ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

Τμήμα Διδακτικής της Τεχνολογίας και Ψηφιακών Συστημάτων

Πτυχιακή Εργασία

Θέμα:

Συστήματα Επικοινωνίας Μικτής Πραγματικότητας σε Πραγματικό Χρόνο

Επιβλέπων καθηγητής: Δρ. Ν. Σγούρος

Ευσταθίου Φίλιππος

Α.Μ. ΜΕ/0456

Περίληψη

Πρωταρχικός στόχος αυτής της εργασίας είναι η μελέτη και η ανάπτυξη ενός συστήματος Μικτής Πραγματικότητας. Για τον σκοπό αυτό αναλύεται πρώτα η Μικτή Πραγματικότητα ως ερευνητική περιοχή με αναφορές στα πρόσφατα επιστημονικά επιτεύγματα καθώς και στις αρχικές ιδέες και τεχνολογίες που την καθιέρωσαν ως αντικείμενο έρευνας. Για την εργασία αυτή έγινε εκτεταμένη έρευνα για τους τομείς που μπορεί να χρησιμοποιηθεί η νέα αυτή τεχνολογία όπως επίσης και για το που πιστεύεται απο τους ειδικούς μπορεί να οδηγήσει. Το σύστημα που αναπτύχθηκε από εμάς είναι ένα απλό συστημα μεταφοράς βίντεο, ήχου, κειμένου, και γραφικών σε πραγματικό χρόνο μέσω δικτύου.

Περιεχόμενα

1. Augmented Reality – Μικτή Πραγματικότητα

Η Μικτή Πραγματικότητα είναι ένα διαρκώς επεκτεινόμενο πεδίο στην έρευνα γύρω από την Εικονική Πραγματικότητα(Virtual Reality). Το περιβάλλον γύρω μας παρέχει έναν αστείρευτο πλούτο πληροφοριών που είναι δύσκολο να αναπαραχθεί σε έναν υπολογιστή. Αυτό αποδεικνύεται από τους «κόσμους» που χρησιμοποιούνται στα εικονικά περιβάλλοντα. Οι «κόσμοι» αυτοί συνήθως είναι πολύ απλοϊκοί, όπως τα περιβάλλοντα που δημιουργούνται για τη ενεργή ψυχαγωγία και τα παιχνίδια, ενώ τα συστήματα που μπορούν να δημιουργήσουν ένα ρεαλιστικότερο περιβάλλον κοστίζουν εκατομμύρια δολάρια, όπως οι προσομοιωτές πτήσης.

Ένα σύστημα μικτής πραγματικότητας παράγει μια σύνθετη άποψη για το χρήστη. Είναι ένας συνδυασμός της πραγματικής σκηνής που βλέπει ο χρήστης και μιας εικονικής σκηνής που παράγεται από τον υπολογιστή που εμπλουτίζει την άποψη του χρήστη με τις πρόσθετες πληροφορίες. Οι περιοχές εφαρμογής που περιγράφονται στο παρακάτω κεφάλαιο αποκαλύπτουν ότι η εμπλούτιση αυτή της πραγματικότητας του χρήστη μπορεί να λάβει πολλές διάφορες μορφές. Σε όλες αυτές τις εφαρμογές η μικτή πραγματικότητα που παρουσιάζεται στο χρήστη ενισχύει την απόδοση του προσώπου και την αντίληψη για τον κόσμο.

Ο απόλυτος στόχος είναι να δημιουργηθεί ένα σύστημα έτσι ώστε ο χρήστης να μην μπορεί να διαφοροποιήσει τον πραγματικό κόσμο από την εικονική εμπλούτισή του από το σύστημα. Στο χρήστη αυτού του απόλυτου συστήματος θα φαίνεται ότι εξετάζει μια ενιαία πραγματική σκηνή. Για παράδειγμα στον ιατρικό τομέα η ακριβής εικονική αποικόνιση των εσωτερικών οργάνων του ασθενή πάνω στο σώμα του θα δημιουργούσε μία άποψη στον χειρούργο ιατρό η οποία, όχι μόνο θα ενίσχυε την απόδοσή του, αλλά και θα εξάλειφε ενδεχομένως την ανάγκη για οποιεσδήποτε άλλες προεγχειριτικές τεχνικές διορθωτικής ευθυγράμμισης κλπ

1.1 Μικτή Πραγματικότητα ή Εικονική Πραγματικότητα;

Η εικονική πραγματικότητα είναι μια τεχνολογία που καλύπτει ένα ευρύ φάσμα ιδεών. Καθορίζει μια ομπρέλα κάτω από την οποία πολλοί ερευνητές και επιχειρήσεις εκφράζουν την εργασία τους. Η φράση δημιουργήθηκε από τον Jaron Lanier τον ιδρυτή της VPL Research, μια από τις πρώτες εταιρίες που πουλούν συστήματα εικονικής πραγματικότητας. Ο όρος Εικονική Πραγματικότητα καθορίστηκε ως «ένα δημιουργούμενο από ηλεκτρονικό υπολογιστή, διαλογικό, τρισδιάστατο περιβάλλον στο οποίο ένα πρόσωπο βυθίζεται»("a computer generated, interactive, three-dimensional environment in which a person is immersed" Aukstakalnis and Blatner 1992")

Υπάρχουν τρία βασικά σημεία σε αυτόν τον ορισμό. Κατ' αρχάς, αυτό το εικονικό περιβάλλον είναι μια παραγμένη από υπολογιστή τρισδιάστατη σκηνή που απαιτεί γραφικά υψηλής απόδοσης για να παρέχει ένα επαρκές επίπεδο ρεαλισμού. Το δεύτερο σημείο είναι ότι ο εικονικός κόσμος είναι διαλογικός. Ένας χρήστης απαιτεί σε πραγματικό χρόνο απαντήσεις από το σύστημα για να είναι σε θέση να αλληλεπιδράσει με αυτό κατά τρόπο αποτελεσματικό. Το τελευταίο σημείο είναι ότι ο χρήστης «βυθίζεται» σε αυτό το εικονικό περιβάλλον. Ένα από τα ευρέως αναγνωρισμένα περιφερειακά ενός συστήματος εικονικής πραγματικότητας είναι το head mounted display που φοριέται από τους χρήστες. Αυτές οι συσκευές εμποδίζουν όλες τις πληροφορίες από τον εξωτερικό κόσμο και το παρόν στον χρήστη και του παρέχουν μια άποψη που είναι υπό τον πλήρη έλεγχο του υπολογιστή. Ο χρήστης «βυθίζεται» εντελώς σε έναν τεχνητό κόσμο και απέχει από το πραγματικό περιβάλλον. Για να εμφανιστεί ρεαλιστική αυτή η «βύθιση» το σύστημα εικονικής πραγματικότητας πρέπει να αισθανθεί με ακρίβεια πώς ο χρήστης κινείται και να καθορίσει ποια επίδραση θα έχει στη σκηνή που προβάλλεται στο head mounted display.

Τα παραπάνω δίνουν έμφαση στις ομοιότητες και τις διαφορές μεταξύ της Εικονικής Πραγματικότητας και της Μικτής Πραγματικότητας. Μια πολύ εύκολα ορατή διαφορά μεταξύ αυτών των δύο τύπων συστημάτων είναι το immersiveness του

συστήματος-ο βαθμός δηλαδή της «βύθισης» του χρήστη στο σύστημα. Η Εικονική Πραγματικότητα προσπαθεί για ένα συνολικά «βυθισμένο»(immersive) περιβάλλον. Η αίσθηση της όρασης, και σε μερικά συστήματα της ακοής και άλλων αισθήσεων, είναι υπό τον έλεγχο του συστήματος. Αντίθετα, ένα σύστημα Μικτής Πραγματικότητας εμπλουτίζει το περιβάλλον του πραγματικού κόσμου, άρα απαιτεί από τον χρήστη να διατηρεί μια αίσθηση της παρουσίας στον κόσμο αυτό. Οι εικονικές σκηνές συγχωνεύονται με τον πραγματικό κόσμο για να δημιουργήσουν την μικτή άποψη. Πρέπει δηλαδή να υπάρχει ένας μηχανισμός για να συνδυαστεί ο πραγματικός και ο εικονικός κόσμος που δεν είναι αναγκαίος ή χρήσιμος σε άλλη εργασία εικονικής πραγματικότητας. Η ανάπτυξη της τεχνολογίας για τη συγχώνευση των πραγματικών και εικονικών streams είναι ένα ενεργό ερευνητικό θέμα και περιγράφεται εν συντομία στην παράγραφο 3.3.

Τα εικονικά αντικείμενα που δημιουργούνται από το σύστημα πρέπει να καταχωρηθούν(register) ακριβώς με τον πραγματικό κόσμο σε όλες τις διαστάσεις. Τα λάθη στην καταχώρηση(registration) θα αποτρέψουν στο χρήστη να δει τις πραγματικές και εικονικές σκηνές όπως θα έπρεπε να συνδυάζονται. Η σωστή καταχώρηση πρέπει επίσης να διατηρείται ενώ ο χρήστης κινείται μέσα στο πραγματικό περιβάλλον. Οι αποκλίσεις ή οι αλλαγές στην καταχώρηση δύνανται τόσο σε απόσπαση, πράγμα που καθιστά την εργασία με την Μικτή Πραγματικότητα δυσκολότερη, όσο και φυσική ενόχληση για το χρήστη, πράγμα που καθιστά το σύστημα απολύτως ακατάλληλο προς χρήση.

Ένα immersive σύστημα εικονικής πραγματικότητας πρέπει να διατηρεί την καταχώρηση έτσι ώστε οι αλλαγές στη σκηνή ταιριάζουν με την αντίληψη του χρήστη. Οποιαδήποτε λάθη εδώ είναι αντικρουόμενες αντιλήψεις μεταξύ του οπτικού συστήματος και των κιναισθητικών ή άλλων αισθητήριων συστημάτων. Το φαινόμενο της οπτικής σύλληψης δίνει στο οπτικό σύστημα μια ισχυρότερη επιρροή στην αντίληψή μας (Welch 1978). Αυτό θα επιτρέψει στον χρήστη να αποδεχτεί ή να προσαρμοστεί σε ένα οπτικό ερέθισμα που αγνοεί τις αποκλίσεις από τα ερεθίσματα των άλλων αισθητήριων συστημάτων.

Αντίθετα, τα λάθη στην καταχώρηση σε ένα σύστημα Μικτής Πραγματικότητας είναι μεταξύ δύο οπτικών ερεθισμάτων που ο χρήστης προσπαθεί να συγχωνεύσει για να

τα δει ως μια σκηνή. Υπάρχει δυστυχώς μια αυξημένη ευαισθησία σε αυτά τα λάθη (Azuma 1993, 1995).

Ο Milgram (Milgram, Kishino 1994) περιγράφει μια ταξινόμηση που δείχνει πώς η Μικτή Πραγματικότητα και η Εικονική Πραγματικότητα συσχετίζονται. Καθορίζει τη συνέχεια Πραγματικότητας-Εικονικότητας(Reality-Virtuality continuum) που παρουσιάζεται στο παρακάτω σχήμα:

Ο πραγματικός κόσμος και ένα συνολικά εικονικό περιβάλλον είναι στις δύο άκρες αυτής της συνέχειας ενώ η μέση περιοχή καλείται Μικτή Πραγματικότητα. Η «αυξημένη πραγματικότητα»(Augmented Reality) βρίσκεται κοντά στην άκρη της γραμμής που είναι ο πραγματικός κόσμος, ο οποίος υπερισχύει στην αντίληψη του χρήστη, ενώ παράλληλα αυξάνεται από τα στοιχεία που παράγονται από τον υπολογιστή. Αυξημένη Εικονικότητα(Augmented Virtuality) είναι ένας όρος που δημιουργείται από το Milgram για να προσδιορίσει τα συστήματα που είναι κυρίως συνθετικά με κάποια στοιχεία του πραγματικού κόσμου που προστίθενται όπως texture mapping video επάνω σε εικονικά αντικείμενα. Αυτή είναι μια διάκριση που τελικά θα εξασθενίσει καθώς η τεχνολογία βελτιώνεται και τα εικονικά στοιχεία στη σκηνή γίνονται λιγότερο διακριτά από τα πραγματικά. Για τον λόγο αυτόν όλο και συχνότερα βλέπουμε τους όρους Mixed Reality και Augmented Reality ουσιαστικά να ταυτίζονται.

Ο Milgram καθορίζει μια περαιτέρω ταξινόμηση για τα συστήματα Μικτής Πραγματικότητας. Οι τρεις άξονες που προτείνει για την ταξινόμηση αυτών των συστημάτων είναι: Πιστότητα Αναπαραγωγής(Reproduction Fidelity), Έκταση της Μεταφοράς Παρουσίας(Extent of Presence Metaphor) και Έκταση της Γνώσης του Κόσμου(Extent of World Knowledge). Η Πιστότητα Αναπαραγωγής αφορά την ποιότητα των στοιχείων που παράγονται από τον υπολογιστή, τα οποία κυμαίνονται

από απλές προσεγγίσεις πλαισίων ως πλήρως φωτορεαλιστικές αποδόσεις. Ο περιορισμός του πραγματικού χρόνου στα συστήματα Μικτής Πραγματικότητας τα αναγκάζει να κυμαίνονται σε χαμηλά επίπεδα στο φάσμα της Πιστότητας Αναπαραγωγής. Οι ικανότητες των επεξεργαστών γραφικών προς το παρόν δεν μπορούν να παράγουν φωτορεαλιστικές αποδόσεις της εικονικής σκηνής σε πραγματικό χρόνο.

Ο Milgram τοποθετεί επίσης τα συστήματα Μικτής Πραγματικότητας σε χαμηλά επίπεδα στον άξονα της Έκτασης της Μεταφοράς Παρουσίας. Αυτός ο άξονας μετρά το επίπεδο «βύθισης» του χρήστη μέσα στην επιδειχθείσα σκηνή. Αυτή η κατηγοριοποίηση είναι στενά συνδεδεμένη στην τεχνολογία απεικόνισης που χρησιμοποιείται από το σύστημα. Υπάρχουν διάφορες κατηγορίες τέτοιων μέσων απεικόνισης που χρησιμοποιούνται στα συστήματα Μικτής Πραγματικότητας, τα οποία εξετάζονται στην παράγραφο 3.3. Κάθε ένα από αυτά δίνει μια διαφορετική αίσθηση της βύθισης στην σκηνή. Σε ένα σύστημα Μικτής Πραγματικότητας, αυτό μπορεί να είναι παραπλανητικό επειδή με μερικές τεχνολογίες απεικόνισης μέρος της «εικόνας» είναι η άμεση άποψη του χρήστη του πραγματικού κόσμου. Η βύθιση σε ένα τέτοιο σύστημα προέρχεται απλά από το να έχει ο χρήστης τα μάτια του ανοιχτά και φυσικά αντιπαραβάλλεται στα συστήματα όπου η συγχωνευμένη άποψη παρουσιάζεται στο χρήστη σε ένα χωριστό όργανο ελέγχου, το οποίο καλείται άποψη «παράθυρο στον κόσμο»("Window on the World" view).

Η τρίτη, και τελευταία, διάσταση που χρησιμοποιεί ο Milgram για την ταξινόμηση των συστημάτων Μικτής Πραγματικότητας είναι Έκταση της Γνώσης του Κόσμου. Η Μικτή Πραγματικότητα δεν σημαίνει απλά την υπέρθεση ενός γραφικού αντικειμένου πάνω σε μια σκηνή του πραγματικού κόσμου. Αυτό είναι τεχνικά ένας εύκολος στόχος. Μια από τις δυσκολίες, όπως καθορίζεται εδώ, είναι η ανάγκη να διατηρηθεί η ακριβής καταχώρηση των εικονικών αντικειμένων με την εικόνα του πραγματικού κόσμου. Όπως θα περιγραφεί στην παράγραφο 3.5, αυτό απαιτεί συχνά τη λεπτομερή γνώση της σχέσης μεταξύ των πλαισίων αναφοράς για τον πραγματικό κόσμο, την κάμερα που τον καταγράφει και το χρήστη. Σε μερικές περιπτώσεις χρήσης αυτές οι σχέσεις είναι γνωστές πράγμα που καθιστά το έργο της μικτής πραγματικότητας ευκολότερο ή οδηγεί τον σχεδιαστή του συστήματος να χρησιμοποιήσει ένα απολύτως εικονικό περιβάλλον.

2. Πεδία εφαρμογής της Μικτής Πραγματικότητας

Μόνο πρόσφατα αναπτύχθηκαν τεχνολογίες, όπως επεξεργασία βίντεο σε πραγματικό χρόνο, τα συστήματα γραφικών των ηλεκτρονικών υπολογιστών και οι νέες τεχνολογίες απεικόνισης, τόσο ώστε να συγκλίνουν για να καταστήσουν δυνατή την απεικόνιση μιας εικονικής σκηνής η οποία είναι σωστά καταχωρημένη με μια άποψη του τρισδιάστατου περιβάλλοντος που περιβάλλει το χρήστη. Οι ερευνητές που εργάζονται με τα συστήματα Μικτής Πραγματικότητας, τα έχουν προτείνει ως λύσεις σε πολλά πεδία. Οι θεματικές περιοχές που έχουν συζητηθεί περιλαμβάνουν από ψυχαγωγία μέχρι στη στρατιωτική εκπαίδευση. Για πολλούς από τους τομείς αυτούς, όπως ο ιατρικός τομέας (Rosen, Laub 1996), έχουν προταθεί επίσης και τα παραδοσιακά συστήματα Εικονικής Πραγματικότητας. Αυτό το κεφάλαιο θα δώσει έμφαση σε μερικές από τις προτεινόμενες εφαρμογές για συστήματα Μικτής Πραγματικότητας. Μεγάλη συλλογή από ongoing projects πάνω στην Μικτή Πραγματικότητα σε όλους τους τομείς που περιγράφονται παρακάτω μπορεί κανείς να βρει στις διευθύνσεις http://vrlab.epfl.ch/ και http://www.augmented-reality.org/, οι οποίες ερευνήθηκαν σε βάθος για την συγγραφή του εγγράφου αυτού.

2.1 Ιατρικός Τομέας

Επειδή η τεχνολογία απεικόνισης είναι τόσο διάχυτη σε όλο τον ιατρικό τομέα, δεν εκπλήσσει το γεγονός ότι ο τομέας αυτός αντιμετωπίζεται ως ένας από τους σημαντικότερους για τα συστήματα Μικτής Πραγματικότητας. Οι περισσότερες από τις ιατρικές εφαρμογές έχουν να κάνουν με χειρουργικές επεμβάσεις καθοδηγούμενες από εικόνα. Προεγχειριτικές μελέτες απεικόνισης, όπως οι εξετάσεις CT ή MRI, από τον ασθενή παρέχουν στο χειρούργο την απαραίτητη άποψη της εσωτερικής ανατομίας. Από αυτές τις εικόνες προγραμματίζεται η χειρουργική επέμβαση. Απεικόνιση της πορείας μέσω της ανατομίας στην επηρεασθείσα περιοχή όπου, παραδείγματος χάριν, ένας όγκος πρέπει να αφαιρεθεί γίνεται πρώτα με την

δημιουργία ενός τρισδιάστατου προτύπου από τις πολλαπλάσιες απόψεις που συγκεντρώθηκαν στην προεγχειριτική μελέτη. Αυτό συχνά γίνεται διανοητικά αν και μερικά συστήματα δημιουργούν τρισδιάστατες απεικονίσεις του όγκου από τη μελέτη της εικόνας. Η Μικτή Πραγματικότητα μπορεί να εφαρμοστεί έτσι ώστε η χειρουργική ομάδα μπορεί να δει τα στοιχεία των εξετάσεων CT ή MRI που καταχωρούνται σωστά στον ασθενή στο χειρουργικό τραπέζι ενώ η διαδικασία προχωρεί. Το να είναι σε θέση το σύστημα να καταχωρεί με ακρίβεια τέτοιες εικόνες θα ενισχύσει την απόδοση της χειρουργικής ομάδας και θα εξαλείψει την ανάγκη για τα επίπονα και δυσκίνητα στερεοστατικά πλαίσια που χρησιμοποιούνται προς το παρόν για την καταχώρηση.

Πολλά ακόμη papers για την συγκεκριμένη εφαρμογή βρέθηκαν αλλά δεν αναλύονται περαιτέρω στο παρόν έγγραφο επειδή ξεφεύγουν από τα όρια του θέματος (Lorensen, Cline 1993, Grimson, Lozano-Perez 1994, Betting, Feldmar 1995, Grimson, Ettinger 1995, Mellor 1995, Uenohara, Kanade 1995).

Μια άλλη εφαρμογή για την Μικτή Πραγματικότητα στον ιατρικό τομέα είναι στην απεικόνιση υπερήχου (State, Chen 1994). Χρησιμοποιώντας μια διάφανη συσκευή απεικόνισης ο τεχνικός υπερήχου μπορεί να δει μια ογκομετρική εικόνα του εμβρύου που επιστρώνεται στην κοιλιά της εγκύου. Η εικόνα εμφανίζεται σαν ήταν μέσα της κοιλίας και δίνεται σωστά καθώς ο χρήστης κινείται όπως φαίνεται και στην παρακάτω εικόνα:

Πολλά ακόμα ongoing projects στον τομέα αυτόν υπάρχουν όπως πχ το EVA Project (http://www.mindtel.com/projects/sa/projects/eva/), το ARIS*ER (www.ariser.info/), το MEDARPA (http://www.medarpa.de/).Επίσης τα: MIAS (Minimally Invasive Articular Surgery), DRAMA (Developments in Rehabilitation of the Arm - A Multimedia Approach), JUST (Training of non professionals in Health Emergency Situations by Virtual Reality), EPFL-UNIL PHOBIA (Virtual Reality in Psychiatry), NCCR CO-ME (Computer-Aided and Image Guided Medical Intervention) για τα οποία περισσότερες πληροφορίες μπορούν να βρεθούν στην διεύθυνση http://vrlab.epfl.ch/Projects/projects_index.html.

2.2 Ψυχαγωγία

Μια απλή μορφή Μικτής Πραγματικότητας στην ψυχαγωγία είναι σε λειτουργία εδώ και αρκετό καιρό. Όποτε παρακολουθούμε την πρόβλεψη του καιρού ο δημοσιογράφος παρουσιάζεται να στέκεται μπροστά από τους μεταβαλλόμενους καιρικούς χάρτες. Στο στούντιο ο δημοσιογράφος στέκεται πραγματικά μπροστά από μια μπλε ή πράσινη οθόνη. Αυτή η πραγματική εικόνα εμπλουτίζεται από

δημιουργημένους από τον ηλεκτρονικό υπολογιστή χάρτες χρησιμοποιώντας μια τεχνική αποκαλούμενη χρωματική διαμόρφωση.

