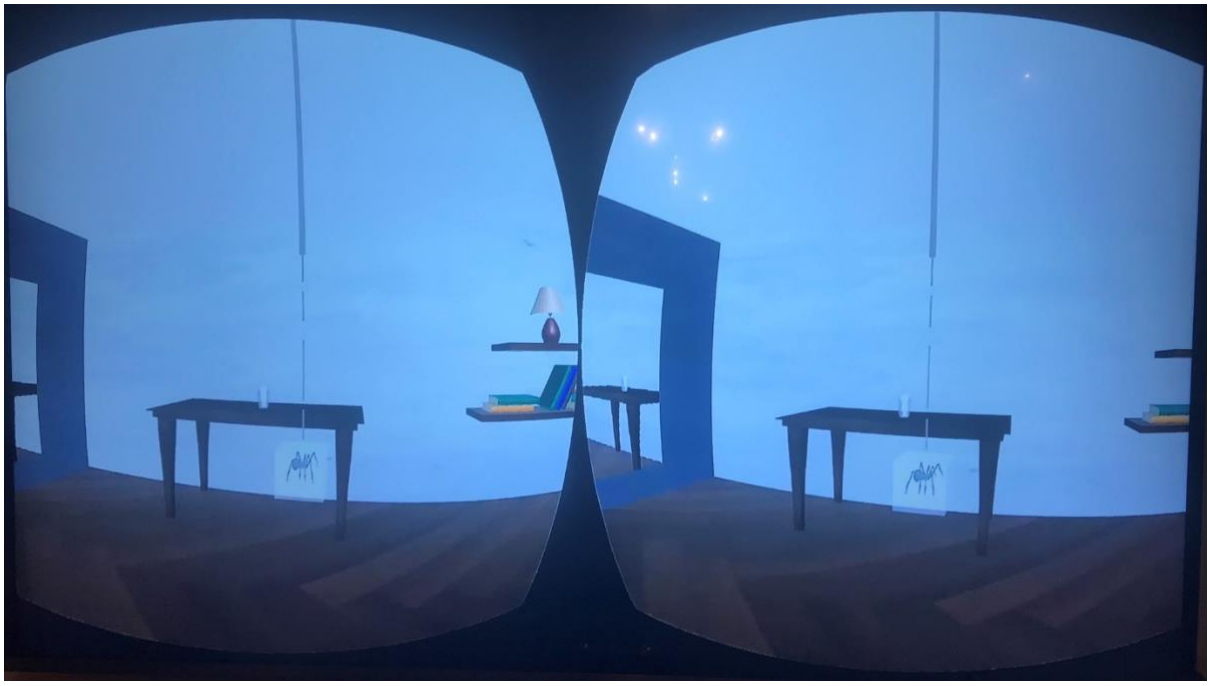


Εικόνα 25

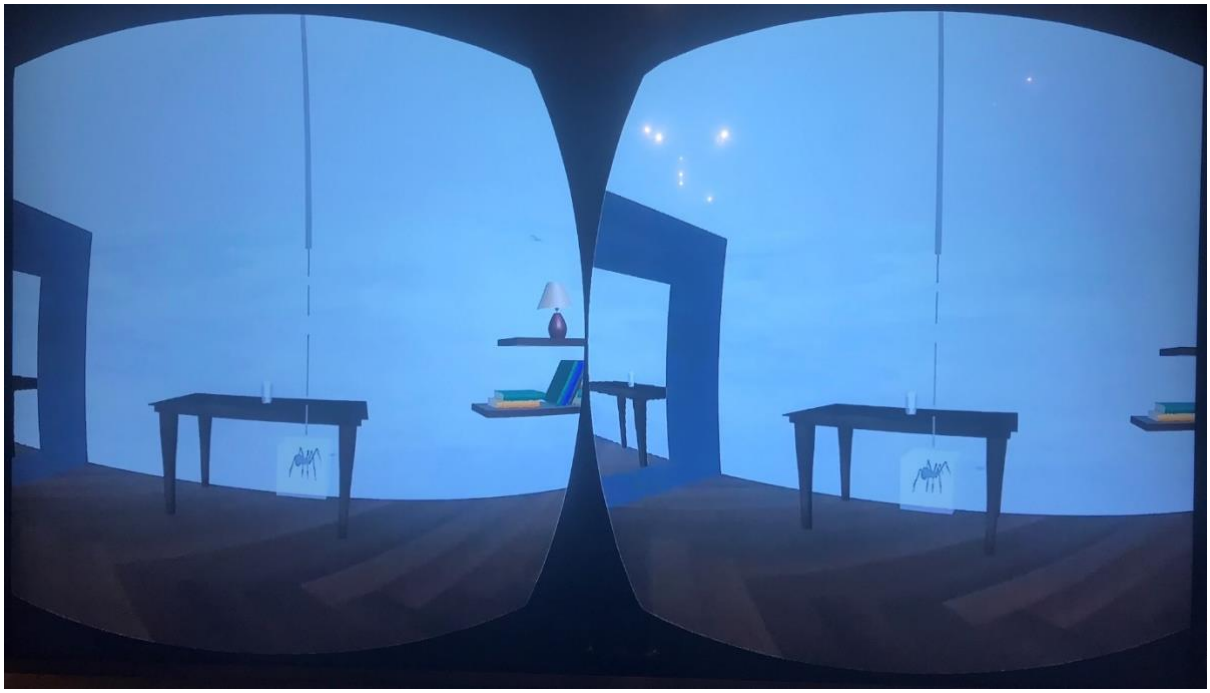


Εικόνα 26

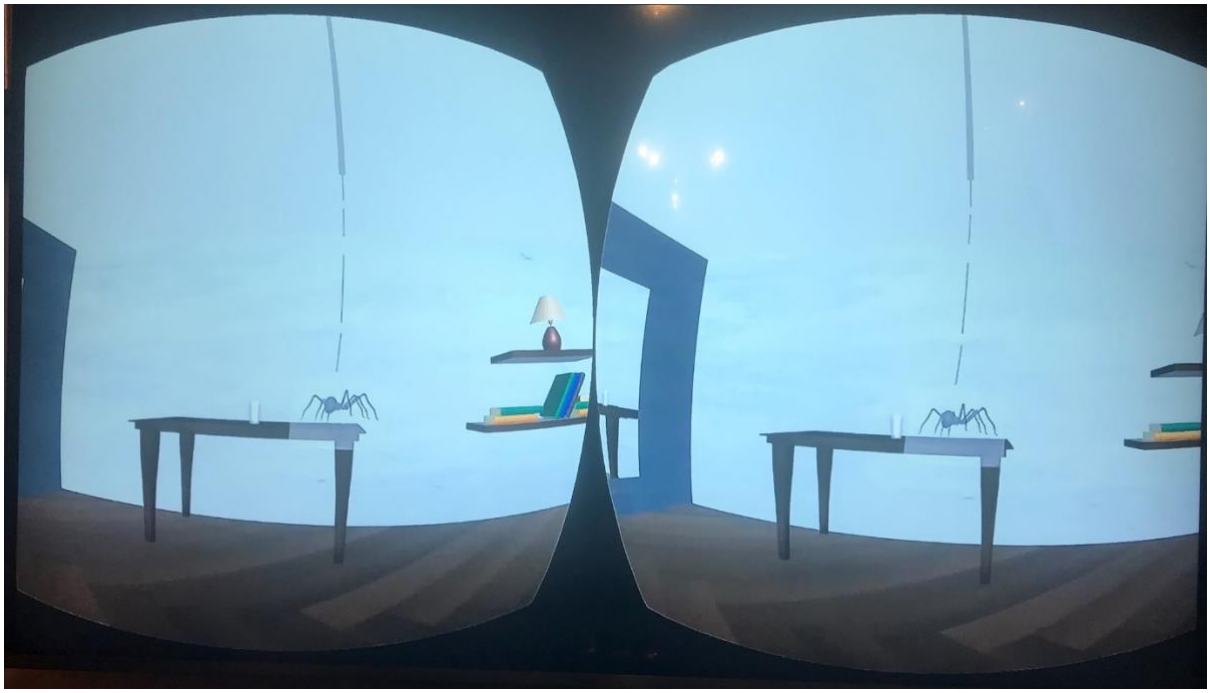
7. Ο ασθενής βλέπει μπροστά μου μια εικονική αράχνη σε ένα κουτί (Εικόνα 27) (Εικόνα 28) (Εικόνα 29).



Εικόνα 27

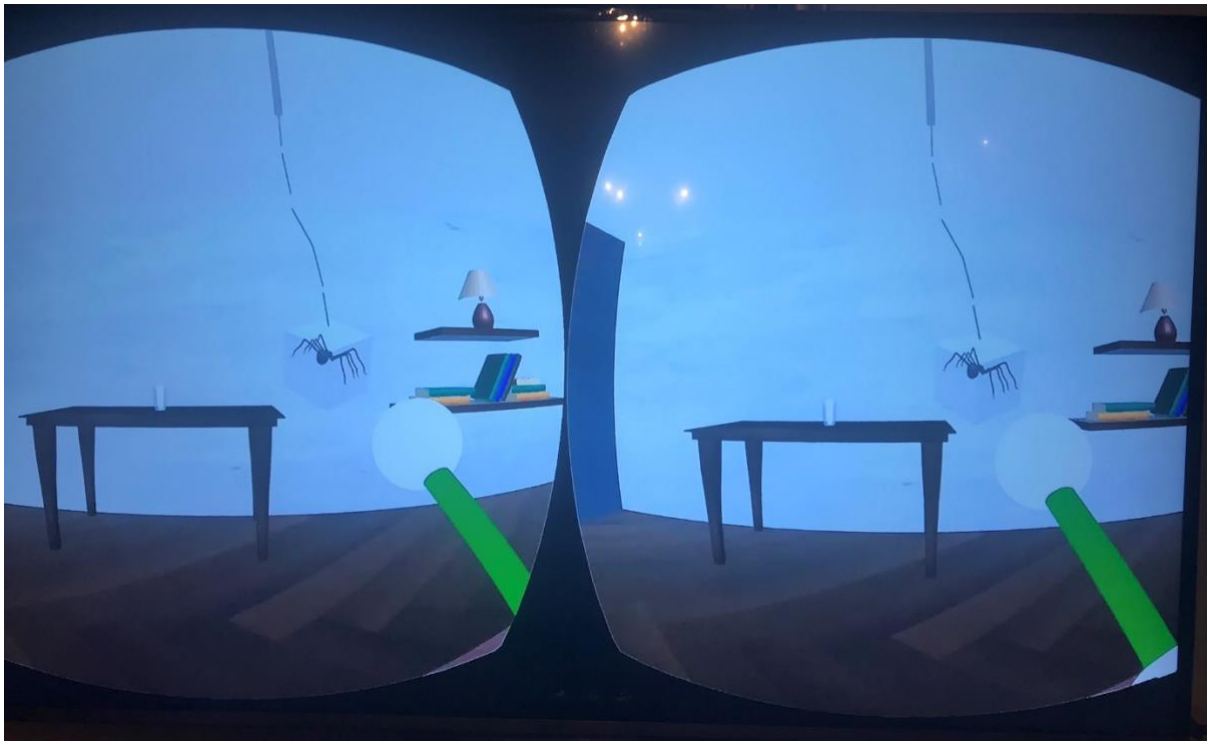


Εικόνα 28



Εικόνα 29

8. Στόχος της εκπαίδευσης που του έχουμε βάλει είναι να μπορεί να χτυπήσει την αράχνη (Εικόνα 30) (Εικόνα 31).



Εικόνα 30

Εικόνα 31

9. Αν καταφέρει να την χτυπήσει η εκπαίδευση ολοκληρώθηκε (Εικόνα 32)

Εικόνα 32

7 Συμπεράσματα

7.1 Σενάριο Υλοποίησης:

Το σύστημα (ΑΤ) μπορεί να προσομοιώσει οποιαδήποτε εκπαιδευτική αλληλουχία. Η επιλογή των ειδικών φοβιών ως δοκιμαστικής νόσου βασίστηκε στον υψηλό επιπολασμό της. Σύμφωνα με το DSM -5 [2] 1 στους 5 ανθρώπους πάσχει από ειδική φοβία.

Αυτή η πλατφόρμα δοκιμάστηκε στην Ψυχιατρική Κλινική (Πανεπιστημιακή) του Γενικού Ογκολογικού Νοσοκομείου Κηφισιάς κατόπιν έγκρισης σχετικού αιτήματός μας.. Είκοσι μέλη του προσωπικού - συμπεριλαμβανομένων των ιατρών, των καθηγητών καιτου νοσηλευτικού προσωπικού παραβρέθηκαν στην δοκιμή και τα σχόλια ήταν αρκετά θετικά. Οι συμμετέχοντες ήταν αρχικά ενθουσιασμένοι που ήταν σε θέση να αλληλοεπιδράσουν με το εικονικό περιβάλλον του δωματίου σαν να ήταν πραγματικό. Επιπλέον, οι ασκήσεις που τους τέθηκαν να εκτελέσουν τους φαινόνταν διασκεδαστικές. Αυτό που ήταν ιδιαίτερα ενδιαφέρον ήταν το πώς οι συμμετέχοντες παρέμειναν απολύτως ήρεμοι κατά το πρώτο επίπεδο - όπου ο ασθενής δεν επιτρέπεται να αλληλοεπιδράσει με την αράχνη. Αντίθετα, κατά τη διάρκεια του δεύτερου επιπέδου - όπου ο ασθενής αναμένεται να αλληλοεπιδράσει με την αράχνη - οι συμμετέχοντες ισχυρίστηκαν ότι ένιωσαν φόβο για τη δυνατότητα αυτή. Από τα προαναφερθέντα γεγονότα μπορούμε να συμπεράνουμε ότι η εικονική αλληλεπίδραση με ένα αντικείμενο που προκαλεί φόβο μπορεί να είναι πολύ πιο εντυπωσιακή και αποτελεσματική από τη συσσώρευση του άγχους του ασθενούς μέσω ζωντανών εικονικών παραστάσεων του ερεθίσματος του φόβου. Ένα άλλο ενδιαφέρον γεγονός ήταν η έλλειψη πολυπλοκότητας κατά την εκτέλεση της εφαρμογής, καθώς δεν χρησιμοποιούμε υπολογιστές. Ως εκ τούτου, η μεταφορά της πλατφόρμας από το ένα δωμάτιο στο άλλο αποδείχθηκε μια γρήγορη και εύκολη διαδικασία. Όπως αναφέρθηκε παραπάνω, η μόνη προϋπόθεση για την δοκιμασίατης εφαρμογής είναι ένας κενός χώρος 15m2. Το μόνο μειονέκτημα που εντοπίσαμε ήταν ένα ορισμένο επίπεδο δυσαρέσκειας από τους συμμετέχοντες όταν τους δόθηκε μόνο ένα λεπτό για να εξοικειωθούν με το εικονικό ή ψηφιακό σώμα τους ώστε να αλληλοεπιδρούν εντός του εικονικού δωματίου. Ενώ αυτό φαινόταν να τους προβληματίζει , , δεν υπήρχαν προβλήματα κατά τη διάρκεια της υπόλοιπης διαδικασίας. Θα επιδιωχθεί περαιτέρω αξιολόγηση από την επιστημονική κοινότητα. Παρατίθενται φωτογραφίες από την δοκιμασία εντός του νοσοκομειακού χώρου (Εικόνα 33) (Εικόνα 34).

Εικόνα 33

Εικόνα 34

Όσον αφορά στο κόστος, υπάρχει μεγάλος όγκος βιβλιογραφίας για τη δημιουργία πιο προσιτών εργαλείων VR για τη θεραπεία των διαταραχών άγχους. Το κόστος μιας εφαρμογής βασισμένης σε επιτραπέζιους υπολογιστές είναι κατά μέσο όρο 3.000 δολάρια. Αποτελείται από μια επιφάνεια εργασίας και ένα ακουστικό VR - όπως το Oculus Rift - το οποίο συνδέεται με την επιφάνεια εργασίας. Το κόστος μιας εφαρμογής βάσει κινητής τηλεφωνίας κυμαίνεται μεταξύ $ 200 και $ 900, ανάλογα με την ποιότητα του ερεθίσματος του ασθενούς. Ενώ αρκετοί ερευνητές έχουν ασχοληθεί με επιτυχία με το πώς να εφαρμόσουν VR στις θεραπείες άγχους, το κόστος των γυαλιών VR μαζί με ένα μια κάμερα αναγνώρισης κινήσεων μπορεί ακόμα να είναι ασαφές. Εκτιμάται ότι κυμαίνεται από $ 500 έως $ 2.500, ανάλογα με την ποιότητα της εκπαίδευσης και τις κινήσεις και τις χειρονομίες που χρειάζονται. Ωστόσο, το κόστος της συγκεκριμένης ερευνητικής προσπάθειας ανέρχεται σε μόλις $ 550 αφού δεν χρησιμοποιήθηκε υπολογιστής απλώς ένας επεξεργαστής ARM.

