Τ.Ε.Ι Δυτικής Ελλάδας
Τμήμα Μηχανικών Πληροφορικής
Τ.Ε.Σ.Υ.Δ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

3D Virtual Environments

(Τρισδιάστατα Εικονικά Περιβάλλοντα)

ΝΙΚΟΛΑΣ ΘΩΜΑΣ

Α.Μ: 178

ΕΠΙΒΛΕΠΩΝ**: Βασίλειος Δ. Τριανταφύλλου**

Απρίλης 2014

Περίληψη

Η ανάπτυξη των Τεχνολογιών Πληροφορικής και Επικοινωνιών έδωσε ώθηση και νέες πτυχές σε κάθε επιστημονική, οικονομική, πολιτική και κοινωνική δραστηριότητα. Μια από τις κυριότερες πτυχές είναι η επικοινωνία και η συνεργατικότητα μεταξύ των εμπλεκόμενων μερών κάθε τέτοιας δραστηριότητας, αυξάνοντας έτσι και την ανάγκη για ανταλλαγή απόψεων και λήψη αποφάσεων. Τα συστήματα ηλεκτρονικής συμμετοχής, τα οποία βασίζονται στις έννοιες του **διαλόγου** και της **διαβούλευσης**, αποτελούν μια από τις σημαντικότερες λύσεις για την γεφύρωση του χάσματος μεταξύ του πολίτη και του Κράτους, των συμμετεχόντων μεταξύ τους και γενικότερα ανάμεσα σε κάθε άλλο φυσικό ή νομικό φορέα. Στην παρούσα εργασία, μελετώνται η έννοια και τα χαρακτηριστικά του διαλόγου (debate), οι τεχνολογίες Εικονικής Πραγματικότητας (Virtual Reality), τεχνολογικά εργαλεία που είναι ήδη σε χρήση και τέλος σχεδιάζεται και υλοποιείται ένα εικονικό περιβάλλον, που υποστηρίζει εικονικές διαβουλεύσεις (virtual debates).

Λέξεις – Κλειδιά: Εικονική, Πραγματικότητα, Χώρος, Διαβούλευση, Κατανεμημένο, Περιβάλλον, Ομαδοσυνεργατικό.

Abstract

The development of ICT has given impetus and new aspects in any scientific, economic, political and social activity. One of the main aspects is communication and cooperativity between the parties of any such activity, thereby increasing the need for discussion and decision making. The electronic participation, based on the concepts of dialogue and consultation, are one of the most important solutions for bridging the gap between citizens and the State, between the participants and generally among any other natural or legal entity. In this paper, studied the concept and characteristics of the dialogue (debate), the technologies of Virtual Reality, technological tools are already in use and then designed and implemented a virtual environment that supports virtual consultations (virtual debates).

Keywords: Virtual, Reality, 3D, Debate, Distributed, Collaborative, Second, Life.

Εισαγωγή

Η ανάπτυξη των Τεχνολογιών Πληροφορικής και Επικοινωνιών έδωσε ώθηση και νέες πτυχές σε κάθε επιστημονική, οικονομική, πολιτική και κοινωνική δραστηριότητα. Μια από τις κυριότερες πτυχές είναι η επικοινωνία και η συνεργατικότητα μεταξύ των εμπλεκόμενων μερών κάθε τέτοιας δραστηριότητας, αυξάνοντας έτσι και την ανάγκη για ανταλλαγή απόψεων και λήψη αποφάσεων. Τα συστήματα ηλεκτρονικής συμμετοχής, τα οποία βασίζονται στις έννοιες του διαλόγου και της διαβούλευσης, αποτελούν μια από τις σημαντικότερες λύσεις για την γεφύρωση του χάσματος μεταξύ του πολίτη και του Κράτους, των συμμετεχόντων μεταξύ τους και γενικότερα ανάμεσα σε κάθε άλλο φυσικό ή νομικό φορέα.

Στην παρούσα εργασία, μελετώνται η έννοια και τα χαρακτηριστικά του διαλόγου (debate), οι τεχνολογίες Εικονικής Πραγματικότητας (Virtual Reality), τεχνολογικά εργαλεία που είναι ήδη σε χρήση και τέλος σχεδιάζεται και υλοποιείται ένα εικονικό περιβάλλον, που υποστηρίζει εικονικές διαβουλεύσεις (virtual debates).

Στο πρώτο κεφάλαιο, αναλύεται η έννοια της διαβούλευσης (debate) σαν ένας μηχανισμός ανταλλαγής απόψεων διαφορετικός από τον διάλογο, μιας που χαρακτηρίζεται από μια πιο δομημένη και αναλυτική μορφή και στοχεύει στην ορθολογική και κριτική αντιπαράθεση όλων των διαφορετικών απόψεων, με στόχο την λήψη αποφάσεων. Παρουσιάζονται απόψεις που εξηγούν τον τρόπο με τον οποίο η διαβούλευση οδηγεί στην διαμόρφωση της κοινής γνώμης και προσδιορίζονται βασικοί κανόνες.

Στο δεύτερο κεφάλαιο, γίνεται μια ανασκόπηση πάνω στην ερευνητική και παραγωγική δραστηριότητα που σχετίζεται με την ανάπτυξη συστημάτων λογισμικού που προσομοιώνουν ή και υποστηρίζουν εικονικές (virtual) διαδικασίες διαβούλευσης.

Στο τρίτο κεφάλαιο, στο τρίτο κεφάλαιο, επιχειρείται μια ανάλυση, από τεχνικής, άποψης των βασικών χαρακτηριστικών τέτοιων συστημάτων λογισμικού. Βασίζονται στην τεχνολογία της εικονικής πραγματικότητας και κατηγοριοποιούνται σε κλάσεις

συστημάτων. Επίσης, γίνεται και μια συνοπτική παρουσίαση των πιο σημαντικών από άποψη δημοτικότητας ή τεχνολογικής καινοτομίας.

Στο τέταρτο κεφάλαιο, επιλέγεται μια συγκεκριμένη πλατφόρμα ανάπτυξης ενός τέτοιου συστήματος και παρουσιάζεται εκτενώς. Σε αυτήν την πλατφόρμα θα στηριχθεί η υλοποίηση που θα πραγματοποιηθεί στα πλαίσια αυτής της εργασίας, και που θα παρουσιαστεί αναλυτικά στα επόμενα κεφάλαια.

Στο πέμπτο κεφάλαιο, παρουσιάζεται βήμα-προς-βήμα ο σχεδιασμός και η ανάπτυξη ενός δικού μας συστήματος εικονικής διαβούλευσης, από την ανάλυση απαιτήσεων έως και την τελική υλοποίηση.

Στο έκτο κεφάλαιο, παρουσιάζεται η λειτουργία και τα χαρακτηριστικά του συστήματος που αναπτύχθηκε.

Πίνακας Περιεχομένων

Ευρετήριο εικόνων

Κεφάλαιο 1 Διαβουλεύσεις και Διαμόρφωση της Κοινής Γνώμης

Στο κεφάλαιο αυτό, αναλύεται η έννοια της διαβούλευσης (debate) σαν ένας μηχανισμός ανταλλαγής απόψεων διαφορετικός από τον διάλογο, μιας που χαρακτηρίζεται από μια πιο δομημένη και αναλυτική μορφή και στοχεύει στην ορθολογική και κριτική αντιπαράθεση όλων των διαφορετικών απόψεων, με στόχο την λήψη αποφάσεων. Παρουσιάζονται απόψεις που εξηγούν τον τρόπο με τον οποίο η διαβούλευση οδηγεί στην διαμόρφωση της κοινής γνώμης και προσδιορίζονται βασικοί κανόνες.

1.1 Διάλογος και Διαβούλευση

Οι έννοιες του διαλόγου και της διαβούλευσης συχνά ταυτίζονται αλλά μια εμβριθής ανάλυση των δύο παραπάνω εννοιών οδηγεί στη διάκρισή τους, η οποία συνίσταται στη φύση των παραγόμενων αποτελεσμάτων. Ο διάλογος είναι μία συνομιλία ανταλλαγής εμπειριών, απόψεων και ιδεών μεταξύ δύο ή περισσοτέρων ανθρώπων, η οποία στοχεύει στην κατανόηση και την μάθηση. Αντίθετα, η διαβούλευση αποτελεί μια περισσότερο **δομημένη** και **αναλυτική** διαδικασία, καθώς στοχεύει στην ορθολογική και κριτική αντιπαράθεση όλων των διαφορετικών απόψεων, με στόχο την **λήψη αποφάσεων** (*Χριστοπούλου, 2011*).

Οι δύο έννοιες όμως συνυπάρχουν, καθώς ο διάλογος ενδυναμώνει την διαδικασία της διαβούλευσης, καλλιεργώντας ένα γόνιμο έδαφος για μία αποτελεσματική διαδικασία βασισμένη στην εμπιστοσύνη, την αμοιβαία κατανόηση και στην ισοτιμία και η διαβούλευση δεν μπορεί να πραγματοποιηθεί χωρίς την συνδρομή ενός γνήσιου διαλόγου.

Στη διεθνή βιβλιογραφία (*Dryzek, 1990*), η διαβούλευση συναντάται με τους όρους deliberation και consultation. Με τη λέξη «consultation», νοείται η διαδικασία εκείνη, η οποία στοχεύει στη λήψη αποφάσεων. Παρά το γεγονός ότι οι δύο αυτές έννοιες διαπλέκονται στενά, μια ετυμολογική ανάλυση ξεκαθαρίζει το τοπίο. Το ρήμα συμβουλεύομαι πάντα συντάσσεται με αντικείμενο, δηλώνοντας μια αναγκαία και ασύμμετρη σχέση προς κάποιο άλλο πρόσωπο ή πηγή υπέρτατου κύρους, ενώ

στο ρήμα διαβουλεύομαι η σχέση με τα άλλα πρόσωπα θα δηλωθεί έμμεσα ως εμπρόθετος προσδιορισμός (διαβουλεύομαι με άλλους) ή δεν θα δηλωθεί καθόλου (οι ένορκοι διαβουλεύθηκαν).

Στα αγγλικά, ο όρος «deliberation» (σε αντίθεση προς το «consultation») δεν συνηθίζεται στον πληθυντικό, δηλώνοντας ότι δεν πρόκειται τόσο για μια διαπροσωπική διαδικασία όσο για μια προσωπική κατάσταση, ενώ μια πιθανολογούμενη ετυμολόγηση του όρου «deliberate», αναφέρεται στο λατινικό «libra» (λόγος), παραπέμποντας στη σημασία της κρίσης και στάθμευσης των πραγμάτων. Ωστόσο, και στις δύο περιπτώσεις, η αναγωγή στην προσωπικότητα του υποκειμένου είναι ταυτόχρονα και μια αναγωγή στην ελευθερία της σκέψης και την ισοτιμία της σχέσης με άλλα υποκείμενα.

Η διαβούλευση θεμελιώνεται στην παραδοχή, ότι οι ιδιώτες αποτελούν ελεύθερους και ίσους συμμετέχοντες που συμμετέχουν ενεργά στον ορισμό και την πραγμάτωση του κοινού αγαθού. Η δημόσια συμμετοχή αναδεικνύεται σε λυδία λίθο της διαβούλευσης, καθώς συναρτάται άμεσα με τη συμμετοχή των πολιτικά ισότιμων ατόμων σε έναν δημόσιο διάλογο για την αύξηση της ποιότητας της λήψης αποφάσεων, τη συμμετοχή δηλαδή, του πολίτη στη διαδικασία του άρχειν σε όλα τα στάδια της πολιτικής εξουσίας και την αύξηση της επιρροής των συμμετεχόντων σε πολιτικές που δημιουργούνται για αυτούς και επηρεάζονται από αυτούς.