Είναι επίσης δυνατό να δημιουργηθεί ένα εικονικό περιβάλλον στούντιο έτσι ώστε οι χρήστες μπορούν να εμφανίζονται τοποθετημένοι σε ένα στούντιο με παραγόμενη από υπολογιστή διακόσμηση. Υπάρχουν πολλά παραδείγματα χρήσης της τεχνικής αυτής(Schmidt 1996) αλλά το πιο γνωστό στο ευρύ κοινό είναι το παιχνίδι EyeToy που κυκλοφορεί για την κονσόλα Playstation 2 το οποίο χρησιμοποιεί μια απλή κάμερα για καταγράψει την πραγματική μορφή των χρηστών και τις κινήσεις τους. Στην συνέχεια τοποθετεί τους χρήστες σε διαφορετικά εικονικά περιβάλλοντα που ποικίλουν από ένα εικονικό στούντιο για να παίξουν ένα εικονικό τηλεπαιχνίδι μέχρι τις όχθες ενός ποταμού για ψάρεμα και την άκρη ενός γκρεμού που προσπαθούν να αποφύγουν αντικείμενα που πέφτουν.

Στα ειδικά εφέ στον κινηματογράφο χρησιμοποιούνται ψηφιακά μέσα για να δημιουργηθούν αυταπάτες στο θεατή (Pyros, Goren 1995). Για την ακρίβεια όμως, με την τρέχουσα τεχνολογία αυτό δεν θεωρείται Μικτή Πραγματικότητα επειδή δεν γίνεται σε πραγματικό χρόνο. Τα περισσότερα ειδικά εφέ δημιουργούνται offline, frame προς frame με μερική αλληλεπίδραση των ηθοποιών και χρήση συστημάτων παραγωγής τρισδιάστατων γραφικών. Παρόλα αυτά, κάποιες από τις εργασίες προχωρούν ως την ανάλυση των πλάνων δράσης για να καθοριστούν οι παράμετροι για τις κάμερες(όπως γωνίες, αποστάσεις κλπ) ώστε να χρησιμοποιηθούν σωστά και να οδηγήσουν στην ορθή και πιστευτή παραγωγή των εικονικών αντικειμένων που συγχωνεύονται στην εικόνα (Zorpette 1994).

Στην συνέχεια ένα ακόμα παράδειγμα που στις μέρες μας έχει διαδεδομένη χρήση. Ο ηλεκτρονικός πίνακας διαφημίσεων Princeton(Princeton Electronic Billboard) είναι ένα σύστημα Μικτής Πραγματικότητας που επιτρέπει στο εκάστοτε κανάλι που μεταδίδει ζωντανά έναν αγώνα να παρεμβάλει διαφημίσεις σε συγκεκριμένους τομείς της εικόνας της μετάδοσης (National Association of Broadcasters 1994). Παραδείγματος χάριν, μεταδίδοντας έναν ποδοσφαιρικό αγώνα αυτό το σύστημα είναι σε θέση να τοποθετήσει μια διαφήμιση στην εικόνα έτσι ώστε εμφανίζεται στον εξωτερικό τοίχο του σταδίου ή ακόμα – όπως συχνότερα εμφανίζεται στην χώρα μας- και στο χορτάρι του γηπέδου.

Ο ηλεκτρονικός πίνακας διαφημίσεων απαιτεί την ευθυγράμμιση στο γήπεδο με τη λήψη εικόνων από τις χαρακτηριστικές γωνίες που βιντεοσκοπούν οι κάμερες και τις ρυθμίσεις ζουμ προκειμένου να δημιουργηθεί ένας χάρτης του γηπέδου συμπεριλαμβανομένων των θέσεων στις λήψεις όπου οι διαφημίσεις θα παρεμβληθούν. Με τη χρησιμοποίηση των προ-διευκρινισμένων σημείων αναφοράς στο γήπεδο, το σύστημα καθορίζει αυτόματα τη γωνία της κάμερας που χρησιμοποιείται και η αναφορά στον προκαθορισμένο χάρτη του γηπέδου παρεμβάλλει τη διαφήμιση στη σωστή θέση, με την κατάλληλη γωνία κλπ.

2.3 Στρατιωτική Εκπαίδευση

Στα πολεμικά αεροσκάφη εδώ και αρκετό καιρό χρησιμοποιούν απεικόνιση που παρουσιάζει τις πληροφορίες στον πιλότο στο γυαλί του πιλοτηρίου ή του κράνους πτήσης τους. Αυτό είναι μια μορφή απεικόνισης Μικτής Πραγματικότητας. Το SIMNET, ένα κατανεμημένο σύστημα προσομοίωσης πολεμικών παιχνιδιών, χρησιμοποιεί επίσης την τεχνολογία Μικτής Πραγματικότητας. Εξοπλίζοντας το στρατιωτικό προσωπικό με τοποθετημένες στο κράνος συσκευές απεικόνισης(helmet mounted visor displays) ή ένα ειδικό σύστημα που μετράει αποστάσεις (Urban 1995) οι δραστηριότητες άλλων μονάδων που συμμετέχουν στην άσκηση μπορούν να είναι εικονικές. Για παράδειγμα, ο εξοπλισμένος με τα συστήματα αυτά στρατιώτης κοιτώντας στον ορίζοντα μπορεί να δει ένα ελικόπτερο να ξεπροβάλει από τα δέντρα (Metzger 1993). Αυτό το ελικόπτερο θα μπορούσε να οδηγείται στην προσομοίωση από έναν άλλο συμμετέχοντα.

Σε πραγματικές συνθήκες, η απεικόνιση κατά αυτό τον τρόπο πάνω από την πραγματική σκηνή της μάχης θα μπορούσε να παρέχει περισσότερες πληροφορίες όπως σχολιασμός(εχθρική ή φιλική μονάδα κλπ) ή «υπογράμμιση» για να τονίσει τις κρυμμένες εχθρικές μονάδες κλπ.

2.4 Εφαρμοσμένη Μηχανική

Ας υποθέσουμε ότι μια ομάδα σχεδιαστών εργάζεται στο πρότυπο μιας σύνθετης συσκευής για τους πελάτες τους. Οι σχεδιαστές και οι πελάτες θέλουν να κάνουν μια κοινή αναθεώρηση σχεδίου ακόμα κι αν είναι φυσικά χωρισμένοι-δεν είναι λίγες οι φορές που τα μεγαλύτερα έργα οπουδήποτε στον κόσμο ανατίθενται σε ομάδες σχεδιαστών που βρίσκονται σε διαφορετική ήπειρο. Εάν πελάτες και σχεδιαστές είχαν μια αίθουσα διασκέψεων που ήταν εξοπλισμένη με μιας μορφής απεικόνισης Μικτής Πραγματικότητας αυτό θα ήταν εύκολο. Το φυσικό πρωτότυπο που οι σχεδιαστές έχουν κατασκευάσει και τοποθετήσει στην αίθουσα τους έχει γίνει παράλληλα και εικονικό ώστε να προβάλλεται στην αίθουσα διασκέψεων των πελατών τρισδιάστατο. Οι πελάτες μπορούν να περπατήσουν γύρω από την τρισδιάστατη απεικόνιση εξετάζοντας τις διαφορετικές πτυχές του έργου. Για να καταθέσει ο πελάτης τις απόψεις του μπορεί να δείξει στο πρωτότυπο τα κυριότερα τμήματα, μέρη κλπ και αυτά θα απεικονιστούν στο πραγματικό πρότυπο με την απεικόνιση του συστήματος Μικτής Πραγματικότητας που οι σχεδιαστές χρησιμοποιούν.

Ένα ακόμα μεγαλύτερο βήμα ίσως σε ένα πιο αρχικό στάδιο του σχεδίου, προτού ακόμα να χτιστεί ένα πρωτότυπο, να απεικονίζεται και στις δύο αίθουσες μια παραγόμενη από τον υπολογιστή εικόνα του τρέχοντος σχεδίου, η οποία δημιουργείται από τα αρχεία CAD που το περιγράφουν. Αυτό θα επέτρεπε μία πραγματικού χρόνου αλληλεπίδραση με τα στοιχεία του σχεδίου έτσι ώστε καθεμία πλευρά να μπορεί να κάνει τις προσαρμογές και τις αλλαγές που θέλει, οι οποίες απεικονίζονται αμέσως και στις δύο αίθουσες (Ahlers, Kramer 1995).

2.5 Ρομποτική και Τηλερομποτική

Στην περιοχή της ρομποτικής και της τηλερομποτικής μια απεικόνιση Μικτής Πραγματικότητας μπορεί να βοηθήσει το χρήστη του συστήματος (Kim, Schenker 1993, Milgram, Zhai 1993). Ένας χειριστής τηλερομπότ χρησιμοποιεί μια οπτική εικόνα του απομακρυσμένου χώρου εργασίας για να καθοδηγήσει το ρομπότ. Οι σημειώσεις στην πραγματική εικόνα θα ήταν χρήσιμες ακόμα και αν η πραγματική σκηνή ήταν μπροστά στο χειριστή. Ειδικά όμως στην περίπτωση του χειρισμού από απόσταση υπάρχει ένα προστιθέμενο πιθανό όφελος. Δεδομένου ότι συχνά η άποψη της απομακρυσμένης σκηνής είναι μονοσκοπική, η μίξη της πραγματικότητας με τα σχέδια των δομών των κατασκευών που εμφανίζονται στην εικόνα μπορεί να διευκολύνει την κατανόηση της μακρινής τρισδιάστατης γεωμετρίας.

Εάν ο χειριστής προσπαθεί να κινήσει το ρομπότ αρχικά η κίνηση αυτή θα μπορούσε να γίνει από ένα εικονικό ρομπότ που απεικονίζεται ως μίξη πραγματικότητας στην πραγματική σκηνή. Ο χειριστής μπορεί να αποφασίσει να συνεχίσει με την κίνηση αφού δει τα αποτελέσματα που είχε με το εικονικό ρομπότ.

2.6 Κατασκευαστικός Τομέας

Ένα ήδη project στον τομέα αυτόν από το Columbia University είναι ένα σύστημα Μικτής Πραγματικότητας που καθοδηγεί τον χρήστη στην σωστή συναρμολόγηση ενός δικτυώματος –μέρος συνήθως των σκελετών διαφόρων μεγάλου μεγέθους κτισμάτων. Η ορθή κατασκευή αυτών έχει μεγάλη σημασία στην βιωσιμότητα των κτισμάτων, αφού αυτά είναι που συγκρατούν τα φορτία στα οποία εκτίθενται τα κτίσματα, ενώ ταυτόχρονα είναι επιρρεπή σε λάθη. Αυτό συμβαίνει γιατί οι δοκοί που χρησιμοποιούνται είναι συνήθως παρόμοιων μεγεθών και πάχους ενώ η αντοχή τους ποικίλει αυξάνοντας τις πιθανότητες για ανθρώπινο λάθος, πράγμα που ενδεχομένως

θα είχε καταστροφικές συνέπειες στο μέλλον. Η συγκεκριμένη πρόταση χρησιμοποιεί εικονικές δοκούς για να δείξει στο χρήστη την σωστή τους θέση στην κατασκευή, ενώ παράλληλα τον καθοδηγεί βήμα προς βήμα ακουστικά και απαιτεί την σάρωση των barcodes των δοκών για να επαληθεύσει την ορθή τους τοποθέτηση. Περισσότερες λεπτομέρειες για το συγκεκριμένο project: http://www1.cs.columbia.edu/graphics/projects/arc/arc.html

Οι ερευνητές της Boeing αναπτύσσουν μια συσκευή απεικόνισης Μικτής Πραγματικότητας για να αντικαταστήσουν τα μεγάλα πλαίσια εργασίας που χρησιμοποιούνται για την παραγωγή των καλωδιώσεων για τα αεροσκάφη τους (Caudell 1994 Sims 1994). Χρησιμοποιώντας αυτό το πειραματικό σύστημα, οι τεχνικοί καθοδηγούνται από την απεικόνιση Μικτής Πραγματικότητας που παρουσιάζει τη δρομολόγηση των καλωδίων. Περισσότερες λεπτομέρειες http://www.boeing.com/defense-space/aerospace/training/instruct/augmented.htm

2.7 Συντήρηση και Επισκευή

Όταν ο τεχνικός συντήρησης πρέπει να δουλέψει σε ένα νέο ή άγνωστο για αυτόν κομμάτι εξοπλισμού αντί να ανοίξει διάφορα εγχειρίδια επισκευής θα μπορούσε να χρησιμοποιήσει ένα σύστημα Μικτής Πραγματικότητας. Σε αυτήν την επίδειξη η εικόνα του εξοπλισμού θα βελτιωνόταν με εικονικούς σχολιασμούς και πληροφορίες σχετικά με την επισκευή. Παραδείγματος χάριν, η θέση των συνδέσμων και του υλικού σύνδεσης που πρέπει να αφαιρεθεί θα τονιζόταν. Κατόπιν η εσωτερική άποψη της μηχανής θα έδινε έμφαση στους πίνακες που πρέπει να αντικατασταθούν κλπ (Feiner, MacIntyre 1993, Uenohara, Kanade 1995).

Ακόμη ένα παράδειγμα χρήσης Μικτής Πραγματικότητας για τη συντήρηση είναι το KARMA(Knowledge-based Augmented Reality for Maintenance Assistance) από μία έρευνα που έκανε το Columbia University. Περισσότερες λεπτομέρειες υπάρχουν στο Διαδίκτυο στην σελίδα http://www1.cs.columbia.edu/graphics/projects/karma/.

Ο αμερικανικός στρατός έχει αναπτύξει ένα ασύρματο γιλέκο που φοριέται από το προσωπικό το οποίο είναι συνδεμένο με μια διάφανη συσκευή απεικόνισης (Urban 1995). Η ασύρματη σύνδεση επιτρέπει στο στρατιώτη για να έχει πρόσβαση στα εγχειρίδια επισκευής και τις εικόνες του εξοπλισμού. Οι μελλοντικές εκδόσεις θα καταχωρούν τις εικόνες αυτές στη πραγματική σκηνή και θα παρέχουν animation με τις διαδικασίες που πρέπει να εκτελεσθούν.

2.8 Καταναλωτικός Τομέας

Τα συστήματα εικονικής πραγματικότητας χρησιμοποιούνται ήδη στον καταναλωτικό τομέα. Χρησιμοποιώντας ίσως περισσότερο συστήματα δημιουργίας γραφικών παρά εικονική πραγματικότητα, όταν πηγαίνει κάποιος σε ένα κατάστημα με σκοπό να εγκαταστήσει ένα ακόμη όροφο στο σπίτι του, του παρουσιάζεται μια τρισδιάστατη

εικόνα με το πώς θα είναι το σπίτι μετά την περάτωση του έργου. Είναι κατανοητό ότι ένα μελλοντικό σύστημα θα επέτρεπε στον πελάτη να φέρει ένα βίντεο με πλάνα του σπιτιού σας από διάφορες γωνίες και σε πραγματικό χρόνο με μικτή πραγματικότητα θα του επέτρεπε να δει το σπίτι του με τον νέο όροφο στην τελική του μορφή. Ή ένα βήμα παρακάτω το σύστημα Μικτής Πραγματικότητας να αντικαθιστούσε τα παλιά έπιπλα με εικονικά καινούρια στην κουζίνα σε πραγματικό χρόνο χρησιμοποιώντας αντίστοιχα το βίντεο από την κουζίνα του πελάτη στην μορφή που είναι τώρα.

Οι εφαρμογές που θα ωφελούσαν τη βιομηχανία μόδας και καλλυντικών με την χρήση ενός συστήματος Μικτής Πραγματικότητας είναι επίσης εύκολα αντιληπτές. Εάν το κατάστημα ρούχων δεν έχει ένα συγκεκριμένο φόρεμα στο μέγεθός του πελάτη, κάποιο άλλο κατάλληλου μεγέθους φόρεμα θα μπορούσε να χρησιμοποιηθεί για να βελτιώσει την εικόνα του με το φόρεμα που επέλεξε. Δεδομένου ότι κοιτάει στον τρίπλευρο καθρέφτη θα βλέπει την εικόνα του νέου φορέματος στο σώμα του Αλλαγές στο μήκος, την μορφή των ώμων ή άλλες λεπτομέρειες του σχεδίου θα μπορούσαν να αντιμετωπισθούν ενώπιον του πριν την παραγγελία.

Όταν πηγαίνει κάποιος σε μερικά καταστήματα ομορφιάς υψηλής τεχνολογίας σήμερα μπορεί να δει πως ένα νέο κούρεμα θα του ταίριαζε σε μια «βελτιωμένη» εικόνα του. Με ένα προηγμένο σύστημα Μικτής Πραγματικότητας θα ήταν σε θέση να δει τον εαυτό του από διαφορετικές απόψεις καθώς κινείται. Εάν η δυναμική των μαλλιών συμπεριλαμβάνεται στην περιγραφή του εικονικού αντικειμένου θα έβλεπε επίσης την κίνηση των μαλλιών του κατά την κίνηση του κεφαλιού του.

3. Περιγραφή ενός Συστήματος Μικτής Πραγματικότητας

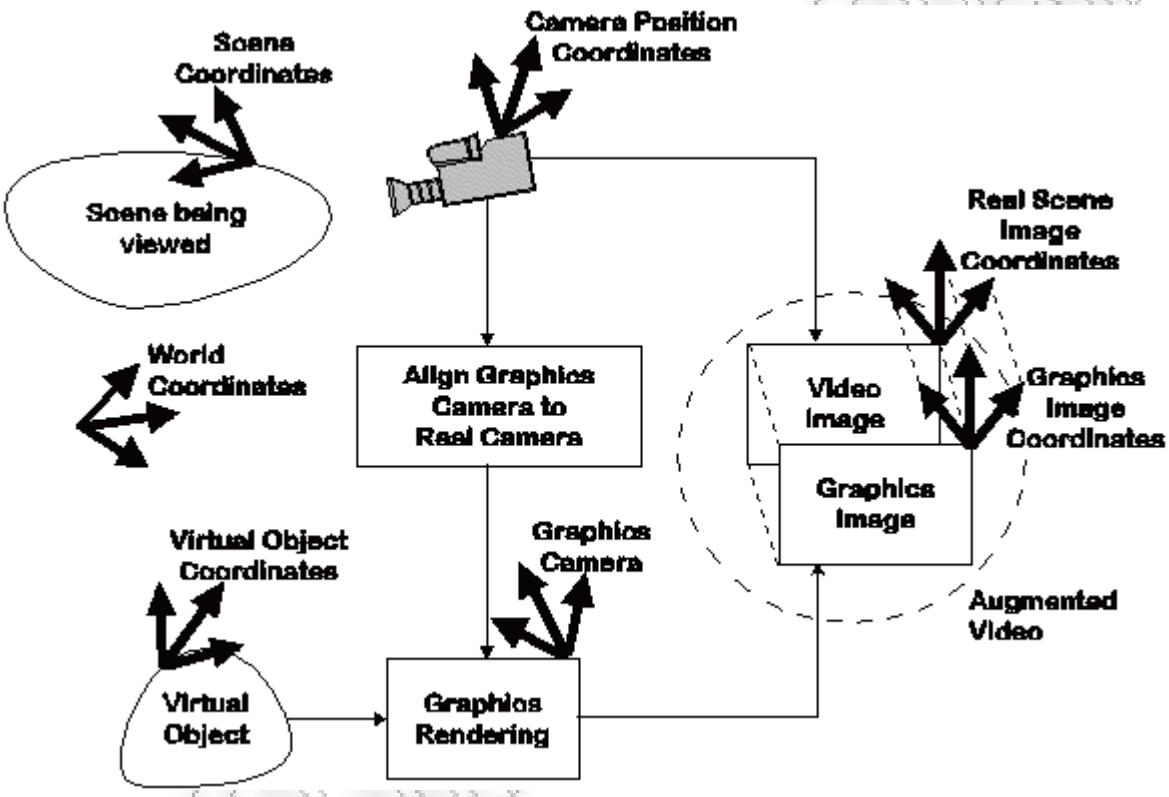
Αυτό το κεφάλαιο θα περιγράψει τα συστατικά που αποτελούν ένα χαρακτηριστικό σύστημα Μικτής Πραγματικότητας. Παρά τα διαφορετικά πεδία που μπορούν να εφαρμοστούν τα οποία αναφέραμε παραπάνω, τα συστήματα Μικτής Πραγματικότητας αποτελούνται από κοινά μικρά εξαρτήματα. Σ' αυτό το κεφάλαιο θα αναδειχθεί επίσης πώς η Μικτή Πραγματικότητα είναι στην ουσία ένα πεδίο έρευνας όπου πολλές τεχνολογίες συνδυάζονται μαζί σε ένα ενιαίο σύστημα. Οι τομείς της μηχανικής όρασης, των γραφικών και των διεπαφών συμβάλλουν ενεργά στην πρόοδο των συστημάτων Μικτής Πραγματικότητας.

3.1 Ένα Τυπικό Σύστημα Μικτής Πραγματικότητας

Ένα τυπικό σύστημα εικονικής πραγματικότητας επιδιώκει να «βυθίσει» εντελώς το χρήστη σε ένα περιβάλλον που έχει παραχθεί από ηλεκτρονικό υπολογιστή. Αυτό το περιβάλλον διατηρείται από το σύστημα σε ένα πλαίσιο αναφοράς, το οποίο καταχωρείται με το σύστημα γραφικών του υπολογιστή, δημιουργώντας την απόδοση του εικονικού κόσμου. Για να είναι αποτελεσματική αυτή η βύθιση, το «εγωκεντρικό» πλαίσιο αναφοράς διατηρείται από το σώμα του χρήστη ενώ ο εγκέφαλος καταχωρεί ένα πλαίσιο αναφοράς με τον εικονικό κόσμο. Αυτό απαιτεί οι κινήσεις ή οι αλλαγές που γίνονται από το χρήστη να οδηγούν σε κατάλληλες αλλαγές στον αντιληπτό εικονικό κόσμο. Επειδή ο χρήστης βλέπει έναν εικονικό κόσμο δεν υπάρχει καμία φυσική σύνδεση μεταξύ αυτών των δύο πλαισίων αναφοράς και μια σύνδεση πρέπει να δημιουργηθεί (Azuma 1993).

Ένα σύστημα Μικτής Πραγματικότητας θα μπορούσε να θεωρηθεί το απόλυτο immersive σύστημα. Ο χρήστης δεν μπορεί να βυθιστεί περισσότερο στον πραγματικό κόσμο. Ο στόχος είναι να καταχωρηθεί τώρα το εικονικό πλαίσιο αναφοράς με αυτό που ο χρήστης βλέπει. Όπως αναφέρεται στην παράγραφο 1.1,

αυτή η καταχώρηση είναι πιο κρίσιμη σε ένα σύστημα Μικτής Πραγματικότητας επειδή είμαστε πιο ευαίσθητοι στα οπτικά λάθη ευθυγράμμισης(misalignments) απ' ότι στα οπτικού-κιναισθητικού τύπου λάθη που παρουσιάζονται σε ένα τυπικό σύστημα Εικονικής Πραγματικότητας. Το παρακάτω σχήμα παρουσιάζει τα διάφορα πλαίσια αναφοράς που πρέπει να συσχετίζονται σε ένα σύστημα Μικτής Πραγματικότητας.