Το ουσιαστικό μέρος της διπλωματικής είναι η προσπάθεια να φτιάξουμε ένα σύστημα που με τον καλύτερο δυνατό τρόπο να προσομοιώνει διάφορες καταστάσεις. Μέχρι σήμερα όλες οι έρευνες έχουν εστιάσει στο να καταφέρουν ρεαλιστική προσομοίωση με όσο το δυνατόν καλύτερη εικόνα, δηλαδή καλύτερο «output», και κατ' επέκταση καλύτερα VR. Η σωστή προσομοίωση επιτυγχάνεται όταν το άτομο αισθάνεται ότι μπορεί να αλληλοεπιδρά με την κατάσταση που θα βρεθεί, δηλαδή με καλύτερο «input», με τη χρήση αισθητήρων. Πέρα από την κάμερα βάθους που χρησιμοποιούμε θα μπορούσαμε να καταφέρουμε το ίδιο αποτέλεσμα με τη χρήση αισθητήρων γυροσκοπίου πάνω στο σώμα του ατόμου, αλλά αυτό θα οδηγούσε σε πολύ μεγαλύτερη αύξηση του κόστους του συστήματος. Κατά συνέπεια το (AT) σύστημα μπορεί να προσομοιώσει οποιαδήποτε κατάσταση αποφασίσει ο επιβλέπων ιατρός και ο ασθενής να νιώθει ότι υπάρχει πραγματικά σε αυτή ακόμα και αν όλα είναι εικονικά.

8 Κώδικας

8.1 Εισαγωγή:

Το σύστημά μας αποτελείται από 2 συσκευές, ένα κινητό τηλέφωνο (Samsung Galaxy S7) και έναν υπολογιστή ARM με κάμερα 3D (Orbbec Persee). Για να το δοκιμάσετε πρέπει να έχετε κάποια κομμάτια εξειδικευμένου υλικού:

1. Samsung Galaxy S7
2. Orbbec Persse
3. GearVR Oculus

8.1.1 Orbbec Persee:

1. Η συσκευή θα πρέπει να τοποθετείται σε ύψος περίπου 70cm. Επίσης, πρέπει να έχετε ελεύθερο χώρο 5x3 μέτρων.

2. Η συσκευή διαθέτει προ-εγκατεστημένο πρόγραμμα Android. Θα πρέπει να τοποθετηθεί το Ubuntu 14.04. Ολοκληρωμένες οδηγίες σχετικά με τη διαδικασία παρέχονται από την εταιρεία. (Μπορείτε να προσθέσετε μια οθόνη με ένα hdmi και πληκτρολόγιο-ποντίκι με usb).

3. Bluetooth library. Στο φάκελο (ble-arm) εκτελέστε:

sudo apt-get install libboost-dev libboost-all-dev libbluetooth-dev build-essential make
sudo make install sudo ldconfig

4. Build & run main program. Στο φάκελο (nuitrack-tracking) εκτελέστε:

sudo apt-get install cmake build-essential
sudo apt-get install freeglut3-dev libgl1-mesa-dri
make
sudo ./nuitrack_gl_sample

8.1.2 Samsung Galaxy S7:

1. Εγκαταστήστε το Unity στον υπολογιστή σας

2. Ανοίξτε το φάκελο 'Unity-VRspider-prototype'

3. Αντικαταστήστε το αρχείο στη διαδρομή ' Unity-VRspider-prototype/Assets/ Plugins/ Android/assets’ με το αρχείο osig της συσκευής σας.

Βιβλιοθήκη Bluetooth
Έχουμε δημιουργήσει μια προσαρμοσμένη συλλογή βιβλιοθηκών Java για να έχουμε πρόσβαση στη χαμηλή ενέργεια του bluetooth του Android. Ο πηγαίος κώδικας περιλαμβάνεται στο φάκελο 'ExampleBluetoothUtil' και βρίσκετε στο ‘Unity-VRspider-prototype/Assets/Plugins/Android/bluetooth-util-debug.aar'.
4. Αλλάξτε το όνομα Bluetooth της συσκευής στο 'GalaxyS7'. (κάνει τη διαδικασία σύνδεσης απλή και είναι χρήσιμη σε αυτή τη φάση)

5. Κάντε Pair το samsung galaxy S7 με το orbbec.

6. Πατήστε το build & run στο unity με το galaxyS7 συνδεδεμένο.

7. Όταν ολοκληρωθεί το build, συνδέστε τη συσκευή σε γάντια gearVR.

8.2 BLE ARM (Bluetooth in Camera):

8.2.1 Bluetooth.cc

```
#include <iostream>
#include <sstream>
#include <iomanip>
#include <blepp/blestatemachine.h>
#include <blepp/float.h>
#include <deque>
#include <sys/time.h>
#include <unistd.h>
#include "cxxgplot.h"  //lolzworthy plotting program
using namespace std;
using namespace BLEPP;

void bin(uint8_t i)
{
      for(int b=7; b >= 0; b--)
            cout << !!(i & (1<<b));

}

//ASCII throbber
string throbber(int i)
{
      string base = " (--------------------)";

      i = i%40;
      if(i >= 20)
            i = 19-(i-20);
      base[i + 2] = 'O';

      return base + string(base.size(), '\b');
}


double get_time_of_day()
{
        struct timeval tv;
        gettimeofday(&tv,NULL);
        return tv.tv_sec+tv.tv_usec * 1e-6;
}
////////////////////////////////////////////////////////////////////////////
/////////
//
// This program demonstrates the use of the library
//
int main(int argc, char **argv)
{
      if(argc != 2 && argc != 3)
      {
            cerr << "Please supply address.\n";
            cerr << "Usage:\n";
            cerr << "prog address [nonblocking]\n";
            exit(1);
      }
```

```
	log_level = Info;


	//This is the interface to the BLW protocol.
	BLEGATTStateMachine gatt;


	//This is a cheap and cheerful plotting system using gnuplot.
	//Ignore this if you don't care about plotting.
	cplot::Plotter plot;
	plot.range = " [ ] [0:] ";
	deque<int> points;

	int count = -1;
	double prev_time = 0;
	float voltage=0;

	////////////////////////////////////////////////////////////////
///////////////
	//
	// This is important! This is an example of a callback which
responds to
	// notifications or indications. Currently, BLEGATTStateMachine
responds
	// automatically to indications. Maybe that will change.
	//
	//Function that reads an indication and formats it for plotting.
	std::function<void(const PDUNotificationOrIndication&)> notify_cb
= [&](const PDUNotificationOrIndication& n)
	{
		if(count == -1)
		{
			prev_time = get_time_of_day();
		}
		count++;

		if(count == 10)
		{
			double t = get_time_of_day();
			cout << 10 / (t-prev_time)  << " packets per
second\n";

			prev_time = t;
			count=0;
		}



		//This particular device sends 16 bit integers.
		//Extract them and both print them in binary and send them
to the plotting program
		const uint8_t* d = n.value().first;
		for(int i=0; i < 7; i++)
		{
			int val = ((0+d[1 + 2*i] *256 + d[0 + 2*i])>>0) ;
			//Format the points and send the results to the
plotting program.
```

```
                points.push_back(val);
                if(points.size() > 300)
                      points.pop_front();
          }

          uint32_t seq = d[14] | (d[15]<<8) | (d[16]<<16) |
(d[17]<<8);
          int16_t bv = d[18] | (d[19] << 8);

          if(bv != -32768)
                voltage = bv / 1000.0;

          //cout << "Hello: "  << dec  << setfill('0') << setw(6) <<
val << dec << " ";
          //bin(d[1]);
          //cout << " ";
          //bin(d[0]);

          //cout << endl;


          plot.newline("line lw 3 lt 1 title \"\"");
          plot.addpts(points);
          ostringstream os;
          os << "set title \"Voltage: " << voltage << " Seq: " << seq
<< "\"";
          plot.add_extra(os.str());

          plot.draw();
     };


     ////////////////////////////////////////////////////////////////
///////////////
     //
     // This is important! This is an example of a callback which is
run when the
     // client characteristic configuration is retreived. Essentially
this is when
     // all the most useful device information has been received and
the device can
     // now be used.
     //
     // At this point you need to search for things to activate. The
code here activates
     // notifications on a device I have. You will need to modify
this!
     //
     // Search for the service and attribute and set up notifications
and the appropriate callback.
     bool enable=true;
     std::function<void()> cb = [&gatt, &notify_cb, &enable](){

          pretty_print_tree(gatt);

          for(auto& service: gatt.primary_services)
                for(auto& characteristic: service.characteristics)
```

```
                        if(service.uuid == UUID("7309203e-349d-4c11-
ac6b-baedd1819764") && characteristic.uuid == UUID("e5f49879-6ee1-479e-
bfec-3d35e13d3b88"))
                        {
                              cout << "woooo\n";
                              characteristic.cb_notify_or_indicate =
notify_cb;

      characteristic.set_notify_and_indicate(enable, false);
                        }
      };

      ////////////////////////////////////////////////////////////////
///////////////
      //
      // This is somewhat important.  Set up callback for disconnection
      //
      // All reasonable errors are handled by a disconnect. The BLE
spec specifies that
      // if the device sends invalid data, then the client must
disconnect.
      //
      // Failure to connect also comes here.
      gatt.cb_disconnected = [](BLEGATTStateMachine::Disconnect d)
      {
            cerr << "Disconnect for reason " <<
BLEGATTStateMachine::get_disconnect_string(d) << endl;
            exit(1);
      };


      ////////////////////////////////////////////////////////////////
///////////////
      //
      // You almost always want to query the tree of things on the
entire device
      // So, there is a function to do this automatically. This is a
helper which
      // sets up all the callbacks necessary to automate the scanning.
You could
      // reduce connection times a little bit by only scanning for soma
attributes.
      gatt.setup_standard_scan(cb);



      ////////////////////////////////////////////////////////////////
///////////////
      //
      // There are two modes, blocking and nonblocking.
      //
      // Blocking is useful for simple commandline programs which just
log a bunch of
      // data from a BLE device.
      //
      // Nonblocking is useful for everything else.
      //
```

```
      // A few errors are handled by exceptions. std::runtime errors
happen if nearly fatal
      // but recoverable-with-effort errors happen, such as a failure
in socket allocation.
      // It is very unlikely you will encounter a runtime error.
      //
      // std::logic_error happens if you abuse the BLEGATTStateMachine.
For example trying
      // to issue a new instruction before the callback indicating the
in progress one has
      // finished has been called. These errors mean the program is
incorrect.
      try
      {
            if(argc >2 && argv[2] == string("nonblocking"))
            {

                  //This is how to use the blocking interface. It is
very simple.
                  gatt.connect_blocking(argv[1]);
                  for(;;)
                  {
                        gatt.read_and_process_next();
                  }
            }
            else
            {
                  //Connect as a non blocking call
                  gatt.connect_nonblocking(argv[1]);



                  //Example of how to use the state machine with
select()
                  //
                  //This just demonstrates the capability and should be
easily
                  //transferrable to poll(), epoll(), libevent and so
on.
                  fd_set write_set, read_set;

                  for(int i=0;;i++)
                  {
                        FD_ZERO(&read_set);
                        FD_ZERO(&write_set);

                        //Reads are always a possibility due to
asynchronus notifications.
                        FD_SET(gatt.socket(), &read_set);

                        //Writes are usually available, so only check
for them when the
                        //state machine wants to write.
                        if(gatt.wait_on_write())
                              FD_SET(gatt.socket(), &write_set);


                        struct timeval tv;
```

```
                        tv.tv_sec = 0;
                        tv.tv_usec = 10000;
                        int result = select(gatt.socket() + 1,
&read_set, &write_set, NULL, & tv);

                        if(FD_ISSET(gatt.socket(), &write_set))
                              gatt.write_and_process_next();

                        if(FD_ISSET(gatt.socket(), &read_set))
                              gatt.read_and_process_next();

                        cout << throbber(i) << flush;
/*
                        if(i % 100 == 0 && gatt.is_idle())
                        {
                              enable = !enable;
                              cb();
                        }*/

                  }
            }
      }
      catch(std::runtime_error e)
      {
            cerr << "Something's stopping bluetooth working: " <<
e.what() << endl;
      }
      catch(std::logic_error e)
      {
            cerr << "Oops, someone fouled up: " << e.what() << endl;
      }

}
```

8.2.2 lescan_simple.cc

```
#include <blepp/lescan.h>

int main()
{
      BLEPP::log_level = BLEPP::LogLevels::Info;
      BLEPP::HCIScanner scanner;
      while (1) {
            std::vector<BLEPP::AdvertisingResponse> ads =
scanner.get_advertisements();
      }
}
```

8.2.3 lescan.cc

```
#include <bluetooth/bluetooth.h>
#include <bluetooth/hci.h>
#include <bluetooth/hci_lib.h>

#include <string.h>
#include <stdio.h>
#include <unistd.h>
#include <cerrno>
#include <array>
#include <iomanip>
#include <vector>
#include <boost/optional.hpp>

#include <signal.h>

#include <stdexcept>

#include <blepp/logging.h>
#include <blepp/pretty_printers.h>
#include <blepp/blestatemachine.h> //for UUID. FIXME mofo
#include <blepp/lescan.h>

using namespace std;
using namespace BLEPP;

void catch_function(int)
{
      cerr << "\nInterrupted!\n";
}

int main(int argc, char** argv)
{
      HCIScanner::ScanType type = HCIScanner::ScanType::Active;
      HCIScanner::FilterDuplicates filter =
HCIScanner::FilterDuplicates::Software;


      int c;
      string help = R"X(-[sHbdhp]:
  -s  software filtering of duplicates (default)
  -H  hardware filtering of duplicates
  -b  both hardware and software filtering
  -d  show duplicates (no filtering)
  -h  show this message
  -p  passive scan
)X";
      while((c=getopt(argc, argv, "sHbdhp")) != -1)
      {
            if(c == 'p')
                  type = HCIScanner::ScanType::Passive;
            else if(c == 's')
                  filter = HCIScanner::FilterDuplicates::Software;
            else if(c == 'H')
                  filter = HCIScanner::FilterDuplicates::Hardware;
            else if(c == 'b')
                  filter = HCIScanner::FilterDuplicates::Both;
```

```
            else if(c == 'd')
                  filter = HCIScanner::FilterDuplicates::Off;
            else if(c == 'h')
            {
                  cout << "Usage: " << argv[0] << " " << help;
                  return 0;
            }
            else
            {
                  cerr << argv[0] << ":  unknown option " << c << endl;
                  return 1;
            }
      }


      log_level = LogLevels::Warning;
      HCIScanner scanner(true, filter, type);

      //Catch the interrupt signal. If the scanner is not
      //cleaned up properly, then it doesn't reset the HCI state.
      signal(SIGINT, catch_function);

      //Something to print to demonstrate the timeout.
      string throbber="/|\\-";

      //hide cursor, to make the throbber look nicer.
      cout << "[?25l" << flush;

      int i=0;
      while (1) {


            //Check to see if there's anything to read from the HCI
            //and wait if there's not.
            struct timeval timeout;
            timeout.tv_sec = 0;
            timeout.tv_usec = 300000;

            fd_set fds;
            FD_ZERO(&fds);
            FD_SET(scanner.get_fd(), &fds);
            int err = select(scanner.get_fd()+1, &fds, NULL, NULL,
&timeout);

            //Interrupted, so quit and clean up properly.
            if(err < 0 && errno == EINTR)
                  break;

            if(FD_ISSET(scanner.get_fd(), &fds))
            {
                  //Only read id there's something to read
                  vector<AdvertisingResponse> ads =
scanner.get_advertisements();

                  for(const auto& ad: ads)
                  {
                        cout << "Found device: " << ad.address << " ";
```

```
                        if(ad.type == LeAdvertisingEventType::ADV_IND)
                              cout << "Connectable undirected" << endl;
                        else if(ad.type ==
LeAdvertisingEventType::ADV_DIRECT_IND)
                              cout << "Connectable directed" << endl;
                        else if(ad.type ==
LeAdvertisingEventType::ADV_SCAN_IND)
                              cout << "Scannable " << endl;
                        else if(ad.type ==
LeAdvertisingEventType::ADV_NONCONN_IND)
                              cout << "Non connectable" << endl;
                        else
                              cout << "Scan response" << endl;
                        for(const auto& uuid: ad.UUIDs)
                              cout << "  Service: " << to_str(uuid) <<
endl;
                        if(ad.local_name)
                              cout << "  Name: " << ad.local_name->name
<< endl;
                        if(ad.rssi == 127)
                              cout << "  RSSI: unavailable" << endl;
                        else if(ad.rssi <= 20)
                              cout << "  RSSI = " << (int) ad.rssi << "
dBm" << endl;
                        else
                              cout << "  RSSI = " <<
to_hex((uint8_t)ad.rssi) << " unknown" << endl;
                  }
            }
            else
                  cout << throbber[i%4] << "\b" << flush;
            i++;
      }

      //show cursor
      cout << "[?25h" << flush;

}
```