Σύμφωνα με έναν πιο εργαλειακό ορισμό, η διαβούλευση είναι μια προσέγγιση στη λήψη αποφάσεων, στην οποία οι συμμετέχοντες εξετάζουν τα σχετικά γεγονότα από πολλαπλές οπτικές γωνίες και συζητούν μεταξύ τους για να σταθμίσουν τις επιλογές τους. Η συμμετοχή των συμμετεχόντων, οι οποίοι αποβάλλουν τις ατομοκεντρικές τους προτιμήσεις και συμμετέχουν στη διαδικασία λήψης δημόσιων αποφάσεων, με σκοπό την εξυπηρέτηση του δημοσίου συμφέροντος, ανάγεται στην πεμπτουσία της διαβούλευσης.

Κοινό σημείο και στους δύο ορισμούς περί διαβούλευσης είναι η παραδοχή, ότι ο ευρύτερος δημόσιος διάλογος που αναπτύσσεται και η συνακόλουθη επιχειρηματολογία βρίσκονται στον πυρήνα της λήψης πολιτικών αποφάσεων. Μέσω της διαβούλευσης η θέληση και οι προτιμήσεις των συμμετεχόντων δεν προηγούνται των διαδικασιών λήψης αποφάσεων αλλά είναι προϊόν τους.

1.2 Διαμόρφωση κοινής γνώμης μέσω διαβουλεύσεων

Όπως προαναφέρθηκε, μια διαβούλευση (debate) είναι, πάνω από όλα, μια διαδικασία συζήτησης, που αποκαλύπτει την θέση των εμπλεκόμενων μερών για την απόφαση που πρέπει να ληφθεί πάνω στο θέμα που διαπραγματεύεται. Συνεπώς, είναι ένα εργαλείο το οποίο παράγει αποφάσεις που λαμβάνουν υπόψη τις διαφορετικές απόψεις και θέσεις μέσα στο αντιπαρατιθέμενο σύνολο.

Από μια νομική αλλά και τεχνική σκοπιά, η απόφαση που βγαίνει μέσω μιας διαδικασίας διαβούλευσης είναι περισσότερο πιθανό να αντιπροσωπεύει το λεγόμενο «κοινό συμφέρον». Έτσι, το κοινό αντιλαμβάνεται σαν πιο λογικές και εύλογες τις αποφάσεις, τις αποδέχεται και εργάζεται για την εφαρμογή τους. Αυτό άλλωστε είναι και μια από τις βασικές αρχές τις δημοκρατίας.

1.3 Μορφές και κανόνες

Υπάρχουν διάφορες μορφές διαβούλευσης, με παραλλαγές όσον αφορά το στυλ, την οργάνωση και τους κανόνες διεξαγωγής. Σε αυτήν την ενότητα παρουσιάζονται οι πιο διαδεδομένες.

1.3.1 Ανταγωνιστική Διαβούλευση (Competitive Debate)

Στην ανταγωνιστική διαβούλευση, οι συμμετέχοντες χωρίζονται σε ομάδες. Πριν την έναρξη της διαβούλευσης, ορίζεται ένα σύνολο κριτηρίων βάσει των οποίων κρίνεται η επικρατούσα ομάδα (και συνεπώς η επικρατούσα άποψη). Χρησιμοποιείται ιδιαίτερα στις περιπτώσεις όπου το ζητούμενο είναι η εκπαίδευση ατόμων που πρόκειται να απασχοληθούν σε θέσεις όπου το αντικείμενο είναι η διαβούλευση που στοχεύει στην επίλυση θεμάτων.

Η ανταγωνιστική διαβούλευση μπορεί να λαμβάνει χώρα τόσο σε τοπικό, όσο και σε εθνικό και διεθνές επίπεδο. Σε σχολεία και ακαδημαϊκά ιδρύματα, συνήθως παίρνει την μορφή διαγωνισμού, με ρητούς κανόνες. Μπορεί να πραγματοποιείται υπό την κρίση ενός ή περισσοτέρων κριτών. Κάθε διαβουλευόμενη πλευρά επιδιώκει την επικράτηση έναντι των υπολοίπων, στα πλαίσια που ορίζουν οι κανόνες. Κάθε πλευρά μπορεί να είναι υπέρ ή κατά μιας πρότασης που τίθεται στην αρχή της διαβούλευσης. Η υποστηρίζουσα πλευρά επιχειρηματολογεί υπέρ της πρότασης και η άλλη προσπαθεί να αποδομήσει τα επιχειρήματα της πρώτης, χωρίς να είναι υποχρεωμένη να προτείνει και να υποστηρίξει μια εναλλακτική πρόταση.

1.3.2 Κοινοβουλευτική Διαβούλευση (Parliamentary debate)

Η κοινοβουλευτική διαβούλευση χρησιμοποιεί τους κανονισμούς που έλκουν την καταγωγή τους από τις διαδικασίες που ακολουθούνται στο Βρετανικό Κοινοβούλιο. Περιλαμβάνει πολλούς συμμετέχοντες, που τοποθετούνται μεμονωμένα. Η πλευρά που υποστηρίζει την πρόταση που είναι το αντικείμενο της διαβούλευσης συνήθως αποκαλείται «κυβέρνηση» ή «πρόταση». Η άλλη πλευρά συνήθως αποκαλείται «αντιπολίτευση».

Αυτή η μορφή διαβούλευσης είναι μια από τις πιο διαδεδομένες παγκοσμίως. Για αυτό το λόγο και αποκαλείται απλά «διαβούλευση». Ιδιαίτερα χρησιμοποιείται στην δημόσια ζωή στο Ηνωμένο Βασίλειο, στην Ινδία και στην Ελλάδα, εκτός των άλλων. Αυτούς τους κανόνες υιοθετεί και το Παγκόσμιο Πανεπιστημιακό Πρωτάθλημα Διαβούλευσης (World Universities Debating Championship).

1.3.3 Δημόσια Διαβούλευση (Public Debate)

Η δημόσια διαβούλευση εισήχθη από το International Public Debate Association (IPDA), που ιδρύθηκε στις ΗΠΑ και διεξάγει σχετικούς αγώνες διαβούλευσης από το 1997. Αυτή η μορφή διαβούλευσης (format) δίνει έμφαση στην δημόσια τοποθέτηση και πειθώ πραγματικών ακροατών. Εκτός από την αποδεικτική επιχειρηματολογία, σημασία έχει και η ταχύτητα συγκρότησης και έκφρασης των επιχειρημάτων.

Στο πλαίσιο αυτό, χρησιμοποιούνται πολλαπλοί κριτές που έχουν τον ρόλο του ζωντανού κοινού της διαβούλευσης. Στα πρωταθλήματα που διοργανώνει το IPDA, προβλέπεται επίσης επιπλέον βραβείο, στους πιο πειστικούς συνομιλητές. Επίσης, προβλέπεται τόσο ο χωρισμός των συμμετεχόντων σε ομάδες, όσο και διαβούλευση μεταξύ μεμονωμένων ομιλητών.

Σε όλες τις περιπτώσεις, δίνεται λίστα με θέματα στους συμμετέχοντες, 30 λεπτά της ώρας πριν ξεκινήσει η διαβούλευση. Στην συνέχεια, ξεκινάει διαπραγμάτευση σχετικά με το πιο θέμα θα συζητηθεί τελικά. Αφού επιλυθεί αυτό, κάθε πλευρά κάνει μια εισηγητική τοποθέτηση, υπέρ ή κατά του θέματος, ακολουθεί αντιπαράθεση της μιας με την άλλη πλευρά και ολοκληρώνεται με το κλείσιμο. Αυτή η μορφή συναντάται συχνά σε κολλέγια και Πανεπιστήμια στις ΗΠΑ.

1.3.4 Κλασσική Διαβούλευση (Classic Debate)

Η κλασσική διαβούλευση, παρά την ονομασία της, είναι μια από τις πιο πρόσφατα εισαχθείσες μορφές, που συστάθηκε στο Πανεπιστήμιο της Μινεσότα, των ΗΠΑ. Εστιάζει στην κατάθεση λογικών επιχειρημάτων και στην πειθώ πραγματικού κοινού. Ξεκινάει με την από κοινού επιλογή ενός θέματος, στο οποίο η μία πλευρά συμφωνεί και η άλλη διαφωνεί. Στην συνέχεια, πραγματοποιείται ανταλλαγή επιχειρημάτων σε γύρους τοποθέτησης.

1.3.5 Αυθόρμητη Διαβούλευση (Extemporaneous debate)

Η μορφή αυτή δεν προβλέπει προετοιμασία και συμπεριλαμβάνει δύο ομάδες, που καθεμία διαθέτει πρώτο και δεύτερο ομιλητή. Επιτρέπεται συνήθως η επίκληση σε τρέχοντα γεγονότα, σχετικά με το θέμα, ή σε στατιστικά στοιχεία και αρθρογραφία. Αυτά τα στοιχεία όμως η άλλη πλευρά έχει το δικαίωμα να τα αμφισβητήσει, ως αναξιόπιστα, ενώ σε κάποιες παραλλαγές επιτρέπεται η χρήση μόνο στοιχείων από μια λίστα πηγών που καθορίζεται ελάχιστα πριν την έναρξη της διαβούλευσης.

Ξεκινάει με την τοποθέτηση του πρώτου ομιλητή της ομάδας που υποστηρίζει την θέση του θέματος. Ακολουθεί η τοποθέτηση του πρώτου ομιλητή της άλλης ομάδας. Στην συνέχεια, το ίδιο γίνεται και με τους δεύτερους ομιλητές. Κάθε τοποθέτηση διαρκεί έξι λεπτά της ώρας, και δύο λεπτά αντιπαράθεσης.

Ακολουθούν δευτερολογίες, με την ίδια σειρά, διάρκειας τεσσάρων λεπτών της ώρας. Στις δευτερολογίες δεν επιτρέπονται παρεμβάσεις και αντιπαράθεση. Νικήτρια είναι η ομάδα (και συνεπώς, άποψη) που καταφέρνει να αποδομήσει όλα τα επιχειρήματα της άλλης ενώ τα δικά της θα είναι «άθικτα».

Κεφάλαιο 2 Βιβλιογραφική Ανασκόπηση

Στο κεφάλαιο αυτό παρουσιάζεται το ερευνητικό και εφαρμοσμένο έργο που έχει παραχθεί πάνω στα συστήματα εικονικών διαβουλεύσεων. Γίνεται μια ανασκόπηση πάνω στην ερευνητική και παραγωγική δραστηριότητα που σχετίζεται με την ανάπτυξη συστημάτων λογισμικού που προσομοιώνουν ή και υποστηρίζουν εικονικές (virtual) διαδικασίες διαβούλευσης.

2.1 Αποδοτικές Στρατηγικές για Συστήματα Διαβούλευσης

Στο (*Yuan et al, 2004*), μελετώνται στρατηγικές διαλόγων κατάλληλες για υλοποίηση από συστήματα λογισμικού που υποστηρίζουν διαβουλεύσεις (computer debating systems). Συγκεκριμένα, προτείνεται ένα σύνολο ευρετικών τεχνικών που μπορούν να καταστήσουν ένα αυτόματο σύστημα σαν έναν εικονικό συνομιλητή. Επίσης, ερευνώνται μέθοδοι υλοποίησης των προτεινόμενων στρατηγικών (υπολογιστικοί πράκτορες - computational agents) καθώς και μέθοδοι αξιολόγησης των παραγόμενων διαλόγων.