Η σκηνή(scene) λαμβάνεται από μια συσκευή καταγραφής εικόνας, η όποια σε αυτήν την περίπτωση απεικονίζεται ως βιντεοκάμερα. Η κάμερα ουσιαστικά προβάλει την προοπτική του τρισδιάστατου κόσμου επάνω σε μια δισδιάστατη εικόνα. Οι εγγενής (εστιακό μήκος και διαστρέβλωση φακών) και εξωγενής (η θέση και πόζα) παράμετροι της συσκευής καθορίζουν ακριβώς τι προβάλλεται στο πλάνο της εικόνας της.

Η παραγωγή της εικονικής σκηνής αντίστοιχα γίνεται με ένα τυπικό σύστημα δημιουργίας γραφικών. Τα εικονικά αντικείμενα διαμορφώνονται σε ένα πλαίσιο

αναφοράς αντικειμένων. Το σύστημα γραφικών απαιτεί πληροφορίες για την απεικόνιση της πραγματικής σκηνής έτσι ώστε να μπορεί να αποδώσει σωστά αυτά τα αντικείμενα. Με βάση αυτές τις πληροφορίες ελέγχεται η εικονική κάμερα που χρησιμοποιείται για να οπτικοποιήσει τα εικονικά αντικείμενα. Αυτή η εικόνα συγχωνεύεται έπειτα με την εικόνα της πραγματικής σκηνής για να διαμορφώσει την εικόνα Μικτής Πραγματικότητας.

Η σύλληψη του βίντεο και η απόδοση των γραφικών όπως αναφέρθηκε ανωτέρω είναι σχετικά απλές διαδικασίες. Οι ερευνητικές δραστηριότητες στην Μικτή Πραγματικότητα συγκεντρώνονται γύρω από δύο πτυχές του προβλήματος. Μια είναι ότι πρέπει να αναπτυχθούν οι μέθοδοι για να καταχωρήσουν τα δύο ξεχωριστά σύνολα εικόνων και να τις κρατήσουν καταχωρημένες σε πραγματικό χρόνο. Κάποια νέες μελέτες σε αυτήν την περιοχή έχουν αρχίσει να χρησιμοποιούν τεχνικές μηχανικής όρασης. Η δεύτερη κατεύθυνση της έρευνας είναι στις τεχνολογίες απεικόνισης για τη συγχώνευση των δύο εικόνων. Στην παράγραφο 3.3 θα συζητηθούν εν συντομία οι πτυχές της έρευνας στις τεχνολογίες απεικόνισης. Οι παράγραφοι 3.4 και 3.5 αναφέρονται στις τρέχουσες προσεγγίσεις στην καταχώρηση των διάφορων πλαισίων αναφοράς στο σύστημα.

3.2 Προβλήματα Απόδοσης στα Συστήματα Μικτής Πραγματικότητας

Στα συστήματα Μικτής Πραγματικότητας απαιτείται η εκτέλεση σε πραγματικό χρόνο έτσι ώστε ένας χρήστης να είναι σε θέση να τριγυρνάει ελεύθερα μέσα στη σκηνή και να βλέπει συνεχώς μια σωστή μίξη του πραγματικού κόσμου με την κατάλληλη όψη των εικονικών αντικειμένων. Αυτό σημαίνει ότι υπάρχουν δύο κριτήρια απόδοσης στο σύστημα τα οποία είναι τα εξής:

- Ταχύτητα αναπροσαρμογής της εικόνας(Update rate for generating the augmenting image),
- Ακρίβεια καταχώρησης της πραγματικής και εικονικής εικόνας(Accuracy of the registration of the real and virtual image)

Οπτικά, ο περιορισμός του πραγματικού χρόνου φανερώνεται στο χρήστη με το να βλέπει μια εικόνα στην οποία τα εικονικά μέρη δίνονται χωρίς οποιαδήποτε ορατά άλματα. Για να εμφανιστούν χωρίς οποιαδήποτε άλματα, μια εμπειρική μέθοδος υπαγορεύει ότι το σύστημα γραφικών πρέπει να είναι σε θέση να ανανεώσει την εικονική σκηνή τουλάχιστον 10 φορές ανά δευτερόλεπτο. Σύμφωνα με τις ικανότητες των τρεχόντων συστημάτων γραφικών αυτό είναι σχετικά απλό.

Ένα βήμα παρακάτω, αν θέλουμε τα εικονικά αντικείμενα να εμφανίζονται ως ένα ρεαλιστικό μέρος της σκηνής, απαιτείται φωτορεαλιστική απόδοση γραφικών. Οι δυνατότητες των μέσων τρεχόντων γραφικών συστημάτων σήμερα δεν μπορούν να αποδώσουν fully lit, shaded και ray-traced εικόνες από πολύπλοκες σκηνές. Η τεχνολογία στα συστήματα αυτά βέβαια, αυξάνεται ραγδαία και ήδη κυκλοφορούν στην αγορά ακόμη και επεξεργαστές που αναλαμβάνουν την ξεχωριστή επεξεργασία των νόμων της φυσικής που υπακούνε τα εικονικά αντικείμενα ώστε να είναι όσο το δυνατόν πιο ρεαλιστική η απόδοση τους(π.χ PhysX www.ageia.com). Πολύ σύντομα δηλαδή όλες αυτές οι απαιτήσεις για γραφικά από τα συστήματα Μικτής πραγματικότητας θα είναι εντός των δυνατοτήτων του υλικού ενός σχετικά φτηνού υπολογιστικού συστήματος.

Ευτυχώς όμως, προς το παρόν, υπάρχουν πολλές εφαρμογές για την αυξημένη πραγματικότητα στην οποία το εικονικό μέρος είτε δεν είναι πολύ σύνθετο είτε δεν απαιτεί υψηλό επίπεδο φωτορεαλισμού.

Οι αποτυχίες στο δεύτερο κριτήριο απόδοσης έχουν δύο πιθανές αιτίες. Μια είναι λάθος καταχώρηση της πραγματικής και εικονικής σκηνής λόγω του θορύβου στο σύστημα. Η εξωγενής παράμετρος της κάμερας, όπως αναφέρθηκε παραπάνω, σε σχέση με την πραγματική σκηνή πρέπει να ελεγχθεί. Οποιοσδήποτε θόρυβος σε αυτήν την μέτρηση δύναται να εκτεθεί ως λάθος στην καταχώρηση της εικονικής εικόνας με την εικόνα της πραγματικής σκηνής. Οι διακυμάνσεις των τιμών ενώ το σύστημα είναι σε λειτουργία θα προκαλέσουν ένα «τρεμούλιασμα» στην τελική εικόνα. Όπως αναφέρθηκε προηγουμένως, το οπτικό σύστημά μας είναι πολύ ευαίσθητο στα οπτικά λάθη που σε αυτήν την περίπτωση θα ήταν η αντίληψη ότι το εικονικό αντικείμενο δεν είναι στάσιμο στην πραγματική σκηνή ή τοποθετείται

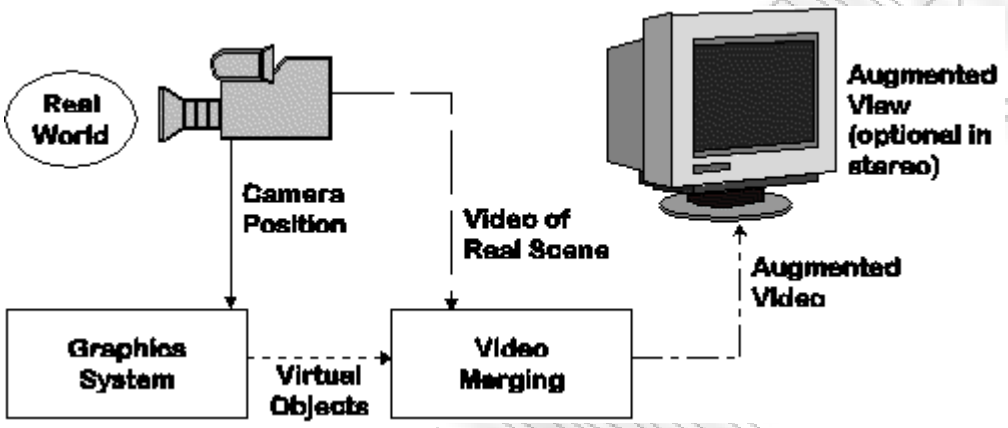
ανακριβώς. Αρκεί και μόνο να φανταστούμε πόσο εύκολα αντιλαμβανόμαστε ότι είναι μη πιστευτή μια εικόνα που προσπαθεί να περάσει ως ενιαία, ενός τραπεζιού πχ που εμφανίζεται με αυτό το «τρεμούλιασμα» λόγω κακής λήψης του πλάνου και ενός εικονικού αντικειμένου πάνω στο τραπέζι, το οποίο είναι τελείως ακίνητο ή ταλαντεύεται κι αυτό ελάχιστα αλλά με διαφορετική φάση. Το λάθος στην καταχώρηση ακόμη και ενός pixel μπορεί να ανιχνευθεί υπό τις κατάλληλες προϋποθέσεις.

Η δεύτερη αιτία λαθών στην καταχώρηση είναι χρονικές καθυστερήσεις στο σύστημα. Όπως αναφέρεται στην προηγούμενη παράγραφο, ελάχιστος ρυθμός ανανέωσης 0.1 δευτερολέπτων απαιτείται για αποδεκτή σε πραγματικό χρόνο απόδοση. Εάν υπάρχουν καθυστερήσεις στον υπολογισμό της θέσης της κάμερας ή της σωστής ευθυγράμμισής της εικονικής κάμερας των γραφικών, τα εικονικά αντικείμενα τείνουν να καθυστερούν σε σχέση με τις κινήσεις στην πραγματική σκηνή. Ο σχεδιασμός του συστήματος πρέπει να ελαχιστοποιεί όσο το δυνατόν τις καθυστερήσεις για να συγκρατεί τη συνολική καθυστέρηση του συστήματος μέσα στις απαιτήσεις της απόδοσης σε πραγματικό χρόνο.

3.3 Τεχνολογίες Απεικόνισης στην Μικτή Πραγματικότητα

Ο συνδυασμός πραγματικών και εικονικών εικόνων σε μια ενιαία εικόνα παρουσιάζουν νέες τεχνικές προκλήσεις για τους σχεδιαστές των συστημάτων Μικτής Πραγματικότητας. Πώς θα γίνει αυτή η συγχώνευση των δύο εικόνων είναι μια βασική απόφαση που ο σχεδιαστής πρέπει να λάβει. Στην παράγραφο 1.1 αναφέραμε τη συνέχεια Πραγματικότητας-Εικονικότητας που ο Milgram χρησιμοποιεί για να ταξινομήσει τα συστήματα Μικτής Πραγματικότητας. Η Έκταση της Μεταφοράς Παρουσίας που όρισε, αφορά άμεσα την συσκευή απεικόνισης που χρησιμοποιείται.

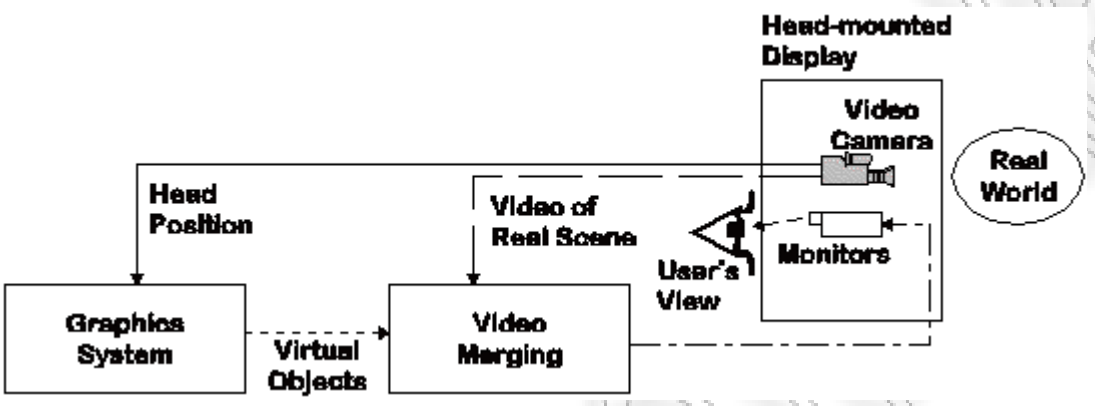
Στο ένα άκρο έχουμε την βασισμένη σε οθόνη προβολή της σκηνής Μικτής Πραγματικότητας. Αυτό αναφέρεται ως «παράθυρο στον κόσμο» («Window on the World»Feiner, MacIntyre 1993) ή «Fish Tank virtual reality» (Ware, Arthur 1993). Ο χρήστης έχει ελάχιστη αίσθηση της βύθισης στο περιβάλλον που δημιουργείται μέσω της απεικόνισης αυτής. Αυτή η τεχνολογία, στο σχήμα παρακάτω, είναι η απλούστερη διαθέσιμη.



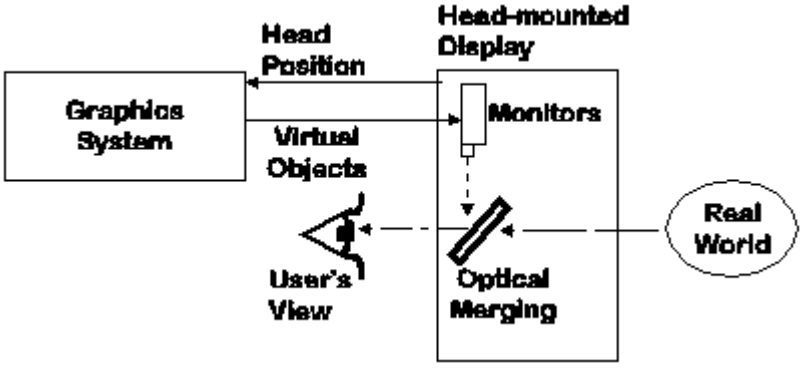
Για να αυξηθεί η αίσθηση της παρουσίας στον κόσμο της Μικτής Πραγματικότητας απαιτούνται άλλες τεχνολογίες απεικόνισης. Οι συσκευές απεικόνισης που τοποθετούνται στο κεφάλι (Head-mounted displays - HMD) χρησιμοποιούνται ευρέως στα συστήματα εικονικής πραγματικότητας. Οι ερευνητές που ασχολούνται με την Μικτή Πραγματικότητα έχουν δοκιμάσει δύο τύπους HMD. Ο ένας ονομάζεται video see-through και ο άλλος optical see-through. Ο χαρακτηρισμός «see-through» προέρχεται από την ανάγκη για το χρήστη να είναι σε θέση να δει την εικόνα του πραγματικού κόσμου που είναι μπροστά του ακόμα και φορώντας το HMD.

Το τυπικό HMD που χρησιμοποιείται στην εικονική πραγματικότητα δίνει στο χρήστη πλήρη οπτική απομόνωση από το γύρω περιβάλλον. Δεδομένου ότι η συσκευή απεικόνισης απομονώνει οπτικά, το σύστημα πρέπει να χρησιμοποιήσει κάμερες που είναι ευθυγραμμισμένες με την εικόνα για να καταγράψουν την άποψη του πραγματικού κόσμου. Ένα διάγραμμα ενός video see-through HMD παρουσιάζεται στο παρακάτω σχήμα. Αυτό που μπορεί εύκολα να διαπιστώσει

κάποιος είναι ότι είναι η ίδια αρχιτεκτονική με τη βασισμένη σε οθόνη απεικόνιση που περιγράφεται παραπάνω εκτός από το ότι τώρα ο χρήστης έχει μεγαλύτερη αίσθηση βύθισης στην σκηνή.



Το optical see-through HMD (Manhart, Malcolm 1993) αποβάλλει το κανάλι που βιντεοσκοπεί την πραγματική σκηνή. Άντ' αυτού, όπως φαίνεται στο σχήμα παρακάτω, η συγχώνευση του πραγματικού κόσμου και των εικονικών αντικειμένων γίνεται οπτικά μπροστά από το χρήστη. Αυτή η τεχνολογία είναι παρόμοια με τις συσκευές απεικόνισης heads up display (HUD) που εμφανίζονται συνήθως στα πιλοτήρια στρατιωτικών αεροπλάνων και πρόσφατα σε μερικά ακριβά αυτοκίνητα. Σε αυτήν την περίπτωση, η οπτική συγχώνευση των δύο εικόνων γίνεται στην τοποθετημένη κεφάλι συσκευή, και όχι στο γυαλί των πιλοτηρίων ή του αυτοκινήτου, το οποίο σκωπτικά αναφέρεται στην αγγλική γλώσσα ως «heads up display on a head»!



Υπάρχουν πλεονεκτήματα και μειονεκτήματα σε κάθε ένας από τους τύπους συσκευών απεικόνισης που αναφέρθηκαν παραπάνω. Αναλύονται με περισσότερες λεπτομέρειες από τον Azuma σε αρκετές από τις δημοσιεύσεις του (Azuma 1993 - 97).

Υπάρχουν μερικά ζητήματα απόδοσης, εντούτοις, τα οποία θα τονιστούν εδώ. Και στις δύο συσκευές απεικόνισης που χρησιμοποιούν κάμερα για να συλλάβουν τον πραγματικό κόσμο υπάρχει μια αναγκαστική καθυστέρηση που φτάνει μέχρι και τον χρόνο ενός πλαισίου(frame) για να εκτελεσθεί η διαδικασία της συγχώνευσης. Σε τυπικούς ρυθμούς frame rate αυτό σημαίνει ότι έχουμε ενδεχομένως 33.33 χιλιοστά του δευτερολέπτου καθυστέρηση στην εικόνα που βλέπει ο χρήστης. Εφόσον όμως όλα όσα βλέπει ο χρήστης είναι κάτω από την καθοδήγηση του συστήματος, αυτή η καθυστέρηση μπορεί να αντισταθμιστεί από σωστά συγχρονισμένες άλλες διαδικασίες του συστήματος. Ή, εναλλακτικά, εάν άλλες διαδικασίες δημιουργούν μεγαλύτερες καθυστερήσεις το βίντεο της πραγματικής σκηνής θα μπορούσε να καθυστερήσει ηθελημένα.

Με ένα optical see-through HMD σύστημα απεικόνισης η άποψη του πραγματικού κόσμου είναι στιγμιαία έτσι δεν καθιστά δυνατό να αντισταθμιστούν οι καθυστερήσεις του συστήματος σε άλλες περιοχές. Παράλληλα, με τα συστήματα video see-through HMD και βασισμένης σε οθόνη προβολή μια κάμερα καταγράφει την πραγματική σκηνή. Ένα πλεονέκτημα αυτού είναι ότι η εικόνα που παράγεται από την κάμερα είναι διαθέσιμη στο σύστημα για να παρέχει πληροφορίες πορείας. Το optical see-through HMD σύστημα απεικόνισης δεν παρέχει αυτές τις πρόσθετες πληροφορίες. Οι μόνες πληροφορίες θέσης διαθέσιμες με αυτό το σύστημα είναι οι πληροφορίες που παρέχουν οι αισθητήρες θέσης που είναι τοποθετημένοι στο HMD.

3.4 Εντοπισμός των Απαιτήσεων στην Μικτή Πραγματικότητα

Το πως ακολουθεί το σύστημα τη θέση και τις κινήσεις του χρήστη στα συστήματα εικονικής πραγματικότητας έχει αποτελέσει το αντικείμενο μιας ευρείας σειράς από έρευνες. Μια από τις συνηθέστερα χρησιμοποιούμενες μεθόδους για το σκοπό αυτό είναι με έναν μαγνητικό αισθητήρα όπως ο Polhemus Isotrac(ο συνηθέστερα χρησιμοποιούμενος). Η καταγραφή της τροχιάς της θέσης απαιτείται στην εικονική πραγματικότητα για να καθοδηγήσει το σύστημα γραφικών ώστε να δώσει μια άποψη του κόσμου από τη νέα θέση του χρήστη. Λόγω του φαινομένου της οπτικής σύλληψης που αναφέρεται στην παράγραφο 1.1, ο χρήστης ενός συστήματος εικονικής πραγματικότητας θα ανεχτεί, και θα προσαρμοστεί ενδεχομένως, σε λάθη μεταξύ της αντιληπτής κίνησής του και των οπτικών αποτελεσμάτων.

Σε ένα σύστημα Μικτής Πραγματικότητας η καταχώρηση είναι σε συνάρτηση με το οπτικό πεδίο του χρήστη. Ο τύπος απεικόνισης που χρησιμοποιείται από το σύστημα Μικτής Πραγματικότητας θα καθορίσει την ακρίβεια που απαιτείται για την καταχώρηση των πραγματικών και των εικονικών εικόνων. Το κεντρικό βοθρίο του ανθρώπινου ματιού έχει ανάλυση περίπου 0.5 λ. του τόξου (Jain 1989). Σε αυτήν την περιοχή το ανθρώπινο μάτι είναι σε θέση να ξεχωρίσει τις εναλλασσόμενες ζώνες φωτεινότητας που υποτείνουν ένα λεπτό του τόξου. Η ικανότητα αυτή καθορίζει τον απόλυτο στόχο καταχώρησης για ένα σύστημα Μικτής Πραγματικότητας. Η ανάλυση της εικονικής εικόνας χαρτογραφείται κατευθείαν πάνω στην άποψη του πραγματικού κόσμου όταν χρησιμοποιείται optical see-through HMD σύστημα απεικόνισης. Εάν χρησιμοποιείται σύστημα απεικόνισης video see-through HMD ή βασισμένο σε οθόνη αναγκαστικά οι αναλύσεις του πραγματικού και του εικονικού κόσμου μειώνονται στην ανάλυση που υποστηρίζεται από το σύστημα απεικόνισης.

Η τρέχουσα τεχνολογία στις συσκευές μαγνητικών αισθητήρων που τοποθετούνται στο κεφάλι εισάγουν στο σύστημα ακόμα περισσότερα προβλήματα, όπως για παράδειγμα λάθη που προκαλούνται από οποιαδήποτε μεταλλικά αντικείμενα στο άμεσο περιβάλλον γύρω τους. Αυτό εμφανίζεται ως λάθος στην θέση ή τον

προσανατολισμό του χρήστη, πράγμα που δεν είναι εύκολο να μοντελοποιηθεί ώστε να προβλέπεται από το σύστημα, ενώ επίσης τα λάθη αυτά θα αλλάξουν εάν οποιοδήποτε από τα παρεμβαίνοντα αντικείμενα κινηθεί.