8.2.4 read_device_name.cc

```
#include <iostream>
#include <iomanip>
#include <blepp/blestatemachine.h>
#include <blepp/float.h>  //BLE uses the obscure IEEE11073 decimal
exponent floating point values
#include <unistd.h>
#include <chrono>
using namespace std;
using namespace chrono;
using namespace BLEPP;

int main(int argc, char **argv)
{
```

```
	if(argc != 2)
	{
		cerr << "Please supply address.\n";
		cerr << "Usage:\n";
		cerr << "prog <addres>";
		exit(1);
	}

	log_level = Debug;

	BLEGATTStateMachine gatt;

	//This is called when a complete scan of the device is done,
giving
	//all services and characteristics. This one simply searches for
the
	//standardised "temperature" characteristic (aggressively
cheating and not
	//bothering to check if the service is correct) and sets up the
device to
	//send us notifications.
	//
	//This will simply sit there happily connected in blissful
ignorance if there's
	//no temperature characteristic.
	std::function<void()> found_services_and_characteristics_cb =
[&gatt](){
		for(auto& service: gatt.primary_services)
			for(auto& characteristic: service.characteristics)
				if(characteristic.uuid == UUID("2a00"))
				{
					characteristic.cb_read = [&](const
PDUReadResponse& r)
					{
						cout << "Hello, my name is: ";
						auto v = r.value();
						cout << string(v.first, v.second) <<
". You killed my father. repare to die." << endl;
						gatt.close();
					};

					characteristic.read_request();
					goto name_found;
				}

		cerr << "No device name found." << endl;
		gatt.close();
		name_found:
		;
	};

	//This is the simplest way of using a bluetooth device. If you
call this
	//helper function, it will put everything in place to do a
complete scan for
	//services and characteristics when you connect. If you want to
save a small amount
```

```
      //of time on a connect and avoid the complete scan (you are
allowed to cache this
      //information in certain cases), then you can provide your own
callbacks.

      gatt.setup_standard_scan(found_services_and_characteristics_cb);

      //I think this one is reasonably clear?
      gatt.cb_disconnected = [](BLEGATTStateMachine::Disconnect d)
      {
            if(d.reason !=
BLEGATTStateMachine::Disconnect::ConnectionClosed)
            {
                  cerr << "Disconnect for reason " <<
BLEGATTStateMachine::get_disconnect_string(d) << endl;
                  exit(1);
            }
            else
                  exit(0);
      };

      //This is how to use the blocking interface. It is very simple.
You provide the main
      //loop and just hammer on the state machine struct.
      gatt.connect_blocking(argv[1]);
      for(;;)
            gatt.read_and_process_next();

}
```

8.3 Android Bluetooth

8.3.1 BleCommunication.java

```
package com.example.bluetooth_util;

/**
 * Created by manolis on 14/12/2017.
 */

import android.app.Activity;
import android.app.Fragment;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothGatt;
import android.bluetooth.BluetoothGattCharacteristic;
import android.bluetooth.BluetoothGattDescriptor;
import android.bluetooth.BluetoothGattServer;
import android.bluetooth.BluetoothGattServerCallback;
import android.bluetooth.BluetoothManager;
import android.bluetooth.BluetoothProfile;
import android.bluetooth.le.AdvertiseCallback;
import android.bluetooth.le.AdvertiseData;
```

```
import android.bluetooth.le.AdvertiseSettings;
import android.bluetooth.le.BluetoothLeAdvertiser;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.content.pm.PackageManager;
import android.os.Bundle;
import android.os.ParcelUuid;
import android.util.Log;
import android.widget.Toast;

import com.unity3d.player.UnityPlayer;

import java.util.Arrays;
import java.util.HashSet;
import java.util.Set;
import java.util.UUID;

//import com.unity3d.player.UnityPlayer;

public class BleCommunication extends Fragment {

    private static final String TAG =
BleCommunication.class.getSimpleName();

    /* Bluetooth API */
    private BluetoothManager mBluetoothManager;
    private BluetoothGattServer mBluetoothGattServer;
    private BluetoothLeAdvertiser mBluetoothLeAdvertiser;
    /* Collection of notification subscribers */
    private Set<BluetoothDevice> mRegisteredDevices = new HashSet<>();

    private int connection_state = BluetoothProfile.STATE_DISCONNECTED;

    private long timer = 0;
    private long update_time = 0;


    /* ----------------------------------------------------------------
Unity functions ------------------------------------------------------
-- */
    /**
     *  Unity specific properties
     */
    // Singleton instance.
    public static BleCommunication instance;

    // Unity context.
    String gameObjectName;

    /**
     *  Initialize an instance of the android java class in the a unity
C# scipt.
     *
     * @param gameObjectName A gameObjectName from unity.
     */
    public static void instantiate(String gameObjectName){
```

```
        // Instantiate and add to Unity Player Activity.
        instance = new BleCommunication();

        // Store 'GameObject' reference\
        instance.gameObjectName = gameObjectName;


UnityPlayer.currentActivity.getFragmentManager().beginTransaction().add
(instance, BleCommunication.TAG).commit();
    }

    public boolean isConnected(){
        if ( instance!= null && connection_state  ==
BluetoothProfile.STATE_CONNECTED)
            return true;
        else
            return false;
    }

    public boolean isConnecting(){
        if ( instance!= null && connection_state  ==
BluetoothProfile.STATE_CONNECTING)
            return true;
        else
            return false;
    }

    /* ------------------------------------------------------------
functionality functions ---------------------------------------------
---------- */

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        mBluetoothManager = (BluetoothManager)
getActivity().getSystemService(Context.BLUETOOTH_SERVICE);
        BluetoothAdapter mBluetoothAdapter =
mBluetoothManager.getAdapter();

        //Activity activity = UnityPlayer.currentActivity;
        Activity activity = getActivity();

        // We can't continue without proper Bluetooth support
        if (mBluetoothAdapter == null) {
            Toast.makeText(activity, "Bluetooth is not available",
Toast.LENGTH_LONG).show();
            activity.finish();
        }
        if
(!getActivity().getPackageManager().hasSystemFeature(PackageManager.FEA
TURE_BLUETOOTH_LE)) {
            Toast.makeText(activity, "BLE is not supported",
Toast.LENGTH_LONG).show();
            activity.finish();
        }

        // Register for system Bluetooth events
```

```
        IntentFilter filter = new
IntentFilter(BluetoothAdapter.ACTION_STATE_CHANGED);
        getActivity().registerReceiver(mBluetoothReceiver, filter);

        if (!mBluetoothAdapter.isEnabled()) {
            Log.d(TAG, "Bluetooth is currently disabled...enabling");
            mBluetoothAdapter.enable();
        } else {
            Log.d(TAG, "Bluetooth enabled...starting services");
            startAdvertising();
            startServer();
        }
    }




    @Override
    public void onStart() {
        super.onStart();
        // Register for system clock events
        IntentFilter filter = new IntentFilter();
        filter.addAction(Intent.ACTION_TIME_TICK);
        filter.addAction(Intent.ACTION_TIME_CHANGED);
        filter.addAction(Intent.ACTION_TIMEZONE_CHANGED);
        getActivity().registerReceiver(mTimeReceiver, filter);
    }

    @Override
    public void onStop() {
        super.onStop();
        getActivity().unregisterReceiver(mTimeReceiver);
    }

    @Override
    public void onDestroy() {
        super.onDestroy();

        BluetoothAdapter bluetoothAdapter =
mBluetoothManager.getAdapter();
        if (bluetoothAdapter.isEnabled()) {
            stopServer();
            stopAdvertising();
        }

        getActivity().unregisterReceiver(mBluetoothReceiver);
    }

    /**
     * Listens for system time changes and triggers a notification to
     * Bluetooth subscribers.
     */
    private BroadcastReceiver mTimeReceiver = new BroadcastReceiver() {
        @Override
        public void onReceive(Context context, Intent intent) {
            byte adjustReason;
```

```
            switch (intent.getAction()) {
                case Intent.ACTION_TIME_CHANGED:
                    adjustReason = TimeProfile.ADJUST_MANUAL;
                    break;
                default:
                case Intent.ACTION_TIME_TICK:
                    adjustReason = TimeProfile.ADJUST_NONE;
                    break;
            }
        }
    };

    /**
     * Listens for Bluetooth adapter events to enable/disable
     * advertising and server functionality.
     */
    private BroadcastReceiver mBluetoothReceiver = new
BroadcastReceiver() {
        @Override
        public void onReceive(Context context, Intent intent) {
            int state =
intent.getIntExtra(BluetoothAdapter.EXTRA_STATE,
BluetoothAdapter.STATE_OFF);

            switch (state) {
                case BluetoothAdapter.STATE_ON:
                    startAdvertising();
                    startServer();
                    break;
                case BluetoothAdapter.STATE_OFF:
                    stopServer();
                    stopAdvertising();
                    break;
                default:
                    // Do nothing
            }
        }
    };

    /**
     * Begin advertising over Bluetooth that this device is connectable
     * and supports the Current Time Service.
     */
    private void startAdvertising() {
        BluetoothAdapter bluetoothAdapter =
mBluetoothManager.getAdapter();
        mBluetoothLeAdvertiser =
bluetoothAdapter.getBluetoothLeAdvertiser();
        if (mBluetoothLeAdvertiser == null) {
            Log.w(TAG, "Failed to create advertiser");
            return;
        }

        AdvertiseSettings settings = new AdvertiseSettings.Builder()

.setAdvertiseMode(AdvertiseSettings.ADVERTISE_MODE_LOW_LATENCY)
                .setConnectable(true)
                .setTimeout(1)
```

```
.setTxPowerLevel(AdvertiseSettings.ADVERTISE_TX_POWER_MEDIUM)
                .build();

        AdvertiseData data = new AdvertiseData.Builder()
                .setIncludeDeviceName(true)
                .setIncludeTxPowerLevel(true)
                .addServiceUuid(new
ParcelUuid(TimeProfile.TIME_SERVICE))
                .build();

        mBluetoothLeAdvertiser
                .startAdvertising(settings, data, mAdvertiseCallback);
    }

    /**
     * Stop Bluetooth advertisements.
     */
    private void stopAdvertising() {
        if (mBluetoothLeAdvertiser == null) return;

        mBluetoothLeAdvertiser.stopAdvertising(mAdvertiseCallback);
    }

    /**
     * Initialize the GATT server instance with the
services/characteristics
     * from the Time Profile.
     */
    private void startServer() {
        mBluetoothGattServer =
mBluetoothManager.openGattServer(getContext(), mGattServerCallback);
        if (mBluetoothGattServer == null) {
            Log.w(TAG, "Unable to create GATT server");
            return;
        }

        mBluetoothGattServer.clearServices();

mBluetoothGattServer.addService(TimeProfile.createTimeService());
        mBluetoothGattServer.addService(new
android.bluetooth.BluetoothGattService(  UUID.fromString("00001845-
0000-1000-8000-00805f9b34fb"),

android.bluetooth.BluetoothGattService.SERVICE_TYPE_PRIMARY));

    }

    /**
     * Shut down the GATT server.
     */
    private void stopServer() {
        if (mBluetoothGattServer == null) return;

        mBluetoothGattServer.close();
    }

    /**
```

```
     * Callback to receive information about the advertisement process.
     */
    private AdvertiseCallback mAdvertiseCallback = new
AdvertiseCallback() {
        @Override
        public void onStartSuccess(AdvertiseSettings settingsInEffect)
{
            Log.i(TAG, "LE Advertise Started.");
        }

        @Override
        public void onStartFailure(int errorCode) {
            Log.w(TAG, "LE Advertise Failed: "+errorCode);
        }
    };




    /**
     * Callback to handle incoming requests to the GATT server.
     * All read/write requests for characteristics and descriptors are
handled here.
     */
    private BluetoothGattServerCallback mGattServerCallback = new
BluetoothGattServerCallback() {

        @Override
        public void onConnectionStateChange(BluetoothDevice device, int
status, int newState) {
            if (newState == BluetoothProfile.STATE_CONNECTED) {
                connection_state = newState;
                Log.i(TAG, "BluetoothDevice CONNECTED: " + device);
                stopAdvertising();
            } else if (newState == BluetoothProfile.STATE_DISCONNECTED)
{
                startAdvertising();
                connection_state = newState;
                Log.i(TAG, "BluetoothDevice DISCONNECTED: " + device);
                //Remove device from any active subscriptions
                mRegisteredDevices.remove(device);
            } else  {
                Log.i(TAG, "Bluetooth Device: Other connection state
change: " );
            }
        }

        @Override
        public void onCharacteristicWriteRequest(BluetoothDevice
device, int requestId, BluetoothGattCharacteristic characteristic,
                                                 boolean preparedWrite,
boolean responseNeeded, int offset, byte[] value) {
            //super.onCharacteristicWriteRequest(device, requestId,
characteristic, preparedWrite, responseNeeded, offset, value);

//            Log.d(TAG, "request id: "  + requestId + "\noffeset: " +
offset + " \nsize: " + value.length);
//            for(int i =0; i<value.length; i++){
```

```
//                Log.d(TAG, String.format("byte %d - 0x%02X", i,
value[i]));
//            }


//            // time debug
//            long millis = System.currentTimeMillis() % 1000;
//            if (TimeProfile.Head.equals(characteristic.getUuid())) {
//                Log.d(TAG,"Head starts here. Previous nuitrack update
time: " + Long.toString(update_time));
//                update_time = 0;
//            }
//            Log.d(TAG,"Delta time in ms: " + Long.toString(millis -
timer) + " --Time: " + millis);
//            timer = millis;
//            update_time += (millis-timer);




            String string_value = new String(value, 0, value.length);
//          Log.d(TAG, "string attmept!: " + string_value);


            if (TimeProfile.Head.equals(characteristic.getUuid())) {
                string_value = "Head" + string_value;

            }else if
(TimeProfile.lCollar.equals(characteristic.getUuid())) {
                string_value = "lCollar" + string_value;

            }else if
(TimeProfile.rWrist.equals(characteristic.getUuid())) {
                string_value = "rWrist" + string_value;

            }else if
(TimeProfile.rElbow.equals(characteristic.getUuid())) {
                string_value = "rElbow" + string_value;

            }else if
(TimeProfile.rShoulder.equals(characteristic.getUuid())) {
                string_value = "rShoulder" + string_value;

            }else if
(TimeProfile.lWrist.equals(characteristic.getUuid())) {
                string_value = "lWrist" + string_value;

            }else if
(TimeProfile.lElbow.equals(characteristic.getUuid())) {
                string_value = "lElbow" + string_value;

            }else if
(TimeProfile.lShoulder.equals(characteristic.getUuid())) {
                string_value = "lShoulder" + string_value;

            }else if
(TimeProfile.Grabs.equals(characteristic.getUuid())) {
                string_value = "Grabs" + string_value;
```

```
            } else {
                return;
            }

            UnityPlayer.UnitySendMessage(gameObjectName,
                    "receiveMessage",
                    string_value);
            }

        @Override
        public void onCharacteristicReadRequest(BluetoothDevice device,
int requestId, int offset,

BluetoothGattCharacteristic characteristic) {
            long now = System.currentTimeMillis();

            // field is the content of a test message
            byte[] field = {1, 2, 3 , 4, 5, 6, 7, 8, 9, 10};
            if (TimeProfile.Head.equals(characteristic.getUuid())) {

                Log.i(TAG, "Read CurrentTime");
                mBluetoothGattServer.sendResponse(device,
                        requestId,
                        BluetoothGatt.GATT_SUCCESS,
                        0,
                        field);
            } else if
(TimeProfile.rWrist.equals(characteristic.getUuid())) {
                Log.i(TAG, "Read LocalTimeInfo");
                mBluetoothGattServer.sendResponse(device,
                        requestId,
                        BluetoothGatt.GATT_SUCCESS,
                        0,
                        field);
            } else {
                // Invalid characteristic
                Log.w(TAG, "Invalid Characteristic Read: " +
characteristic.getUuid());
                mBluetoothGattServer.sendResponse(device,
                        requestId,
                        BluetoothGatt.GATT_FAILURE,
                        0,
                        null);
            }
        }

        @Override
        public void onDescriptorReadRequest(BluetoothDevice device, int
requestId, int offset,
                                            BluetoothGattDescriptor
descriptor) {
            if (TimeProfile.CLIENT_CONFIG.equals(descriptor.getUuid()))
{
                Log.d(TAG, "Config descriptor read");
                byte[] returnValue;
                if (mRegisteredDevices.contains(device)) {
```

```
                    returnValue =
BluetoothGattDescriptor.ENABLE_NOTIFICATION_VALUE;
                } else {
                    returnValue =
BluetoothGattDescriptor.DISABLE_NOTIFICATION_VALUE;
                }
                mBluetoothGattServer.sendResponse(device,
                        requestId,
                        BluetoothGatt.GATT_SUCCESS,
                        0,
                        returnValue);
            } else {
                Log.w(TAG, "Unknown descriptor read request");
                mBluetoothGattServer.sendResponse(device,
                        requestId,
                        BluetoothGatt.GATT_FAILURE,
                        0,
                        null);
            }
        }

        @Override
        public void onDescriptorWriteRequest(BluetoothDevice device,
int requestId,
                                             BluetoothGattDescriptor
descriptor,
                                             boolean preparedWrite,
boolean responseNeeded,
                                             int offset, byte[] value)
{
            if (TimeProfile.CLIENT_CONFIG.equals(descriptor.getUuid()))
{
                if
(Arrays.equals(BluetoothGattDescriptor.ENABLE_NOTIFICATION_VALUE,
value)) {
                    Log.d(TAG, "Subscribe device to notifications: " +
device);
                    mRegisteredDevices.add(device);
                } else if
(Arrays.equals(BluetoothGattDescriptor.DISABLE_NOTIFICATION_VALUE,
value)) {
                    Log.d(TAG, "Unsubscribe device from notifications:
" + device);
                    mRegisteredDevices.remove(device);
                }

                if (responseNeeded) {
                    mBluetoothGattServer.sendResponse(device,
                            requestId,
                            BluetoothGatt.GATT_SUCCESS,
                            0,
                            null);
                }
            } else {
                Log.w(TAG, "Unknown descriptor write request");
                if (responseNeeded) {
                    mBluetoothGattServer.sendResponse(device,
                            requestId,
```

```
                                BluetoothGatt.GATT_FAILURE,
                                0,
                                null);
                    }
                }
            }
        };

}
```