Στα κλασσικά διαλογικά συστήματα, οι κανόνες του διαλόγου συνήθως αφήνουν έναν βαθμό ελευθερίας όσον αφορά την κίνηση των συμμετεχόντων και το περιεχόμενο. Συνεπώς, είναι σημαντικό για ένα ψηφιακό διαλογικό σύστημα να επιτρέπει στους , πραγματικούς και εικονικούς, αυτόν τον βαθμό ελευθερίας.

2.2 Αναπτύσσοντας «έξυπνους» εικονικούς συνομιλητές

Στο (Cassell, Vilhjálmsson*,1999*), το ενδιαφέρον επικεντρώνεται στους ψηφιακούς συνομιλητές (avatars). Αν και η ανάπτυξη των συστημάτων Εικονικής Πραγματικότητας έχει βοηθήσει ώστε οπτικά να αποτελούν αξιόλογες προσομοιώσεις πραγματικών συνομιλητών, εντούτοις μικρή πρόοδος έχει επιτευχθεί στο να αποκτήσουν πραγματική αυτονομία μέσα στα όρια των ψηφιακών διαλογικών συστημάτων. Έτσι, σε αντίθεση με τους ανθρώπους, ελάχιστα χρησιμοποιούν τα μη-λεκτικά, εκφραστικά μέσα.

Η μοντελοποίηση και η σχεδιοκίνηση μιας τέτοιας, σύνθετης, συμπεριφοράς, είναι καθοριστική για την πιστότητα και την ποιότητα της εικονικής αλληλεπίδρασης. Προτείνεται μια μέθοδος αυτοματοποίησης της σχεδιοκίνησης σημαντικών στοιχείων επικοινωνίας που σχετίζονται με μη-λεκτικά εκφραστικά μέσα, όπως η κίνηση του σώματος και οι εκφράσεις του προσώπου. Αποτέλεσμα της συγκεκριμένης έρευνας είναι το σύστημα **BodyChat**, που επιτρέπει στους χρήστες να επικοινωνούν με μηνύματα κειμένου ενώ ταυτόχρονα τα avatars πραγματοποιούν σύνθετες κινήσεις, όπως χαιρετισμούς, στάση σώματος, και εκφράσεις προσώπου.

Η αξιολόγηση της μεθόδου έδειξε πως τα «έξυπνα» avatars έχουν συμπεριφορά πιο κοντινή στην πραγματική ανθρώπινη συμπεριφορά, σε σχέση με τα κλασσικά, απόλυτα ελεγχόμενα avatars, αυξάνοντας έτσι την ποιότητα της εικονικής επικοινωνίας. Επίσης, οι χρήστες του συστήματος εξέφρασαν την πεποίθηση πως τα έξυπνα avatars τους έδιναν ισχυρότερη αίσθηση ελέγχου.

2.3 Συνεργατική Λήψη Αποφάσεων

Στο (*Karakapilidis, Papadias, 2001*), μελετώνται προβλήματα συνεργατικής λήψης αποφάσεων και προτείνονται λύσεις επαυξητικής συνομιλίας και συνεργασίας. Η συναίνεση μεταξύ των χρηστών επιτυγχάνεται μέσω της διαδικασίας μέσω του συνυπολογισμού διαφορετικών όψεων του υπό συζήτηση θέματος, αντικρουόμενων συμφερόντων, περιορισμών και άλλων.

Αποτέλεσμα αυτής της έρευνας είναι το σύστημα **HERMES**, που επαυξάνει τις κλασσικές προσεγγίσεις στην λήψη αποφάσεων. Είναι εξ'ολοκλήρου υλοποιημένο στην γλώσσα προγραμματισμού Java και μπορεί να εκτελείται σε διαδικτυακό περιβάλλον και συνεπώς είναι δυνατή η οικονομική και ευρεία χρήση του από το κοινό.

2.4 Convinceme.net – Διαδικτυακό σύστημα διαβούλευσης

Ο δικτυακός κόμβος ConvinceMe.net είναι ένα χαρακτηριστικό παράδειγμα online συστήματος δημόσιας διαβούλευσης. Κάθε χρήστης του Διαδικτύου έχει δικαίωμα να εγγραφεί και να συμμετάσχει στις συζητήσεις που φιλοξενεί το συγκεκριμένο

σύστημα. Οι σχεδιαστές του ConvinceMe.net ορίζουν τρείς κύριες κατηγορίες διαβουλεύσεων, που φέρουν τις χαρακτηριστικές ονομασίες **Open**, **Battle** και **King of the Hill**.

Η κατηγορία «Open» περιλαμβάνει debates περιλαμβάνει διαβουλεύσεις μεταξύ πολλών συμμετεχόντων, που κάθε ένας υποστηρίζει την άποψη του πάνω στο συγκεκριμένο θέμα, χωρίς προκαθορισμένο τέλος.

Στην κατηγορία «Battle», συμμετέχουν δύο χρήστες που διαβουλεύονται ένα προκαθορισμένο θέμα για ένα προκαθορισμένο χρονικό διάστημα. Οι υπόλοιποι χρήστες ψηφίζουν για έναν από τους δύο συμμετέχοντες και έτσι αναδεικνύεται κάποιος «νικητής».

Τέλος, στην κατηγορία «King of the hill», ο συμμετέχων εκφράζει ένα κεντρικό επιχείρημα και στην συνέχεια προσπαθεί να πείσει τους υπόλοιπους συμμετέχοντες να του δώσουν πόντους, στοχεύοντας στους 10, που θα τον αναδείξουν «νικητή» της διαβούλευσης.

Τα χαρακτηριστικά του ConvinceMe.net θα μπορούσαν να συνοψιστούν στα παρακάτω σημεία:

- Παρέχεται η δυνατότητα παράθεσης παραπομπών σε άλλες ιστοσελίδες και πηγές.
- Υποστηρίζεται η κατάρτιση «ιδωτικών» debates για ομάδες χρηστών.
- Υποστηρίζονται RSS feeds.
- Καταρτίζονται rankings σε μηνιαία βάση.
- Παρέχονται χαρακτηριστικά Web 2.0 (widgets, social networking, δημιουργία προσωπικού προφίλ, blogging κλπ).
- Υποστηρίζονται ελεύθερες συζητήσεις σε ειδικό χώρο (forum).

Οι βασικοί περιορισμοί του ConvinceMe.net θα μπορούσαν να συνοψιστούν στα παρακάτω σημεία:

- Δεν επιτρέπονται οι προσωπικές επιθέσεις.

- Δεν επιτρέπονται ρατσιστικά σχόλια.
- Δεν επιτρέπονται άμεσες ή υπονοούμενες απειλές κατά χρηστών του μέσου.
- Κάθε παράβαση οδηγεί σε διαγραφή του λογαριασμού του χρήστη.

Κεφάλαιο 3 Εικονική Πραγματικότητα

Στο κεφάλαιο αυτό, επιχειρείται μια ανάλυση, από τεχνικής, άποψης των βασικών χαρακτηριστικών τέτοιων συστημάτων λογισμικού. Βασίζονται στην τεχνολογία της εικονικής πραγματικότητας και κατηγοριοποιούνται σε κλάσεις συστημάτων. Επίσης, γίνεται και μια συνοπτική παρουσίαση των πιο σημαντικών από άποψη δημοτικότητας ή τεχνολογικής καινοτομίας.

3.1 Ορισμός

Η Εικονική Πραγματικότητα, είναι ένα πεδίο έρευνας που σκοπό έχει να δημιουργήσει ένα σύστημα, που παρέχει μια συνθετική εμπειρία στους χρήστες της. Η εμπειρία ονομάζεται «εικονική» (επίσης αναφέρεται και ως «συνθετική» ή «απατηλή») επειδή η αισθητήρια διέγερση στον χρήστη εξομοιώνεται και παράγεται από το «σύστημα».

Για πρακτικούς λόγους, το σύστημα συνήθως αποτελείται από διάφορους τύπους συσκευών προβολής, για να υπάρξει η διέγερση, αισθητήρες που εντοπίζουν τις κινήσεις του χρήστη, και ένας υπολογιστής που επεξεργάζεται τις κινήσεις αυτές και παράγει την έξοδο της οθόνης.

Εικόνα 1 – Τυπικό Σύστημα Εικονικής Πραγματικότητας.

Για να εξομοιώσουν και να παράγουν εικονικές εμπειρίες, οι προγραμματιστές, συχνά δομούν ένα υπολογιστικό μοντέλο, γνωστό και ως «εικονικός κόσμος» ή «εικονικό περιβάλλον» που είναι χωρικά οργανωμένα υπολογιστικά αντικείμενα (εν συντομία «εικονικά αντικείμενα»), που παρουσιάζονται στον χρήστη μέσω διαφόρων αισθητήριων συσκευών προβολής, όπως οι οθόνες, τα ηχεία και συσκευές ανάδρασης.

Σημαντικό συστατικό ενός επιτυχημένου συστήματος εικονικής πραγματικότητας είναι η παροχή αλληλεπίδρασης, που επιτρέπει στον χρήστη όχι απλά να νιώσει κάποια ερεθίσματα, αλλά επιπλέον, να μπορεί να αλλάξει και να επηρεάσει τον εικονικό κόσμο σε κάποιο βαθμό.

Με την χρήση της τεχνολογίας της εικονικής πραγματικότητας μπορούν να δημιουργηθούν Εικονικά περιβάλλοντα. Εικονικό Περιβάλλον είναι το αποτέλεσμα που παράγει ένα σύστημα Εικονικής Πραγματικότητας. Αποτελεί μια προσομοίωση ενός πραγματικού ή φανταστικού κόσμου που σκοπό έχει την παροχή μιας αυξημένης αίσθησης ρεαλισμού στους χρήστες, μέσω της αναπαράστασης τους από εικονικές οντότητες γνωστές και ως «Avatars».

3.2 Ιστορικά Στοιχεία

Η τεχνολογία της Εικονικής Πραγματικότητας δεν θα μπορούσε να αναπτυχθεί και να αποτελέσει ένα υλοποιήσιμο όραμα, χωρίς την ανάπτυξη των τεχνολογιών Πληροφορικής και Επικοινωνιών (ΤΠΕ). Το 1946 κατασκευάζεται ο πρώτος ηλεκτρονικός υπολογιστής, με την ονομασία ENIAC, από το Πανεπιστήμιο της Πενσυλβάνια, για τον αμερικάνικο στρατό.

Στη δεκαετία του 1950 ο Αμερικανός κινηματογραφιστής Morton Heilig προτείνει "το σινεμά του μέλλοντος", το οποίο θα περικυκλώνει το θεατή με αισθήσεις φτιαγμένες από μηχανήματα και θα μεταφέρει τους θεατές σε μια άλλη διάσταση. Το Sensorama που κατασκευάζεται από τον ίδιο το 1956, προσφέρει μια βόλτα με μοτοσυκλέτα στους δρόμους του Μανχάταν. Χρησιμοποιούνται 3Δ γραφικά, στερεοσκοπικός ήχος και δονητές. Ο χρήστης του μπορεί επίσης να νοιώσει τον αέρα να τον χτυπάει στο πρόσωπο και να μυρίσει αρώματα της πόλης, όπως γιασεμί και

ιβίσκο. Τελικά όμως το Sensorama αποδεικνύεται πολύ επαναστατικό για την εποχή του και αποτυγχάνει.