Επιπλέον, οι συσκευές αυτές εισάγουν στο σύστημα καθυστέρηση της μέτρησης που κυμαίνονται από 40 έως 100 msec για στους τυπικούς αισθητήρες θέσης (Adelstein, Johnston 1992) το οποίο είναι ένα σημαντικό μέρος του ρυθμού ανανέωσης των 100 msec που απαιτείται για τη σε πραγματικό χρόνο λειτουργία του συστήματος. Οι ερευνητές Μικτής Πραγματικότητας εξετάζουν τις υβριδικές τεχνικές για τον εντοπισμό της θέσης (Azuma 1993, Zikan, Curtis 1994). Μια ενδιαφέρουσα σελίδα για την ανάλυση των υβριδικών τεχνικών είναι βρίσκεται στην διεύθυνση http://www.cs.unc.edu/Research/us/hybrid.html

3.5 Διαφορετικές Προσεγγίσεις της Μικτής Πραγματικότητας

Ένα σύστημα Μικτής Πραγματικότητας μπορεί να αντιμετωπισθεί ως μια συλλογή των σχετικών πλαισίων αναφοράς όπως παρουσιάζεται στο κεφάλαιο 1.3.1. Η σωστή καταχώρηση εικονικών αντικειμένων πάνω από την πραγματική σκηνή απαιτεί το σύστημα να παραθέσει τις δύο εικόνες στο ίδιο πλαίσιο αναφοράς. Στον πραγματικό κόσμο αυτά τα πλαίσια μπορούν να εκφραστούν με τρισδιάστατη ευκλείδεια γεωμετρία. Στις περισσότερες παλαιότερες εργασίες πάνω στην Μικτή Πραγματικότητα χρησιμοποιούν αυτήν την ευκλείδεια γεωμετρία και κάνουν προσεκτικές μετρήσεις στις σχέσεις μεταξύ των διάφορων πλαισίων αναφοράς. Για να γίνει αυτό απαιτείται η θέση του χρήστη στον τρισδιάστατο χώρο και ακριβή γνώση των παραμέτρων της κάμερας σε όλη την ακολουθία (Azuma 1993 Janin, Mizell 1993, Azuma,Bishop 1994 Tuceryan, Greer 1995).

Ο εντοπισμός θέσης, όπως με το συχνά χρησιμοποιημένο ηλεκτρομαγνητικό αισθητήρα Polhemus, και οι τεχνικές διόρθωσης/ευθυγράμμισης της κάμερας είναι επιρρεπείς σε λάθη, με συνέπεια να υπάρχουν λάθη στην καταχώρηση των πραγματικών και εικονικών εικόνων. Σε μια ομιλία στο IEEE 1996 Virtual Reality International Symposium, ο Fred Brooks -από το τμήμα πληροφορικής του πανεπιστημίου της βόρειας Καρολίνας- γνωστός για τις έρευνές του πάνω στην εικονική πραγματικότητα, δήλωσε ότι δεν θεωρεί ότι ο εντοπισμός θέσης στα συστήματα Μικτής Πραγματικότητας θα λειτουργήσει ποτέ αρκετά καλά ώστε να είναι η μόνη τεχνολογία που θα χρησιμοποιείται για αυτόν τον σκοπό, λόγω των ανακριβειών και των καθυστερήσεων στο σύστημα. Οι Durlach και Mavor (1995) καταλήγουν στο ίδιο συμπέρασμα και προτείνουν ότι η πιο ελπιδοφόρος τεχνική είναι ο συνδυασμός του τυπικού εντοπισμού θέσης για την «χονδρική» καταχώρηση και μια βασισμένη στην εικόνα μέθοδο για τον τελικό ακριβή συντονισμό.

Λίγη μόνο εργασία έχει γίνει στην προσπάθεια να μετριαστούν ή να εξαλειφθούν τα λάθη που προκαλούνται από τον εντοπισμό θέσης και της λάθος ρύθμισης της κάμερας, χρησιμοποιώντας επεξεργασία εικόνας των ζωντανών λήψεων (Bajura, Neumann 1995, Wloka, Anderson 1995). Ορισμένα άλλα συστήματα Μικτής

Πραγματικότητας (Mellor 1995, Ravela, Draper 1995) ούτε στηρίζονται σε μέθοδο για εύρεσης της θέσης κάμερας, ούτε απαιτούν τις πληροφορίες των παραμέτρων της κάμερας. Το πρόβλημα των εικονικών αντικειμένων πάνω από το ζωντανό βίντεο αντικαθίσταται με το πρόβλημα εκτίμησης της σκηνής. Με τον εντοπισμό σημείων με συγκεκριμένα χαρακτηριστικά γνωρίσματα στο βίντεο αυτά τα συστήματα αναστρέφουν τη λειτουργία σύλληψης που εκτελείται από τη κάμερα και υπολογίζουν τις παραμέτρους της. Αυτό, εντούτοις, απαιτεί τη γνώση της τρισδιάστατης θέσης των σημείων με τα χαρακτηριστικά γνωρίσματα αυτά έτσι ώστε οι παράμετροι της κάμερας να μπορούν να υπολογιστούν στο πλαίσιο της ευκλείδειας γεωμετρίας.

Ορισμένα συστήματα (Grimson, Lozano-Perez 1994, Grimson, Ettinger1995, Mellor 1995) χρησιμοποιούν ένα αποστασιόμετρο λέιζερ για να λάβουν αυτά τα τρισδιάστατα δεδομένα. Η απαίτηση της ακριβούς θέσης των σημείων αυτών περιορίζει τις επιλογές των χαρακτηριστικών σημείων που μπορούν να χρησιμοποιηθούν για τον εντοπισμό, επιστρέφοντας έτσι ένα κάπως διαφορετικό πρόβλημα.

Ένα βασικό πρόβλημα αυτής της θεωρίας είναι ότι όλα τα πλαίσια αναφοράς καθορίζονται σε ένα τρισδιάστατο χώρο ευκλείδειας γεωμετρίας. Η εξαγωγή αυτών των πληροφοριών από την εικόνα της πραγματικής σκηνής είναι μια επιρρεπής σε λάθη διαδικασία. Με τη χαλάρωση της απαίτησης ότι όλα τα πλαίσια πρέπει να έχουν έναν ευκλείδειο καθορισμό είναι δυνατό να εξαλείψουμε την ανάγκη για ακριβή καθορισμό θέσης και παραμέτρων της κάμερας. Μια τέτοια προσέγγιση συστήματος Μικτής Πραγματικότητας περιγράφεται από τους Uenohara και Kanade (1995). Ακολουθούν οπτικά δείκτες σε μια δισδιάστατη επιφάνεια και χρησιμοποιούν τα αποτελέσματα για την καταχώρηση των εικονικών αντικειμένων.

Μια παρόμοια προσέγγιση χρησιμοποιείται από τον Vallino (1997). Το U of R augmented reality system, που προτείνει δεν απαιτεί καμία προηγούμενη μετρική πληροφορία για τις εγγενείς και εξωγενείς παραμέτρους της κάμερας, την θέση του χρήστη ή των αντικειμένων στον πραγματικό κόσμο.

4. Η Μικτή Πραγματικότητα στο Μέλλον

Η ορθή πρόβλεψη των μελλοντικών τάσεων στην Μικτή Πραγματικότητα μπορεί να γίνει μόνο από τους ερευνητές που έχουν ασχοληθεί σε βάθος με την τεχνολογία αυτή. Για τον λόγο αυτό στο κεφάλαιο αυτό παραθέτουμε τα συμπεράσματα που εξήχθησαν από τις ομιλίες και συζητήσεις στο IEEE and ACM International Symposium on Mixed and Augmented Reality (http://campar.in.tum.de/ISMAR/WebHome).

Καταρχήν το ερώτημα που τίθεται με μία πρώτη ματιά στην Μικτή πραγματικότητα είναι το τί κόστος θα έχει ένα σύστημα Μικτής Πραγματικότητας σε βάθος χρόνου 5-10 χρόνων. Αν αναλογιστούμε ότι οι τιμές του υλικού για ένα τέτοιο σύστημα συνεχώς μειώνονται, ενώ παράλληλα η υποστηριζόμενη από αυτό τεχνολογία διευρύνεται, καθώς και το γεγονός ότι ήδη κοστίζει πολύ λιγότερο από ένα Πληροφοριακό σύστημα δεν είναι ουτοπικό να πούμε ότι στο σύντομο μέλλον θα είναι συστήματα ευρείας χρήσης.

Στο σημείο αυτό πρέπει να εξετάσουμε επίσης τις ενέργειες που πρέπει να γίνουν για την δημιουργία πραγματικά χρήσιμων εφαρμογών Μικτής Πραγματικότητας. Τα συστήματα αυτά σίγουρα περιλαμβάνονται στον γενικότερο όρο «υπολογιστικών συστημάτων». Αν αντιμετωπιστούν ως τέτοια, εύκολα μπορούμε να συμπεράνουμε ότι ουσιαστικά οι ζητούμενες ενέργειες δεν διαφέρουν καθόλου από τα βήματα που ακολουθούμε για να δημιουργήσουμε ένα, οποιουδήποτε άλλου τύπου, «πραγματικά χρήσιμο» σύστημα. Ως πρώτο βήμα λοιπόν οι ερευνητές πρέπει να καταλάβουν τις πραγματικές προσδοκίες των πιθανών τελικών χρηστών. Στην συνέχεια πρέπει ερευνηθεί σε ποιους τομείς η Μικτή Πραγματικότητα πραγματικά βελτιώνει την απόδοση των χρηστών, ώστε να μην δαπανηθεί περισσότερος χρόνος και εργασία ερευνώντας τομείς που λειτουργούν αρκετά καλά και χωρίς την παρουσία της Μικτής Πραγματικότητας.

Οι ερευνητές στο σύνολό τους συμφωνούν ότι η πρόοδος στις συσκευές απεικόνισης (latency, resolution) καθώς και στις γενικότερες δυνατότητες (repeatability, speed)

των υπολογιστικών συστημάτων είναι αναγκαία για την δημιουργία αποδεκτών συστημάτων Μικτής πραγματικότητας. Τα συστήματα αυτά θα πρέπει να έχουν ως βασικό σκοπό την ελαχιστοποίηση της πιθανότητας ανθρώπινου λάθους και την παροχή μέσων για να γίνονται εργασίες ταχύτερα και καλύτερα από ότι η παρούσα τεχνολογία επιτρέπει.

Παρόλο που δεν έχει φανεί ακόμα καθαρά ποιες τεχνολογίες αλληλεπίδρασης με τον χρήστη θα επικρατήσουν στα μελλοντικά συστήματα Μικτής Πραγματικότητας οι επιστήμονες συμφωνούν ότι όχι μόνο δεν θα υπάρχει πληκτρολόγιο αλλά ούτε καν περιορισμοί σε συγκεκριμένους τύπους υλικού για τον σκοπό αυτό (πχ συνδυασμοί κινητικών και φωνητικών εντολών).

Τέλος υπάρχει το κυριότερο ίσως ερώτημα που πρέπει να απαντηθεί ως κατακλείδα: υπάρχει κάποιος κοινός στόχος για τα συστήματα Μικτής Πραγματικότητας στο μέλλον; Η απάντηση που δίνουν οι ερευνητές είναι όχι. Αυτό συμβαίνει γιατί ένας και μόνο στόχος θα ήταν ούτως ή άλλως λίγος για να χρησιμοποιηθεί τόση τεχνολογία, έρευνα και εργασία σε αυτόν και μόνο τον τομέα. Σ' αυτό όμως που συμφωνούν όλοι οι ερευνητές είναι ότι όλη η έρευνα τείνει προς συστήματα τα οποία μπορούν να φορεθούν(wearable), πανταχού παρόντα(ubiquitous), που συνδυάζουν και χρησιμοποιούν όλες τις ανθρώπινες αισθήσεις και συνδέουν το περιβάλλον με πληροφορίες. Ο ορισμός και μόνο της Μικτής Πραγματικότητας ως μίας και μόνο τεχνολογίας θεωρείται τουλάχιστον «περιορισμένη» αντίληψη.

5. Το σύστημα που αναπτύχθηκε στα πλαίσια της εργασίας

Παρόλο που ως θέμα η Μικτή Πραγματικότητα είναι ένας τομέας πού πραγματικά συναρπάζει οποιονδήποτε ασχοληθεί μ' αυτήν, ο χρόνος που είχαμε στην διάθεσή μας για την περάτωση της εργασίας σίγουρα δεν ήταν αρκετός για τον σχεδιασμό και την δημιουργία ενός ολοκληρωμένου συστήματος Μικτής Πραγματικότητας. Παρόλα αυτά αναπτύξαμε ένα σύστημα το οποίο, αν και σχετικά απλό, καλύπτει όλες τις απαιτήσεις ενός συστήματος Μικτής Πραγματικότητας.

Εδώ πρέπει να τονίσουμε ότι ως «απλό» σαν σύστημα Μικτής Πραγματικότητας, σίγουρα δεν είναι μια απλή εφαρμογή στην υλοποίηση της, αφού περιλαμβάνει πολλές και διαφορετικές τεχνολογίες όπως τρισδιάστατα γραφικά, πολυμέσα, δικτυακά πρωτόκολλα, text to speech και φυσικά προγραμματισμό, αρκετά πολύπλοκο, για να συνδυαστούν όλα αυτά. Αυτό φαίνεται και από τις απαιτήσεις της εφαρμογής, η οποία απλά και μόνο για μια επίδειξη απαιτεί δυο διασυνδεδεμένους μέσω δικτύου υπολογιστές (το δίκτυο πρέπει να είναι αρκετά γρήγορο για να καλύπτει Real-Time Video και Audio Streams παράλληλα με ικανοποιητικές αποδόσεις) οι οποίοι πρέπει να έχουν προεγκατεστημένη γλώσσα προγραμματισμού Java, Java Media Framework(JMF), Java for OpenGL libraries(jogl), FREE-TTS καθώς και αρκετά γρήγορες κάρτες γραφικών(σε ορισμένες περιπτώσεις και επεξεργαστές) για την υποστήριξη των τρισδιάστατων γραφικών.

5.1 Σύντομη περιγραφή του συστήματος

Το σύστημα που αναπτύξαμε σε γενικές γραμμές είναι μια client-server εφαρμογή η οποία στέλνει σε πραγματικό χρόνο video και ήχο τα οποία συνδυάζει με τρισδιάστατα γραφικά, δυνατότητες σημείωσης και εισαγωγής κειμένου πάνω στο βίντεο ενώ το κείμενο αυτό ταυτόχρονα μετατρέπεται σε ήχο(TTS). Πιο

συγκεκριμένα ας δούμε πως λειτουργεί το σύστημα από την άποψη του χρήστη ως πελάτη ή εξυπηρετητή.

5.1.1 Server

Η υλοποίηση του Server λειτουργεί ως εξής:

Από τη στιγμή που θα τρέξει το πρόγραμμα ο server περιμένει ακούγοντας τα κανάλια που του προκαθορίζονται για clients που θέλουν να συνδεθούν μαζί του. Από τη στιγμή που ένας client στέλνει αίτηση για σύνδεση ο server καταρχήν ειδοποιεί τον χρήστη, προσπαθεί να τον αναγνωρίσει(IP, port κλπ) και τον αποδέχεται. Ο κάθε client αρχίζει από την στιγμή αυτή να στέλνει RTP δεδομένα. Το πρόγραμμα του server αναγνωρίζει τα δεδομένα αυτά και ανοίγει τα ανάλογα παράθυρα για να τα αναπαράγει ενώ συνεχίζει να ακούει στα προκαθορισμένα κανάλια για άλλους clients. Παράλληλα με τα RTP δεδομένα περιμένει να του αποσταλεί κάποιο κείμενο που εμφανίζεται ταυτόχρονα και στο βίντεο –που όπως θα φανεί παρακάτω είναι μέρος των RTP δεδομένων που στέλνει ο χρήστης του client. Κάθε φορά που αναγνωρίζει εισερχόμενο κείμενο ενεργοποιείται ο μηχανισμός text to speech και αποστέλλεται στην κάρτα ήχου του συστήματος μια ανάγνωση του κειμένου από το σύστημα.

5.1.2. Client

Η υλοποίηση του client λειτουργεί ως εξής:

Για την ορθή εκτέλεση του προγράμματος αυτού απαιτείται καταρχήν να είναι ήδη σε λειτουργία (up and running) ο server. Εφόσον συμβαίνει αυτό ο χρήστης του client επιλέγει ένα αρχείο βίντεο (ή μια συσκευή καταγραφής βίντεο) και ένα αρχείο ήχου (ή αντίστοιχα μια συσκευή καταγραφής ήχου) και ο client αποστέλλει στον server

αίτηση για να συνδεθεί μαζί του. Παράλληλα δημιουργεί ένα παράθυρο, που λειτουργεί και ως διεπαφή χρήστη, στο οποίο αρχίζει να αναπαράγει τοπικά το βίντεο.

Αφού ο server αποδεχθεί την σύνδεση ο client αρχίζει να αποστέλλει δύο RTP streams. Το πρώτο stream περιέχει το βίντεο ακριβώς όπως προβάλλεται στον client ενώ το δεύτερο το αρχείο ήχου που ο χρήστης προεπέλεξε. Από την στιγμή αυτή ο χρήστης έχει την δυνατότητα να «επεξεργαστεί» το βίντεο σε πραγματικό χρόνο, προσθέτοντας τρισδιάστατα γραφικά και μετακινώντας τα, προσθέτοντας κείμενο σε οποιοδήποτε μέρος του βίντεο, επιλέγοντας font κλπ, ή ακόμα και σημειώνοντας κάτι πάνω στο βίντεο. Οποιεσδήποτε μετατροπές αυτού του είδους εμφανίζονται σε πραγματικό χρόνο και στην μεριά του server. Ειδικά στην περίπτωση του κειμένου εκτός από την εμφάνιση του πάνω στο βίντεο ο χρήστης του server έχει την δυνατότητα να το ακούσει και αναγνωσμένο από το σύστημα.

5.2 Εγχειρίδιο Χρήσης

Η εφαρμογή αποτελείται από 9 αρχεία java τα οποία απαιτούνται για την εκτέλεση της. Πιο συγκεκριμένα στον υπολογιστή που δρα ως server απαιτούνται τα αρχεία:

- AVReceive2.java
- Receivestring.java
- Dektis.bat

Αντίστοιχα στον υπολογιστή που δρα ως client απαιτούνται τα αρχεία:

- AudioStreamer.java
- VideoStreamer.java
- mysocket.java
- DataSource.java

- LiveStream.java
- Transmitter.bat

5.2.1 Προαπαιτούμενη εγκατάσταση

Όπως αναφέρθηκε και στην εισαγωγή του κεφαλαίου αυτού για να τρέξει η εφαρμογή απαιτούνται δύο υπολογιστές διασυνδεδεμένοι μέσω δικτύου οι οποίοι έχουν προεγκατεστημένα τα εξής:

- γλώσσα προγραμματισμού Java,
- Java Media Framework(JMF),
- Java for OpenGL libraries(jogl),

Ειδικά στην περίπτωση του server πρέπει επίσης να έχει εγκατασταθεί το freetts για την μετατροπή κειμένου σε ήχο.

5.2.2 Προετοιμασία για την εκτέλεση της εφαρμογής

Πριν το compilation των αρχείων θα πρέπει από την μεριά του client να τροποποιήσουμε τα αρχεία VideoStreamer.java και mysocket.java με την IP προορισμού(ο υπολογιστής που τρέχει ο server). Στην συνέχεια σε command line στον φάκελο που περιέχει τα αρχεία της εφαρμογής(πρέπει να βρίσκονται στον ίδιο φάκελο) πληκτρολογούμε την εντολή “javac –d . DataSource.java LiveStream.java”. Ακολούθως κάνουμε compile τα αρχεία mysocket.java, AudioStreamer.java και VideoStreamer.java.

Αντιστοίχως στον server απλά κάνουμε compile και τα δυο αρχεία(επίσης απαιτείται να είναι στον ίδιο φάκελο). Στην συνέχεια τρέχουμε το dektis.bat(αφού το

τροποποιήσουμε κατάλληλα με τα IP και port του client) και ο server αρχίζει να ακούει στα δοθέντα κανάλια.

Ο client τώρα μπορεί να τρέξει είτε με το transmitter.bat αρχείο είτε από command line με τα ακόλουθα ορίσματα: το πρώτο όρισμα πρέπει να είναι μια πηγή βίντεο και το δεύτερο μια πηγή ήχου.

5.2.3 User Interface

Ακολουθώντας τα παραπάνω βήματα η εφαρμογή από την μεριά του client εμφανίζει ένα παράθυρο στο οποίο το μεγαλύτερο μέρος του καλύπτεται από το βίντεο που ο χρήστης έδωσε ως όρισμα κατά την εκτέλεση του προγράμματος, ενώ γύρω από αυτό εμφανίζεται το user interface. Για την εισαγωγή τρισδιάστατων γραφικών πάνω στο βίντεο ο χρήστης μπορεί να πατήσει το κουμπί "Insert Graphics" και από το σημείο αυτό μπορεί με drag πάνω στα τρισδιάστατα γραφικά να τα περιστρέψει.

Για την εισαγωγή κειμένου πάνω στο βίντεο ο χρήστης μπορεί να κάνει κλικ σε οποιοδήποτε σημείο του βίντεο θέλει να εμφανίζεται, γράφει το κείμενο που θέλει στο textbox που βρίσκεται από κάτω, επιλέγει γραμματοσειρά και μέγεθος από τις αντίστοιχες λίστες και πατάει το κουμπί "Insert Text/Annotation". Πατώντας το κουμπί αυτό ενεργοποιεί ταυτόχρονα και την λειτουργία σημειώσεων με την οποία μπορεί να σημειώνει σε οποιοδήποτε σημείο του βίντεο αυτό που θέλει.

Όλες οι παραπάνω ενέργειες του χρήστη από την μεριά του Client εμφανίζονται σε πραγματικό χρόνο και στον server ο οποίος παράλληλα με το βίντεο αναπαράγει και την πηγή ήχου που του αποστέλλεται από τον client με την μορφή RTP stream. Τέλος κάθε φορά που εισάγεται κείμενο πάνω στο βίντεο ο server το μετατρέπει σε ομιλία(Text to Speech).