8.3.2 BluetoothCommunication.java

```
package com.example.bluetooth_util;

import android.app.ActionBar;
import android.app.Activity;
import android.app.Fragment;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.content.Intent;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.util.Log;
import android.widget.Toast;

import com.unity3d.player.UnityPlayer;

import java.util.Arrays;

/**
 * Created by manolis on 26/10/2017.
 */


public class BluetoothCommunication extends Fragment{

    private static final String TAG = "BluetoothCommunication";

    private static final int REQUEST_ENABLE_BT = 3;

    /**
     * Name of the connected device
     */
    private String mConnectedDeviceName = null;


    /**
     * Local Bluetooth adapter
     */
    private BluetoothAdapter mBluetoothAdapter = null;

    /**
     * Member object for the chat services
     */
    private BluetoothService mChatService = null;
```

```
    /**
     * ---------------------------------------------- Public functions
for Unity ----------------------------------------------
     */

    /**
     *  Unity specific properties
     */
    // Singleton instance.
    public static BluetoothCommunication instance;

    // Unity context.
    String gameObjectName;


    /**
     *  Initialize an instance of the android java class in the a unity
C# scipt.
     *
     * @param gameObjectName A gameObjectName from unity.
     */
    public static void instantiate(String gameObjectName){
        // Instantiate and add to Unity Player Activity.
        instance = new BluetoothCommunication();

        // Store 'GameObject' reference\
        instance.gameObjectName = gameObjectName;


UnityPlayer.currentActivity.getFragmentManager().beginTransaction().add
(instance, BluetoothCommunication.TAG).commit();
    }



    /**
     * Establish connection with other device
     *
     * @param macaddress A string with the macaddress of the device in
uppercase.
     * @param secure Socket Security type - Secure (true) , Insecure
(false)
     */
    public void connectDevice(String macaddress, boolean secure) {
        Log.e("BT", "Connect Device called, with mac: " + macaddress);

        // Get the BluetoothDevice object
        BluetoothDevice device =
mBluetoothAdapter.getRemoteDevice(macaddress);

        // Attempt to connect to the device
        mChatService.connect(device, secure);
    }

    /**
     *  Establishes a bluetooth server
     */
    public void acceptServer() {
```

```
        // Performing this check in onResume() covers the case in which
BT was
        // not enabled during onStart(), so we were paused to enable
it...
        // onResume() will be called when ACTION_REQUEST_ENABLE
activity returns.
        if (mChatService != null) {
            // Only if the state is STATE_NONE, do we know that we
haven't started already
            if (mChatService.getState() == BluetoothService.STATE_NONE)
{
                // Start the Bluetooth chat services
                mChatService.start();
            }
        }
    }

    public boolean isConnected(){
        if ( mChatService!= null && mChatService.getState() ==
BluetoothService.STATE_CONNECTED)
            return true;
        else
            return false;
    }

    public boolean isConnecting(){
        if ( mChatService!= null && mChatService.getState() ==
BluetoothService.STATE_CONNECTING)
            return true;
        else
            return false;
    }

    public boolean isListening(){
        if ( mChatService!= null && mChatService.getState() ==
BluetoothService.STATE_LISTEN)
            return true;
        else
            return false;
    }

    /**
     * Sends a message.
     *
     * @param message A string of text to send.
     */
    public void sendMessage(String message) {

        // Check that we're actually connected before trying anything
        if (mChatService.getState() !=
BluetoothService.STATE_CONNECTED) {
            Toast.makeText(getActivity(), R.string.not_connected,
Toast.LENGTH_SHORT).show();
            return;
        }

        // Check that there's actually something to send
        if (message.length() > 0) {
```

```
            // Get the message bytes and tell the BluetoothService to
write
            byte[] send = message.getBytes();
            byte[] enc_send = HalfCutter.encode(send);

            Log.d("Encode", "Real string : "  + message + "-----len: "
+ message.length() + "----------------------------------" );
            for(int i = 0; i < enc_send.length; i++) {
                Log.d("Encode", "Compressed Byte: " + i +  " - " +
String.format("%8s", Integer.toBinaryString(enc_send[i] &
0xFF)).replace(' ', '0'));
            }
            Log.d("Encode", "Compressed: "  + " len: " +
enc_send.length + "----------------------------------"  );
            mChatService.write(enc_send);
        }
    }


    /**
     * ---------------------------------------------- Functionality
Android functions ----------------------------------------------
     */


    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // Get local Bluetooth adapter
        mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();

        // If the adapter is null, then Bluetooth is not supported
        if (mBluetoothAdapter == null) {
            Activity activity = UnityPlayer.currentActivity;
            Toast.makeText(activity, "Bluetooth is not available",
Toast.LENGTH_LONG).show();
            activity.finish();
        }
    }


    @Override
    public void onStart() {
        super.onStart();
        // If BT is not on, request that it be enabled.
        //  BluetoothService object  will then be created during
onActivityResult
        if (!mBluetoothAdapter.isEnabled()) {

            Intent enableIntent = new
Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
            startActivityForResult(enableIntent, REQUEST_ENABLE_BT);
            // Otherwise, setup the chat session
        } else if (mChatService == null) {

            // Initialize the BluetoothService to perform bluetooth
connections
```

```
            mChatService = new BluetoothService(getActivity(),
mHandler);
        }
    }

    @Override
    public void onDestroy() {
        super.onDestroy();
        if (mChatService != null) {
            mChatService.stop();
        }
    }


    /**
     * Updates the status on the action bar.
     *
     * @param resId a string resource ID
     */
    private void setStatus(int resId) {
        Activity activity = UnityPlayer.currentActivity;
        if (null == activity) {
            return;
        }
        final ActionBar actionBar = activity.getActionBar();
        if (null == actionBar) {
            return;
        }
        actionBar.setSubtitle(resId);
    }

    /**
     * Updates the status on the action bar.
     *
     * @param subTitle status
     */
    private void setStatus(CharSequence subTitle) {
        Activity activity = UnityPlayer.currentActivity;
        if (null == activity) {
            return;
        }
        final ActionBar actionBar = activity.getActionBar();
        if (null == actionBar) {
            return;
        }
        actionBar.setSubtitle(subTitle);
    }

    /**
     * The Handler that gets information back from the BluetoothService
     */
    private final Handler mHandler = new Handler() {
        @Override
        public void handleMessage(Message msg) {
            Activity activity = UnityPlayer.currentActivity;
            switch (msg.what) {
                case Constants.MESSAGE_STATE_CHANGE:
                    switch (msg.arg1) {
```

```
                        case BluetoothService.STATE_CONNECTED:

setStatus(getString(R.string.title_connected_to,
mConnectedDeviceName));
                            break;
                        case BluetoothService.STATE_CONNECTING:
                            setStatus(R.string.title_connecting);
                            break;
                        case BluetoothService.STATE_LISTEN:
                        case BluetoothService.STATE_NONE:
                            setStatus(R.string.title_not_connected);
                            break;
                    }
                    break;
                case Constants.MESSAGE_READ:
                    byte[] readBuf = (byte[]) msg.obj;
                    // construct a string from the valid bytes in the
buffer
//                    String readMessage = new String(readBuf, 0,
msg.arg1);
                    byte[] slice = Arrays.copyOfRange(readBuf, 0,
msg.arg1);
                    byte[] uncompressed = HalfCutter.decode(slice);

                    UnityPlayer.UnitySendMessage(gameObjectName,
                            "receiveMessage",
                            new String(uncompressed, 0,
uncompressed.length));

                    for(int i = 0; i < uncompressed.length; i++) {
                        Log.d("Decode", "Compressed Byte
recevied(sliced): " + i +  " - " +uncompressed[i]);
                    }
                    Log.d("Decode", "Real: " + new String(uncompressed)
+ "--len:" + uncompressed.length);
//                    Log.d("Output", "Uncompressed: "  + " len: " +
uncompressed.length + "----------------------------------"  );
                    break;
                case Constants.MESSAGE_DEVICE_NAME:
                    // save the connected device's name
                    mConnectedDeviceName =
msg.getData().getString(Constants.DEVICE_NAME);
                    if (null != activity) {
                        Toast.makeText(activity, "Connected to "
                                + mConnectedDeviceName,
Toast.LENGTH_SHORT).show();
                    }
                    break;
                case Constants.MESSAGE_TOAST:
                    if (null != activity) {
                        Toast.makeText(activity,
msg.getData().getString(Constants.TOAST),
                                Toast.LENGTH_SHORT).show();
                    }
                    break;
            }
        }
    };
```

```
}
```

8.3.3 BluetoothService.java

```
/*
 * Copyright (C) 2014 The Android Open Source Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

package com.example.bluetooth_util;

import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothServerSocket;
import android.bluetooth.BluetoothSocket;
import android.content.Context;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.util.Log;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.util.UUID;

/**
 * This class does all the work for setting up and managing Bluetooth
 * connections with other devices. It has a thread that listens for
 * incoming connections, a thread for connecting with a device, and a
 * thread for performing data transmissions when connected.
 */
public class BluetoothService {
    // Debugging
    private static final String TAG = "BluetoothService";

    // Name for the SDP record when creating server socket
    private static final String NAME_SECURE = "BluetoothChatSecure";
    private static final String NAME_INSECURE =
"BluetoothChatInsecure";

    // Unique UUID for this application
    private static final UUID MY_UUID_SECURE =
            UUID.fromString("fa87c0d0-afac-11de-8a39-0800200c9a66");
    private static final UUID MY_UUID_INSECURE =
```

```
            UUID.fromString("8ce255c0-200a-11e0-ac64-0800200c9a66");

    // Member fields
    private final BluetoothAdapter mAdapter;
    private final Handler mHandler;
    private AcceptThread mSecureAcceptThread;
    private AcceptThread mInsecureAcceptThread;
    private ConnectThread mConnectThread;
    private ConnectedThread mConnectedThread;
    private int mState;
    private int mNewState;

    // Constants that indicate the current connection state
    public static final int STATE_NONE = 0;       // we're doing
nothing
    public static final int STATE_LISTEN = 1;     // now listening for
incoming connections
    public static final int STATE_CONNECTING = 2; // now initiating an
outgoing connection
    public static final int STATE_CONNECTED = 3;  // now connected to a
remote device

    /**
     * Constructor. Prepares a new BluetoothChat session.
     *
     * @param context The UI Activity Context
     * @param handler A Handler to send messages back to the UI
Activity
     */
    public BluetoothService(Context context, Handler handler) {
        mAdapter = BluetoothAdapter.getDefaultAdapter();
        mState = STATE_NONE;
        mNewState = mState;
        mHandler = handler;
    }

    /**
     * Update UI title according to the current state of the chat
connection
     */
    private synchronized void updateUserInterfaceTitle() {
        mState = getState();
//        Log.d(TAG, "updateUserInterfaceTitle() " + mNewState + " -> "
+ mState);
        mNewState = mState;

        // Give the new state to the Handler so the UI Activity can
update
        mHandler.obtainMessage(Constants.MESSAGE_STATE_CHANGE,
mNewState, -1).sendToTarget();
    }

    /**
     * Return the current connection state.
     */
    public synchronized int getState() {
        return mState;
    }
```

```
    /**
     * Start the chat service. Specifically start AcceptThread to begin
a
     * session in listening (server) mode. Called by the Activity
onResume()
     */
    public synchronized void start() {
//        Log.d(TAG, "start");

        // Cancel any thread attempting to make a connection
        if (mConnectThread != null) {
            mConnectThread.cancel();
            mConnectThread = null;
        }

        // Cancel any thread currently running a connection
        if (mConnectedThread != null) {
            mConnectedThread.cancel();
            mConnectedThread = null;
        }

        // Start the thread to listen on a BluetoothServerSocket
        if (mSecureAcceptThread == null) {
            mSecureAcceptThread = new AcceptThread(true);
            mSecureAcceptThread.start();
        }
        if (mInsecureAcceptThread == null) {
            mInsecureAcceptThread = new AcceptThread(false);
            mInsecureAcceptThread.start();
        }
        // Update UI title
        updateUserInterfaceTitle();
    }

    /**
     * Start the ConnectThread to initiate a connection to a remote
device.
     *
     * @param device The BluetoothDevice to connect
     * @param secure Socket Security type - Secure (true) , Insecure
(false)
     */
    public synchronized void connect(BluetoothDevice device, boolean
secure) {
//        Log.d(TAG, "connect to: " + device);

        // Cancel any thread attempting to make a connection
        if (mState == STATE_CONNECTING) {
            if (mConnectThread != null) {
                mConnectThread.cancel();
                mConnectThread = null;
            }
        }

        // Cancel any thread currently running a connection
        if (mConnectedThread != null) {
            mConnectedThread.cancel();
```

```
            mConnectedThread = null;
        }

        // Start the thread to connect with the given device
        mConnectThread = new ConnectThread(device, secure);
        mConnectThread.start();
        // Update UI title
        updateUserInterfaceTitle();
    }

    /**
     * Start the ConnectedThread to begin managing a Bluetooth
connection
     *
     * @param socket The BluetoothSocket on which the connection was
made
     * @param device The BluetoothDevice that has been connected
     */
    public synchronized void connected(BluetoothSocket socket,
BluetoothDevice
            device, final String socketType) {
//        Log.d(TAG, "connected, Socket Type:" + socketType);

        // Cancel the thread that completed the connection
        if (mConnectThread != null) {
            mConnectThread.cancel();
            mConnectThread = null;
        }

        // Cancel any thread currently running a connection
        if (mConnectedThread != null) {
            mConnectedThread.cancel();
            mConnectedThread = null;
        }

        // Cancel the accept thread because we only want to connect to
one device
        if (mSecureAcceptThread != null) {
            mSecureAcceptThread.cancel();
            mSecureAcceptThread = null;
        }
        if (mInsecureAcceptThread != null) {
            mInsecureAcceptThread.cancel();
            mInsecureAcceptThread = null;
        }

        // Start the thread to manage the connection and perform
transmissions
        mConnectedThread = new ConnectedThread(socket, socketType);
        mConnectedThread.start();

        // Send the name of the connected device back to the UI
Activity
        Message msg =
mHandler.obtainMessage(Constants.MESSAGE_DEVICE_NAME);
        Bundle bundle = new Bundle();
        bundle.putString(Constants.DEVICE_NAME, device.getName());
        msg.setData(bundle);
```

```
        mHandler.sendMessage(msg);
        // Update UI title
        updateUserInterfaceTitle();
    }

    /**
     * Stop all threads
     */
    public synchronized void stop() {
//        Log.d(TAG, "stop");

        if (mConnectThread != null) {
            mConnectThread.cancel();
            mConnectThread = null;
        }

        if (mConnectedThread != null) {
            mConnectedThread.cancel();
            mConnectedThread = null;
        }

        if (mSecureAcceptThread != null) {
            mSecureAcceptThread.cancel();
            mSecureAcceptThread = null;
        }

        if (mInsecureAcceptThread != null) {
            mInsecureAcceptThread.cancel();
            mInsecureAcceptThread = null;
        }
        mState = STATE_NONE;
        // Update UI title
        updateUserInterfaceTitle();
    }

    /**
     * Write to the ConnectedThread in an unsynchronized manner
     *
     * @param out The bytes to write
     * @see ConnectedThread#write(byte[])
     */
    public void write(byte[] out) {
        // Create temporary object
        ConnectedThread r;
        // Synchronize a copy of the ConnectedThread
        synchronized (this) {
            if (mState != STATE_CONNECTED) return;
            r = mConnectedThread;
        }
        // Perform the write unsynchronized
        r.write(out);
    }

    /**
     * Indicate that the connection attempt failed and notify the UI
Activity.
     */
    private void connectionFailed() {
```

```
        // Send a failure message back to the Activity
        Message msg = mHandler.obtainMessage(Constants.MESSAGE_TOAST);
        Bundle bundle = new Bundle();
        bundle.putString(Constants.TOAST, "Unable to connect device");
        msg.setData(bundle);
        mHandler.sendMessage(msg);

        mState = STATE_NONE;
        // Update UI title
        updateUserInterfaceTitle();

        // Start the service over to restart listening mode
        BluetoothService.this.start();
    }

    /**
     * Indicate that the connection was lost and notify the UI
Activity.
     */
    private void connectionLost() {
        // Send a failure message back to the Activity
        Message msg = mHandler.obtainMessage(Constants.MESSAGE_TOAST);
        Bundle bundle = new Bundle();
        bundle.putString(Constants.TOAST, "Device connection was
lost");
        msg.setData(bundle);
        mHandler.sendMessage(msg);

        mState = STATE_NONE;
        // Update UI title
        updateUserInterfaceTitle();

        // Start the service over to restart listening mode
        BluetoothService.this.start();
    }

    /**
     * This thread runs while listening for incoming connections. It
behaves
     * like a server-side client. It runs until a connection is
accepted
     * (or until cancelled).
     */
    private class AcceptThread extends Thread {
        // The local server socket
        private final BluetoothServerSocket mmServerSocket;
        private String mSocketType;

        public AcceptThread(boolean secure) {
            BluetoothServerSocket tmp = null;
            mSocketType = secure ? "Secure" : "Insecure";

            // Create a new listening server socket
            try {
                if (secure) {
                    tmp =
mAdapter.listenUsingRfcommWithServiceRecord(NAME_SECURE,
                            MY_UUID_SECURE);
```

```
                } else {
                    tmp =
mAdapter.listenUsingInsecureRfcommWithServiceRecord(
                            NAME_INSECURE, MY_UUID_INSECURE);
                }
            } catch (IOException e) {
//                Log.e(TAG, "Socket Type: " + mSocketType + "listen()
failed", e);
            }
            mmServerSocket = tmp;
            mState = STATE_LISTEN;
        }

        public void run() {
//            Log.d(TAG, "Socket Type: " + mSocketType +
//                    "BEGIN mAcceptThread" + this);
            setName("AcceptThread" + mSocketType);

            BluetoothSocket socket = null;

            // Listen to the server socket if we're not connected
            while (mState != STATE_CONNECTED) {
                try {
                    // This is a blocking call and will only return on
a
                    // successful connection or an exception
                    socket = mmServerSocket.accept();
                } catch (IOException e) {
//                    Log.e(TAG, "Socket Type: " + mSocketType +
"accept() failed", e);
                    break;
                }

                // If a connection was accepted
                if (socket != null) {
                    synchronized (BluetoothService.this) {
                        switch (mState) {
                            case STATE_LISTEN:
                            case STATE_CONNECTING:
                                // Situation normal. Start the
connected thread.
                                connected(socket,
socket.getRemoteDevice(),
                                        mSocketType);

                                // adde d this line
                                this.cancel();

                                break;
                            case STATE_NONE:
                            case STATE_CONNECTED:
                                // Either not ready or already
connected. Terminate new socket.
                                try {
                                    socket.close();
                                } catch (IOException e) {
//                                    Log.e(TAG, "Could not close
unwanted socket", e);
```

```
                                }
                                break;
                            }
                        }
                    }
                }
//              Log.i(TAG, "END mAcceptThread, socket Type: " +
mSocketType);

            }

            public void cancel() {
//                Log.d(TAG, "Socket Type" + mSocketType + "cancel " +
this);
                try {
                    mmServerSocket.close();
                } catch (IOException e) {
//                    Log.e(TAG, "Socket Type" + mSocketType + "close() of
server failed", e);
                }
            }
        }


        /**
         * This thread runs while attempting to make an outgoing connection
         * with a device. It runs straight through; the connection either
         * succeeds or fails.
         */
        private class ConnectThread extends Thread {
            private final BluetoothSocket mmSocket;
            private final BluetoothDevice mmDevice;
            private String mSocketType;

            public ConnectThread(BluetoothDevice device, boolean secure) {
                mmDevice = device;
                BluetoothSocket tmp = null;
                mSocketType = secure ? "Secure" : "Insecure";

                // Get a BluetoothSocket for a connection with the
                // given BluetoothDevice
                try {
                    if (secure) {
                        tmp = device.createRfcommSocketToServiceRecord(
                                MY_UUID_SECURE);
                    } else {
                        tmp =
device.createInsecureRfcommSocketToServiceRecord(
                                MY_UUID_INSECURE);
                    }
                } catch (IOException e) {
//                    Log.e(TAG, "Socket Type: " + mSocketType + "create()
failed", e);
                }
                mmSocket = tmp;
                mState = STATE_CONNECTING;
            }
```

```
        public void run() {
//            Log.i(TAG, "BEGIN mConnectThread SocketType:" +
mSocketType);
            setName("ConnectThread" + mSocketType);

            // Always cancel discovery because it will slow down a
connection
//-------------------------------------------------------------------
--------            mAdapter.cancelDiscovery();              ----------
-----------------------------------------

            // Make a connection to the BluetoothSocket
            try {
                // This is a blocking call and will only return on a
                // successful connection or an exception
                mmSocket.connect();
            } catch (IOException e) {
                // Close the socket
                try {
                    mmSocket.close();
                } catch (IOException e2) {
//                    Log.e(TAG, "unable to close() " + mSocketType +
//                            " socket during connection failure", e2);
                }
                connectionFailed();
                return;
            }

            // Reset the ConnectThread because we're done
            synchronized (BluetoothService.this) {
                mConnectThread = null;
            }

            // Start the connected thread
            connected(mmSocket, mmDevice, mSocketType);
        }

        public void cancel() {
            try {
                mmSocket.close();
            } catch (IOException e) {
//                Log.e(TAG, "close() of connect " + mSocketType + "
socket failed", e);
            }
        }
    }

    /**
     * This thread runs during a connection with a remote device.
     * It handles all incoming and outgoing transmissions.
     */
    private class ConnectedThread extends Thread {
        private final BluetoothSocket mmSocket;
        private final InputStream mmInStream;
        private final OutputStream mmOutStream;

        public ConnectedThread(BluetoothSocket socket, String
socketType) {
```

```
//            Log.d(TAG, "create ConnectedThread: " + socketType);
            mmSocket = socket;
            InputStream tmpIn = null;
            OutputStream tmpOut = null;

            // Get the BluetoothSocket input and output streams
            try {
                tmpIn = socket.getInputStream();
                tmpOut = socket.getOutputStream();
            } catch (IOException e) {
//                Log.e(TAG, "temp sockets not created", e);
            }

            mmInStream = tmpIn;
            mmOutStream = tmpOut;
            mState = STATE_CONNECTED;
        }

        public void run() {
//            Log.i(TAG, "BEGIN mConnectedThread");
            byte[] buffer = new byte[512];
            int bytes;
            long timer1, timer2;


            // Keep listening to the InputStream while connected
            while (mState == STATE_CONNECTED) {
                try {
                    timer1 = System.currentTimeMillis();;
                    // Read from the InputStream
                    bytes = mmInStream.read(buffer);
                    timer2 = System.currentTimeMillis();
                    Log.e("BTr", "Read Delay--:" +
Long.toString(timer2-timer1) + "----Data size: " + bytes );

                    // Send the obtained bytes to the UI Activity
                    mHandler.obtainMessage(Constants.MESSAGE_READ,
bytes, -1, buffer)
                            .sendToTarget();
                } catch (IOException e) {
//                    Log.e(TAG, "disconnected", e);
                    connectionLost();
                    break;
                }
            }
        }

        /**
         * Write to the connected OutStream.
         *
         * @param buffer The bytes to write
         */
        public void write(byte[] buffer) {
            long timer1, timer2;
            try {

                timer1 = System.currentTimeMillis();;
                // Read from the InputStream
```

```
                mmOutStream.write(buffer);
//                mmOutStream.flush();
                timer2 = System.currentTimeMillis();
                Log.e("BTw", "Write Delay--:" + Long.toString(timer2-
timer1)  );

                // Share the sent message back to the UI Activity
                mHandler.obtainMessage(Constants.MESSAGE_WRITE, -1, -1,
buffer)
                        .sendToTarget();
            } catch (IOException e) {
//                Log.e(TAG, "Exception during write", e);
            }
        }

        public void cancel() {
            try {
                mmSocket.close();
            } catch (IOException e) {
//                Log.e(TAG, "close() of connect socket failed", e);
            }
        }
    }
}
```