Το 1961 οι μηχανικοί της εταιρίας Philco Comeau και Bryan δημιουργούν ένα HMD (Head Mounted Display) με την ονομασία Headsight TV Surveillance System απομακρυσμένης παρακολούθησης, με ανίχνευση της κίνησης του κεφαλιού. Για να το επιτύχουν αυτό χρησιμοποιούν ένα ειδικά κατασκευσμένο ηλεκτρομαγνητικό σύστημα. Το HMD αυτό χρησιμοποιήθηκε για την απομακρυσμένη παρακολούθηση επικίνδυνων καταστάσεων.

Το 1963 ο διδακτορικός φοιτητής του MIT Ivan Sutherland εισάγει τα αλληλεπιδραστικά γραφικά μέσω υπολογιστή με την εφαρμογή του Sketchpad. Η συγκεκριμένη εφαρμογή χρησιμοποιεί ένα ελαφρύ στυλό για την επιλογή αντικειμένων, παράλληλα με τη χρήση του πληκτρολογίου. Ο ίδιος το 1965 κάνει τα πρώτα βήματα στο να συνδυάσει τους υπολογιστές και τη δημιουργία Εικονικών Κόσμων με την εργασία του "The ultimate display". Στην εργασία αυτή ουσιαστικά περιγράφει ένα δωμάτιο, όπου τα πάντα ελέγχονται από τον υπολογιστή και όλες οι ενέργειες του χρήστη μέσα σε αυτό έχουν τον ίδιο αντίκτυπο που θα είχαν και στον πραγματικό κόσμο.

Το 1967, ο Fred Brooks επιρεασμένος από την εργασία του Sutherland, ξεκινάει το project GROPE, που έχει σαν στόχο να εξερευνήσει τη χρήση απτικής αλληλεπίδρασης για να βοηθήσει τους βιοχημικούς να "αισθανθούν" τις αλληλεπιδράσεις μεταξύ μορίων πρωτεΐνης. Το 1968, ο Sutherland κατασκευάζει το Sword of Damocles (Σπαθί του Δαμοκλή), ένα HDM το οποίο πήρε το όνομα του από το γεγονός ότι κρεμόταν από το ταβάνι. Χρησιμοποιούσε καθοδικές λυχνίες, είχε μηχανική ανίχνευση της κίνησης του κεφαλιού και πρόβαλλε εικόνες πάνω στον πραγματικό κόσμο.

Το εύρος πεδίου του ήταν 40 μοίρες και ο χρήστης μπορούσε να δει σε πραγματικό χρόνο, αντικείμενα σε wireframe μορφή να προβάλλονται πάνω στον πραγματικό κόσμο. Την ίδια χρονιά ο ίδιος και ο David Evans ιδρύουν την εταιρία Evans and Sutherland Computer Corp. (E&S), η οποία ασχολείται με συστήματα οπτικοποίησης

τα οποία χρησιμοποιούνται στο στρατό, σε εμπορικούς εξομοιωτές καθώς και σε πλανητάρια και αλληλεπιδραστικά θέατρα.

Το 1972, η εταιρία Atari προσφέρει στο ευρύ κοινό αλληλεπιδραστικά γραφικά πραγματικού χρόνου, με το παιχνίδι Pong. Η ίδια εταιρία στη συνέχεια θα συγκεντρώσει στους κόλπους της πολλούς μελλοντικούς πρωτοπόρους της Εικονικής Πραγματικότητας, όπως είναι οι Alan Kay, Fisher, Bricken, Foster, Laurel, Walser, Robinett και Zimmerman.

Το 1974 ο Myron Krueger δημιουργεί τα πρωτοποριακά του έργα, Metaplay και Videoplace, όπου εξερευνά τις δυνατότητες της αλληλεπίδρασης με τη βοήθεια υπολογιστή. Δημιουργούνται έτσι αλληλεπιδραστικά καλλιτεχνικά περιβάλλοντα, σχεδιασμένα με τέτοιο τρόπο ώστε να δίνουν στους χρήστες τους τη δυνατότητα ελευθερίας επιλογής και προσωπικής έκφρασης.

Το 1976 κατασκευάζεται το GROPE II, από τους P. J. Kilpatrick και Fred Brooks, το οποίο παρείχε force feedback (ανάδραση δύναμης) και χρησιμοποιούσε μηχανικούς βραχίονες, για να μεταφερθούν οι κινήσεις των χεριών των χημικών που χρησιμοποιούσαν το σύστημα, στα άτομα φαρμάκων και να μεταβάλλουν τη συμπεριφορά τους.

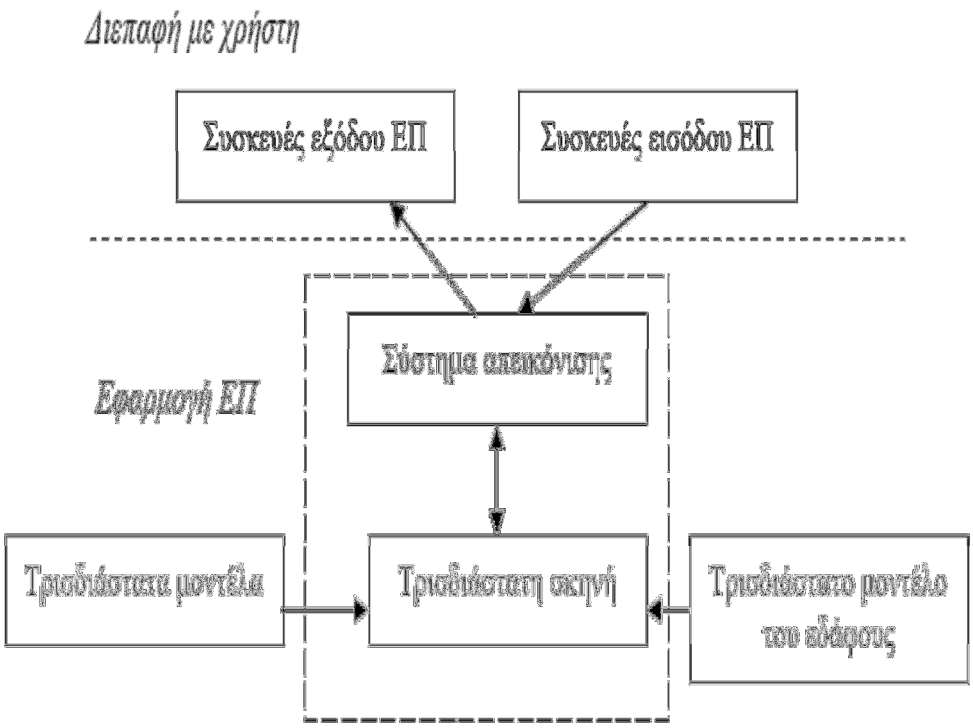
3.3 Χαρακτηριστικά

Η αρχιτεκτονική ενός συστήματος εικονικής πραγματικότητας αποτελείται από συγκεκριμένα συστατικά στοιχεία, που φαίνονται σχηματικά στην εικόνα που ακολουθεί.

- Το σύστημα απεικόνισης (viewer) και τρισδιάστατη σκηνή αποτελούν στοιχεία που συνδέονται στενά μεταξύ τους, αφού η επιλογή του τρισδιάστατου περιβάλλοντος απεικόνισης ως 3D viewer υποδηλώνει μια τρισδιάστατη υλοποίηση του σκηνικού (3D scene). Η τρισδιάστατη σκηνή λαμβάνει συνεισφορές από ένα τρισδιάστατο μοντέλο του εδάφους και τρισδιάστατες απεικονίσεις των αντικειμένων του πραγματικού κόσμου. Και

τα δυο μαζί αποτελούν την τρισδιάστατη μηχανή απεικόνισης (3D player engine).

- Το μοντέλο εδάφους είναι μια γεωγραφική βάση δεδομένων του εδάφους σε τρισδιάστατη μορφή.
- Τρισδιάστατα μοντέλα του πραγματικού κόσμου.
- Συσκευές εισόδου.
- Συσκευές εξόδου ή απεικόνισης.



Εικόνα 2 – Τυπική δομή εφαρμογής εικονικής πραγματικότητας.

Οι χρήστες βλέπουν έναν τρισδιάστατο εικονικό κόσμο στις συσκευές εξόδου εικονικής πραγματικότητας και μπορούν ν' αλληλεπιδράσουν μ' αυτόν μέσω συσκευών εισόδου εικονικής πραγματικότητας. Ένα σύστημα απεικόνισης (viewer) περιέχει μια τρισδιάστατη σκηνή η οποία αποτελείται από τρισδιάστατα μοντέλα και (πιθανώς) από ένα μοντέλο του περιβάλλοντος που καθοδηγεί τις συσκευές εισόδου και εξόδου.

Η τρισδιάστατη σκηνή είναι μια δυναμική δομή δεδομένων η οποία περιέχει όλη την πληροφορία που η εφαρμογή εικονικής πραγματικότητας πρόκειται να δείξει στο χρήστη. Τα τρισδιάστατα μοντέλα περιγράφουν τις κλάσεις των ορατών αντικειμένων της τρισδιάστατης σκηνής. Το μοντέλο του περιβάλλοντος περιγράφει το τοπίο σε τρισδιάστατη μορφή και η μηχανή απεικόνισης το απεικονίζει.

3.4 Συστήματα εικονικής πραγματικότητας (Virtual Reality - VR)

Τα συστήματα Εικονικής Πραγματικότητας έχουν κύριο στόχο την πλήρη ενσωμάτωση του χρήστη στο περιβάλλον που αναπαρίσταται. Για την επίτευξη αυτού του σκοπού χρησιμοποιείται συνδυασμός ειδικών τεχνολογιών όπως στερεοσκοπικές οθόνες προβολής, ανιχνευτών θέσης, 3D γυαλιών, ακουστικής και υπολογιστικού συστήματος. Οι επιλογές και τεχνολογίες χρήσης είναι πολλές.

Για αυτό, σε κάθε σύστημα είναι σημαντική η ουσιαστική μελέτη της χρήσης, των προδιαγραφών και του διαθέσιμου χώρου ώστε να προμηθευτεί ο σωστός εξοπλισμός. Ένα εκπαιδευτικό εικονικό περιβάλλον (Educational Virtual Environment - EVE) είναι μια ειδική περίπτωση ενός συστήματος VR, όπου η έμφαση δίνεται περισσότερο στην εκπαίδευση και τη συνεργασία παρά στην προσομοίωση.