5.3 Αρχιτεκτονική του συστήματος

Παρόλο που το σύστημα εκ πρώτης όψεως φαίνεται απλό, εφόσον δεν περιλαμβάνει κάποια βάση δεδομένων, χρησιμοποιεί αρκετές διαδικτυακές τεχνολογίες καθώς και τεχνολογίες γραφικών και TTS. Τα δεδομένα τόσο του βίντεο όσο και του ήχου στέλνονται με πρωτόκολλο RTP ενώ παράλληλα χρησιμοποιείται TCP socket για την ασφαλή μεταφορά των δεδομένων του κειμένου, η οποία και διατηρείται ανοιχτή

καθόλη την διάρκεια της εκτέλεσης του προγράμματος. Τα γραφικά είναι της μορφής OpenGL ενώ το TTS υλοποιήθηκε με βάση το έτοιμο πακέτο FreeTTS. Πιο συγκεκριμένα οι λειτουργικές απαιτήσεις που λάβαμε υπόψη για την δημιουργία τις εφαρμογής θα μελετηθούν στο ακόλουθο κεφάλαιο.

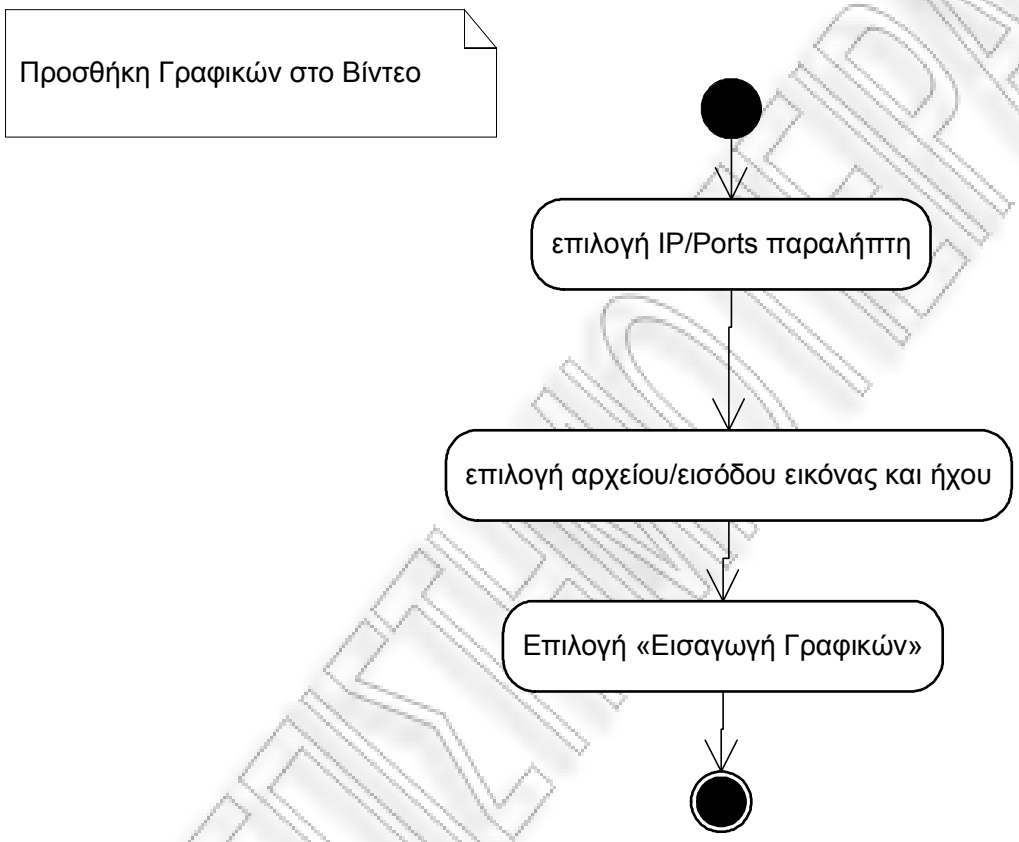
5.3.1 Λειτουργικές απαιτήσεις

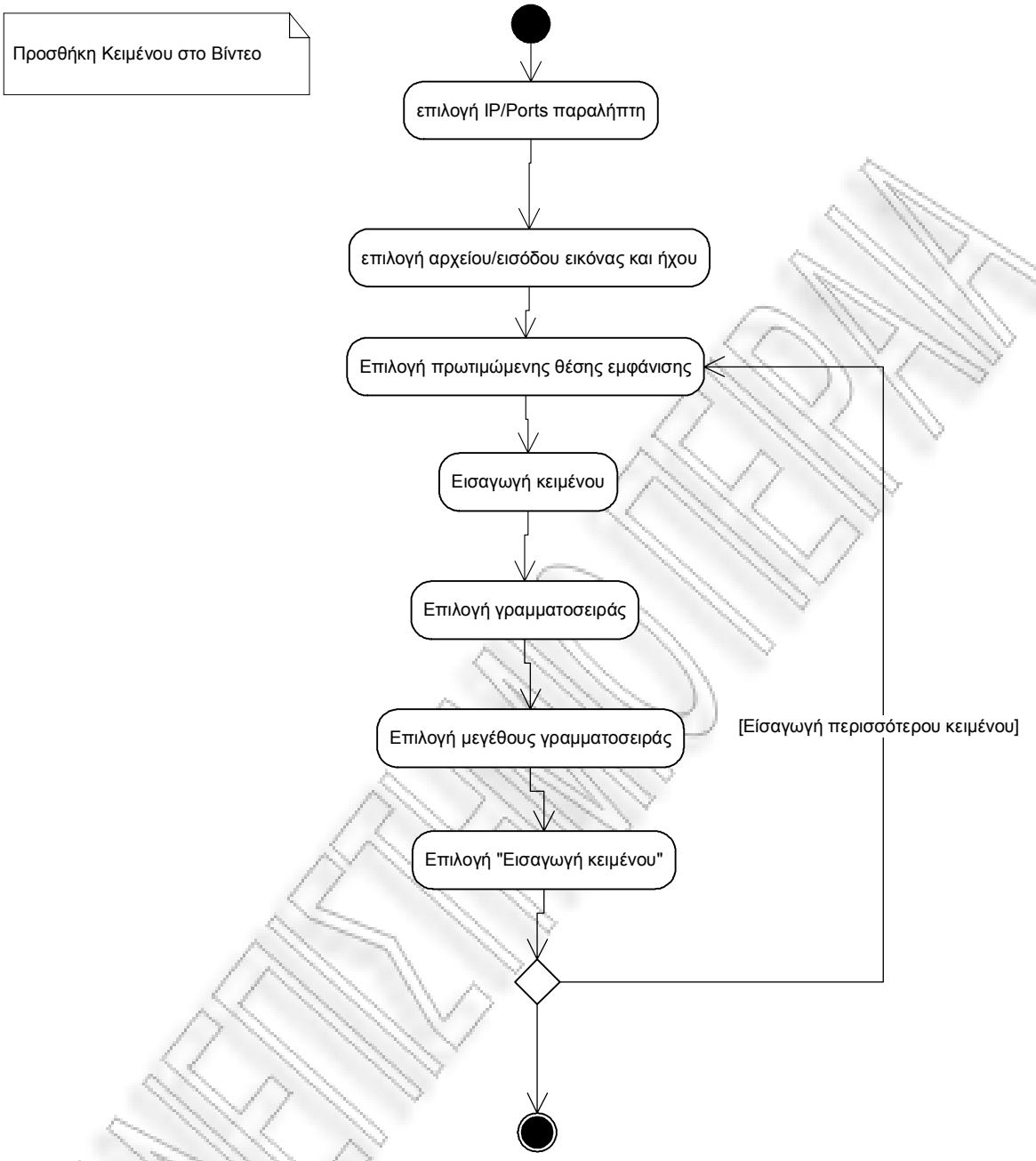
Οι λειτουργικές απαιτήσεις καθορίστηκαν με βάση τις δυνατότητες που θέλουμε να έχει στην διάθεση του ο χρήστης. Αυτές είναι:

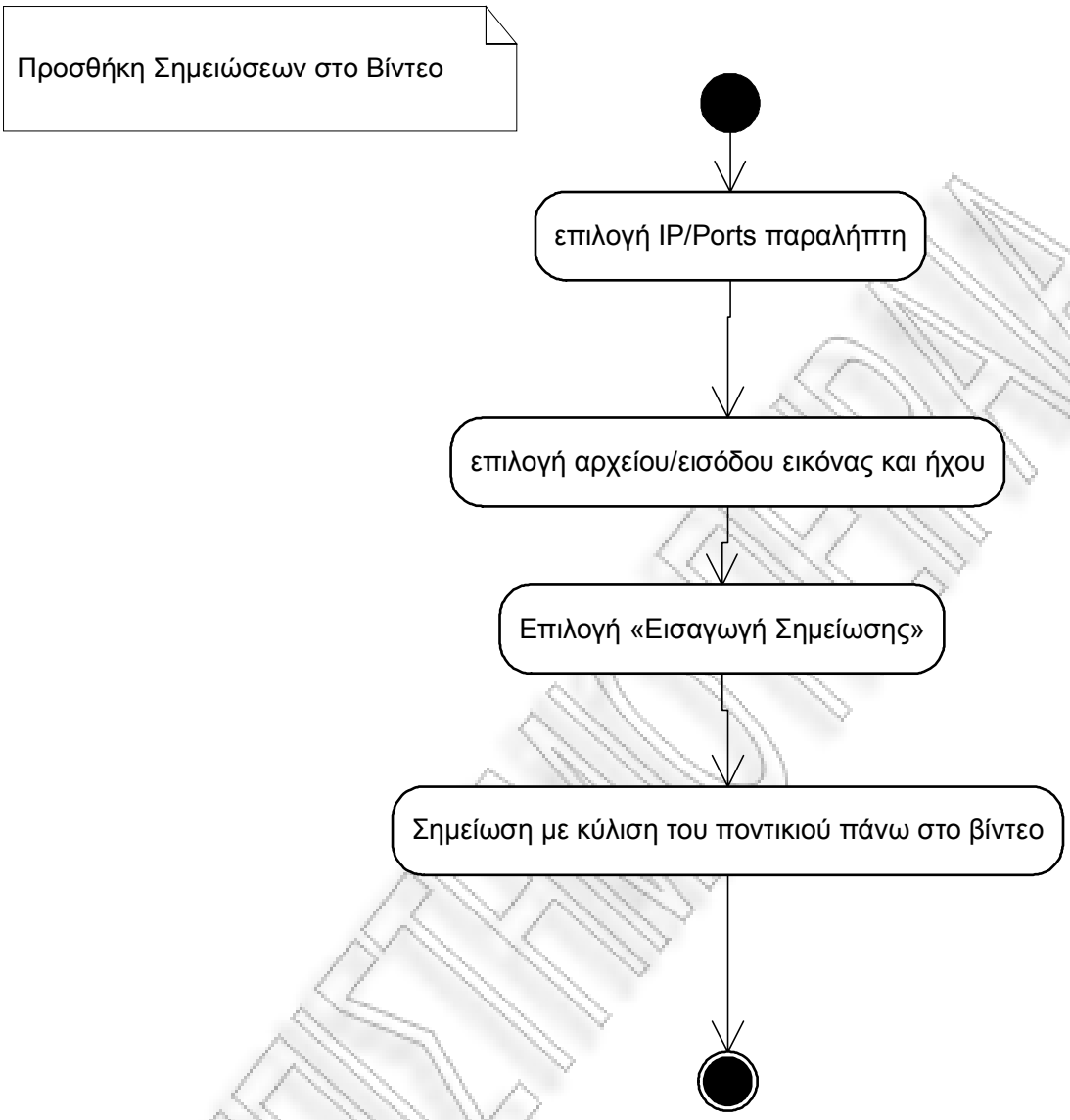
- Αποστολή ζωντανού RTP stream βίντεο(από αρχείο ή συσκευή εισόδου) προς άλλο υπολογιστή μέσω δικτύου
- Αποστολή ζωντανού RTP stream ήχου(από αρχείο ή συσκευή εισόδου) προς άλλο υπολογιστή μέσω δικτύου
- Δυνατότητα εισαγωγής Τρισδιάστατων Γραφικών πάνω στο βίντεο που αποστέλλεται σε πραγματικό χρόνο
- Δυνατότητα περιστροφής των γραφικών σε όλους τους άξονες σε πραγματικό χρόνο
- Δυνατότητα σημείωσης πάνω σε οποιοδήποτε σημείο της εικόνας του βίντεο που μεταδίδεται σε πραγματικό χρόνο
- Δυνατότητα εισαγωγής κειμένου σε οποιοδήποτε σημείο της εικόνας του βίντεο που μεταδίδεται σε πραγματικό χρόνο
- Δυνατότητα επιλογής της γραμματοσειράς του κειμένου που εισάγεται και του μεγέθους της
- Μετατροπή του κειμένου που αποστέλλεται σε πραγματικό χρόνο σε speech
- Δυνατότητα αναπαραγωγής/πάυσης/μετακίνησης πάνω στο βίντεο που αποστέλλεται σε πραγματικό χρόνο

5.3.2 Σχεδιασμός της Εφαρμογής (Διαγράμματα UML)

Με βάση τις παραπάνω λειτουργικές απαιτήσεις της εφαρμογής καταλήξαμε στα παρακάτω διαγράμματα δραστηριοτήτων:




Προσθήκη Κειμένου στο Βίντεο
επιλογή IP/Ports παραλήπτη
επιλογή αρχείου/εισόδου εικόνας και ήχου
Επιλογή πρωτιμώμενης θέσης εμφάνισης
Εισαγωγή κειμένου
Επιλογή γραμματοσειράς
Επιλογή μεγέθους γραμματοσειράς
[Είσαγωγή περισσότερου κειμένου]
Επιλογή "Εισαγωγή κειμένου"



5.4 Υλοποίηση της εφαρμογής

Η εφαρμογή που υλοποιήθηκε περιλαμβάνει ένα μεγάλο αριθμό από modules τα οποία στο κεφάλαιο αυτό θα αναλύσουμε περιληπτικά. Στον παρακάτω πίνακα εμφανίζεται το όνομα των συναρτήσεων, το όνομα του προγράμματος που ανήκουν, μια σύντομη περιγραφή του τι ακριβώς κάνουν και σε ποιο βαθμό τροποποιήθηκαν ή βρέθηκαν έτοιμα. Τα αυθεντικά αρχεία των προγραμμάτων και modules που

χρησιμοποιήθηκαν για την δημιουργία της εφαρμογής μας, βρίσκονται στο CD της εργασίας στον φάκελο “Original programs”.

Module	Program	Περιγραφή	Τροποποίηση
main	VideoStreamer.java	Ελέγχει την ορθότητα των ορισμάτων, και καλεί τις υπόλοιπες συναρτήσεις για την αναπαραγωγή και αποστολή των δεδομένων	Τροποποιήθηκε κατάλληλα από την αντίστοιχη συνάρτηση του αυθεντικού προγράμματος
StateListener	VideoStreamer.java	Ελέγχει για λάθη κατά την προετοιμασία της αναπαραγωγής και τα εμφανίζει στην οθόνη	Δεν τροποποιήθηκε από το αυθεντικό πρόγραμμα
prUsage	VideoStreamer.java	Καλείται σε περίπτωση λάθους στην εισαγωγή των ορισμάτων και δίνει οδηγίες για την ορθή σύνταξη τους	Δεν τροποποιήθηκε από το αυθεντικό πρόγραμμα
VideoPlay	VideoStreamer.java	Δημιουργεί το παράθυρο με την διεπαφή του χρήστη και τοποθετεί όλα τα elements στα σημεία που πρέπει. Τα elements αυτά περιλαμβάνουν από τα γραφικά μέχρι τα MouseListener κλπ. Επίσης με την	Ξεκίνησε από το ήδη έτοιμο πρόγραμμα VideoPlay.java αλλά τροποποιήθηκε σε τέτοιο βαθμό που τελικά μόνο το όνομα παρέμεινε ίδιο

		βοήθεια των επόμενων συναρτήσεων είναι υπεύθυνη για την τοπική αναπαραγωγή του βίντεο καθώς και των μετατροπών που κάνει ο χρήστης σ'αυτόν	
ControllerUpdate	VideoStreamer.java	Τοποθετεί το βίντεο στην οθόνη και τον πίνακα ελέγχου του και εμφανίζει μηνύματα λάθους που σχετίζονται με αυτά	Χρησιμοποιήθηκε από έτοιμο πρόγραμμα αλλά τροποποιήθηκαν αρκετά πράγματα για να εξαλειφθούν προβλήματα συμβατότητας
paint	VideoStreamer.java	Δημιουργεί τα γραφικά του κειμένου και των σημειώσεων για να τοποθετηθούν πάνω στο βίντεο	Δημιουργήθηκε εξ' ολοκλήρου από εμάς
Gears	VideoStreamer.java	Δημιουργεί τα τρισδιάστατα γραφικά με σκοπό να τοποθετηθούν πάνω στο βίντεο	Τροποποιήθηκε κατάλληλα από το έτοιμο πρόγραμμα Gears.java
Main	AudioStreamer.java	Είναι υπεύθυνη για την αποστολή των δεδομένων ήχου σε μορφή RTP.Κάνει την σύνδεση με τον	Μικρές αλλαγές έγιναν από το αυθεντικό για να αποστέλλονται τα δεδομένα στον ίδιο

		εξυπηρετητή και αρχίζει να μεταδίδει τα δεδομένα αυτά. Οι λοιπές συναρτήσεις στο πρόγραμμα αυτό είναι ίδιες με τις αντίστοιχες του VideoStreamer.java	υπολογιστή με τα δεδομένα της εικόνας
DataSource	DataSource.java	Καλεί την LiveStream για να συλλάβει τα screenshots και τα μετατρέπει σε RTP για να μεταδοθούν ως stream	Χρησιμοποιήθηκε ως ήταν από το site της JMF
LiveStream	LiveStream.java	Είναι υπεύθυνη για τον κατακερματισμό της οθόνης και την εξαγωγή από αυτήν screenshots με οποιονδήποτε ρυθμό	Χρησιμοποιήθηκε ως ήταν από το site της JMF
AVReceive2	AVReceive2.java	Περιμένει RTP δεδομένα από τα προκαθορισμένα κανάλια και ελέγχει τα παράθυρα αναπαραγωγής μόλις τα λάβει. Επίσης εμφανίζει τα δεδομένα των υπολογιστούν που συνδέονται πάνω του	Μικρές αλλαγές έγιναν από το αυθεντικό

PlayerWindow	AVReceive2.java	Είναι υπεύθυνη για το παράθυρο που εμφανίζεται όταν τα δεδομένα αρχίζουν να λαμβάνονται	Χρησιμοποιήθηκε από το αυθεντικό
PlayerPanel	AVReceive2.java	Δημιουργεί και τοποθετεί τα elements όπως τον πίνακα ελέγχου μέσα στο παράθυρο που δημιουργήθηκε παραπάνω	Χρησιμοποιήθηκε από το αυθεντικό
main	AVReceive2.java	Αρχικοποιεί τις συναρτήσεις που χειρίζονται τα εισερχόμενα δεδομένα	Τροποποιήθηκε κατάλληλα για να υποστηρίζονται επιπλέον τύποι δεδομένων(TTS κλπ)
Info2	AVReceive2.java	Ακούει στο προκαθορισμένο socket για δεδομένα κειμένου και καλεί τις αντίστοιχες συναρτήσεις για να τα λάβουν και να τα επεξεργαστούν.	Δημιουργήθηκε εξ'ολοκλήρου από εμάς
initJSAPI	AVReceive2.java	Προετοιμάζει το text to speech feature ώστε να μετατρέψει το κείμενο σε ήχο σε πραγματικό χρόνο μόλις παραληφθεί από την προηγούμενη	Τροποποιήθηκε κατάλληλα από το πρόγραμμα SpeechKeyboard.java

		συνάρτηση	
Receivestring	Receivestring.java	Ακούει στο socket και περιμένει εισερχόμενο κείμενο. Στη συνέχεια το τοποθετεί σε μεταβλητή και το προωθεί ως έξοδο	Αναπτύχθηκε από εμάς
mysocket	mysocket.java	Ανοίγει socket για την μεταφορά του κειμένου και το αποστέλλει	Αναπτύχθηκε από εμάς

Επίσης στο CD της εργασίας στον φάκελο «background problem» υπάρχουν τα τροποποιημένα αρχεία με τις προσπάθειες για την απόρριψη του φόντου των γραφικών που δυστυχώς δεν κατάφερε να λυθεί παρά την εκτεταμένη εργασία και κατανάλωση χρόνου που προσφέρθηκε. Τα προγράμματα λειτουργούν κανονικά με μόνο πρόβλημα ότι τα γραφικά εμφανίζονται πίσω από το βίντεο με αποτέλεσμα να μην είναι ορατά. Για τον σκοπό αυτό αντί για το demo πρόγραμμα Gears.java χρησιμοποιήθηκε το JGears.java και με τις κατάλληλες τροποποιήσεις που κάναμε γίνεται compile και εκτελείται κανονικά.

5.5 Κώδικας των αρχείων της εφαρμογής

Όλα τα αρχεία που δημιουργήθηκαν στο πλαίσιο της εργασίας αυτής περιέχουν σύντομα επεξηγηματικά σχόλια για την κατανόηση της φιλοσοφίας ανάπτυξης.