8.3.4 Constants.java

```
/*
 * Copyright (C) 2014 The Android Open Source Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

package com.example.bluetooth_util;

/**
 * Defines several constants used between {@link BluetoothService} and
the UI.
 */
public interface Constants {

    // Message types sent from the BluetoothService Handler
    public static final int MESSAGE_STATE_CHANGE = 1;
```

```
    public static final int MESSAGE_READ = 2;
    public static final int MESSAGE_WRITE = 3;
    public static final int MESSAGE_DEVICE_NAME = 4;
    public static final int MESSAGE_TOAST = 5;

    // Key names received from the BluetoothService Handler
    public static final String DEVICE_NAME = "device_name";
    public static final String TOAST = "toast";

}
```

8.3.5 HalfCutter.java

```
package com.example.bluetooth_util;

import android.util.Log;

/**
 * Created by manolis on 30/11/2017.
 */

public class HalfCutter {
    private static final String TAG = "HalfCutter";

    static public byte[] decode(byte[] data){
        byte[] fullData = new byte[data.length*2];
        byte part1 = 15, part2 = 15;

        int i = 0, j = 0;
        while (i < data.length) {
            part1 = (byte) (data[i] & 0xF0);
            part1 = (byte) (part1 >>> 0x04);
            part1 = (byte) (part1 & 0x0F);
            part2 = (byte) (data[i] & 0x0F);
            Log.d(TAG, "part1: "+ part1 + " Part2: " + part2);

            fullData[j] = identify(part1);
            j++;
            fullData[j] = identify(part2);
            j++;
            i++;
        }
        return fullData;
    }

    static private byte identify(byte data) {
        switch (data) {
            case (byte) 0:
                return (byte) '0';
            case (byte) 1:
                return (byte) '1';
            case (byte) 2:
                return (byte) '2';
            case (byte) 3:
                return (byte) '3';
            case (byte) 4:
```

```
                return (byte) '4';
            case (byte) 5:
                return (byte) '5';
            case (byte) 6:
                return (byte) '6';
            case (byte) 7:
                return (byte) '7';
            case (byte) 8:
                return (byte) '8';
            case (byte) 9:
                return (byte) '9';
            case (byte) 10:
                return (byte) '!'; //(byte) '!';
            case (byte) 11:
                return (byte) ',';
            case (byte) 12:
                return (byte) '.';
            case (byte) 13:
                return (byte) ' ';
            case (byte) 14:
                return (byte) '-';
            default:
                Log.d(TAG, "Case default selected!: ");
                // 63 is ?
                return 63;
        }
    }



    static public byte[] encode(byte[] data) {
        byte[] halfData = new byte[data.length/2 + 1];
        byte[] part = new byte [2];
        part[0] = 15;
        part[1] = 15;

        int i = 0, j = 0;
        while (i < data.length){
//            if (i != 0 && data[i] == (byte) '!') {
//                // special ocassion in the point of concatinating 2
strings
//                part[i % 2] = 15;
//                data[i] = (byte) '*';
//            } else {
                // normal occasion
                switch (data[i]) {
                    case (byte) '0':
                        part[i % 2] = 0;
                        break;
                    case (byte) '1':
                        part[i % 2] = 1;
                        break;
                    case (byte) '2':
                        part[i % 2] = 2;
                        break;
                    case (byte) '3':
                        part[i % 2] = 3;
                        break;
```

```
                    case (byte) '4':
                        part[i % 2] = 4;
                        break;
                    case (byte) '5':
                        part[i % 2] = 5;
                        break;
                    case (byte) '6':
                        part[i % 2] = 6;
                        break;
                    case (byte) '7':
                        part[i % 2] = 7;
                        break;
                    case (byte) '8':
                        part[i % 2] = 8;
                        break;
                    case (byte) '9':
                        part[i % 2] = 9;
                        break;
                    case (byte) '!':
                        part[i % 2] = 10;
                        break;
//                    case (byte) ('*'):
//                        // secial occasion to add ? for 2 subsequent
point in the same string
//                        part[i % 2] = 10;
//                        break;
                    case (byte) ',':
                        part[i % 2] = 11;
                        break;
                    case (byte) '.':
                        part[i % 2] = 12;
                        break;
                    case (byte) ' ':
                        part[i % 2] = 13;
                        break;
                    case (byte) '-':
                        part[i % 2] = 14;
                        break;
                    default:
                        Log.d(TAG, "Case default selected!: ");
                        // 15 is NaN value
                        part[i % 2] = 15;
                        break;
            }

            if (i % 2 == 1 ){
                byte front = (byte) (part[0] << 0x04);
                halfData[j] = (byte) (front |  part[1]);
                j++;
                // refresh parts to nan
                part[0] = 15;
                part[1] = 15;
            }
            i++;
        }

        // for even string
        if (i%2 == 1) {
```

```
            byte front = (byte) (part[0] << 0x04);
            halfData[j] = (byte) (front |  15);
        } else {
            halfData[j] = (byte) 255;
        }
        Log.d(TAG, "Data proccessing: " + String.format("%8s",
Integer.toBinaryString(halfData[j] & 0xFF)).replace(' ', '0'));
        return halfData;
    }

}
```

8.3.6 TimeProfile.java

```
package com.example.bluetooth_util;

/**
 * Created by manolis on 14/12/2017.
 */

import android.bluetooth.BluetoothGattCharacteristic;
import android.bluetooth.BluetoothGattDescriptor;
import android.bluetooth.BluetoothGattService;

import java.util.Calendar;
import java.util.UUID;

/**
 * Implementation of the Bluetooth GATT Time Profile.
 * https://www.bluetooth.com/specifications/adopted-specifications
 */
public class TimeProfile {
    private static final String TAG =
TimeProfile.class.getSimpleName();

    /* Service UUID */
    public static UUID TIME_SERVICE = UUID.fromString("00001899-0000-
1000-8000-00805f9b34fb");

    /* _______________Characteristics UUID */

    // upper body
    public static UUID Head    = UUID.fromString("00002a2b-0000-1000-
8000-00805f9b34fb");
    public static UUID lCollar    = UUID.fromString("00002a2c-0000-
1000-8000-00805f9b34fb");

    // right upper body
    public static UUID rWrist = UUID.fromString("00002a0f-0000-1000-
8000-00805f9b34fb");
    public static UUID rElbow = UUID.fromString("00002a10-0000-1000-
8000-00805f9b34fb");
    public static UUID rShoulder = UUID.fromString("00002a11-0000-1000-
8000-00805f9b34fb");
```

```
    // left upper body
    public static UUID lWrist = UUID.fromString("00002a12-0000-1000-
8000-00805f9b34fb");
    public static UUID lElbow = UUID.fromString("00002a13-0000-1000-
8000-00805f9b34fb");
    public static UUID lShoulder = UUID.fromString("00002a14-0000-1000-
8000-00805f9b34fb");

    // Grabs
    public static UUID Grabs = UUID.fromString("00002a20-0000-1000-
8000-00805f9b34fb");


    /* _____Client Config Descriptor */
    public static UUID CLIENT_CONFIG = UUID.fromString("00002902-0000-
1000-8000-00805f9b34fb");

    // Adjustment Flags
    public static final byte ADJUST_NONE     = 0x0;
    public static final byte ADJUST_MANUAL   = 0x1;

    /**
     * Return a configured {@link BluetoothGattService} instance for
the
     * Current Time Service.
     */
    public static BluetoothGattService createTimeService() {
        BluetoothGattService service = new
BluetoothGattService(TIME_SERVICE,
                BluetoothGattService.SERVICE_TYPE_PRIMARY);

        // Head characteristic
        BluetoothGattCharacteristic head = new
BluetoothGattCharacteristic(Head,
                BluetoothGattCharacteristic.PROPERTY_READ |
BluetoothGattCharacteristic.PROPERTY_WRITE_NO_RESPONSE,
                BluetoothGattCharacteristic.PERMISSION_READ |
BluetoothGattCharacteristic.PERMISSION_WRITE);

        BluetoothGattDescriptor configDescriptor = new
BluetoothGattDescriptor(CLIENT_CONFIG,
                //Read/write descriptor
                BluetoothGattDescriptor.PERMISSION_READ |
BluetoothGattDescriptor.PERMISSION_WRITE);
        head.addDescriptor(configDescriptor);

        // left Collar characteristic
        BluetoothGattCharacteristic lcollar = new
BluetoothGattCharacteristic(lCollar,
                BluetoothGattCharacteristic.PROPERTY_READ |
BluetoothGattCharacteristic.PROPERTY_WRITE_NO_RESPONSE,
                BluetoothGattCharacteristic.PERMISSION_READ |
BluetoothGattCharacteristic.PERMISSION_WRITE);


        // Right wrist characteristic
```

```
        BluetoothGattCharacteristic rwrist = new
BluetoothGattCharacteristic(rWrist,
                BluetoothGattCharacteristic.PROPERTY_READ |
BluetoothGattCharacteristic.PROPERTY_WRITE_NO_RESPONSE,
                BluetoothGattCharacteristic.PERMISSION_READ |
BluetoothGattCharacteristic.PERMISSION_WRITE);

        // Right elbow characteristic
        BluetoothGattCharacteristic relbow = new
BluetoothGattCharacteristic(rElbow,
                BluetoothGattCharacteristic.PROPERTY_READ |
BluetoothGattCharacteristic.PROPERTY_WRITE_NO_RESPONSE,
                BluetoothGattCharacteristic.PERMISSION_READ |
BluetoothGattCharacteristic.PERMISSION_WRITE);

        // Right shoulder characteristic
        BluetoothGattCharacteristic rshoulder = new
BluetoothGattCharacteristic(rShoulder,
                BluetoothGattCharacteristic.PROPERTY_READ |
BluetoothGattCharacteristic.PROPERTY_WRITE_NO_RESPONSE,
                BluetoothGattCharacteristic.PERMISSION_READ |
BluetoothGattCharacteristic.PERMISSION_WRITE);


        // Left wrist characteristic
        BluetoothGattCharacteristic lwrist = new
BluetoothGattCharacteristic(lWrist,
                BluetoothGattCharacteristic.PROPERTY_READ |
BluetoothGattCharacteristic.PROPERTY_WRITE_NO_RESPONSE,
                BluetoothGattCharacteristic.PERMISSION_READ |
BluetoothGattCharacteristic.PERMISSION_WRITE);

        // Left elbow characteristic
        BluetoothGattCharacteristic lelbow = new
BluetoothGattCharacteristic(lElbow,
                BluetoothGattCharacteristic.PROPERTY_READ |
BluetoothGattCharacteristic.PROPERTY_WRITE_NO_RESPONSE,
                BluetoothGattCharacteristic.PERMISSION_READ |
BluetoothGattCharacteristic.PERMISSION_WRITE);

        // Left shoulder characteristic
        BluetoothGattCharacteristic lshoulder = new
BluetoothGattCharacteristic(lShoulder,
                BluetoothGattCharacteristic.PROPERTY_READ |
BluetoothGattCharacteristic.PROPERTY_WRITE_NO_RESPONSE,
                BluetoothGattCharacteristic.PERMISSION_READ |
BluetoothGattCharacteristic.PERMISSION_WRITE);

        // Right and Left Grabs characteristic
        BluetoothGattCharacteristic grabs = new
BluetoothGattCharacteristic(Grabs,
                BluetoothGattCharacteristic.PROPERTY_READ |
BluetoothGattCharacteristic.PROPERTY_WRITE_NO_RESPONSE,
                BluetoothGattCharacteristic.PERMISSION_READ |
BluetoothGattCharacteristic.PERMISSION_WRITE);
```

```
        service.addCharacteristic(head);
//        service.addCharacteristic(lcollar);

        service.addCharacteristic(rwrist);
        service.addCharacteristic(relbow);
        service.addCharacteristic(rshoulder);

        service.addCharacteristic(lwrist);
        service.addCharacteristic(lelbow);
        service.addCharacteristic(lshoulder);

//        service.addCharacteristic(grabs);

        return service;
    }

}
```