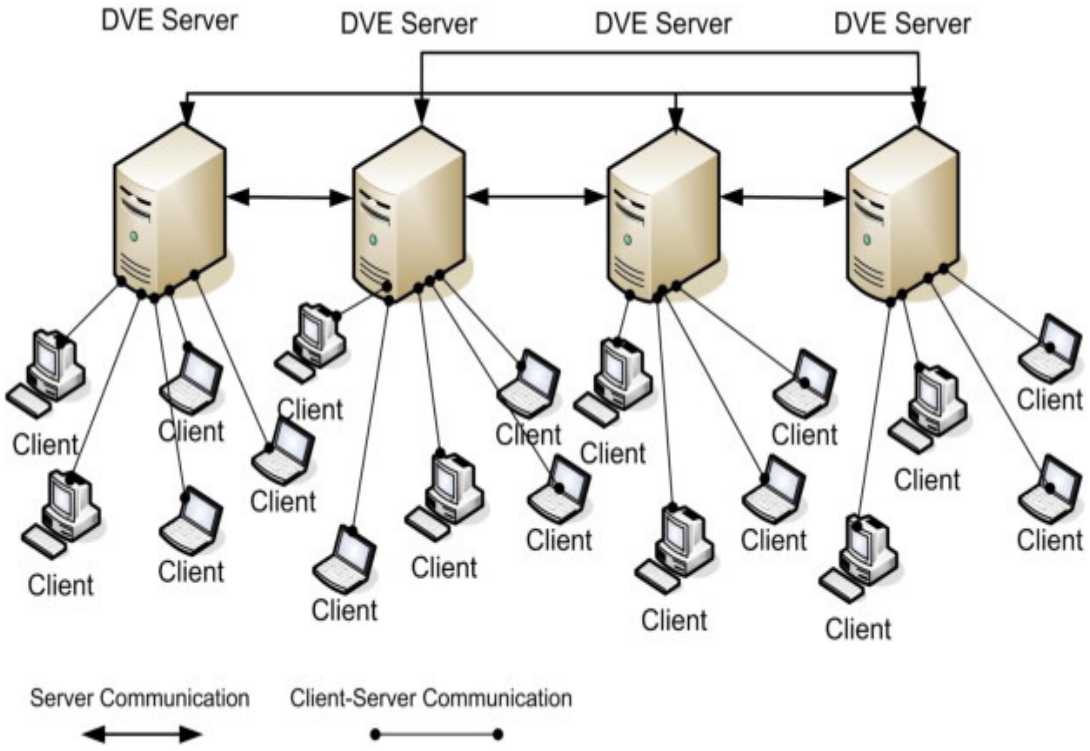
3.5 Κατανεμημένα εικονικά περιβάλλοντα (Distributed Virtual Environments - DVE)

Η ανάπτυξη και εξάπλωση των δικτύων υπολογιστών, έδωσε την δυνατότητα στα εικονικά περιβάλλοντα να προσαρμοστούν και να υποστηρίξουν την χρήση πολλαπλών συστημάτων φιλοξενίας (hosts) και φυσικά την επικοινωνία μεταξύ των στιγμιοτύπων τους πάνω από ένα τέτοιο δίκτυο δεδομένων. Αυτή η κατηγορία συστημάτων ονομάζεται «networked virtual environment» ή, συνηθέστερα, «distributed virtual environment».

Ένα DVE διαφέρει από ένα κεντρικοποιημένο virtual environment (VE) ως προς την δυνατότητα του να διατηρεί την τρέχουσα κατάσταση του (state) διαμέσου δύο ή περισσοτέρων κόμβων, πάνω από ένα δίκτυο δεδομένων. Ένα server-based VE που

πραγματοποιεί κάποιου είδους προσομοίωση, υπολογισμό καταστάσεων και αποθήκευση σε απομακρυσμένους πελάτες (remote clients) θεωρείται επίσης DVE.

Γενικά, ένα DVE μπορεί να θεωρηθεί σαν ένα σύνολο από καταστάσεις και κανόνες για την μετάβαση από μια κατάσταση στην άλλη. Η διαχείριση των καταστάσεων και η απεικόνιση (rendering) στα DVE, είναι μια ιδιαίτερα δύσκολη διαδικασία. Οι χρήστες αλληλεπιδρούν με το DVE σε πραγματικό χρόνο και αναμένουν άμεση ανταπόκριση από το σύστημα.



Εικόνα 3 – Τοπολογία ενός DVE.

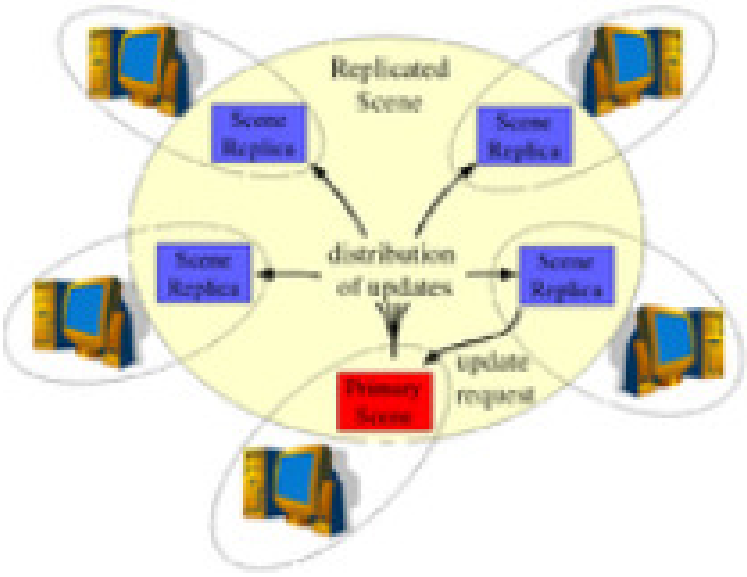
Η καθυστέρηση στο δίκτυο και οι καθυστερήσεις που προκαλούν οι υπολογισμοί είναι σημαντικά εμπόδια στην καλή λειτουργία ενός real-time DVE. Έτσι, τα DVE χρησιμοποιούν σύνθετες στρατηγικές προκειμένου να δώσουν την αίσθηση ενός ενιαίου συστήματος. Τα DVE χρησιμοποιούνται σε ένα μεγάλο εύρος εφαρμογών. Χαρακτηριστικό παράδειγμα, είναι τα στρατιωτικά λογισμικά προσομοίωσης μάχης, ιατρικά λογισμικά, όπως το Therapy World και ψυχαγωγικά, όπως το Second Life.

3.6 Συνεργατικά Εικονικά Περιβάλλοντα (Collaborative Virtual Environments - CVE)

Τα Συνεργατικά Εικονικά Περιβάλλοντα (Collaborative Virtual Environments - CVE), χρησιμοποιούνται για αλληλεπίδραση και ομαδοσυνεργατικότητα (collaboration) μεγάλου πλήθους συμμετεχόντων, κατανεμημένους σε μεγάλες, γεωγραφικές αποστάσεις.

Τυπικά παραδείγματα είναι οι κατανεμημένες προσομοιώσεις, τα 3D παιχνίδια πολλών χρηστών, την συνεργατική ανάπτυξη λογισμικού και πολλά άλλα. Οι εφαρμογές συνήθως βασίζονται σε διαμοιραζόμενα εικονικά περιβάλλοντα. Λόγω της διασποράς των χρηστών, είναι απαραίτητη η χρήση κάποιου μοντέλου που διασφαλίζει την συνεκτικότητα των δεδομένων.

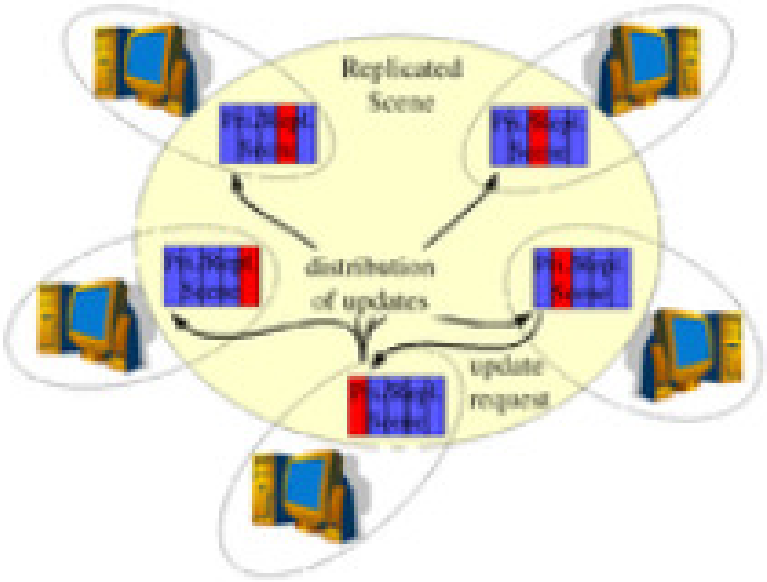
Το μοντέλο συνεκτικότητας επηρεάζει σημαντικά το προγραμματιστικό μοντέλο της εφαρμογής. Σύμφωνα με το (*Pečiva, 2007*), τα CVE κατηγοριοποιούνται στις κλάσεις που περιγράφονται αμέσως παρακάτω. Τα κριτήρια κατηγοριοποίησης είναι η αρχιτεκτονική (κεντρικοποιημένη/κατανεμημένη), ο τύπος της αναπαραγωγής των δεδομένων (replication) και η απόδοση.



Εικόνα 4 – Τυπική αρχιτεκτονική CVE.

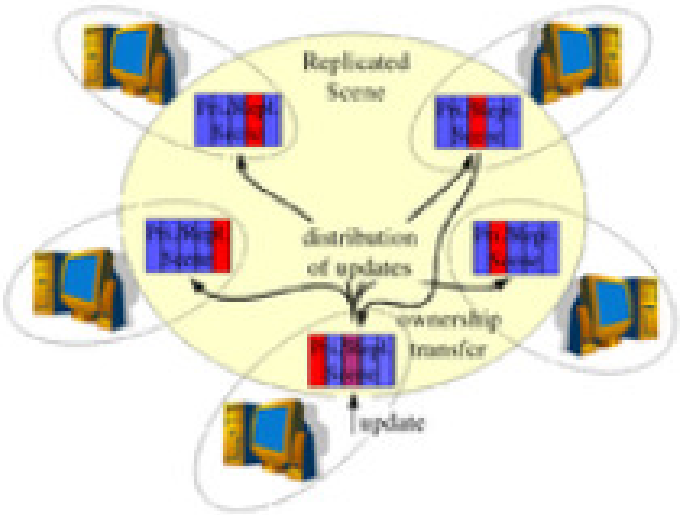
Κεντρικοποιημένα (Centralized). Όλα τα αντίγραφα των δεδομένων αποθηκεύονται στο ίδιο μηχάνημα που βρίσκονται τα πρωτότυπα δεδομένα. Η κατηγορία αυτή πλεονεκτεί στο ότι ο κόμβος έχει απόλυτο έλεγχο πάνω στην σκηνή. Το μειονέκτημα του είναι η συνολική απόδοση.

Κατανεμημένα (Distributed). Τα αντίγραφα των δεδομένων διαμοιράζονται σε όλους τους κόμβους. Αυτή η προσέγγιση αυξάνει την απόδοση του συστήματος και του προσφέρει δυνατότητες κλιμάκωσης. Ωστόσο, δυσκολεύει πολύ τον προγραμματισμό και αδυνατίζει την συνεκτικότητα των δεδομένων.