5.5.1 AVReceive2.java

```
/*
 * @(#)AVReceive2.java  1.3 01/03/13
 *
 * Copyright (c) 1999-2001 Sun Microsystems, Inc. All Rights Reserved.
 *
 * Sun grants you ("Licensee") a non-exclusive, royalty free, license to use,
 * modify and redistribute this software in source and binary code form,
 * provided that i) this copyright notice and license appear on all copies of
 * the software; and ii) Licensee does not utilize the software in a manner
 * which is disparaging to Sun.
 *
 * This software is provided "AS IS," without a warranty of any kind. ALL
 * EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES,
INCLUDING ANY
 * IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE
OR
 * NON-INFRINGEMENT, ARE HEREBY EXCLUDED. SUN AND ITS LICENSORS SHALL NOT
BE
 * LIABLE FOR ANY DAMAGES SUFFERED BY LICENSEE AS A RESULT OF USING,
MODIFYING
 * OR DISTRIBUTING THE SOFTWARE OR ITS DERIVATIVES. IN NO EVENT WILL SUN OR
ITS
 * LICENSORS BE LIABLE FOR ANY LOST REVENUE, PROFIT OR DATA, OR FOR DIRECT,
 * INDIRECT, SPECIAL, CONSEQUENTIAL, INCIDENTAL OR PUNITIVE DAMAGES,
HOWEVER
 * CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF THE
USE OF
 * OR INABILITY TO USE SOFTWARE, EVEN IF SUN HAS BEEN ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 *
 * This software is not designed or intended for use in on-line control of
 * aircraft, air traffic, aircraft navigation or aircraft communications; or in
 * the design, construction, operation or maintenance of any nuclear
 * facility. Licensee represents and warrants that it will not use or
 * redistribute the Software for such purposes.
 */
```

```
import java.io.*;
import java.io.File;
import java.awt.*;
import java.net.*;
import java.awt.event.*;
import java.util.*;
import java.util.Locale;
import java.util.Vector;

import javax.media.*;
import javax.media.rtp.*;
import javax.media.rtp.event.*;
import javax.media.rtp.rtcp.*;
import javax.media.protocol.*;
import javax.media.protocol.DataSource;
import javax.media.format.AudioFormat;
import javax.media.format.VideoFormat;
import javax.media.Format;
import javax.media.format.FormatChangeEvent;
import javax.media.control.BufferControl;


//import java.util.Vector;

import javax.speech.Central;
import javax.speech.Engine;
import javax.speech.EngineList;
import javax.speech.synthesis.Synthesizer;
import javax.speech.synthesis.SynthesizerModeDesc;
import javax.speech.synthesis.SynthesizerProperties;
import javax.speech.synthesis.Voice;


/**
 * AVReceive2 to receive RTP transmission using the new RTP API.
 */
public class AVReceive2 implements ReceiveStreamListener, SessionListener,
        ControllerListener
{
```

```
public static     int   serverPort   = 3456;
   String sessions[] = null;
   RTPManager mgrs[] = null;
   Vector playerWindows = null;


public static Synthesizer synthesizer;



   boolean dataReceived = false;
   Object dataSync = new Object();


   public AVReceive2(String sessions[]) {
         this.sessions = sessions;
   }

   protected boolean initialize() {

      try {
            InetAddress ipAddr;
            SessionAddress localAddr = new SessionAddress();
            SessionAddress destAddr;

            mgrs = new RTPManager[sessions.length];
            playerWindows = new Vector();

            SessionLabel session;

            // Open the RTP sessions.
            for (int i = 0; i < sessions.length; i++) {

                  // Parse the session addresses.
                  try {
                     session = new SessionLabel(sessions[i]);
                  } catch (IllegalArgumentException e) {
                     System.err.println("Failed to parse the session address given: " + sessions[i]);
                     return false;
                  }
```

```
                System.err.println("  - Open RTP session for: addr: " + session.addr + " port: " +
session.port + " ttl: " + session.ttl);

                mgrs[i] = (RTPManager) RTPManager.newInstance();
                mgrs[i].addSessionListener(this);
                mgrs[i].addReceiveStreamListener(this);

                ipAddr = InetAddress.getByName(session.addr);

                if( ipAddr.isMulticastAddress()) {
                   // local and remote address pairs are identical:
                   localAddr= new SessionAddress( ipAddr,
                                                  session.port,
                                                  session.ttl);
                   destAddr = new SessionAddress( ipAddr,
                                                  session.port,
                                                  session.ttl);
                } else {
                   localAddr= new SessionAddress( InetAddress.getLocalHost(),
                                                  session.port);
            destAddr = new SessionAddress( ipAddr, session.port);
                }

                mgrs[i].initialize( localAddr);

                // You can try out some other buffer size to see
                // if you can get better smoothness.
                BufferControl bc =
(BufferControl)mgrs[i].getControl("javax.media.control.BufferControl");
                if (bc != null)
                  bc.setBufferLength(350);

                mgrs[i].addTarget(destAddr);
            }

        } catch (Exception e){
           System.err.println("Cannot create the RTP Session: " + e.getMessage());
           return false;
        }
```

```
        // Wait for data to arrive before moving on.


        long then = System.currentTimeMillis();
        long waitingPeriod = 30000;  // wait for a maximum of 30 secs.


        try{
           synchronized (dataSync) {
                  while (!dataReceived &&
                           System.currentTimeMillis() - then < waitingPeriod) {
                     if (!dataReceived)
                           System.err.println("  - Waiting for RTP data to arrive...");
                     dataSync.wait(1000);
                  }
           }
        } catch (Exception e) {}

        if (!dataReceived) {
           System.err.println("No RTP data was received.");
           close();
           return false;
        }

     return true;
}


public boolean isDone() {
        return playerWindows.size() == 0;
}


/**
 * Close the players and the session managers.
 */
protected void close() {

        for (int i = 0; i < playerWindows.size(); i++) {
```

```
            try {
                ((PlayerWindow)playerWindows.elementAt(i)).close();
            } catch (Exception e) {}
        }

        playerWindows.removeAllElements();

        // close the RTP session.
        for (int i = 0; i < mgrs.length; i++) {
            if (mgrs[i] != null) {
             mgrs[i].removeTargets( "Closing session from AVReceive2");
             mgrs[i].dispose();
                mgrs[i] = null;
            }
        }
}


PlayerWindow find(Player p) {
        for (int i = 0; i < playerWindows.size(); i++) {
            PlayerWindow pw = (PlayerWindow)playerWindows.elementAt(i);
            if (pw.player == p)
                return pw;
        }
        return null;
}


PlayerWindow find(ReceiveStream strm) {
        for (int i = 0; i < playerWindows.size(); i++) {
            PlayerWindow pw = (PlayerWindow)playerWindows.elementAt(i);
            if (pw.stream == strm)
                return pw;
        }
        return null;
}


/**
 * SessionListener.
```

```
 */
public synchronized void update(SessionEvent evt) {
        if (evt instanceof NewParticipantEvent) {
           Participant p = ((NewParticipantEvent)evt).getParticipant();
           System.err.println("  - A new participant had just joined: " + p.getCNAME());
        }
}


/**
 * ReceiveStreamListener
 */
public synchronized void update( ReceiveStreamEvent evt) {

        RTPManager mgr = (RTPManager)evt.getSource();
        Participant participant = evt.getParticipant(); // could be null.
        ReceiveStream stream = evt.getReceiveStream();  // could be null.

        if (evt instanceof RemotePayloadChangeEvent) {

           System.err.println("  - Received an RTP PayloadChangeEvent.");
           System.err.println("Sorry, cannot handle payload change.");
           System.exit(0);

        }

        else if (evt instanceof NewReceiveStreamEvent) {

           try {
                stream = ((NewReceiveStreamEvent)evt).getReceiveStream();
                DataSource ds = stream.getDataSource();

                // Find out the formats.
                RTPControl ctl = (RTPControl)ds.getControl("javax.media.rtp.RTPControl");
                if (ctl != null){
                   System.err.println("  - Recevied new RTP stream: " + ctl.getFormat());
                } else
                   System.err.println("  - Recevied new RTP stream");

                if (participant == null)
```

```
                System.err.println("      The sender of this stream had yet to be identified.");
            else {
                System.err.println("      The stream comes from: " + participant.getCNAME());
            }

            // create a player by passing datasource to the Media Manager
            Player p = javax.media.Manager.createPlayer(ds);
            if (p == null)
                return;

            p.addControllerListener(this);
            p.realize();
            PlayerWindow pw = new PlayerWindow(p, stream);
            playerWindows.addElement(pw);

            // Notify intialize() that a new stream had arrived.
            synchronized (dataSync) {
                dataReceived = true;
                dataSync.notifyAll();
            }

        } catch (Exception e) {
            System.err.println("NewReceiveStreamEvent exception " + e.getMessage());
            return;
        }

    }

    else if (evt instanceof StreamMappedEvent) {

        if (stream != null && stream.getDataSource() != null) {
            DataSource ds = stream.getDataSource();
            // Find out the formats.
            RTPControl ctl = (RTPControl)ds.getControl("javax.media.rtp.RTPControl");
            System.err.println("  - The previously unidentified stream ");
            if (ctl != null)
                System.err.println("      " + ctl.getFormat());
            System.err.println("      had now been identified as sent by: " +
participant.getCNAME());
        }
```

```
        }

        else if (evt instanceof ByeEvent) {

            System.err.println("  - Got \"bye\" from: " + participant.getCNAME());
            PlayerWindow pw = find(stream);
            if (pw != null) {
                    pw.close();
                    playerWindows.removeElement(pw);
                      }
                    }
}


/**
 * ControllerListener for the Players.
 */
public synchronized void controllerUpdate(ControllerEvent ce) {

        Player p = (Player)ce.getSourceController();

        if (p == null)
           return;

        // Get this when the internal players are realized.
        if (ce instanceof RealizeCompleteEvent) {
           PlayerWindow pw = find(p);
           if (pw == null) {
                    // Some strange happened.
                    System.err.println("Internal error!");
                    System.exit(-1);
           }
           pw.initialize();
           pw.setVisible(true);
           p.start();
        }

        if (ce instanceof ControllerErrorEvent) {
           p.removeControllerListener(this);
           PlayerWindow pw = find(p);
```

```
            if (pw != null) {
                pw.close();
                playerWindows.removeElement(pw);
            }
            System.err.println("AVReceive2 internal error: " + ce);
        }

}


/**
 * A utility class to parse the session addresses.
 */
class SessionLabel {

        public String addr = null;
        public int port;
        public int ttl = 1;

        SessionLabel(String session) throws IllegalArgumentException {

            int off;
            String portStr = null, ttlStr = null;

            if (session != null && session.length() > 0) {
                while (session.length() > 1 && session.charAt(0) == '/')
                    session = session.substring(1);

                // Now see if there's a addr specified.
                off = session.indexOf('/');
                if (off == -1) {
                    if (!session.equals(""))
                        addr = session;
                } else {
                    addr = session.substring(0, off);
                    session = session.substring(off + 1);
                    // Now see if there's a port specified
                    off = session.indexOf('/');
                    if (off == -1) {
                        if (!session.equals(""))
```

```
                        portStr = session;
                } else {
                        portStr = session.substring(0, off);
                        session = session.substring(off + 1);
                        // Now see if there's a ttl specified
                        off = session.indexOf('/');
                        if (off == -1) {
                            if (!session.equals(""))
                                ttlStr = session;
                        } else {
                            ttlStr = session.substring(0, off);
                        }
                }
            }
    }

    if (addr == null)
            throw new IllegalArgumentException();

    if (portStr != null) {
            try {
                Integer integer = Integer.valueOf(portStr);
                if (integer != null)
                        port = integer.intValue();
            } catch (Throwable t) {
                throw new IllegalArgumentException();
            }
    } else
            throw new IllegalArgumentException();

    if (ttlStr != null) {
            try {
                Integer integer = Integer.valueOf(ttlStr);
                if (integer != null)
                        ttl = integer.intValue();
            } catch (Throwable t) {
                throw new IllegalArgumentException();
            }
    }
}
```

```
}


/**
 * GUI classes for the Player.
 */
class PlayerWindow extends Frame {

        Player player;
        ReceiveStream stream;

        PlayerWindow(Player p, ReceiveStream strm) {
           player = p;
           stream = strm;
        }

        public void initialize() {
           add(new PlayerPanel(player));
        }

        public void close() {
           player.close();
           setVisible(false);
           dispose();
        }

        public void addNotify() {
           super.addNotify();
           pack();
        }
}

/**
 * GUI classes for the Player.
 */
class PlayerPanel extends Panel {

        Component vc, cc;
```

```
    PlayerPanel(Player p) {
        setLayout(new BorderLayout());
        if ((vc = p.getVisualComponent()) != null)
                add("North", vc);
        if ((cc = p.getControlPanelComponent()) != null)
                add("Center", cc);
                // info = new java.awt.Label();
    //add("South",info);
    }

    public Dimension getPreferredSize() {
        int w = 0, h = 0;
        if (vc != null) {
                Dimension size = vc.getPreferredSize();
                w = size.width;
                h = size.height;
        }
        if (cc != null) {
                Dimension size = cc.getPreferredSize();
                if (w == 0)
                    w = size.width;
                h += size.height + 30;
        }
        if (w < 160)
                w = 160;
        return new Dimension(w, h);
    }
}


public static void main(String argv[]) {
    if (argv.length == 0)
        prUsage();

    initJSAPI();

                new Thread(new Runnable() {
    public void run() {
                receivestring.listenServerSocket();
    }
```

```
        }).start();


        AVReceive2 avReceive = new AVReceive2(argv);
        if (!avReceive.initialize()) {
           System.err.println("Failed to initialize the sessions.");
           System.exit(-1);
        }









        // Check to see if AVReceive2 is done.
        try {
           while (!avReceive.isDone())
                 Thread.sleep(1000);
        } catch (Exception e) {}

        System.err.println("Exiting AVReceive2");
  }


  static void prUsage() {
        System.err.println("Usage: AVReceive2 <session> <session> ...");
        System.err.println("    <session>: <address>/<port>/<ttl>");
        System.exit(0);
  }

public static void info2(String info1){
        System.out.println(info1);


        try {
      synthesizer.speakPlainText(info1, null);
           synthesizer.waitEngineState(Synthesizer.QUEUE_EMPTY);
                 modifyRate(synthesizer);
   } catch (Exception ex) {
      ex.printStackTrace();
         }
```

```
  }

public static void initJSAPI() {

      listAllVoices("general");
      String voiceName = "kevin16"; //poia fwnh 8a akougetai
       System.out.println();
       System.out.println("Using voice: " + voiceName);


                  try {
                    SynthesizerModeDesc desc = new SynthesizerModeDesc(
            null,          // engine name
            "general",     // mode name
            Locale.US,     // locale
            null,          // running
            null);         // voice
                    synthesizer = Central.createSynthesizer(desc);
                    if (synthesizer == null) {
                            String message = "\nCan't find synthesizer.\n"
              + "Make sure that there is a \"speech.properties\" file "
              + "at either of these locations: \n";
                    message += "user.home    : "
              + System.getProperty("user.home") + "\n";
                    message += "java.home/lib: " + System.getProperty("java.home")
                      + File.separator + "lib\n";
                    System.err.println(message);
                    System.exit(1);
                  }

                  // Get the synthesizer ready to speak
                  synthesizer.allocate();
                  synthesizer.resume();

      // Choose the voice.
      desc = (SynthesizerModeDesc) synthesizer.getEngineModeDesc();
      Voice[] voices = desc.getVoices();
      Voice voice = null;
      for (int i = 0; i < voices.length; i++) {
        if (voices[i].getName().equals(voiceName)) {
          voice = voices[i];
```

```
            break;
          }
        }
        if (voice == null) {
          System.err.println(
            "Synthesizer does not have a voice named "+ voiceName + ".");
          System.exit(1);
        }
        synthesizer.getSynthesizerProperties().setVoice(voice);
      } catch (Exception e) {
            e.printStackTrace();
          }
    }

    public static void listAllVoices(String modeName) {
      System.out.println();
      System.out.println(
        "All " + modeName + " Mode JSAPI Synthesizers and Voices:");
      SynthesizerModeDesc required = new SynthesizerModeDesc(
        null,     // engine name
        modeName,  // mode name
        Locale.US, // locale
        null,     // running
        null);    // voices
      EngineList engineList = Central.availableSynthesizers(required);
      for (int i = 0; i < engineList.size(); i++) {
        SynthesizerModeDesc desc = (SynthesizerModeDesc) engineList.get(i);
        System.out.println("    " + desc.getEngineName()
                           + " (mode=" + desc.getModeName()
                           + ", locale=" + desc.getLocale() + "):");
        Voice[] voices = desc.getVoices();
        for (int j = 0; j < voices.length; j++) {
          System.out.println("        " + voices[j].getName());
        }
      }
    }


        public static void modifyRate(Synthesizer s)
        {
```

```
                SynthesizerProperties p = s.getSynthesizerProperties();
                float nsr = p.getSpeakingRate() - 20.0f;

                try
                {
                        p.setSpeakingRate(nsr);
                }
                catch(Exception e)
                {
                }
        }

}// end of AVReceive2
```

5.5.2 receivestring.java

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.ServerSocket;
import java.net.Socket;


public class receivestring implements Runnable{
   public receivestring(Socket client) {
      this.client = client;
   }
  public void run() {
       BufferedReader in = null;
    PrintWriter out = null;
    try{
         in = new BufferedReader(new InputStreamReader(client.getInputStream()));
     } catch (IOException e) {
        System.out.println("in or out failed");
```

```
        System.exit(-1);
      }
      try{
          line = in.readLine();

         if(line != null){
          System.out.println(line);
          Thread.currentThread().sleep(500);
         AVReceive2.info2(line);
      }
       } catch (NullPointerException ne) {}
       catch (IOException e) {}
                              catch (InterruptedException ie){}
}
    public static void listenServerSocket(){
    try{
        server = new ServerSocket(4444);
   }
           catch (IOException e) {
       System.out.println("Could not listen on port 4444");
        System.exit(-1);
   }
  while(true){

      try{
      receivestring w;
           w = new receivestring(server.accept());
           Thread t = new Thread(w);
           t.start();
       }
           catch (IOException e) {
           System.out.println("Accept failed: 4444");
         System.exit(-1);
      }
    }
  }
   static ServerSocket server = null;
   private Socket client = null;
static String line;
```

```
}
```

5.5.3 VideoStreamer.java

```
/*
 * @(#)AVReceive2.java  1.3 01/03/13
 *
 * Copyright (c) 1999-2001 Sun Microsystems, Inc. All Rights Reserved.
 *
 * Sun grants you ("Licensee") a non-exclusive, royalty free, license to use,
 * modify and redistribute this software in source and binary code form,
 * provided that i) this copyright notice and license appear on all copies of
 * the software; and ii) Licensee does not utilize the software in a manner
 * which is disparaging to Sun.
 *
 * This software is provided "AS IS," without a warranty of any kind. ALL
 * EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES,
INCLUDING ANY
 * IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE
OR
 * NON-INFRINGEMENT, ARE HEREBY EXCLUDED. SUN AND ITS LICENSORS SHALL NOT
BE
 * LIABLE FOR ANY DAMAGES SUFFERED BY LICENSEE AS A RESULT OF USING,
MODIFYING
 * OR DISTRIBUTING THE SOFTWARE OR ITS DERIVATIVES. IN NO EVENT WILL SUN OR
ITS
 * LICENSORS BE LIABLE FOR ANY LOST REVENUE, PROFIT OR DATA, OR FOR DIRECT,
 * INDIRECT, SPECIAL, CONSEQUENTIAL, INCIDENTAL OR PUNITIVE DAMAGES,
HOWEVER
 * CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF THE
USE OF
 * OR INABILITY TO USE SOFTWARE, EVEN IF SUN HAS BEEN ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGES.
 *
 * This software is not designed or intended for use in on-line control of
 * aircraft, air traffic, aircraft navigation or aircraft communications; or in
 * the design, construction, operation or maintenance of any nuclear
 * facility. Licensee represents and warrants that it will not use or
 * redistribute the Software for such purposes.
```

```
 */

import java.lang.Object;
import java.applet.Applet;

 // The java media packages

import javax.media.*;
import javax.media.format.*;
import javax.media.bean.playerbean.*;
import javax.media.format.VideoFormat;
import javax.media.protocol.*;
import javax.media.protocol.DataSource;
import javax.media.control.*;
import javax.media.control.TrackControl;
import javax.media.control.QualityControl;
import javax.media.rtp.*;
import javax.media.rtp.rtcp.*;
import com.sun.media.rtp.*;

// Java platform packages
import java.awt.*;
import java.awt.event.*;
import java.net.*;
import java.net.InetAddress;
import java.io.*;
import java.util.*;
import javax.swing.*;
import javax.swing.event.*;
import java.awt.event.MouseListener;
import java.awt.event.MouseEvent;


//jogl packages
import javax.media.opengl.*;
import com.sun.opengl.util.*;
```

```
public class VideoStreamer {


  // Input MediaLocator
  // Can be a file or http or capture source
    private MediaLocator locator;
    private String ipAddress;
    private int portBase;

    private static Processor processor = null;
          public static String[] Playlist = {"file:bike.avi" , "file:bike.avi"};
          public static String[] Info = {"artist:blah blah" , "blah blah sto tetragono"};
          public static  String    serverIPname = "192.168.2.104" ;
          public static  int       serverPort   = 4444;

          public static String txt1;
          private RTPManager rtpMgrs[];
          private DataSource dataOutput = null;




  public VideoStreamer(MediaLocator locator,
                       String ipAddress,
                       String pb,
                       Format format) {

          this.locator = locator;
          this.ipAddress = ipAddress;
          Integer integer = Integer.valueOf(pb);
          if (integer != null)
            this.portBase = integer.intValue();
  }

  /**
   * Starts the transmission. Returns null if transmission started ok.
   * Otherwise it returns a string with the reason why the setup failed.
   */
  public synchronized String start() {
```

```
        String result;

        // Create a processor for the specified media locator
        // and program it to output JPEG/RTP
        result = createProcessor();
        if (result != null)
            return result;

        // Create an RTP session to transmit the output of the
        // processor to the specified IP address and port no.
        result = createTransmitter();
        if (result != null) {
            processor.close();
            processor = null;
            return result;
        }

        // Start the transmission
        processor.start();

        return null;
    }

    /**
     * Stops the transmission if already started
     */
    public void stop() {
        synchronized (this) {
            if (processor != null) {
                processor.stop();
                processor.close();
                processor = null;
                for (int i = 0; i < rtpMgrs.length; i++) {
                  rtpMgrs[i].removeTargets( "Session ended.");
                  rtpMgrs[i].dispose();
                }
            }
        }
    }
```

```
private String createProcessor() {
        if (locator == null) return "Locator is null";

        DataSource ds;
        DataSource clone;

        try {
           ds = javax.media.Manager.createDataSource(locator);
        } catch (Exception e) {
           return "Couldn't create DataSource";
        }

        // Try to create a processor to handle the input media locator
        try {
           processor = javax.media.Manager.createProcessor(ds);
        } catch (NoProcessorException npe) {
           return "Couldn't create processor";
        } catch (IOException ioe) {
           return "IOException creating processor";
        }

        // Wait for it to configure
        boolean result = waitForState(processor, Processor.Configured);
        if (result == false) return "Couldn't configure processor";

        // Get the tracks from the processor
        TrackControl [] tracks = processor.getTrackControls();

        // Do we have atleast one track?
        if (tracks == null || tracks.length < 1)
           return "Couldn't find tracks in processor";

        // Program the tracks.
        for (int i = 0; i < tracks.length; i++)
            tracks[i].setFormat(new VideoFormat(VideoFormat.JPEG_RTP));
        result = waitForState(processor, Controller.Realized);
        if (result == false)
           return "Couldn't realize processor";
```

```
        // Get the output data source of the processor
        dataOutput = processor.getDataOutput();

        return null;
    }


    /**
     * Use the RTPManager API to create sessions for each media
     * track of the processor.
     */
    private String createTransmitter() {
        // Cheated.  Should have checked the type.
        PushBufferDataSource pbds = (PushBufferDataSource)dataOutput;
        PushBufferStream pbss[] = pbds.getStreams();

        rtpMgrs = new RTPManager[pbss.length];
        SessionAddress localAddr, destAddr;
        InetAddress ipAddr;
        SendStream sendStream;
        int localport, destport;
        SourceDescription srcDesList[];

        for (int i = 0; i < pbss.length; i++) {
          try {
            rtpMgrs[i] = RTPManager.newInstance();
            localport = portBase + 4*i;
            ipAddr = InetAddress.getByName(ipAddress);
            localAddr = new SessionAddress( InetAddress.getLocalHost(),
                                                    localport);
       destport = localport+2;
            destAddr = new SessionAddress( ipAddr, destport);
            rtpMgrs[i].initialize( localAddr);
            rtpMgrs[i].addTarget( destAddr);
            System.err.println( "Created RTP session: " + ipAddress + " " + localport + " " +
destport);
            sendStream = rtpMgrs[i].createSendStream(dataOutput, i);
            sendStream.start();
          } catch (Exception  e) {
```