8.4 Image Recognition in Camera

8.4.1 Main.cpp

```
#include <iostream>
#include <GL/glut.h>

#include "NuitrackGLSample.h"


#include <blepp/blestatemachine.h>
#include <blepp/float.h>  //BLE uses the obscure IEEE11073 decimal
exponent floating point values
#include <unistd.h>
#include <iomanip>
#include <chrono>


std::shared_ptr<NuitrackGLSample> sample;

using namespace BLEPP;
using namespace std;
using namespace chrono;


void keyboard(unsigned char key, int x, int y)
{
      switch (key)
      {
      case 27:
      {
            sample->release();

            glutDestroyWindow(glutGetWindow());

            break;
      }
```

```
	default:
	{
		break;
	}
	}
}

void display()
{


	bool update = sample->update();

	if (!update)
	{
		sample->release();
		glutDestroyWindow(glutGetWindow());

		return;
	}

	glFlush();
	glutSwapBuffers();
}

void idle()
{
	glutPostRedisplay();
}


void help()
{
	std::clog << "Usage: nuitrack_gl_sample" << std::endl;
}

int main(int argc, char** argv)
{
	if (argc != 1)
	{
		help();
		exit(-1);
	}


	// bluetooth staff
	BLEGATTStateMachine gatt;

	log_level = Warning;

	std::function<void(const PDUNotificationOrIndication&)> notify_cb
= [&](const PDUNotificationOrIndication& n)
	{
		auto ms_since_epoch =
duration_cast<milliseconds>(system_clock::now().time_since_epoch());
		float temp = bluetooth_float_to_IEEE754(n.value().first+1);
```

```
            cout << setprecision(15) << ms_since_epoch.count()/1000. <<
" " << setprecision(5) << temp << endl;
      };

      std::function<void()> found_services_and_characteristics_cb =
[&gatt, &notify_cb, &argc, &argv](){
            cout << "Calling our our cb callback!!\n";

            // initializing openGL window, etc
            glutInit(&argc, argv);
            glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE | GLUT_DEPTH);
            glutInitWindowSize(640, 480);
            glutCreateWindow("Nuitrack GL Sample");
            glutSetCursor(GLUT_CURSOR_NONE);

            glutKeyboardFunc(keyboard);
            glutDisplayFunc(display);
            glutIdleFunc(idle);

            glDisable(GL_DEPTH_TEST);
            glEnable(GL_TEXTURE_2D);

            glEnableClientState(GL_VERTEX_ARRAY);
            glDisableClientState(GL_COLOR_ARRAY);

            glOrtho(0, 640, 480, 0, -1.0, 1.0);
            glMatrixMode(GL_PROJECTION);
            glPushMatrix();
            glLoadIdentity();

                sample = std::make_shared<NuitrackGLSample>();
            sample->init("",&gatt);
            glutMainLoop();

      };

      gatt.setup_standard_scan(found_services_and_characteristics_cb);

      gatt.cb_disconnected = [](BLEGATTStateMachine::Disconnect d)
      {
            cerr << "Disconnect for reason " <<
BLEGATTStateMachine::get_disconnect_string(d) << endl;
            exit(1);
      };

      gatt.connect_blocking_S7();

      for(;;)
            gatt.read_and_process_next();


      return 0;
}
```

8.4.2 NuitrackGLSample.cpp

```
#include "NuitrackGLSample.h"
#include <blepp/blestatemachine.h>
#include <blepp/float.h>  //BLE uses the obscure IEEE11073 decimal
exponent floating point values
#include <math.h>
#include <iostream>
#include <ctime>
#include <cstdio>

NuitrackGLSample::NuitrackGLSample() :
    _textureID(0), _textureBuffer(0)
{
      _width = 640;
      _height = 480;
}

NuitrackGLSample::~NuitrackGLSample()
{
      Nuitrack::release();
}

/**
 * Initialize Nuitrack sample
 */
void NuitrackGLSample::init(const std::string& config,
BLEPP::BLEGATTStateMachine* gatt)
{
      // Save bluetooth gatt object if given
      if (gatt != NULL){
            gatt_obj = gatt;
      }

      // Before initialize Nuitrack, after create Nuitrack modules
      Nuitrack::init(config);

      // Create all needed Nuitrack modules
      _depthSensor = DepthSensor::create();
      // Bind to event new frame
      _depthSensor-
>connectOnNewFrame(std::bind(&NuitrackGLSample::onNewDepthFrame, this,
std::placeholders::_1));

      _outputMode = _depthSensor->getOutputMode();
      // Create texture by output mode DepthSensor
      initTexture(_width, _height);

      _userTracker = UserTracker::create();
      // Bind to event update user tracker
      _userTracker-
>connectOnUpdate(std::bind(&NuitrackGLSample::onUserUpdate, this,
std::placeholders::_1));

      _skeletonTracker = SkeletonTracker::create();
      // Bind to event update skeleton tracker
```

```
	_skeletonTracker-
>connectOnUpdate(std::bind(&NuitrackGLSample::onSkeletonUpdate, this,
std::placeholders::_1));

	_handTracker = HandTracker::create();
	// Bind to event update Hand tracker
	_handTracker-
>connectOnUpdate(std::bind(&NuitrackGLSample::onHandUpdate, this,
std::placeholders::_1));

	_gestureRecognizer = GestureRecognizer::create();
	_gestureRecognizer-
>connectOnNewGestures(std::bind(&NuitrackGLSample::onNewGesture, this,
std::placeholders::_1));

	// After create Nuitrack modules, calling Nuitrack::run() to
start processing all modules
	Nuitrack::run();
}

bool NuitrackGLSample::update()
{
	// Wait and update Nuitrack
	try
	{
			//std::clock_t start;
			//double duration;
			//start = std::clock();

		Nuitrack::waitUpdate(_skeletonTracker);

			//duration = ( std::clock() - start ) / (double)
CLOCKS_PER_SEC;
			//std::cout<<"printf: "<< duration <<'\n';

		renderTexture();
		renderLines();
	}
	catch (...)
	{
		return false;
	}

	return true;
}

void NuitrackGLSample::release()
{
	// Release Nuitrack remove all modules
	Nuitrack::release();

	if (_textureBuffer)
	{
		delete[] _textureBuffer;
		_textureBuffer = 0;
	}
}
```

```
void NuitrackGLSample::onNewDepthFrame(DepthFrame::Ptr frame)
{
      uint8_t* texturePtr = _textureBuffer;
      const uint16_t* depthPtr = frame->getData();

      int w_step = _width / frame->getCols();
      int h_step = _height / frame->getRows();

      int next_i_border = h_step;

      for (size_t i = 0; i < _height; ++i)
      {
            if (i == next_i_border)
            {
                  next_i_border += h_step;
                  depthPtr += frame->getCols();
            }

            int col = 0;
            int next_j_border = w_step;
            uint16_t depthValue = *depthPtr >> 5;

            for (size_t j = 0; j < _width; ++j, texturePtr += 3)
            {
                  if (j == next_j_border)
                  {
                        ++col;
                        next_j_border += w_step;
                        depthValue = *(depthPtr + col) >> 5;
                  }

                  texturePtr[0] = depthValue;
                  texturePtr[1] = depthValue;
                  texturePtr[2] = depthValue;
            }
      }
}

void NuitrackGLSample::onUserUpdate(UserFrame::Ptr frame)
{
      static uint8_t colors[8][3] =
      {
          {255, 0, 0},
          {0, 255, 0},
          {0, 0, 255},
          {255, 255, 0},

          {0, 255, 255},
          {255, 0, 255},
          {255, 255, 255},
          {127, 255, 0}
      };

      uint8_t* texturePtr = _textureBuffer;
      const uint16_t* labelPtr = frame->getData();

      int w_step = _width / frame->getCols();
      int h_step = _height / frame->getRows();
```

```
	int next_i_border = h_step;

	for (size_t i = 0; i < _height; ++i)
	{
		if (i == next_i_border)
		{
			next_i_border += h_step;
			labelPtr += frame->getCols();
		}

		int col = 0;
		int next_j_border = w_step;
		uint16_t label = *labelPtr;

		for (size_t j = 0; j < _width; ++j, texturePtr += 3)
		{
			if (j == next_j_border)
			{
				++col;
				next_j_border += w_step;
				label = *(labelPtr + col);
			}

			if (label)
			{
				texturePtr[0] = colors[label & 7][0];
				texturePtr[1] = colors[label & 7][1];
				texturePtr[2] = colors[label & 7][2];
			}
		}
	}
}

void NuitrackGLSample::onSkeletonUpdate(SkeletonData::Ptr
userSkeletons)
{
	_lines.clear();

	auto skeletons = userSkeletons->getSkeletons();
	for (auto skeleton: skeletons)
	{
		drawSkeleton(skeleton.joints);
	}
}

void NuitrackGLSample::onHandUpdate(HandTrackerData::Ptr handData)
{
	_leftHandPointers.clear();
	_rightHandPointers.clear();

	for (auto userHands: handData->getUsersHands())
	{
		std::string lGrab, rGrab;

		if (userHands.leftHand)
		{
```

```
                  _leftHandPointers.push_back(_width *
userHands.leftHand->x);
                  _leftHandPointers.push_back(_height *
userHands.leftHand->y);
                  if (userHands.leftHand->click)
                  {
                        //printf("Left hand clicked!\n");
                        lGrab = "1";

                        _leftHandPointers.push_back(_width *
userHands.leftHand->x);
                        _leftHandPointers.push_back(_height *
userHands.leftHand->y);
                  } else {
                        lGrab = "0";
                  }

                  uint8_t buffer[1];
                  std::strcpy((char*) buffer, lGrab.c_str());

                  //send to bluetooth
                  if (this->gatt_obj == NULL){
                        std::cout << "gatt_obj not passed properly\n";
                  } else
                        for(auto& service: this->gatt_obj-
>primary_services)
                              for(auto& characteristic:
service.characteristics)
                              if(characteristic.uuid ==
BLEPP::UUID("2a21")){
                                    characteristic.write_command(buffer,
lGrab.size());
                  }

            }

            if (userHands.rightHand)
            {
                  _rightHandPointers.push_back(_width *
userHands.rightHand->x);
                  _rightHandPointers.push_back(_height *
userHands.rightHand->y);
                  if (userHands.rightHand->click)
                  {
                        //printf("Right hand clicked!\n");
                        rGrab = "1";

                        _rightHandPointers.push_back(_width *
userHands.rightHand->x);
                        _rightHandPointers.push_back(_height *
userHands.rightHand->y);
                  } else {
                        rGrab = "0";
                  }

                  uint8_t buffer[1];
                  std::strcpy((char*) buffer, rGrab.c_str());
```

```
                  //send to bluetooth
                  if (this->gatt_obj == NULL){
                        std::cout << "gatt_obj not passed properly\n";
                  } else
                        for(auto& service: this->gatt_obj-
>primary_services)
                              for(auto& characteristic:
service.characteristics)
                              if(characteristic.uuid ==
BLEPP::UUID("2a20")){
                                    characteristic.write_command(buffer,
rGrab.size());
                  }

            }
      }
}

void NuitrackGLSample::onNewGesture(GestureData::Ptr gestureData)
{
      _userGestures = gestureData->getGestures();
      for (int i = 0; i < _userGestures.size(); ++i)
      {
//          printf("Recognized %d from %d\n", _userGestures[i].type,
_userGestures[i].userId);
      }
}

void NuitrackGLSample::drawBone(const Joint& j1, const Joint& j2)
{
      if (j1.confidence > 0.15 && j2.confidence > 0.15)
      {
            _lines.push_back(_width * j1.proj.x);
            _lines.push_back(_height * j1.proj.y);
            _lines.push_back(_width * j2.proj.x);
            _lines.push_back(_height * j2.proj.y);
      }
}

void NuitrackGLSample::send_vector3(GLfloat x, GLfloat y, GLfloat z,
std::string joint_UUID){
      x = 0.001* x;
      y = 0.001* y;
      z = 0.001* z;

      std::string x_str = "12345";
      std::string y_str = "12345";
      std::string z_str = "12345";
      std::string vector_str;

      int written;
      std::string sign_x, sign_y, sign_z;

      if (x < 0) {
            sign_x = "-";
            x = -x;
      } else
            sign_x = " ";
```

```
      if (y < 0) {
            sign_y = "-";
            y = -y;
      } else
            sign_y = " ";

      if (z < 0) {
            sign_z = "-";
            z = -z;
      } else
            sign_z = " ";



      written = snprintf(&x_str[0], x_str.size(), "%.2f", x);
      x_str.resize(written);
      written = snprintf(&y_str[0], y_str.size(), "%.2f", y);
      y_str.resize(written);
      written = snprintf(&z_str[0], z_str.size(), "%.2f", z);
      z_str.resize(written);

      vector_str = "!" + sign_x + x_str +  ", " + sign_y + y_str + ", "
+  z_str + '?';
      //std::cout << vector_str << std::endl;
      //std::cout << "size: " + vector_str.size() << std::endl;

      uint8_t buffer[20];
      std::strcpy((char*) buffer, vector_str.c_str());

      //send to bluetooth
      if (this->gatt_obj == NULL){
            std::cout << "gatt_obj not passed properly\n";
      } else
            for(auto& service: this->gatt_obj->primary_services)
                  for(auto& characteristic: service.characteristics)
                  if(characteristic.uuid == BLEPP::UUID(joint_UUID)){
                        characteristic.write_command(buffer,
vector_str.size());
                  }

}

void NuitrackGLSample::drawSkeleton(const std::vector<Joint>& joints)
{

      // center uppder body
      send_vector3(joints[JOINT_HEAD].real.x,
                  joints[JOINT_HEAD].real.y,
                  joints[JOINT_HEAD].real.z,
                  "2a2b");


      send_vector3(joints[JOINT_LEFT_COLLAR].real.x,
                  joints[JOINT_LEFT_COLLAR].real.y,
                  joints[JOINT_LEFT_COLLAR].real.z,
                  "2a2c");
```

```
	// right upper body
	if(joints[JOINT_RIGHT_WRIST].confidence > 0.5)
		send_vector3(joints[JOINT_RIGHT_WRIST].real.x,
				joints[JOINT_RIGHT_WRIST].real.y,
				joints[JOINT_RIGHT_WRIST].real.z,
				"2a0f");

	if(joints[JOINT_RIGHT_ELBOW].confidence > 0.5)
		send_vector3(joints[JOINT_RIGHT_ELBOW].real.x,
				joints[JOINT_RIGHT_ELBOW].real.y,
				joints[JOINT_RIGHT_ELBOW].real.z,
				"2a10");

	if(joints[JOINT_RIGHT_SHOULDER].confidence > 0.5)
		send_vector3(joints[JOINT_RIGHT_SHOULDER].real.x,
				joints[JOINT_RIGHT_SHOULDER].real.y,
				joints[JOINT_RIGHT_SHOULDER].real.z,
				"2a11");

	// left upper body
	if(joints[JOINT_LEFT_WRIST].confidence > 0.5)
		send_vector3(joints[JOINT_LEFT_WRIST].real.x,
				joints[JOINT_LEFT_WRIST].real.y,
				joints[JOINT_LEFT_WRIST].real.z,
				"2a12");
	if(joints[JOINT_LEFT_ELBOW].confidence > 0.5)
		send_vector3(joints[JOINT_LEFT_ELBOW].real.x,
				joints[JOINT_LEFT_ELBOW].real.y,
				joints[JOINT_LEFT_ELBOW].real.z,
				"2a13");
	if(joints[JOINT_LEFT_SHOULDER].confidence > 0.5)
		send_vector3(joints[JOINT_LEFT_SHOULDER].real.x,
				joints[JOINT_LEFT_SHOULDER].real.y,
				joints[JOINT_LEFT_SHOULDER].real.z,
				"2a14");

	drawBone(joints[JOINT_HEAD], joints[JOINT_NECK]);
	drawBone(joints[JOINT_NECK], joints[JOINT_LEFT_COLLAR]);
	drawBone(joints[JOINT_LEFT_COLLAR], joints[JOINT_TORSO]);
	drawBone(joints[JOINT_LEFT_COLLAR], joints[JOINT_LEFT_SHOULDER]);
	drawBone(joints[JOINT_LEFT_COLLAR],
joints[JOINT_RIGHT_SHOULDER]);
	drawBone(joints[JOINT_WAIST], joints[JOINT_LEFT_HIP]);
	drawBone(joints[JOINT_WAIST], joints[JOINT_RIGHT_HIP]);
	drawBone(joints[JOINT_TORSO], joints[JOINT_WAIST]);
	drawBone(joints[JOINT_LEFT_SHOULDER], joints[JOINT_LEFT_ELBOW]);
	drawBone(joints[JOINT_LEFT_ELBOW], joints[JOINT_LEFT_WRIST]);
	drawBone(joints[JOINT_LEFT_WRIST], joints[JOINT_LEFT_HAND]);
	drawBone(joints[JOINT_RIGHT_SHOULDER],
joints[JOINT_RIGHT_ELBOW]);
	drawBone(joints[JOINT_RIGHT_ELBOW], joints[JOINT_RIGHT_WRIST]);
	drawBone(joints[JOINT_RIGHT_WRIST], joints[JOINT_RIGHT_HAND]);
	drawBone(joints[JOINT_RIGHT_HIP], joints[JOINT_RIGHT_KNEE]);
	drawBone(joints[JOINT_LEFT_HIP], joints[JOINT_LEFT_KNEE]);
	drawBone(joints[JOINT_RIGHT_KNEE], joints[JOINT_RIGHT_ANKLE]);
	drawBone(joints[JOINT_LEFT_KNEE], joints[JOINT_LEFT_ANKLE]);
}
```

```
void NuitrackGLSample::renderTexture()
{
	glClearColor(1, 1, 1, 1);
	glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

	glEnable(GL_TEXTURE_2D);
	glColor4f(1, 1, 1, 1);

	glBindTexture(GL_TEXTURE_2D, _textureID);
	glTexSubImage2D(GL_TEXTURE_2D, 0, 0, 0, _width, _height, GL_RGB,
GL_UNSIGNED_BYTE, _textureBuffer);

	glEnableClientState(GL_VERTEX_ARRAY);
	glEnableClientState(GL_TEXTURE_COORD_ARRAY);

	glVertexPointer(2, GL_FLOAT, 0, _vertexes);
	glTexCoordPointer(2, GL_FLOAT, 0, _textureCoords);

	glDrawArrays(GL_TRIANGLE_FAN, 0, 4);

	glDisableClientState(GL_VERTEX_ARRAY);
	glDisableClientState(GL_TEXTURE_COORD_ARRAY);

	glDisable(GL_TEXTURE_2D);
}

int NuitrackGLSample::power2(int n)
{
	unsigned int m = 2;
	while (m < n)
		m <<= 1;

	return m;
}

void NuitrackGLSample::renderLines()
{
	if (_lines.empty())
		return;

	glEnableClientState(GL_VERTEX_ARRAY);

	glColor4f(1, 1, 1, 1);

	glLineWidth(6);

	glVertexPointer(2, GL_FLOAT, 0, _lines.data());
	glDrawArrays(GL_LINES, 0, _lines.size() / 2);

	glLineWidth(1);

	glEnable(GL_POINT_SMOOTH);
	glPointSize(16);

	glVertexPointer(2, GL_FLOAT, 0, _lines.data());
	glDrawArrays(GL_POINTS, 0, _lines.size() / 2);
```

```
	if (!_leftHandPointers.empty())
	{
		glColor4f(1, 0, 0, 1);
		glPointSize(16);
		glVertexPointer(2, GL_FLOAT, 0, _leftHandPointers.data());
		glDrawArrays(GL_POINTS, 0, 1);
		if (_leftHandPointers.size() > 2)
		{
			glPointSize(24);
			glVertexPointer(2, GL_FLOAT, 0,
_leftHandPointers.data() + 2);
			glDrawArrays(GL_POINTS, 0, 1);
		}
	}

	if (!_rightHandPointers.empty())
	{
		glColor4f(0, 0, 1, 1);
		glPointSize(16);
		glVertexPointer(2, GL_FLOAT, 0, _rightHandPointers.data());
		glDrawArrays(GL_POINTS, 0, 1);
		if (_rightHandPointers.size() > 2)
		{
			glPointSize(24);
			glVertexPointer(2, GL_FLOAT, 0,
_rightHandPointers.data() + 2);
			glDrawArrays(GL_POINTS, 0, 1);
		}
	}

	glColor4f(1, 1, 1, 1);
	glPointSize(1);
	glDisable(GL_POINT_SMOOTH);

	glDisableClientState(GL_VERTEX_ARRAY);
}

void NuitrackGLSample::initTexture(int width, int height)
{
	glGenTextures(1, &_textureID);

	width = power2(width);
	height = power2(height);

	if (_textureBuffer != 0)
		delete[] _textureBuffer;

	_textureBuffer = new uint8_t[width * height * 3];
	memset(_textureBuffer, 0, sizeof(uint8_t) * width * height * 3);

	glBindTexture(GL_TEXTURE_2D, _textureID);
	glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, width, height, 0, GL_RGB,
GL_UNSIGNED_BYTE, NULL);

	glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
	glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);

	// Set texture coordinates [0, 1] and vertexes position
```

```
	{
		_textureCoords[0] = (float) _width / width;
		_textureCoords[1] = (float) _height / height;
		_textureCoords[2] = (float) _width / width;
		_textureCoords[3] = 0.0;
		_textureCoords[4] = 0.0;
		_textureCoords[5] = 0.0;
		_textureCoords[6] = 0.0;
		_textureCoords[7] = (float) _height / height;

		_vertexes[0] = _width;
		_vertexes[1] = _height;
		_vertexes[2] = _width;
		_vertexes[3] = 0.0;
		_vertexes[4] = 0.0;
		_vertexes[5] = 0.0;
		_vertexes[6] = 0.0;
		_vertexes[7] = _height;
	}
}
```