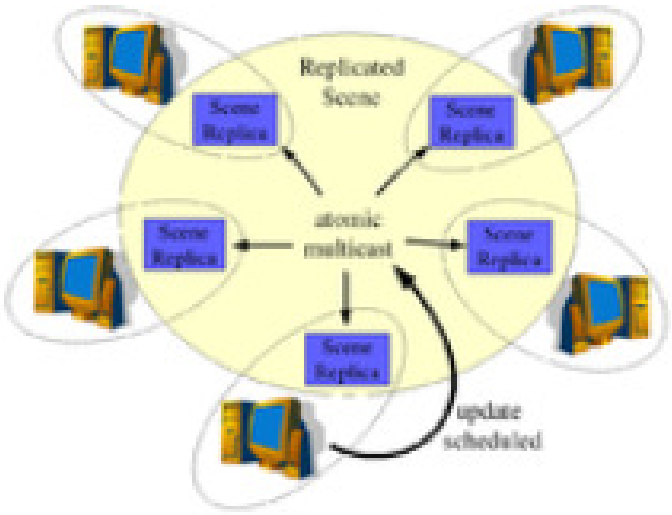
Εικόνα 5 – Τυπική αρχιτεκτονική κατανεμημένου CVE

Data ownership. Εδώ ακολουθείται μετατροπή (migration) των δεδομένων από κόμβο σε κόμβο. Αυτή η προσέγγιση είναι πιο ευέλικτη σε σχέση με την προηγούμενη κατηγορία. Ωστόσο, εισάγει μεγαλύτερη πολυπλοκότητα στο σύστημα, πράγμα που επηρεάζει αρνητικά την απόδοση.



Εικόνα 6 – Data ownership.

Ενεργητική αντιγραφή (Active replication). Αυτή η κατηγορία ακολουθεί μια peer-to-peer προσέγγιση, όπου όλα τα αντίγραφα είναι ισοδύναμα. Συνήθως, χρησιμοποιείται ατομικό broadcast για την ανανέωση των δεδομένων. Έτσι επιτυγχάνεται και ο επιθυμητός συγχρονισμός. Το πλεονέκτημα έγκειται στον πλήρη και συνεχή συγχρονισμό σε όλους τους κόμβους, μειώνει όμως την απόδοση.



Εικόνα 7 – Active replication.

Τα CVE στοχεύουν να παρέχουν ένα ολοκληρωμένο πλαίσιο εργασίας (framework) το οποίο υποστηρίζει την συνεργατικότητα μεταξύ των μερών που συμμετέχουν. Αυτό το χαρακτηριστικό κάνει τα συστήματα αυτά κατάλληλα για χρήση στην εκπαίδευση. Η χρήση των συνεργατικών εικονικών περιβαλλόντων στην εκμάθηση είναι μια από τις πιο ελπιδοφόρες χρήσεις της τεχνολογίας της εικονικής πραγματικότητας.

3.7 Λογισμικά για DVE, CVE

Η δυνατότητα της χρήσης ενός εικονικού περιβάλλοντος από έναν χρήστη βαθμιαία οδήγησε στην ανάγκη της ταυτόχρονης χρήσης του συστήματος από περισσότερους χρήστες και επιπλέον την αλληλεπίδραση μεταξύ τους, και όχι μόνο μεταξύ του χρήστη και του συστήματος. Οι πρώτες προσπάθειες πολυχρηστικότητας εστίαζαν στην ανάθεση ενός συνόλου από συσκευές εισόδου/εξόδου ανά χρήστη και στην σύνδεση όλων στο ίδιο, αυτόνομο λογισμικό.

Έτσι επιτυγχάνονταν η παραγωγή διαφορετικών όψεων του εικονικού περιβάλλοντος. Ωστόσο, απαιτούνταν η φυσική παρουσία όλων των χρηστών στον ίδιο γεωγραφικό χώρο και μάλιστα στο χώρο όπου το σύστημα ήταν εγκατεστημένο. Η προφανής λύση σε αυτό το πρόβλημα είναι η χρήση πολλαπλών στιγμιότυπων του λογισμικού και η επικοινωνία μεταξύ τους μέσω ενός δικτύου δεδομένων.

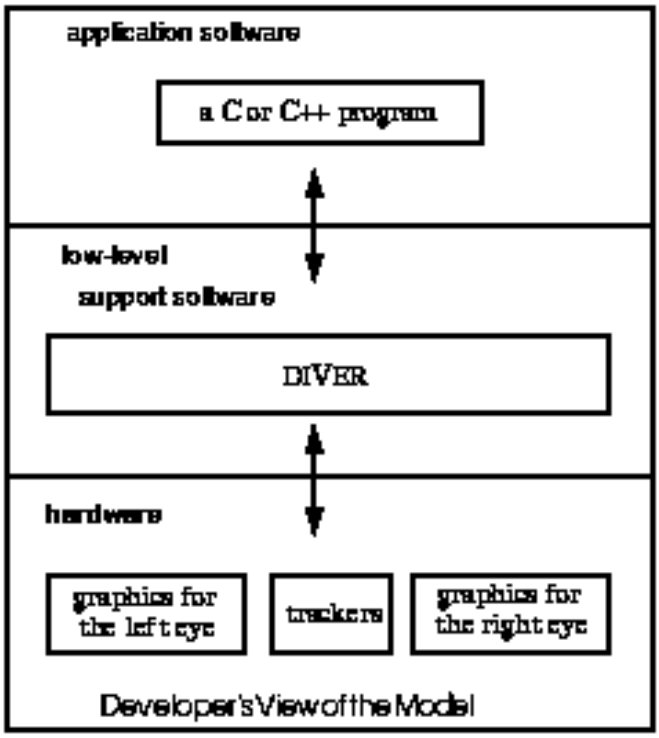
3.7.1 DVE

Στην ενότητα αυτή παρουσιάζονται συνοπτικά τα πιο σημαντικά, από άποψη δημοτικότητας ή τεχνολογικής καινοτομίας, λογισμικά εικονικού περιβάλλοντος.

3.7.1.1 Distributed Virtual Environment Research Platform (DIVER)

Το σύστημα DIVER είναι ένα εικονικό περιβάλλον που παρέχει ένα εύχρηστο περιβάλλον διεπαφής για προγραμματιστές εφαρμογών. Το DIVER κατατμεί αδιαφανώς τους υπολογισμούς που απαιτεί η επεξεργασία των δεδομένων εισόδου και το rendering, εκτελώντας τους σε διαφορετικούς κόμβους που μπορεί να είναι και τελείως ανομοιογενείς από άποψη υλικού. Για αυτό το σκοπό, το DIVER παρέχει μια βιβλιοθήκη της γλώσσας προγραμματισμού C, η οποία περιέχει συναρτήσεις που

πραγματοποιούν αδιαφανώς απομακρυσμένους υπολογισμούς. Στην επόμενη εικόνα φαίνεται το διάγραμμα της αρχιτεκτονικής του DIVER.

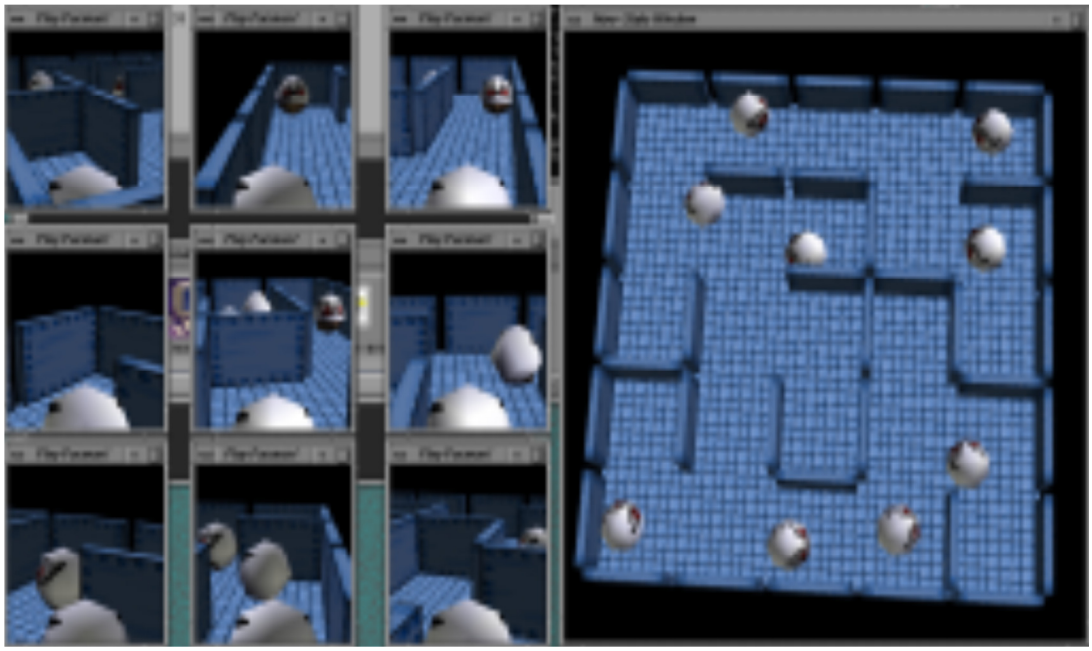


Εικόνα 8 – Η αρχιτεκτονική του DIVER.

Η επικοινωνία γίνεται με την χρήση Remote Procedure Calls (RPC) στην διεργασία-εξυπηρετητή, δημιουργώντας έτσι ένα υπόστρωμα ασύγχρονης επικοινωνίας. Το DIVER επιτρέπει την σχετικά εύκολη ανάπτυξη νέων εφαρμογών εικονικού περιβάλλοντος.

3.7.1.2 AVOCADO

Το λογισμικό Avocado είναι ένα πλαίσιο εργασίας (framework) για εφαρμογές κατανεμημένων εικονικών περιβαλλόντων. Επιτελεί τον ρόλο της κεντρικής πλατφόρμας, επί της οποίας μπορούν να αναπτύσσονται άλλες εφαρμογές εικονικών περιβαλλόντων. Χαρακτηρίζεται από ένα ιεραρχημένο μοντέλο αντικειμένων και γεγονότων, υποστηρίζει πλήθος συσκευών απεικόνισης, είναι επεκτάσιμο και πολύ αποδοτικό.



Εικόνα 9 – Στιγμιότυπο εφαρμογής που έχει υλοποιηθεί στην πλατφόρμα AVOCADO.

Τα κύρια χαρακτηριστικά της πλατφόρμας είναι τα εξής:

- Η κύρια τεχνολογία υλοποίησης του Avocado είναι το OpenGL Performer, που υλοποιεί την αναπαράσταση δεδομένων και την απεικόνιση.
- Το μοντέλο αντικειμένων ακολουθεί το πρότυπο «field container» και υλοποιεί τις απαραίτητες διεπαφές, φτιάχνοντας υποκλάσεις των κλάσεων-κόμβων του performer. Οι κόμβοι (Nodes) και οι αισθητήρες (sensors) αποτελούν τις δύο κύριες κλάσεις αντικειμένων. Οι κόμβοι αναπαριστούν τα αντικείμενα που χρησιμοποιούνται για το χτίσιμο της τρισδιάστατης σκηνής, ενώ οι αισθητήρες είναι abstraction για είσοδο δεδομένων.
- Η πλατφόρμα χρησιμοποιεί μια ξεχωριστή γλώσσα σεναρίων (scripting language) για την κατασκευή εφαρμογών, που ονομάζεται Elk. Υπάρχουν binding στην γλώσσα αυτή για κάθε στοιχείο του API.

3.7.1.3 Second Life

Το SecondLife είναι ένας εικονικός κόσμος που δημιουργήθηκε το 2003 και είναι προσβάσιμος μέσω διαδικτύου. Οι κάτοικοι μπορούν να εξερευνούν, να συναντούν και να συναναστρέφονται άλλους κατοίκους, να συμμετέχουν σε ατομικές και

ομαδικές δραστηριότητες καθώς και να δημιουργούν και να ανταλλάσσουν εικονική ιδιοκτησία και υπηρεσίες ή να ταξιδεύουν παντού σε αυτόν τον κόσμο.