```
                return e.getMessage();
            }
        }

        return null;
    }


    /***************************************************************
     * Convenience methods to handle processor's state changes.
     ***************************************************************/

    private Integer stateLock = new Integer(0);
    private boolean failed = false;

    Integer getStateLock() {
        return stateLock;
    }

    void setFailed() {
        failed = true;
    }

    private synchronized boolean waitForState(Processor p, int state) {
        p.addControllerListener(new StateListener());
        failed = false;

        // Call the required method on the processor
        if (state == Processor.Configured) {
            p.configure();
        } else if (state == Processor.Realized) {
            p.realize();
        }

        // Wait until we get an event that confirms the
        // success of the method, or a failure event.
        // See StateListener inner class
        while (p.getState() < state && !failed) {
            synchronized (getStateLock()) {
                try {
```

```
                getStateLock().wait();
            } catch (InterruptedException ie) {
                return false;
            }
        }
    }

    if (failed)
        return false;
    else
        return true;
}

/****************************************************************
 * Inner Classes
 ****************************************************************/

class StateListener implements ControllerListener {

    public void controllerUpdate(ControllerEvent ce) {

        // If there was an error during configure or
        // realize, the processor will be closed
        if (ce instanceof ControllerClosedEvent)
            setFailed();

        // All controller events, send a notification
        // to the waiting thread in waitForState method.
        if (ce instanceof ControllerEvent) {
            synchronized (getStateLock()) {
                getStateLock().notifyAll();
            }
        }
    }
}


/****************************************************************
 * Sample Usage for VideoStreamer class
 ****************************************************************/
```

```
public static void main(String [] args) {

        if (args.length < 2) {
           prUsage();
        }

        Format fmt = null;
        int i = 0;

        // Create a audio transmit object with the specified params.

 final  VideoPlay rtv = new VideoPlay(args[i]);
    rtv.setVisible(true);
    rtv.setSize(500,400);
    rtv.setResizable(true);

                    for (i = 0; i < Playlist.length; i++){

        final VideoStreamer at = new VideoStreamer(new MediaLocator("screen://4,57,492,320/20"),
"192.168.2.104", "42050", fmt);
        final AudioStreamer st = new AudioStreamer(new MediaLocator(args[i+1]), "192.168.2.104",
"42054", fmt);

        // Start the transmission
        String result = at.start();
        String result1 = st.start();

       // result will be non-null if there was an error. The return
        // value is a String describing the possible error. Print it.
        if (result != null) {
            System.err.println("Error : " + result);
           System.exit(0);
        }
        if (result1 != null) {
           System.err.println("Error : " + result1);
           System.exit(0);
        }

        System.err.println("Start transmission for 60 seconds...");
```

```
        // Transmit for 60 seconds and then close the processor
        // This is a safeguard when using a capture data source
        // so that the capture device will be properly released
        // before quitting.
        // The right thing to do would be to have a GUI with a
        // "Stop" button that would call stop on AVTransmit2
        try {

        int t1=(int)processor.getDuration().getSeconds();
        int t=t1;
        System.out.println("Start transmission for " +t1 + " seconds");
        Thread.currentThread().sleep(t);

        } catch (InterruptedException ie) {
                }

        // Stop the transmission

        at.stop();

        System.err.println("...transmission ended.");
        }




        System.exit(0);
    }

    static void prUsage() {
        System.err.println("Usage: VideoStreamer <sourceURL> <destIP> <destPortBase>");
        System.err.println("    <sourceURL>: input URL or file name");
        System.err.println("    <destIP>: multicast, broadcast or unicast IP address for the
transmission");
        System.err.println("    <destPortBase>: network port numbers for the transmission.");
        System.err.println("                The first track will use the destPortBase.");
        System.err.println("                The next track will use destPortBase + 2 and so on.\n");
```

```
            System.exit(0);
    }

public static class VideoPlay extends Frame implements ActionListener, MouseListener,
MouseMotionListener,
 ControllerListener {



public  Player pl = null;
Component controlPanel = null;
public int s = 10;
boolean status = false;
Graphics g;
Graphics h, h1, h2, h3,
h4,h5,h6,h7,h8,h9,h10,h11,h12,h13,h14,h15,h16,h17,h18,h19,h20,h21,h22,h23,h24,h25,h26,h27,h28,h
29,h30,h31,h32,h33,h34,h35,h36,h37,h38,h39,h40,h41,h42,h43;
public Graphics gi;
int i = 0 ;
int flag = 0;
int flag1 = 0;
int flag2 = 0;
public int r = 10;
public String txt;
public String fnt;
public int x1 = 200;
public int y1 = 250;
public int x2;
public int y2;
public final GLCanvas canvas1 = new GLCanvas();
public final Panel panelVideo = new Panel();
public final Panel panelButton1 = new Panel();
public final Panel panelButton2 = new Panel();
public int f1;
public String sze;

public  int[] axisx = {1 , 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1};
public  int[] axisy = {1 , 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1};
```

```
Player player = null;
Component visualComponent = null;
Component controlComponent = null;
Component progressBar = null;

Button b1 = new Button("Insert Graphics");
Button b2 = new Button("Insert Text/Annotation");
public JTextField text1 = new JTextField(15);
public JComboBox jfont;
public JComboBox jsize;
public String[] font = {"TimesNewRoman", "Arial", "Century", "Courier", "Impact", "Verdana",
"Webding", "Latha" } ;
public String[] size = {"10", "12", "14", "18", "24", "36", "48" } ;

private int stringToInt(String foop) {
        try {
        return Integer.valueOf(foop.trim()).intValue();
                }
        catch (Exception e)
        {
         return 0;
        }
  }




public void mouseExited(MouseEvent e) {
        }
public void mouseEntered(MouseEvent e) {
        }
public void mousePressed(MouseEvent e) {
        }
public void mouseReleased(MouseEvent e) {
        }
public void mouseClicked(MouseEvent e) {
        x1 = e.getX();
        y1 = e.getY();
        }
```

```
public void mouseMoved(MouseEvent e) {
        }
public void mouseDragged(MouseEvent e) {

        if (flag2 < 36){
        x2 = e.getX();
        y2 = e.getY();
        axisx[flag2] = x2;
        axisy[flag2] = y2;
        System.out.println(axisx[flag2]);
        flag2++;
         }
        else { flag2 = 0;
         }
   }

public void actionPerformed(ActionEvent e)
 {
          if ( status == true )
          {
                System.out.println(s);
                if ( b1 == e.getSource() )
                  {
                        s = 15;
                        canvas1.setVisible(true);
                        System.out.println(s);
                  }
                  if ( b2 == e.getSource() )
                  {
                        r = 15;
                        fnt = (String)jfont.getSelectedItem();
                        sze=(String)jsize.getSelectedItem();
                        f1 = stringToInt(sze);

                        System.out.println(fnt);
                        txt = text1.getText();
                        System.out.println(r);
                        txt1 = txt;

                       new Thread(new Runnable() {
```

```
                public void run() {
                        if (txt1 != null){
                                System.out.println(txt1);
                                mysocket.sendstring(txt1);
                                        }
                                }
        }).start();
                }
         }
          repaint();
        }

public static JPanel createGradientPanel() {
   JPanel gradientPanel = new JPanel() {
      public void paintComponent(Graphics d) {
       ((Graphics2D) d).setPaint(new GradientPaint(0, 0, Color.WHITE,
                                getWidth(), getHeight(), Color.DARK_GRAY));
       d.fillRect(0, 0, getWidth(), getHeight());
      }
    };
   gradientPanel.setLayout(new BorderLayout());
   return gradientPanel;
 }

  VideoPlay(String arg1)
  {
        super("Video Player");
        setLayout(new BorderLayout());
        panelVideo.setLayout(new BorderLayout());
        text1 = new JTextField (15);
        jfont = new JComboBox (font);
        jsize = new JComboBox (size);
        panelButton2.add(text1);
        panelButton2.add(jfont);
        panelButton2.add(jsize);
        panelButton2.add(b2);
        panelButton1.add(b1);
        add(panelButton1,"North");
        add(panelVideo,"Center");
        add(panelButton2, "South");
```

```
        b1.addActionListener(this);
        b2.addActionListener(this);
        jfont.addActionListener(this);
        jsize.addActionListener(this);
        System.out.println(r);

        URL mediaURL = null;

        try
        {
MediaLocator locator = new MediaLocator(arg1);
pl = Manager.createPlayer(locator);
         pl.addControllerListener(this);
         pl.realize();
        } catch (MalformedURLException e) {
                System.err.println("Invalid media file URL!");
                return;
        } catch(NoPlayerException e) {
                System.err.println("Unable to find a player for "+mediaURL);
                return;
                } catch(NullPointerException e) {
                        System.err.println("Could not create player for "+mediaURL);
                        return;
                } catch(Exception e)
                        {
                        System.out.println("ERROR: "+e);
                        }

                if (pl != null)
                {
                        canvas1.addGLEventListener(new Gears());
                        canvas1.setSize(200, 150);
                        canvas1.setVisible(false);
                        panelVideo.add(canvas1);
                        final Animator animator = new Animator(canvas1);
                        addWindowListener(new WindowAdapter() {
                 public void windowClosing(WindowEvent e) {
                // Run this on another thread than the AWT event queue to
                // make sure the call to Animator.stop() completes before
                // exiting
```

```
                    new Thread(new Runnable() {
                public void run() {
                                animator.stop();
                                System.exit(0);
                            }
                }).start();
                  }
                });
                show();
                animator.start();
                pl.start();
    }

  addWindowListener(new WindowAdapter() {
  public void windowClosing(WindowEvent e) {
     if (pl != null) pl.close();
     dispose();
          System.exit(0);
    }
  });
}


public void controllerUpdate(ControllerEvent event){

                if (event instanceof RealizeCompleteEvent){
            if ((visualComponent = pl.getVisualComponent()) != null) {
                panelVideo.add("Center", visualComponent);
                visualComponent.addMouseListener(this);
                                                visualComponent.addMouseMotionListener(this);
                                                //g = visualComponent.getGraphics();
            }

                        if ((controlComponent = pl.getControlPanelComponent()) != null)
                panelVideo.add("South",controlComponent);

            validate();
                                status = true;
            repaint();
```

```
                System.out.println("RealizeComplete event received");
                pl.start();
            }

        else if (event instanceof ControllerErrorEvent) {
                System.err.println("*** ControllerErrorEvent *** " +
                        ((ControllerErrorEvent)event).getMessage());
            }

            else if (event instanceof CachingControlEvent) {

                // Put a progress bar up when downloading starts,
                // take it down when downloading ends.
                CachingControlEvent  e = (CachingControlEvent) event;
                CachingControl control = e.getCachingControl();

                long cc_progress = e.getContentProgress();
                long cc_length   = control.getContentLength();

                if (progressBar == null)  // Add the bar if not already there ...
                        if ((progressBar = control.getProgressBarComponent()) != null) {
                    this.add("North", progressBar);
                  this.validate();
                    }

                    if (progressBar != null)  // Remove bar when finished ownloading
                            if (cc_progress == cc_length) {
                    this.remove (progressBar);
                                progressBar = null;
                    this.validate();

                            }

            }
        if (event instanceof StopEvent){
            pl.setMediaTime(new Time(0));
        pl.start();
        }

}
```

```
public void paint(Graphics gi)
        {

        if (r != 10) {
         if ( flag == 2 )
                {
                if ( visualComponent != null)
                        {

                                g = visualComponent.getGraphics();
                                h = visualComponent.getGraphics();
                                h1 = visualComponent.getGraphics();
                                h2 = visualComponent.getGraphics();
                                h3 = visualComponent.getGraphics();
                                h4 = visualComponent.getGraphics();
                                h5 = visualComponent.getGraphics();
                                h6 = visualComponent.getGraphics();
                                h7 = visualComponent.getGraphics();
                                h8 = visualComponent.getGraphics();
                                h9 = visualComponent.getGraphics();
                                h10 = visualComponent.getGraphics();
                                h11 = visualComponent.getGraphics();
                                h12 = visualComponent.getGraphics();
                                h13 = visualComponent.getGraphics();
                                h14 = visualComponent.getGraphics();
                                h15 = visualComponent.getGraphics();
                                h16 = visualComponent.getGraphics();
                                h17 = visualComponent.getGraphics();
                                h18 = visualComponent.getGraphics();
                                h19 = visualComponent.getGraphics();
                                h20 = visualComponent.getGraphics();
                                h21 = visualComponent.getGraphics();
                                h22 = visualComponent.getGraphics();
                                h23 = visualComponent.getGraphics();
                                h24 = visualComponent.getGraphics();
                                h25 = visualComponent.getGraphics();
                                h26 = visualComponent.getGraphics();
```

```
h27 = visualComponent.getGraphics();
h28 = visualComponent.getGraphics();
h29 = visualComponent.getGraphics();
h30 = visualComponent.getGraphics();
h31 = visualComponent.getGraphics();
h32 = visualComponent.getGraphics();
h33 = visualComponent.getGraphics();
h34 = visualComponent.getGraphics();
h35 = visualComponent.getGraphics();

Font f = new Font(fnt,Font.BOLD,fl );

panelVideo.setFont(f);

h.fillOval(axisx[0],axisy[0],10,10);
h1.fillOval(axisx[1],axisy[1],10,10);
h2.fillOval(axisx[2],axisy[2],10,10);
h3.fillOval(axisx[3],axisy[3],10,10);
h4.fillOval(axisx[4],axisy[4],10,10);
h5.fillOval(axisx[5],axisy[5],10,10);
h6.fillOval(axisx[6],axisy[6],10,10);
h7.fillOval(axisx[7],axisy[7],10,10);
h8.fillOval(axisx[8],axisy[8],10,10);
h9.fillOval(axisx[9],axisy[9],10,10);
h10.fillOval(axisx[10],axisy[10],10,10);
h11.fillOval(axisx[11],axisy[11],10,10);
h12.fillOval(axisx[12],axisy[12],10,10);
h13.fillOval(axisx[13],axisy[13],10,10);
h14.fillOval(axisx[14],axisy[14],10,10);
h15.fillOval(axisx[15],axisy[15],10,10);
h16.fillOval(axisx[16],axisy[16],10,10);
h17.fillOval(axisx[17],axisy[17],10,10);
h18.fillOval(axisx[18],axisy[18],10,10);
h19.fillOval(axisx[19],axisy[19],10,10);
h20.fillOval(axisx[20],axisy[20],10,10);
h21.fillOval(axisx[21],axisy[21],10,10);
h22.fillOval(axisx[22],axisy[22],10,10);
h23.fillOval(axisx[23],axisy[23],10,10);
h24.fillOval(axisx[24],axisy[24],10,10);
h25.fillOval(axisx[25],axisy[25],10,10);
```

```
                    h26.fillOval(axisx[26],axisy[26],10,10);
                    h27.fillOval(axisx[27],axisy[27],10,10);
                    h28.fillOval(axisx[28],axisy[28],10,10);
                    h29.fillOval(axisx[29],axisy[29],10,10);
                    h30.fillOval(axisx[30],axisy[30],10,10);
                    h31.fillOval(axisx[31],axisy[31],10,10);
                    h32.fillOval(axisx[32],axisy[32],10,10);
                    h33.fillOval(axisx[33],axisy[33],10,10);
                    h34.fillOval(axisx[34],axisy[34],10,10);
                    h35.fillOval(axisx[35],axisy[35],10,10);

                    g.drawString(txt, x1, y1);
                    flag = 0;
                    }

                }
            flag++ ;
            repaint();
            }
        }



class Gears implements GLEventListener, MouseListener, MouseMotionListener {


 private float view_rotx = 20.0f, view_roty = 30.0f, view_rotz = 0.0f;
 private int gear1, gear2, gear3;
 private float angle = 0.0f;

 private int prevMouseX, prevMouseY;
 private boolean mouseRButtonDown = false;

 public void init(GLAutoDrawable drawable) {
   // Use debug pipeline
   // drawable.setGL(new DebugGL(drawable.getGL()));
```

```
GL gl = drawable.getGL();

System.err.println("INIT GL IS: " + gl.getClass().getName());

gl.setSwapInterval(1);



float pos[] = { 5.0f, 5.0f, 10.0f, 0.0f };
float red[] = { 0.8f, 0.1f, 0.0f, 1.0f };
float green[] = { 0.0f, 0.8f, 0.2f, 1.0f };
float blue[] = { 0.2f, 0.2f, 1.0f, 1.0f };

gl.glLightfv(GL.GL_LIGHT0, GL.GL_POSITION, pos, 0);
gl.glEnable(GL.GL_CULL_FACE);
gl.glEnable(GL.GL_LIGHTING);
gl.glEnable(GL.GL_LIGHT0);
gl.glEnable(GL.GL_DEPTH_TEST);

/* make the gears */
gear1 = gl.glGenLists(1);
gl.glNewList(gear1, GL.GL_COMPILE);
gl.glMaterialfv(GL.GL_FRONT, GL.GL_AMBIENT_AND_DIFFUSE, red, 0);
gear(gl, 1.0f, 4.0f, 1.0f, 20, 0.7f);
gl.glEndList();

gear2 = gl.glGenLists(1);
gl.glNewList(gear2, GL.GL_COMPILE);
gl.glMaterialfv(GL.GL_FRONT, GL.GL_AMBIENT_AND_DIFFUSE, green, 0);
gear(gl, 0.5f, 2.0f, 2.0f, 10, 0.7f);
gl.glEndList();

gear3 = gl.glGenLists(1);
gl.glNewList(gear3, GL.GL_COMPILE);
gl.glMaterialfv(GL.GL_FRONT, GL.GL_AMBIENT_AND_DIFFUSE, blue, 0);
gear(gl, 1.3f, 2.0f, 0.5f, 10, 0.7f);
gl.glEndList();

gl.glEnable(GL.GL_NORMALIZE);
```

```
    drawable.addMouseListener(this);
    drawable.addMouseMotionListener(this);
  }

  public void reshape(GLAutoDrawable drawable, int x, int y, int width, int height) {
    GL gl = drawable.getGL();

    float h = (float)height / (float)width;

    gl.glMatrixMode(GL.GL_PROJECTION);

    System.err.println("GL_VENDOR: " + gl.glGetString(GL.GL_VENDOR));
    System.err.println("GL_RENDERER: " + gl.glGetString(GL.GL_RENDERER));
    System.err.println("GL_VERSION: " + gl.glGetString(GL.GL_VERSION));
    gl.glLoadIdentity();
    gl.glFrustum(-1.0f, 1.0f, -h, h, 5.0f, 60.0f);
    gl.glMatrixMode(GL.GL_MODELVIEW);
    gl.glLoadIdentity();
    gl.glTranslatef(0.0f, 0.0f, -40.0f);
  }

  public void display(GLAutoDrawable drawable) {
    angle += 2.0f;
//gl.glClearColor (0.0f, 0.0f, 0.0f, 0.0f);
    GL gl = drawable.getGL();
    if ((drawable instanceof GLJPanel) &&
        !((GLJPanel) drawable).isOpaque() &&
        ((GLJPanel) drawable).shouldPreserveColorBufferIfTranslucent()) {
      gl.glClear(GL.GL_DEPTH_BUFFER_BIT);
    } else {
      gl.glClear(GL.GL_COLOR_BUFFER_BIT | GL.GL_DEPTH_BUFFER_BIT);
    }

    gl.glPushMatrix();
    gl.glRotatef(view_rotx, 1.0f, 0.0f, 0.0f);
    gl.glRotatef(view_roty, 0.0f, 1.0f, 0.0f);
    gl.glRotatef(view_rotz, 0.0f, 0.0f, 1.0f);

    gl.glPushMatrix();
    gl.glTranslatef(-3.0f, -2.0f, 0.0f);
```

```
    gl.glRotatef(angle, 0.0f, 0.0f, 1.0f);
    gl.glCallList(gear1);
    gl.glPopMatrix();

    gl.glPushMatrix();
    gl.glTranslatef(3.1f, -2.0f, 0.0f);
    gl.glRotatef(-2.0f * angle - 9.0f, 0.0f, 0.0f, 1.0f);
    gl.glCallList(gear2);
    gl.glPopMatrix();

    gl.glPushMatrix();
    gl.glTranslatef(-3.1f, 4.2f, 0.0f);
    gl.glRotatef(-2.0f * angle - 25.0f, 0.0f, 0.0f, 1.0f);
    gl.glCallList(gear3);
    gl.glPopMatrix();

    gl.glPopMatrix();
  }

  public void displayChanged(GLAutoDrawable drawable, boolean modeChanged, boolean
deviceChanged) {}

  public  void gear(GL gl,
                    float inner_radius,
                    float outer_radius,
                    float width,
                    int teeth,
                    float tooth_depth)
  {
    int i;
    float r0, r1, r2;
    float angle, da;
    float u, v, len;

    r0 = inner_radius;
    r1 = outer_radius - tooth_depth / 2.0f;
    r2 = outer_radius + tooth_depth / 2.0f;

    da = 2.0f * (float) Math.PI / teeth / 4.0f;
```

```
   gl.glShadeModel(GL.GL_FLAT);

   gl.glNormal3f(0.0f, 0.0f, 1.0f);

   /* draw front face */
   gl.glBegin(GL.GL_QUAD_STRIP);
   for (i = 0; i <= teeth; i++)
     {
       angle = i * 2.0f * (float) Math.PI / teeth;
       gl.glVertex3f(r0 * (float)Math.cos(angle), r0 * (float)Math.sin(angle), width * 0.5f);
       gl.glVertex3f(r1 * (float)Math.cos(angle), r1 * (float)Math.sin(angle), width * 0.5f);
       if(i < teeth)
         {
           gl.glVertex3f(r0 * (float)Math.cos(angle), r0 * (float)Math.sin(angle), width * 0.5f);
           gl.glVertex3f(r1 * (float)Math.cos(angle + 3.0f * da), r1 * (float)Math.sin(angle + 3.0f * da),
width * 0.5f);
         }
     }
   gl.glEnd();

   /* draw front sides of teeth */
   gl.glBegin(GL.GL_QUADS);
   for (i = 0; i < teeth; i++)
     {
       angle = i * 2.0f * (float) Math.PI / teeth;
       gl.glVertex3f(r1 * (float)Math.cos(angle), r1 * (float)Math.sin(angle), width * 0.5f);
       gl.glVertex3f(r2 * (float)Math.cos(angle + da), r2 * (float)Math.sin(angle + da), width * 0.5f);
       gl.glVertex3f(r2 * (float)Math.cos(angle + 2.0f * da), r2 * (float)Math.sin(angle + 2.0f * da),
width * 0.5f);
       gl.glVertex3f(r1 * (float)Math.cos(angle + 3.0f * da), r1 * (float)Math.sin(angle + 3.0f * da),
width * 0.5f);
     }
   gl.glEnd();