8.4.3 NuitrackGLSample.h

```
#ifndef NUITRACKGLSAMPLE_H_
#define NUITRACKGLSAMPLE_H_

#ifndef ANDROID
#include <GL/gl.h>
#else
#include <GLES/gl.h>
#endif

#ifdef _WIN32
#include <windows.h>
#endif

#include <cstring>

#include <nuitrack/Nuitrack.h>
#include <blepp/blestatemachine.h>
#include <blepp/float.h>  //BLE uses the obscure IEEE11073 decimal
exponent floating point values



using namespace tdv::nuitrack;

/**
 * Crossplatform Nuitrack GL sample
 */
class NuitrackGLSample  //: public GLSample
{
public:
	NuitrackGLSample();
	virtual ~NuitrackGLSample();

	/**
```

```
        * Initialize sample
        */
       virtual void init(const std::string& config = "",
BLEPP::BLEGATTStateMachine* gatt = NULL);

       /**
        * Update sample
        */
       virtual bool update();

       /**
        * Release sample
        */
       virtual void release();

       BLEPP::BLEGATTStateMachine* gatt_obj;

private:
       int _width, _height;


       // GL data
       GLuint _textureID;
       uint8_t* _textureBuffer;
       GLfloat _textureCoords[8];
       GLfloat _vertexes[8];
       std::vector<GLfloat> _lines;
       std::vector<GLfloat> _leftHandPointers;
       std::vector<GLfloat> _rightHandPointers;
       std::vector<Gesture> _userGestures;

       OutputMode _outputMode;

       DepthSensor::Ptr _depthSensor;
       UserTracker::Ptr _userTracker;
       SkeletonTracker::Ptr _skeletonTracker;
       HandTracker::Ptr _handTracker;
       GestureRecognizer::Ptr _gestureRecognizer;

       /**
        * Nuitrack callbacks
        */
       void onNewDepthFrame(DepthFrame::Ptr frame);
       void onUserUpdate(UserFrame::Ptr frame);
       void onSkeletonUpdate(SkeletonData::Ptr userSkeletons);
       void onHandUpdate(HandTrackerData::Ptr handData);
       void onNewGesture(GestureData::Ptr gestureData);

       /**
        * Draw methods
        */
       void drawSkeleton(const std::vector<Joint>& joints);
       void drawBone(const Joint& j1, const Joint& j2);
       void renderTexture();
       void renderLines();

       /**
        * Bluetooth and formating functions
```

```
	 */
	void send_vector3(GLfloat x, GLfloat y, GLfloat z, std::string
joint_type);

	/**
	 * Other functions
	 */
	int power2(int n);
	void initTexture(int width, int height);
};

#endif /* NUITRACKGLSAMPLE_H_ */
```

8.5 Android VR

8.5.1 BluetoothAndroid.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using System.IO;

public class BluetoothAndroid : MonoBehaviour {

	public static BluetoothAndroid me;

	AndroidJavaClass _class;
	AndroidJavaObject instance { get { return
_class.GetStatic<AndroidJavaObject>("instance"); } }

	private string message;
	public string Message { get { return message; } }

	Dictionary <string, string> joints = new Dictionary<string,
string>();
	public Dictionary<string, string> Joints { get { return joints; }
}

	public bool hasNewData;

	StreamReader reader;

	void Awake () {
		// singleton instance
		if(me == null) {
			me = this;
			DontDestroyOnLoad(gameObject);
		}
		else Destroy(this); // or gameObject
	}

	void Start()
	{
```

```
            print ("Blutooth Android Called From game object: " +
gameObject.transform.name);
            _class = new
AndroidJavaClass("com.example.bluetooth_util.BleCommunication");
            _class.CallStatic("instantiate", gameObject.transform.name);

            hasNewData = false;
      }

      public void sendMessage(string Message){
            instance.Call("sendMessage", Message);
      }

      public void receiveMessage(string answer){
            message = answer;
            hasNewData = true;

            string[] parts = message.Split ('!');
            if (message.Length >= 2) {
                  joints [parts [0]] = parts [1];

                  if (parts[0] == "rGrab" && parts[1] == "1")
                        print ("right hand grab");

                  if (parts[0] == "lGrab" && parts[1] == "1")
                        print ("left hand clicked");

            }
      }

}
```

8.5.2 grab_release.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class grab_release : MonoBehaviour {

      bool open_hand;

      // Use this for initialization
      void Start () {

      }

      void Update(){
            open_hand = false;
      }


      void FixedUpdate() {



      }
```

```
//    void OnTriggerEnter(Collider ball){
//          print ("Trigger started");
//
//    }

      void OnTriggerStay(Collider ball){
            string grabs;
            BluetoothAndroid.me.Joints.TryGetValue("Grabs", out grabs);

            if (grabs == "01" || grabs == "00") {

            } else {
                  print ("Trigger on with hand closed");
                  ball.attachedRigidbody.MovePosition
(transform.position);
            }

//          ball.attachedRigidbody.MovePosition (transform.position);
//                transform.position;
      }

      void OnCollisionStay(Collision ball){
            string grabs;
            BluetoothAndroid.me.Joints.TryGetValue("Grabs", out grabs);
            if (grabs == "01" || grabs == "00") {
                  print ("Collision on with hand open");
            } else {
                  print ("Collision on with hand closed");
                  ball.collider.attachedRigidbody.MovePosition
(transform.position);
            }

            //                transform.position;
      }

      void OnCollisionEnter(Collision ball){
            string grabs;
            BluetoothAndroid.me.Joints.TryGetValue("Grabs", out grabs);
            if (grabs == "01" || grabs == "00") {
                  print ("Collision on with hand open");
            } else {
                  print ("Collision on with hand closed");
                  ball.collider.attachedRigidbody.MovePosition
(transform.position);
            }

            //                transform.position;
      }



//    void OnCollisionEnter(Collision ball){
//          print ("Trigger on collider");
//
//          string grabs;
//          BluetoothAndroid.me.Joints.TryGetValue("Grabs", out grabs);
//          if (grabs == "11" || grabs == "10") {
```

```
//                ball.collider.attachedRigidbody.MovePosition
(transform.position);
//                print ("Trigger on collider with closed hand");
//          } else {
//                print ("Trigger on collider with open hand");
//          }
//    }

}
```

8.5.3 MirrorReflection.cs

```
using UnityEngine;
using System.Collections;

// This is in fact just the Water script from Pro Standard Assets,
// just with refraction stuff removed.

[ExecuteInEditMode] // Make mirror live-update even when not in play
mode
public class MirrorReflection : MonoBehaviour
{
      public bool m_DisablePixelLights = true;
      public int m_TextureSizeX = 256;
      public int m_TextureSizeY = 256;
      public float m_ClipPlaneOffset = 0.07f;

      public LayerMask m_ReflectLayers = -1;

      private Hashtable m_ReflectionCameras = new Hashtable(); //
Camera -> Camera table

      private RenderTexture m_ReflectionTexture = null;
      private int m_OldReflectionTextureSizeX = 0;
      private int m_OldReflectionTextureSizeY = 0;

      private static bool s_InsideRendering = false;

      // This is called when it's known that the object will be
rendered by some
      // camera. We render reflections and do other updates here.
      // Because the script executes in edit mode, reflections for the
scene view
      // camera will just work!
      public void OnWillRenderObject()
      {
            Renderer rend = GetComponent<Renderer>();
            if (!enabled || !rend || !rend.sharedMaterial ||
!rend.enabled)
                  return;

            Camera cam = Camera.current;
            if (!cam) return;

            // Safeguard from recursive reflections.
            if( s_InsideRendering )
```

```
            return;
        s_InsideRendering = true;

        Camera reflectionCamera;
        CreateMirrorObjects( cam, out reflectionCamera );

        // find out the reflection plane: position and normal in
world space
        Vector3 pos = transform.position;
        Vector3 normal = transform.up;

        // Optionally disable pixel lights for reflection
        int oldPixelLightCount = QualitySettings.pixelLightCount;
        if( m_DisablePixelLights )
            QualitySettings.pixelLightCount = 0;

        UpdateCameraModes( cam, reflectionCamera );

        // Render reflection
        // Reflect camera around reflection plane
        float d = -Vector3.Dot (normal, pos) - m_ClipPlaneOffset;
        Vector4 reflectionPlane = new Vector4 (normal.x, normal.y,
normal.z, d);

        Matrix4x4 reflection = Matrix4x4.zero;
        CalculateReflectionMatrix (ref reflection, reflectionPlane);
        Vector3 oldpos = cam.transform.position;
        Vector3 newpos = reflection.MultiplyPoint( oldpos );
        reflectionCamera.worldToCameraMatrix =
cam.worldToCameraMatrix * reflection;

        // Setup oblique projection matrix so that near plane is our
reflection
        // plane. This way we clip everything below/above it for
free.
        Vector4 clipPlane = CameraSpacePlane( reflectionCamera, pos,
normal, 1.0f );
        //Matrix4x4 projection = cam.projectionMatrix;
        Matrix4x4 projection =
cam.CalculateObliqueMatrix(clipPlane);
        reflectionCamera.projectionMatrix = projection;

        reflectionCamera.cullingMask = ~(1<<4) &
m_ReflectLayers.value; // never render water layer
        reflectionCamera.targetTexture = m_ReflectionTexture;
        GL.SetRevertBackfacing (true);
        reflectionCamera.transform.position = newpos;
        Vector3 euler = cam.transform.eulerAngles;
        reflectionCamera.transform.eulerAngles = new Vector3(0,
euler.y, euler.z);
        reflectionCamera.Render();
        reflectionCamera.transform.position = oldpos;
        GL.SetRevertBackfacing (false);
        Material[] materials = rend.sharedMaterials;
        foreach( Material mat in materials ) {
            if( mat.HasProperty("_ReflectionTex") )
                mat.SetTexture( "_ReflectionTex",
m_ReflectionTexture );
```

```
            }

            // Restore pixel light count
            if( m_DisablePixelLights )
                  QualitySettings.pixelLightCount = oldPixelLightCount;

            s_InsideRendering = false;
      }


      // Cleanup all the objects we possibly have created
      void OnDisable()
      {
            if( m_ReflectionTexture ) {
                  DestroyImmediate( m_ReflectionTexture );
                  m_ReflectionTexture = null;
            }
            foreach( DictionaryEntry kvp in m_ReflectionCameras )
                  DestroyImmediate( ((Camera)kvp.Value).gameObject );
            m_ReflectionCameras.Clear();
      }


      private void UpdateCameraModes( Camera src, Camera dest )
      {
            if( dest == null )
                  return;
            // set camera to clear the same way as current camera
            dest.clearFlags = src.clearFlags;
            dest.backgroundColor = src.backgroundColor;
            if( src.clearFlags == CameraClearFlags.Skybox )
            {
                  Skybox sky = src.GetComponent(typeof(Skybox)) as
Skybox;
                  Skybox mysky = dest.GetComponent(typeof(Skybox)) as
Skybox;
                  if( !sky || !sky.material )
                  {
                        mysky.enabled = false;
                  }
                  else
                  {
                        mysky.enabled = true;
                        mysky.material = sky.material;
                  }
            }
            // update other values to match current camera.
            // even if we are supplying custom camera&projection
matrices,
            // some of values are used elsewhere (e.g. skybox uses far
plane)
            dest.farClipPlane = src.farClipPlane;
            dest.nearClipPlane = src.nearClipPlane;
            dest.orthographic = src.orthographic;
            dest.fieldOfView = src.fieldOfView;
            dest.aspect = src.aspect;
            dest.orthographicSize = src.orthographicSize;
      }
```

```
	// On-demand create any objects we need
	private void CreateMirrorObjects( Camera currentCamera, out
Camera reflectionCamera )
	{
		reflectionCamera = null;

		// Reflection render texture
		if( !m_ReflectionTexture || m_OldReflectionTextureSizeX !=
m_TextureSizeX  || m_OldReflectionTextureSizeY != m_TextureSizeY)
		{
			if( m_ReflectionTexture )
				DestroyImmediate( m_ReflectionTexture );
			m_ReflectionTexture = new RenderTexture(
m_TextureSizeX, m_TextureSizeY, 16 );
			m_ReflectionTexture.name = "__MirrorReflection" +
GetInstanceID();
			m_ReflectionTexture.isPowerOfTwo = true;
			m_ReflectionTexture.hideFlags = HideFlags.DontSave;
			m_OldReflectionTextureSizeX = m_TextureSizeX;
			m_OldReflectionTextureSizeY = m_TextureSizeY;
		}

		// Camera for reflection
		reflectionCamera = m_ReflectionCameras[currentCamera] as
Camera;
		if( !reflectionCamera ) // catch both not-in-dictionary and
in-dictionary-but-deleted-GO
		{
			GameObject go = new GameObject( "Mirror Refl Camera
id" + GetInstanceID() + " for " + currentCamera.GetInstanceID(),
typeof(Camera), typeof(Skybox) );
			reflectionCamera = go.GetComponent<Camera>();
			reflectionCamera.enabled = false;
			reflectionCamera.transform.position =
transform.position;
			reflectionCamera.transform.rotation =
transform.rotation;

	reflectionCamera.gameObject.AddComponent<FlareLayer>();
			go.hideFlags = HideFlags.HideAndDontSave;
			m_ReflectionCameras[currentCamera] = reflectionCamera;
		}
	}

	// Extended sign: returns -1, 0 or 1 based on sign of a
	private static float sgn(float a)
	{
		if (a > 0.0f) return 1.0f;
		if (a < 0.0f) return -1.0f;
		return 0.0f;
	}

	// Given position/normal of the plane, calculates plane in camera
space.
	private Vector4 CameraSpacePlane (Camera cam, Vector3 pos,
Vector3 normal, float sideSign)
	{
```

```
            Vector3 offsetPos = pos + normal * m_ClipPlaneOffset;
            Matrix4x4 m = cam.worldToCameraMatrix;
            Vector3 cpos = m.MultiplyPoint( offsetPos );
            Vector3 cnormal = m.MultiplyVector( normal ).normalized *
sideSign;
            return new Vector4( cnormal.x, cnormal.y, cnormal.z, -
Vector3.Dot(cpos,cnormal) );
      }

      // Calculates reflection matrix around the given plane
      private static void CalculateReflectionMatrix (ref Matrix4x4
reflectionMat, Vector4 plane)
      {
            reflectionMat.m00 = (1F - 2F*plane[0]*plane[0]);
            reflectionMat.m01 = (   - 2F*plane[0]*plane[1]);
            reflectionMat.m02 = (   - 2F*plane[0]*plane[2]);
            reflectionMat.m03 = (   - 2F*plane[3]*plane[0]);

            reflectionMat.m10 = (   - 2F*plane[1]*plane[0]);
            reflectionMat.m11 = (1F - 2F*plane[1]*plane[1]);
            reflectionMat.m12 = (   - 2F*plane[1]*plane[2]);
            reflectionMat.m13 = (   - 2F*plane[3]*plane[1]);

            reflectionMat.m20 = (   - 2F*plane[2]*plane[0]);
            reflectionMat.m21 = (   - 2F*plane[2]*plane[1]);
            reflectionMat.m22 = (1F - 2F*plane[2]*plane[2]);
            reflectionMat.m23 = (   - 2F*plane[3]*plane[2]);

            reflectionMat.m30 = 0F;
            reflectionMat.m31 = 0F;
            reflectionMat.m32 = 0F;
            reflectionMat.m33 = 1F;
      }
}
```