Εικόνα 10 – Στιγμιότυπο χρήσης του λογισμικού Second Life.

Το Second Life παρέχει έναν αριθμό από ελεύθερα διανεμόμενους πελάτες (client), που επιτρέπουν στους χρήστες (Residents), να αλληλεπιδράσουν μεταξύ τους και με το σύστημα μέσω τρισδιάστατων, εικονικών πλασμάτων (avatars). Οι residents μπορούν να περιηγούνται στο εικονικό περιβάλλον (που ονομάζεται grid) και να συμμετάσχουν σε ατομικές και ομαδικές δραστηριότητες κάθε είδους.

Το λογισμικό στηρίζεται σε ένα τρισδιάστατο εργαλείο μοντελοποίησης, που χρησιμοποιεί απλά γεωμετρικά σχήματα. Με αυτά τα σχήματα, οι residents μπορούν να δημιουργούν πιο σύνθετα εικονικά αντικείμενα. Μια διαδικαστική, scripting γλώσσα προγραμματισμού, που ονομάζεται Linden, χρησιμοποιείται για να προσδώσει διαδραστικότητα στα αντικείμενα.

Πολλά είδη αντικειμένων, όπως πλέγματα (mesh), textures και animations μπορούν να δημιουργηθούν με εξωτερικό λογισμικό και να εισαχθούν στο Second Life. Μέσω ειδικών όρων στην άδεια χρήσης του λογισμικού, οι χρήστες διατηρούν τα πνευματικά δικαιώματα των χαρακτήρων και των αντικειμένων που δημιουργούν και χρησιμοποιούν στο λογισμικό.

3.7.1.4 SIMNET

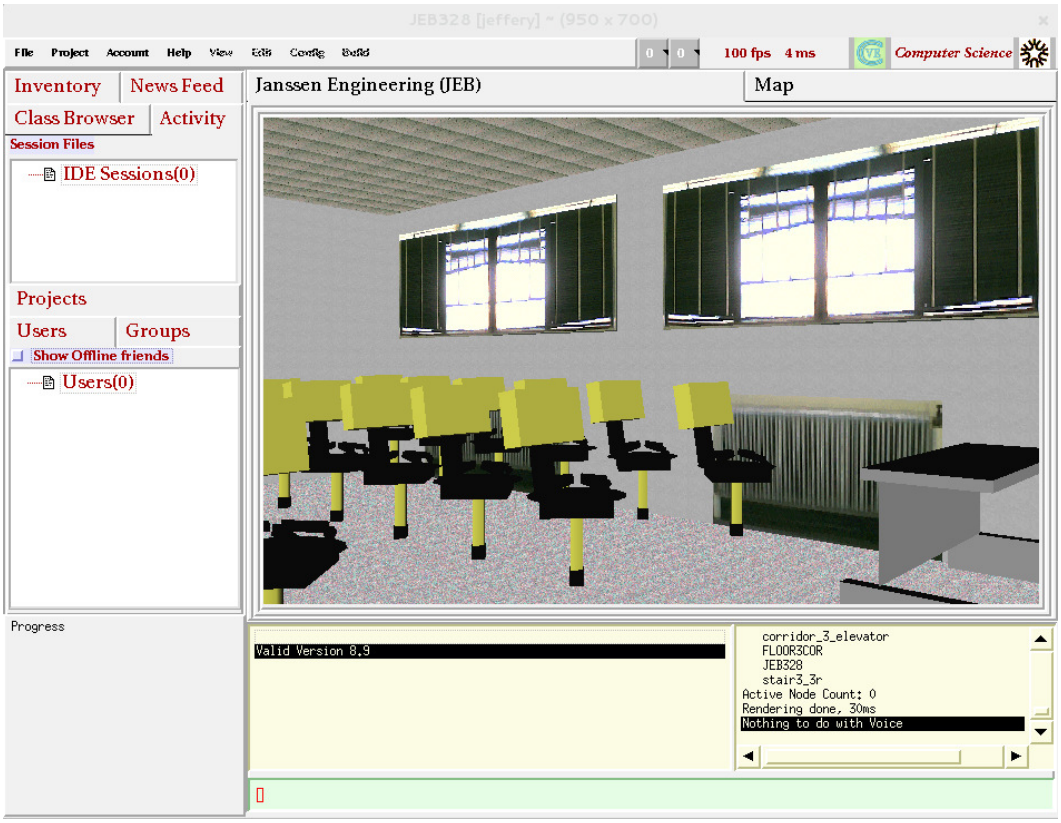
Το SIMNET είναι ένα από τα πρώτα λογισμικά στρατιωτικής προσομοίωσης μεγάλης κλίμακας. Η ονομασία προέρχεται από τα αρχικά «Simulator Networking project» και ξεκίνησε να αναπτύσσεται το 1983. Στην δεκαετία του ’90 το SIMNET αποτελούταν από 250 διασυνδεδεμένους προσομοιωτές, σε 11 κόμβους. Οι σχεδιαστικές αρχές που ακολουθεί το SIMNET είναι ακόμα σε χρήση, όπως η αντικειμενοστραφής προσέγγιση των κόμβων.

Το SIMNET έδωσε το έναυσμα για την ανάπτυξη του πρωτοκόλλου Distributed Interactive Simulation (DIS). Το SIMNET έδειξε τις τεράστιες δυνατότητες που υπάρχουν στην τεχνολογία των κατανεμημένων εικονικών περιβαλλόντων, ωστόσο δεν αποτέλεσε ποτέ ένα ανοικτό, καλώς τεκμηριωμένο πρότυπο. Αυτό το κενό ήρθε να καλύψει το DIS, διατηρώντας την δυνατότητα υποστήριξης πολλών κόμβων, στιγμιοτύπων, χρηστών και αντικειμένων.

3.7.2 CVE

3.7.2.1 CVE

Το CVE είναι ένα συνεργατικό εικονικό περιβάλλον, ανοικτού κώδικα, που υποστηρίζει πολλές διαφορετικές πλατφόρμες και είναι ικανό να υποστηριξει την εύκολη και οικονομική ανάπτυξη συνεργατικών εφαρμογών εικονικού περιβάλλοντος.

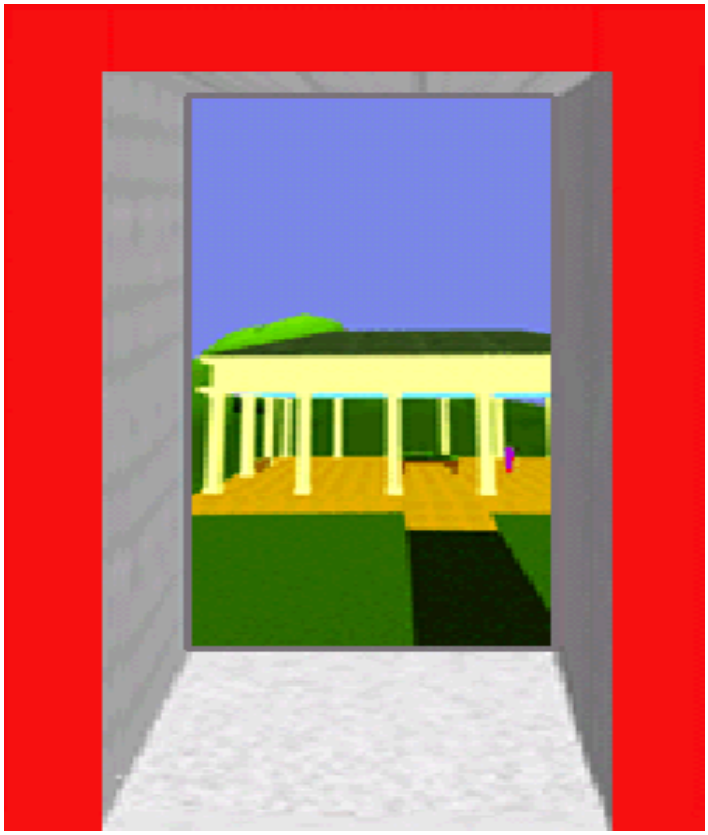


Εικόνα 11 – Στιγμιότυπο του CVE.

Αναπτύσσεται και υποστηρίζεται κατά κύριο λόγο από το τμήμα Επιστήμης Υπολογιστών, στο Πανεπιστήμιο του Idaho των ΗΠΑ. Έχει υλοποιηθεί σε OpenGL και διανέμεται δωρέαν, μαζί με τον πηγαίο του κώδικα υπό την GNU GPL άδεια χρήσης.

3.7.2.2 Scalable Platform for Large Interactive Networked Environments (SPLINE)

Το SPLINE είναι ένα σύστημα CVE που αναπτύσσεται και υποστηρίζεται από την Mitsubishi Electric Research Laboratories, υπό την καθοδήγηση των Richard Waters και David Anderson. Το SPLINE αποτελεί ουσιαστικά ένα ενδιάμεσο λογισμικό (middleware) που περιλαμβάνει εργαλεία για την κατασκευή εφαρμογών για τους τελικούς χρήστες. Για αυτό το λόγο, παρέχει ένα API που έχει υλοποιηθεί στην γλώσσα προγραμματισμού JAVA και χρησιμοποιείται για την ανάπτυξη εικονικών κόσμων.



Εικόνα 12 – Εφαρμογή υλοποιημένη με το SPLINE API.

Ένα από τα πιο ενδιαφέροντα χαρακτηριστικά του SPLINE είναι ότι δεν ορίζει ολοκληρωμένους εικονικούς κόσμους, αλλά μικρές περιοχές (που τις ονομάζει regions ή locales) και οι οποίες μπορούν να συνενώνονται και να σχηματίζουν ολοκληρωμένους κόσμους και δίκτυα.

Η χρήση των locales επιτρέπει την διαλειτουργικότητα μεταξύ αντικειμένων που μπορεί να έχουν δημιουργηθεί από διαφορετικούς χρήστες, σε διαφορετικές πλατφόρμες. Η πρώτη χρήση του λογισμικού ήταν η ανάπτυξη του περιβάλλοντος «Diamond Park», στιγμιότυπο του οποίου φαίνεται στην παραπάνω εικόνα.

Κεφάλαιο 4 Το περιβάλλον του Second Life (SL)

Στο παρόν κεφάλαιο, επιλέγεται μια συγκεκριμένη πλατφόρμα ανάπτυξης ενός τέτοιου συστήματος και παρουσιάζεται εκτενώς. Σε αυτήν την πλατφόρμα θα στηριχθεί η υλοποίηση που θα πραγματοποιηθεί στα πλαίσια αυτής της εργασίας, και που θα παρουσιαστεί αναλυτικά στα επόμενα κεφάλαια.

4.1 Γενικά

Το λογισμικό Second Life (συντομογραφικά SL) είναι ένα λογισμικό εικονικού κόσμου που λειτουργεί σε δικτυακό περιβάλλον. Αναπτύχθηκε και συντηρείται από την εταιρεία Linden Lab και διανεμήθηκε για πρώτη φορά στις 23 Ιουνίου του 2003. Οι χρήστες του λογισμικού (Residents) χρησιμοποιούν οποιονδήποτε από τα λογισμικά-πελάτες που κυκλοφορούν (Viewers), προκειμένου να αποκτήσουν πρόσβαση στον εικονικό αυτό κόσμο.