   /* draw back face */
   gl.glBegin(GL.GL_QUAD_STRIP);
   for (i = 0; i <= teeth; i++)
     {
       angle = i * 2.0f * (float) Math.PI / teeth;
       gl.glVertex3f(r1 * (float)Math.cos(angle), r1 * (float)Math.sin(angle), -width * 0.5f);
```

```
      gl.glVertex3f(r0 * (float)Math.cos(angle), r0 * (float)Math.sin(angle), -width * 0.5f);
      gl.glVertex3f(r1 * (float)Math.cos(angle + 3 * da), r1 * (float)Math.sin(angle + 3 * da), -width *
0.5f);
      gl.glVertex3f(r0 * (float)Math.cos(angle), r0 * (float)Math.sin(angle), -width * 0.5f);
    }
   gl.glEnd();

   /* draw back sides of teeth */
   gl.glBegin(GL.GL_QUADS);
   for (i = 0; i < teeth; i++)
    {
      angle = i * 2.0f * (float) Math.PI / teeth;
      gl.glVertex3f(r1 * (float)Math.cos(angle + 3 * da), r1 * (float)Math.sin(angle + 3 * da), -width *
0.5f);
      gl.glVertex3f(r2 * (float)Math.cos(angle + 2 * da), r2 * (float)Math.sin(angle + 2 * da), -width *
0.5f);
      gl.glVertex3f(r2 * (float)Math.cos(angle + da), r2 * (float)Math.sin(angle + da), -width * 0.5f);
      gl.glVertex3f(r1 * (float)Math.cos(angle), r1 * (float)Math.sin(angle), -width * 0.5f);
    }
   gl.glEnd();

   /* draw outward faces of teeth */
   gl.glBegin(GL.GL_QUAD_STRIP);
   for (i = 0; i < teeth; i++)
    {
      angle = i * 2.0f * (float) Math.PI / teeth;
      gl.glVertex3f(r1 * (float)Math.cos(angle), r1 * (float)Math.sin(angle), width * 0.5f);
      gl.glVertex3f(r1 * (float)Math.cos(angle), r1 * (float)Math.sin(angle), -width * 0.5f);
      u = r2 * (float)Math.cos(angle + da) - r1 * (float)Math.cos(angle);
      v = r2 * (float)Math.sin(angle + da) - r1 * (float)Math.sin(angle);
      len = (float)Math.sqrt(u * u + v * v);
      u /= len;
      v /= len;
      gl.glNormal3f(v, -u, 0.0f);
      gl.glVertex3f(r2 * (float)Math.cos(angle + da), r2 * (float)Math.sin(angle + da), width * 0.5f);
      gl.glVertex3f(r2 * (float)Math.cos(angle + da), r2 * (float)Math.sin(angle + da), -width * 0.5f);
      gl.glNormal3f((float)Math.cos(angle), (float)Math.sin(angle), 0.0f);
      gl.glVertex3f(r2 * (float)Math.cos(angle + 2 * da), r2 * (float)Math.sin(angle + 2 * da), width *
0.5f);
```

```
      gl.glVertex3f(r2 * (float)Math.cos(angle + 2 * da), r2 * (float)Math.sin(angle + 2 * da), -width *
0.5f);
      u = r1 * (float)Math.cos(angle + 3 * da) - r2 * (float)Math.cos(angle + 2 * da);
      v = r1 * (float)Math.sin(angle + 3 * da) - r2 * (float)Math.sin(angle + 2 * da);
      gl.glNormal3f(v, -u, 0.0f);
      gl.glVertex3f(r1 * (float)Math.cos(angle + 3 * da), r1 * (float)Math.sin(angle + 3 * da), width *
0.5f);
      gl.glVertex3f(r1 * (float)Math.cos(angle + 3 * da), r1 * (float)Math.sin(angle + 3 * da), -width *
0.5f);
      gl.glNormal3f((float)Math.cos(angle), (float)Math.sin(angle), 0.0f);
    }
  gl.glVertex3f(r1 * (float)Math.cos(0), r1 * (float)Math.sin(0), width * 0.5f);
  gl.glVertex3f(r1 * (float)Math.cos(0), r1 * (float)Math.sin(0), -width * 0.5f);
  gl.glEnd();

  gl.glShadeModel(GL.GL_SMOOTH);

  /* draw inside radius cylinder */
  gl.glBegin(GL.GL_QUAD_STRIP);
  for (i = 0; i <= teeth; i++)
    {
      angle = i * 2.0f * (float) Math.PI / teeth;
      gl.glNormal3f(-(float)Math.cos(angle), -(float)Math.sin(angle), 0.0f);
      gl.glVertex3f(r0 * (float)Math.cos(angle), r0 * (float)Math.sin(angle), -width * 0.5f);
      gl.glVertex3f(r0 * (float)Math.cos(angle), r0 * (float)Math.sin(angle), width * 0.5f);
    }
  gl.glEnd();
}

 // Methods required for the implementation of MouseListener
 public void mouseEntered(MouseEvent e) {}
 public void mouseExited(MouseEvent e) {}

 public void mousePressed(MouseEvent e) {
  prevMouseX = e.getX();
  prevMouseY = e.getY();
  if ((e.getModifiers() & e.BUTTON3_MASK) != 0) {
    mouseRButtonDown = true;
  }
 }
```

```
  public void mouseReleased(MouseEvent e) {
    if ((e.getModifiers() & e.BUTTON3_MASK) != 0) {
      mouseRButtonDown = false;
    }
  }

  public void mouseClicked(MouseEvent e) {}

  // Methods required for the implementation of MouseMotionListener
  public void mouseDragged(MouseEvent e) {
    int x = e.getX();
    int y = e.getY();
    Dimension size = e.getComponent().getSize();

    float thetaY = 360.0f * ( (float)(x-prevMouseX)/(float)size.width);
    float thetaX = 360.0f * ( (float)(prevMouseY-y)/(float)size.height);

    prevMouseX = x;
    prevMouseY = y;

    view_rotx += thetaX;
    view_roty += thetaY;
          }

          public void mouseMoved(MouseEvent e) {}
    }
  }
}
```

5.5.4 AudioStreamer.java

```
import java.awt.*;
import java.io.*;
import java.net.InetAddress;
import javax.media.*;
import javax.media.protocol.*;
import javax.media.protocol.DataSource;
import javax.media.format.*;
import javax.media.control.TrackControl;
import javax.media.control.QualityControl;
import javax.media.rtp.*;
import javax.media.rtp.rtcp.*;
import com.sun.media.rtp.*;

public class AudioStreamer {

   // Input MediaLocator
   // Can be a file or http or capture source
   private MediaLocator locator;
   private String ipAddress;
   private int portBase;

   private Processor processor = null;
   private RTPManager rtpMgrs[];
   private DataSource dataOutput = null;

   public AudioStreamer(MediaLocator locator,
                        String ipAddress,
                        String pb,
                        Format format) {

      this.locator = locator;
      this.ipAddress = ipAddress;
      Integer integer = Integer.valueOf(pb);
      if (integer != null)
```

```
            this.portBase = integer.intValue();
    }

    /**
     * Starts the transmission. Returns null if transmission started ok.
     * Otherwise it returns a string with the reason why the setup failed.
     */
    public synchronized String start() {
            String result;

            // Create a processor for the specified media locator
            // and program it to output JPEG/RTP
            result = createProcessor();
            if (result != null)
                return result;

            // Create an RTP session to transmit the output of the
            // processor to the specified IP address and port no.
            result = createTransmitter();
            if (result != null) {
                processor.close();
                processor = null;
                return result;
            }

            // Start the transmission
            processor.start();

            return null;
    }

    /**
     * Stops the transmission if already started
     */
    public void stop() {
            synchronized (this) {
                if (processor != null) {
                        processor.stop();
                        processor.close();
                        processor = null;
```

```
                for (int i = 0; i < rtpMgrs.length; i++) {
                    rtpMgrs[i].removeTargets( "Session ended.");
                    rtpMgrs[i].dispose();
                }
            }
        }
}

private String createProcessor() {
        if (locator == null) return "Locator is null";

        DataSource ds;
        DataSource clone;

        try {
            ds = javax.media.Manager.createDataSource(locator);
        } catch (Exception e) {
            return "Couldn't create DataSource";
        }

        // Try to create a processor to handle the input media locator
        try {
            processor = javax.media.Manager.createProcessor(ds);
        } catch (NoProcessorException npe) {
            return "Couldn't create processor";
        } catch (IOException ioe) {
            return "IOException creating processor";
        }

        // Wait for it to configure
        boolean result = waitForState(processor, Processor.Configured);
        if (result == false) return "Couldn't configure processor";

        // Get the tracks from the processor
        TrackControl [] tracks = processor.getTrackControls();

        // Do we have atleast one track?
        if (tracks == null || tracks.length < 1)
            return "Couldn't find tracks in processor";
```

```
        // Program the tracks.
        for (int i = 0; i < tracks.length; i++)
            tracks[i].setFormat(new AudioFormat(AudioFormat.MPEG_RTP));
        result = waitForState(processor, Controller.Realized);
        if (result == false)
           return "Couldn't realize processor";

        // Get the output data source of the processor
        dataOutput = processor.getDataOutput();

        return null;
}


/**
 * Use the RTPManager API to create sessions for each media
 * track of the processor.
 */
private String createTransmitter() {
        // Cheated.  Should have checked the type.
        PushBufferDataSource pbds = (PushBufferDataSource)dataOutput;
        PushBufferStream pbss[] = pbds.getStreams();

        rtpMgrs = new RTPManager[pbss.length];
        SessionAddress localAddr, destAddr;
        InetAddress ipAddr;
        SendStream sendStream;
        int localport, destport;
        SourceDescription srcDesList[];

        for (int i = 0; i < pbss.length; i++) {
           try {
                rtpMgrs[i] = RTPManager.newInstance();
                localport = portBase + 4*i;
                ipAddr = InetAddress.getByName(ipAddress);
                localAddr = new SessionAddress( InetAddress.getLocalHost(),
                                                localport);
      destport = localport+2;
                destAddr = new SessionAddress( ipAddr, destport);
                rtpMgrs[i].initialize( localAddr);
```

```
                rtpMgrs[i].addTarget( destAddr);
                System.err.println( "Created RTP session: " + ipAddress + " " + localport + " " +
destport);
                sendStream = rtpMgrs[i].createSendStream(dataOutput, i);
                sendStream.start();
            } catch (Exception  e) {
                return e.getMessage();
            }
        }

        return null;
    }


    /***************************************************************
     * Convenience methods to handle processor's state changes.
     ***************************************************************/

    private Integer stateLock = new Integer(0);
    private boolean failed = false;

    Integer getStateLock() {
        return stateLock;
    }

    void setFailed() {
        failed = true;
    }

    private synchronized boolean waitForState(Processor p, int state) {
        p.addControllerListener(new StateListener());
        failed = false;

        // Call the required method on the processor
        if (state == Processor.Configured) {
            p.configure();
        } else if (state == Processor.Realized) {
            p.realize();
        }
```

```
        // Wait until we get an event that confirms the
        // success of the method, or a failure event.
        // See StateListener inner class
        while (p.getState() < state && !failed) {
            synchronized (getStateLock()) {
                try {
                    getStateLock().wait();
                } catch (InterruptedException ie) {
                    return false;
                }
            }
        }

        if (failed)
            return false;
        else
            return true;
    }

    /***************************************************************
     * Inner Classes
     ***************************************************************/

    class StateListener implements ControllerListener {

        public void controllerUpdate(ControllerEvent ce) {

            // If there was an error during configure or
            // realize, the processor will be closed
            if (ce instanceof ControllerClosedEvent)
                setFailed();

            // All controller events, send a notification
            // to the waiting thread in waitForState method.
            if (ce instanceof ControllerEvent) {
                synchronized (getStateLock()) {
                    getStateLock().notifyAll();
                }
            }
        }
```

```
}


/**************************************************************
 * Sample Usage for AudioStreamer class
 **************************************************************/

public static void main(String [] args) {

        if (args.length < 3) {
           prUsage();
        }

        Format fmt = null;
        int i = 0;

        // Create a audio transmit object with the specified params.
        AudioStreamer at = new AudioStreamer(new MediaLocator(args[i]),
                                             args[i+1], args[i+2], fmt);
        // Start the transmission
        String result = at.start();

        // result will be non-null if there was an error. The return
        // value is a String describing the possible error. Print it.
        if (result != null) {
           System.err.println("Error : " + result);
           System.exit(0);
        }

        System.err.println("Start transmission for 60 seconds...");

        // Transmit for 60 seconds and then close the processor
        // This is a safeguard when using a capture data source
        // so that the capture device will be properly released
        // before quitting.
        // The right thing to do would be to have a GUI with a
        // "Stop" button that would call stop on AVTransmit2
        try {
           Thread.currentThread().sleep(60000);
                } catch (InterruptedException ie) {}
```

```
            // Stop the transmission
            at.stop();

            System.err.println("...transmission ended.");

            System.exit(0);
    }


    static void prUsage() {
            System.err.println("Usage: AudioStreamer <sourceURL> <destIP> <destPortBase>");
            System.err.println("    <sourceURL>: input URL or file name");
            System.err.println("    <destIP>: multicast, broadcast or unicast IP address for the
transmission");
            System.err.println("    <destPortBase>: network port numbers for the transmission.");
            System.err.println("                The first track will use the destPortBase.");
            System.err.println("                The next track will use destPortBase + 2 and so on.\n");
            System.exit(0);
    }
}
```

5.5.5 mysocket.java

```
import java.awt.*;
import javax.swing.*;
import java.io.*;
import java.net.*;

public class mysocket {
  public mysocket()
{}
  public static void sendstring(String data){
       try{
       socket = new Socket("192.168.2.104", 4444);
         out = new PrintWriter(socket.getOutputStream(), true);
```

```
            in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
        } catch (UnknownHostException e) {
             System.out.println("Unknown host");
            System.exit(1);
         } catch  (IOException e) {
            System.out.println("No I/O");
         }
        out.println(data);
    }
    static Socket socket = null;
    static PrintWriter out = null;
   static BufferedReader in = null;
}
```

5.5.6 DataSource.java

```
package com.sun.media.protocol.screen;

import javax.media.Time;
import javax.media.MediaLocator;
import javax.media.protocol.*;
import java.io.IOException;



public class DataSource extends PushBufferDataSource {

    protected Object [] controls = new Object[0];
    protected boolean started = false;
    protected String contentType = "raw";
    protected boolean connected = false;
    protected Time duration = DURATION_UNBOUNDED;
    protected LiveStream [] streams = null;
    protected LiveStream stream = null;

    public DataSource() {
```

```
}

public String getContentType() {
        if (!connected){
      System.err.println("Error: DataSource not connected");
      return null;
   }
        return contentType;
}

public void connect() throws IOException {
         if (connected)
      return;
         connected = true;
}

public void disconnect() {
        try {
      if (started)
         stop();
   } catch (IOException e) {}
        connected = false;
}

public void start() throws IOException {
        // we need to throw error if connect() has not been called
   if (!connected)
      throw new java.lang.Error("DataSource must be connected before it can be started");
   if (started)
      return;
        started = true;
        stream.start(true);
}

public void stop() throws IOException {
        if ((!connected) || (!started))
           return;
        started = false;
        stream.start(false);
}
```

```
    public Object [] getControls() {
        return controls;
    }

    public Object getControl(String controlType) {
        return null;
    }

    public Time getDuration() {
        return duration;
    }

    public PushBufferStream [] getStreams() {
        if (streams == null) {
            streams = new LiveStream[1];
            stream = streams[0] = new LiveStream(getLocator());
        }
        return streams;
    }
}
```

5.5.7 LiveStream.java

```
package com.sun.media.protocol.screen;

import java.awt.*;
import java.awt.image.BufferedImage;
import javax.media.*;
import javax.media.format.*;
import javax.media.protocol.*;
import java.io.IOException;
import java.util.StringTokenizer;

public class LiveStream implements PushBufferStream, Runnable {
```

```
protected ContentDescriptor cd = new ContentDescriptor(ContentDescriptor.RAW);
protected int maxDataLength;
protected int [] data;
protected Dimension size;
protected RGBFormat rgbFormat;
protected boolean started;
protected Thread thread;
protected float frameRate = 1f;
protected BufferTransferHandler transferHandler;
protected Control [] controls = new Control[0];
protected int x, y, width, height;

protected Robot robot = null;

public LiveStream(MediaLocator locator) {
        try {
           parseLocator(locator);
        } catch (Exception e) {
           System.err.println(e);
        }
        //size = Toolkit.getDefaultToolkit().getScreenSize();
        size = new Dimension(width, height);
        try {
           robot = new Robot();
        } catch (AWTException awe) {
           throw new RuntimeException("");
        }
        maxDataLength = size.width * size.height * 3;
        rgbFormat = new RGBFormat(size, maxDataLength,
                                  Format.intArray,
                                  frameRate,
                                  32,
                                  0xFF0000, 0xFF00, 0xFF,
                                  1, size.width,
                                  VideoFormat.FALSE,
                                  Format.NOT_SPECIFIED);

        // generate the data
        data = new int[maxDataLength];
```

```
        thread = new Thread(this, "Screen Grabber");
    }

    protected void parseLocator(MediaLocator locator) {
        String rem = locator.getRemainder();
        // Strip off starting slashes
        while (rem.startsWith("/") && rem.length() > 1)
            rem = rem.substring(1);
        StringTokenizer st = new StringTokenizer(rem, "/");
        if (st.hasMoreTokens()) {
            // Parse the position
            String position = st.nextToken();
            StringTokenizer nums = new StringTokenizer(position, ",");
            String stX = nums.nextToken();
            String stY = nums.nextToken();
            String stW = nums.nextToken();
            String stH = nums.nextToken();
            x = Integer.parseInt(stX);
            y = Integer.parseInt(stY);
            width = Integer.parseInt(stW);
            height = Integer.parseInt(stH);
        }
        if (st.hasMoreTokens()) {
            // Parse the frame rate
            String stFPS = st.nextToken();
            frameRate = (Double.valueOf(stFPS)).floatValue();
        }
    }

    /**************************************************************************
     * SourceStream
     *************************************************************************/

    public ContentDescriptor getContentDescriptor() {
        return cd;
    }

    public long getContentLength() {
        return LENGTH_UNKNOWN;
    }
```

```
public boolean endOfStream() {
        return false;
}

/**************************************************************************
 * PushBufferStream
 *************************************************************************/

int seqNo = 0;

public Format getFormat() {
        return rgbFormat;
}

public void read(Buffer buffer) throws IOException {
        synchronized (this) {
           Object outdata = buffer.getData();
           if (outdata == null || !(outdata.getClass() == Format.intArray) ||
                   ((int[])outdata).length < maxDataLength) {
                   outdata = new int[maxDataLength];
                   buffer.setData(outdata);
           }
           buffer.setFormat( rgbFormat );
           buffer.setTimeStamp( (long) (seqNo * (1000 / frameRate) * 1000000) );
           BufferedImage bi = robot.createScreenCapture(
                   new Rectangle(x, y, width, height));
           bi.getRGB(0, 0, width, height,
                        (int[])outdata, 0, width);
           buffer.setSequenceNumber( seqNo );
           buffer.setLength(maxDataLength);
           buffer.setFlags(Buffer.FLAG_KEY_FRAME);
           buffer.setHeader( null );
           seqNo++;
        }
}

public void setTransferHandler(BufferTransferHandler transferHandler) {
        synchronized (this) {
           this.transferHandler = transferHandler;
```

```
            notifyAll();
        }
}

void start(boolean started) {
        synchronized ( this ) {
            this.started = started;
            if (started && !thread.isAlive()) {
                    thread = new Thread(this);
                    thread.start();
            }
            notifyAll();
        }
}

/*****************************************************************************
 * Runnable
 *****************************************************************************/

public void run() {
        while (started) {
            synchronized (this) {
                    while (transferHandler == null && started) {
                        try {
                                wait(1000);
                        } catch (InterruptedException ie) {
                        }
                    } // while
            }

            if (started && transferHandler != null) {
                    transferHandler.transferData(this);
                    try {
                        Thread.currentThread().sleep( 10 );
                    } catch (InterruptedException ise) {
                    }
            }
        } // while (started)
} // run
```

```
    // Controls

    public Object [] getControls() {
            return controls;
    }

    public Object getControl(String controlType) {
      try {
        Class  cls = Class.forName(controlType);
        Object cs[] = getControls();
        for (int i = 0; i < cs.length; i++) {
          if (cls.isInstance(cs[i]))
            return cs[i];
        }
        return null;

      } catch (Exception e) {   // no such controlType or such control
       return null;
      }
    }
}
```

Βιβλιογραφία- Πηγές

Brooks, F. Augmented Reality, http://www.cs.unc.edu/Research/graphics

Feiner, S. KARMA, http://www.cs.columbia.edu/graphics/projects/karma.

Feiner, S., T. Webster, Architectural Anatomy, http://www1.cs.columbia.edu/graphics/

Kanade, T. Z-Key: A New Method for Creating Virtual Reality, http://www.cs.cmu.edu/afs/cs/project/stereo-machine/www/z-key.html

Milgram, P. (1995). Augmented Reality, http://vered.rose.utoronto.ca/people/anu_dir/papers/atc/atcDND.html.

T. Azuma, R., A Survey of Augmented Reality

J. Vallino PHD thesis, Interactive Augmented Reality

J. Haniff, D., Baber, C., H. Edmonson, W., Categorizing Augmented Reality Systems

Milgram, P., Drascic, D., Perceptual Effects in Aligning Virtual and Real Objects in Augmented Reality Displays

Rauterberg, New directions in User-System Interaction: augmented reality, ubiquitous and mobile computing

Piekarski., H. Thomas, B., Unifying Augmented Reality and Virtual Reality User Interfaces

Milgram, P., Takemura, H., Utsumi, A., Kishino, F., Augmented Reality: A Class

of Displays on the Reality-Virtuality Continuum

Milgram, P., Drascic, D., Perceptual Issues in Augmented Reality

ETC-Lab Augmented and Virtual Reality Home Page
http://vered.rose.utoronto.ca/etc-lab.html

Registration Errors in Augmented Reality
http://www.cs.unc.edu/~azuma/azuma_AR.html

The Augmented Reality Homepage
http://www.augmented-reality.org

ARVIKA:Augmented Reality or Development, Production and Servicing
http://www.arvika.de/www/index.htm

W. Piekarski, PHD thesis “Interactive 3d Modelling in Outdoor Augmented Reality Worlds”

Breen, D. E., E. Rose, et al. . Interactive Occlusion and Collision of Real and Virtual Objects in Augmented Reality