8.5.4 SkeletonHead.cs

```
using System;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using System.IO;

public class SkeletonHead: MonoBehaviour {

      public Text state;
      [SerializeField] GameObject jointHeadPrefab;
      [SerializeField] GameObject jointPrefab;
      [SerializeField] GameObject connectionPrefab;

      public Slider max_speed;
      public Slider lerp_factor;
      public Slider step_size;
```

```
	private Dictionary<string, GameObject> joints;
	string[] availableJoints;
	string[,] jointConnections;
	GameObject[] connections;


	// Use this for initialization
	void Start () {
		state.text = "Head";

		availableJoints = new string[]
		{
			"Head",
			"rElbow", "rWrist" , "rShoulder",
			"lElbow", "lWrist", "lShoulder",
		};

		joints = new Dictionary<string, GameObject> ();
		foreach (string joint_type in availableJoints) {
			if (joint_type == "Head"){
				GameObject tmp = (GameObject)Instantiate
(jointHeadPrefab, Vector3.zero, Quaternion.identity);
				tmp.SetActive (true);
				joints.Add(joint_type, tmp);
			} else {
				GameObject tmp = (GameObject)Instantiate
(jointPrefab, Vector3.zero, Quaternion.identity);
				tmp.SetActive (true);
				joints.Add (joint_type, tmp);
			}
		}


		connections = new GameObject[5];
		for (int i = 0; i < connections.Length; i++)
		{
			connections[i] =
(GameObject)Instantiate(connectionPrefab, Vector3.zero,
Quaternion.identity);
			connections[i].SetActive(false);
		}

		jointConnections = new string[5, 2];
		{
			jointConnections[0, 0] = "lWrist";
			jointConnections[0, 1] = "lElbow";

			jointConnections[1, 0] = "rWrist";
			jointConnections[1, 1] = "rElbow";

			jointConnections[2, 0] = "lShoulder";
			jointConnections[2, 1] = "lElbow";

			jointConnections[3, 0] = "rShoulder";
			jointConnections[3, 1] = "rElbow";

			jointConnections[4, 0] = "lShoulder";
			jointConnections[4, 1] = "rShoulder";
```

```
            }
      }

      // Update is called once per physics step
      void FixedUpdate () {
            testHeadandHandFromDict ();
      }

      void testHeadandHandFromDict () {
            string answer = BluetoothAndroid.me.Message;

            if (answer == null) {
                  state.text = "null";
                  return;
            }

            //update joints-points;
            foreach (string joint_type in availableJoints) {

                  //answer = BluetoothAndroid.me.Joints [joint_type];
                  BluetoothAndroid.me.Joints.TryGetValue(joint_type, out
answer);
                  if (answer == null)
                        continue;

                  if (answer.StartsWith ("(") && answer.EndsWith (")"))
{
                        answer = answer.Substring (1, answer.Length -
2);
                  } else if (answer.EndsWith ("?")) {
                        answer = answer.Split ('?') [0];

                        string[] positions = answer.Split (',');

                        // store as a Vector3
                        Vector3 result = new Vector3 (
                                                float.Parse (positions
[0]),
                                                float.Parse (positions
[1]),
                                                float.Parse (positions
[2]));

                        // fix the problem with the width of the joints
//                      result += new Vector3(0f, 0f, +0.007f);
                        if (joint_type == "Head") {
                              JointRepositionCalc (joints[joint_type],
result + new Vector3 (0.0f, 0.03f, 0.0f), joint_type);
                              transform.position =
joints[joint_type].GetComponent<Rigidbody> ().position;
                        } else {
                              if (joint_type == "lCollar")
                                    result += new Vector3 (0.0f, 0.05f,
0.0f);
                              if (joint_type == "lShoulder" || joint_type
== "rShoulder" )
                                    result += new Vector3(0.0f, -0.03f,
0.0f);
```

```
                              if (joint_type == "rWrist" || joint_type ==
"lWrist") {
                                    GameObject rElbow_obj;
                                    joints.TryGetValue( "rElbow", out
rElbow_obj);
                                    result +=
(Vector3.Normalize(rElbow_obj.transform.position -result) ) * -0.05f ;
                              }

                              GameObject cur_joint;
                              joints.TryGetValue (joint_type, out
cur_joint);
                              JointRepositionCalc (cur_joint, result,
joint_type);
                        }
                  }
            }

            // update connections
            for (int i = 0; i < jointConnections.Length/2; i++) {
                  Vector3 delta = joints[jointConnections[i,
1]].transform.position -
                        joints[jointConnections[i,
0]].transform.position;
                  if (delta.magnitude > 0.01f)
                  {
                        connections[i].transform.position =
joints[jointConnections[i, 0]].transform.position;
                        connections[i].transform.rotation =
Quaternion.LookRotation(delta);
                        connections[i].transform.localScale = new
Vector3(connections[i].transform.localScale.x,
connections[i].transform.localScale.y, delta.magnitude);

                        if (!connections[i].activeSelf)
connections[i].SetActive(true);
                  }
                  else
                  {
                        connections[i].SetActive(false); //joints are
too close, no need to render
                  }

            }
      }


      void JointRepositionCalc (GameObject jointPhysics, Vector3
jointPos, string joint_type){
            if (joint_type == "Head") {
                  Vector3 next_pos = Vector3.Lerp
(jointPhysics.GetComponent<Rigidbody> ().position, jointPos, 0.28f);
                  if ((next_pos - jointPhysics.GetComponent<Rigidbody>
().position).magnitude > 2.4f * Time.deltaTime) {
                        next_pos = Vector3.MoveTowards
(jointPhysics.GetComponent<Rigidbody> ().position, jointPos, 2.4f *
0.9f * Time.deltaTime);
```

```
                }
                jointPhysics.GetComponent<Rigidbody> ().MovePosition
(next_pos);
            } else {
                Vector3 next_pos = Vector3.Lerp
(jointPhysics.GetComponent<Rigidbody> ().position, jointPos,
lerp_factor.value);

                if ((next_pos - jointPhysics.GetComponent<Rigidbody>
().position).magnitude > max_speed.value * Time.deltaTime) {
                    next_pos = Vector3.MoveTowards
(jointPhysics.GetComponent<Rigidbody> ().position, jointPos,
step_size.value * Time.deltaTime);
                    state.text = "Moving hand fast!!";
                } else {
                    state.text = "Moving hand slow!!";
                }

                jointPhysics.GetComponent<Rigidbody> ().MovePosition
(next_pos);
            }
        }
}
```

9 Αναφορές

[1] I. M. Marks, *Fears and Phobias*, 1st ed. 1969.

[2] American Psychiatric Association, *American Psychiatric Association, 2013. Diagnostic and statistical manual of mental disorders (5th ed.)*. 2013.

[3] F. Amthor, *Neurobiology For Dummies*. Wiley, 2014.

[4] R. C. Kessler, P. Berglund, O. Demler, R. Jin, K. R. Merikangas, and E. E. Walters, "Lifetime Prevalence and Age-of-Onset Distributions of DSM-IV Disorders in the National Comorbidity Survey Replication," *Arch. Gen. Psychiatry*, vol. 62, no. 6, p. 593, 2005.

[5] L.-G. (Eds. . Davis III, Thompson E., Ollendick, Thomas H., Öst, *Intensive One-Session Treatment of Specific Phobias*, 2012th ed. 2012.

[6] P. Emmelkamp and T. Ehring, *The Wiley Handbook of Anxiety Disorders*, vol. I. 2014.

[7] F. S. Stinson *et al.*, "The epidemiology of DSM-IV specific phobia in the USA: results from the National Epidemiologic Survey on Alcohol and Related Conditions.," *Psychol. Med.*, vol. 37, no. March, pp. 1047–1059, 2007.

[8] "Treatment - (ADAA) Anxiety and Depression Association of America." [Online]. Available: https://www.adaa.org/finding-help/treatment.

[9] "CBT - (ADAA) Anxiety and Depression Association of America." [Online]. Available: https://www.adaa.org/finding-help/treatment/therapy.

[10] N. Paunovic and L. G. Ost, "Cognitive-behavior therapy vs exposure therapy in the treatment of PTSD in refugees," *Behav. Res. Ther.*, vol. 39, no. 10, pp. 1183–1197, 2001.

[11] C. Botella, M. Á. Pérez-Ara, J. Bretón-López, S. Quero, A. García-Palacios, and R. M. Baños, "In Vivo versus augmented reality exposure in the treatment of small animal phobia: A randomized controlled trial," *PLoS One*, vol. 11, no. 2, 2016.

[12] Y. Shiban, I. Schelhorn, P. Pauli, and A. Mühlberger, "Effect of combined multiple contexts and multiple stimuli exposure in spider phobia: A randomized clinical trial in virtual reality," *Behav. Res. Ther.*, vol. 71, pp. 45–53, 2015.

[13] A. Healey, W. Mansell, and S. Tai, "An experimental test of the role of control in spider fear," *J. Anxiety Disord.*, vol. 49, pp. 12–20, 2017.

[14] M. K. Rowe and M. G. Craske, "Effects of an expanding-spaced vs massed exposure schedule on fear reduction and return of fear," *Behav. Res. Ther.*, vol. 36, no. 7–8, pp. 701–717, 1998.

[15] B. O. Rothbaum, L. Hodges, S. Smith, J. H. Lee, and L. Price, "A controlled study of virtual reality exposure therapy for the fear of flying.," *J. Consult. Clin. Psychol.*, vol. 68, no. 6, pp. 1020–1026, 2000.

[16] G. W. Alpers, "Avoiding treatment failures in specific phobias," in *Avoiding treatment failures in the anxiety disorders*, 2010, pp. 209–227.

[17] S. M. Ryan, M. V. Strege, E. L. Oar, and T. H. Ollendick, "One session treatment for specific phobias in children: Comorbid anxiety disorders and treatment outcome," *J. Behav. Ther. Exp. Psychiatry*, vol. 54, pp. 128–134, 2017.

[18] J. Lass-Hennemann and T. Michael, "Endogenous cortisol levels influence exposure therapy in spider phobia," *Behav. Res. Ther.*, vol. 60, pp. 39–45, 2014.

[19] Y. Shiban, J. Brütting, P. Pauli, and A. Mühlberger, "Fear reactivation prior to exposure therapy: Does it facilitate the effects of VR exposure in a randomized clinical sample?," *J. Behav. Ther. Exp. Psychiatry*, vol. 46, pp. 133–140, 2015.
[20] A. M. Waters *et al.*, "Augmenting one-session treatment of children's specific phobias with attention training to positive stimuli," *Behav. Res. Ther.*, vol. 62, pp. 107–119, 2014.
[21] T. H. Ollendick, L. G. Öst, S. M. Ryan, N. N. Capriola, and L. Reuterskiöld, "Harm beliefs and coping expectancies in youth with specific phobias," *Behav. Res. Ther.*, vol. 91, pp. 51–57, 2017.
[22] L. G. Öst, "One-session treatment for specific phobias," *Behav. Res. Ther.*, vol. 27, no. 1, pp. 1–7, 1989.
[23] W. Peñate, A. Fumero, C. Viña, M. Herrero, R. J. Marrero, and F. Rivero, "A meta-analytic review of neuroimaging studies of specific phobia to small animals," *Eur. J. Psychiat*, vol. 31, no. 1, pp. 23–36, 2017.
[24] D. Freeman *et al.*, "Virtual reality in the assessment, understanding, and treatment of mental health disorders," *Psychol. Med.*, pp. 1–8, 2017.
[25] P. A. Linley, S. Harrington, and N. Garcea, *Oxford Handbook of Positive Psychology and Work*. 2009.
[26] E. R. Smith and D. M. Mackie, "Social psychology (3rd ed.).," *Social Psychology (3rd ed.)*. 2007.
[27] S. L. Marshall, P. D. Parker, J. Ciarrochi, B. Sahdra, C. J. Jackson, and P. C. L. Heaven, "Reprint of 'Self-compassion protects against the negative effects of low self-esteem: A longitudinal study in a large adolescent sample,'" *Pers. Individ. Dif.*, vol. 81, pp. 201–206, 2015.
[28] U. Orth, R. W. Robins, and L. L. Meier, "Disentangling the effects of low self-esteem and stressful events on depression: findings from three longitudinal studies.," *J. Pers. Soc. Psychol.*, vol. 97, no. 2, pp. 307–321, 2009.
[29] T. S. Hiller, M. C. Steffens, V. Ritter, and U. Stangier, "On the context dependency of implicit self-esteem in social anxiety disorder," *J. Behav. Ther. Exp. Psychiatry*, vol. 57, pp. 118–125, 2017.
[30] Dr. Sorensen, "Symptoms of Low Self-Esteem." [Online]. Available: http://www.getesteem.com/lse-symptoms/symptom-details.html.
[31] E. G. Hepper, "Self-Esteem," in *Encyclopedia of Mental Health*, 2016, pp. 80–91.
[32] D. C. Beidel, S. M. Turner, and J. C. Taylor-Ferreira, "Teaching study skills and test-taking strategies to elementary school students. The Testbusters Program.," *Behav. Modif.*, vol. 23, no. 4, pp. 630–646, 1999.
[33] C. M. H. H. Van Houtem, M. L. Laine, D. I. Boomsma, L. Ligthart, A. J. van Wijk, and A. De Jongh, "A review and meta-analysis of the heritability of specific phobia subtypes and corresponding fears," *Journal of Anxiety Disorders*, vol. 27, no. 4. pp. 379–388, 2013.