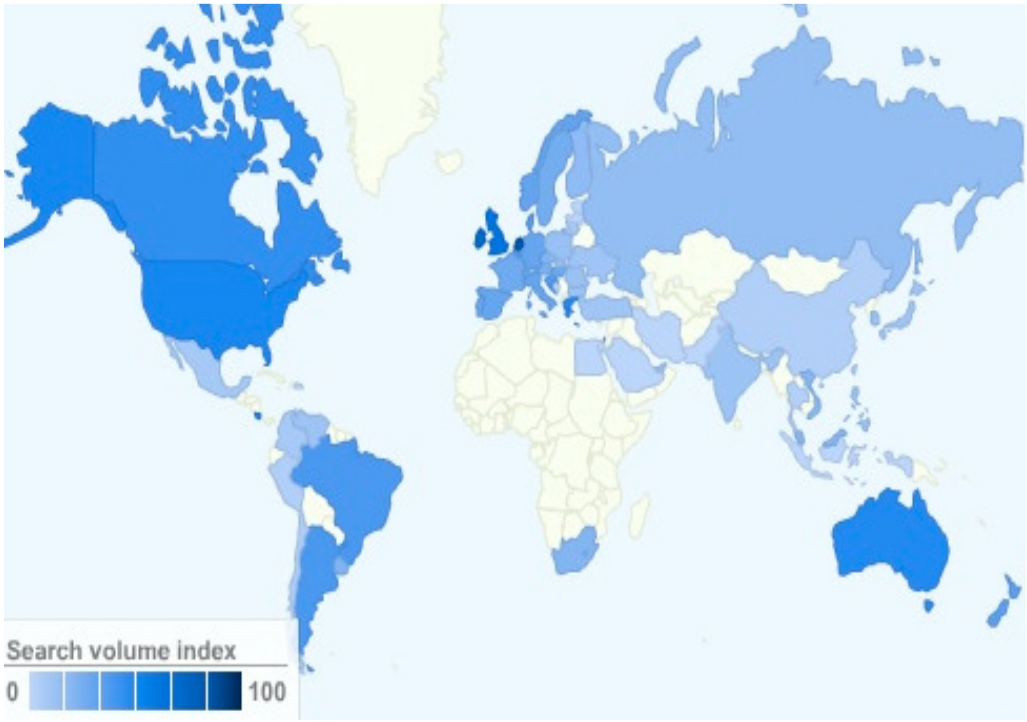
Κάθε χρήστης αναπαρίσταται από ένα τρισδιάστατο αντικείμενο (avatar). Καθοδηγώντας το avatar ο χρήστης πλοηγείται στον κόσμο (που ονομάζεται «grid»), και επιτελεί διάφορες δραστηριότητες που μιμούνται την πραγματική, κοινωνική δραστηριότητα, όπως για παράδειγμα να συνομιλήσει, να συναντηθεί με άλλους, να δημιουργήσει και να εμπορευτεί εικονικά προϊόντα και υπηρεσίες. Το Second Life απευθύνεται σε χρήστες ηλικίας 16 ετών και άνω.

Στο λογισμικό βρίσκεται ενσωματωμένο ένα εργαλείο τρισδιάστατης μοντελοποίησης, το οποίο μπορεί ο χρήστης να χρησιμοποιεί για να κατασκευάζει, με συνδυασμό απλών γεωμετρικών σχημάτων, περίπλοκα εικονικά αντικείμενα. Στα αντικείμενα αυτά είναι δυνατή η πρόσθεση διαδραστικών ικανοτήτων, πράγμα που επιτυγχάνεται με την συγγραφή προγραμμάτων στην ειδική, διαδικαστική, διερμηνευόμενη γλώσσα σεναρίων, με την ονομασία Linden Scripting Language.

Είναι επίσης δυνατή η χρήση τρίτου λογισμικού για την δημιουργία αντικειμένων και η εισαγωγή τους στο περιβάλλον του Second Life. Η άδεια χρήσης του λογισμικού προβλέπει ότι αυτά τα αντικείμενα αποτελούν πνευματική ιδιοκτησία των δημιουργών τους. Είναι αξιοσημείωτο ότι το ίδιο το λογισμικό παρέχει λειτουργίες διαχείρισης ψηφιακών δικαιωμάτων.

4.2 Ιστορία

Το 1999, ο Philip Rosedale ίδρυσε την εταιρεία Linden Lab. Το Second Life προέκυψε σαν εξέλιξη της έρευνας που πραγματοποίησε, πάνω στην ανάπτυξη υλικού Η/Υ για εικονικούς τρισδιάστατους κόσμους. Στην πρώτη του μορφή, το λογισμικό ονομαζόταν "The Rig", και συνόδευε ένα σύστημα εικονικής πραγματικότητας που προσαρμόζονταν στο σώμα του χρήστη και συνοδεύονταν από μια οθόνη. Αυτό μετεξελίχθηκε στην απλή εφαρμογή υπολογιστή Linden World, όπου οι χρήστες συμμετείχαν σε task-based παιχνίδια σε ένα τρισδιάστατο, εικονικό περιβάλλον.



Εικόνα 13 – Η δημοτικότητα του SL ανά γεωγραφική περιοχή (το πιο σκούρο χρώμα σημαίνει μεγαλύτερη παρουσία εγγεγραμμένων χρηστών).

Τα έτη 2005 και 2006, το Second Life έφτασε στο αποκορύφωμα της δημοτικότητας και της προβολής του από τα ΜΜΕ. Την ίδια περίοδο, οι χρήστες του λογισμικού αυξήθηκαν εκθετικά. Το αποκορύφωμα της ταυτόχρονης πρόσβασης στον εικονικό κόσμο σημειώθηκε τον Ιανουάριο του 2008. Από το 2007 μέχρι το 2010 πραγματοποιήθηκαν αρκετές αλλαγές στα υψηλά κλιμάκια της διοίκησης της εταιρείας, που απηχούσαν και την πτώση στην δημοτικότητα του λογισμικού.

4.3 Κατηγοριοποίηση

Το Second Life αρχικά εστίαζε στον χαρακτήρα του σαν ηλεκτρονικό παιχνίδια, σύντομα όμως διαπιστώθηκε ότι οι χρήστες του ενδιαφέρονταν περισσότερο για τις δυνατότητες του σαν εργαλεία συνεργατικότητας και οι επόμενες εκδόσεις του έστρεψαν εκεί το ενδιαφέρον τους. Έτσι το λογισμικό εξελίχθηκε και έδωσε βάρος στην user-created και community-driven εμπειρία χρήσης.

Το Second Life διαφέρει πλέον από τα κλασσικά, ηλεκτρονικά παιχνίδια, ως προς το ότι οι χρήστες δεν προσπαθούν να επιτύχουν έναν προκαθορισμένο στόχο και συνεπώς δεν υπάρχουν μηχανισμοί και κανόνες που να τον κατευθύνουν προς τα εκεί. Επίσης, το Second Life συγκαταλέγεται στους πολύ-χρηστικούς, εικονικούς κόσμους μιας που η δραστηριότητα περιστρέφεται γύρω από την αλληλεπίδραση μεταξύ των χρηστών.

4.4 Τεχνικά Χαρακτηριστικά

Το Second Life αποτελείται από δύο συστατικά στοιχεία:

- Το λογισμικό πελάτη (viewer ή client) που εκτελείται στο υπολογιστικό σύστημα του χρήστη.
- Το λογισμικό εξυπηρετητή (server) που εκτελείται σε υπολογιστικά συστήματα – εξυπηρετητές που εγκαθίστανται και συντηρούνται από την εταιρεία Linden Lab.

4.4.1 Το λογισμικό - πελάτης (client)

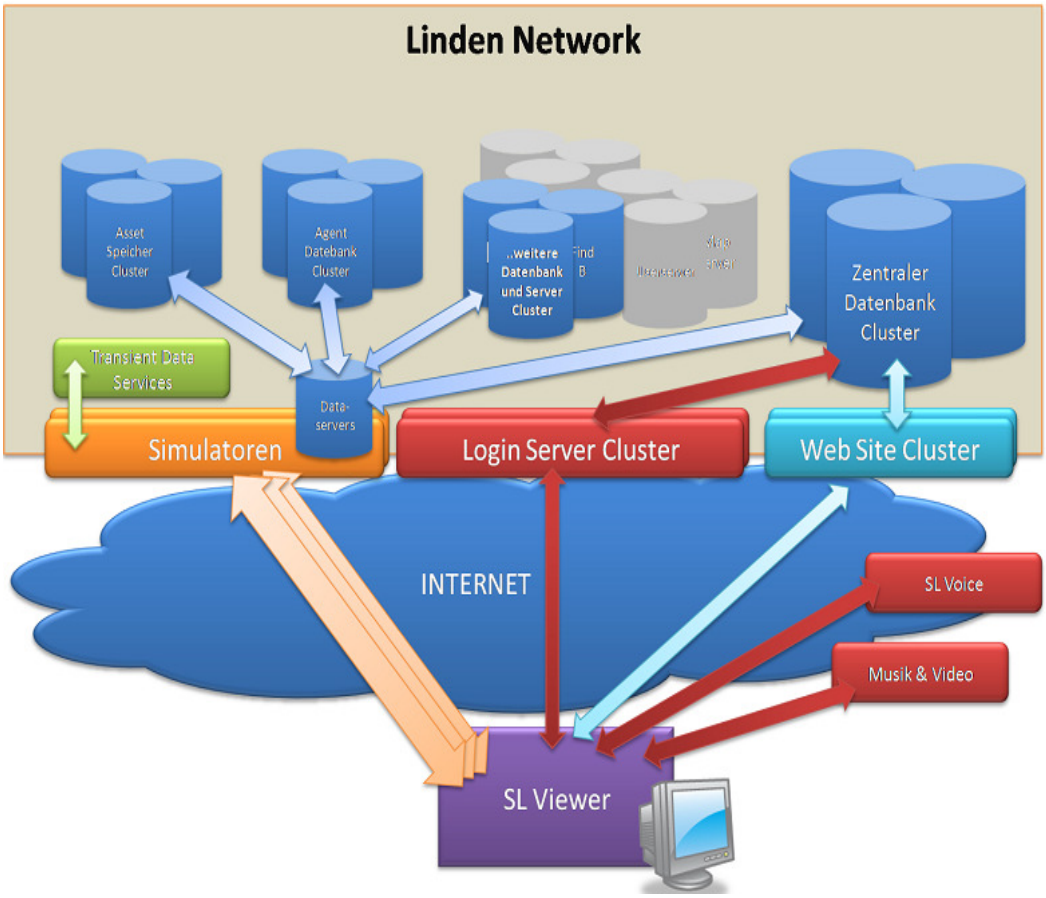
Η Linden Lab παρέχει και υποστηρίζει viewers για λειτουργικά συστήματα Microsoft Windows XP / Vista / 7, για Mac OS X, καθώς επίσης και για τις πιο διαδεδομένες διανομές Linux. Επίσης, υπάρχουν viewers για τα λειτουργικά συστήματα Solaris και OpenSolaris, που όμως αναπτύσσονται και συντηρούνται από τρίτους. Ο viewer απεικονίζει τα γραφικά με την χρήση της OpenGL.

Εφόσον ο πηγαίος κώδικας του viewer είναι ανοικτός, οι χρήστες με τις ανάλογες τεχνικές γνώσεις μπορούν να μεταγλωττίζουν και να τροποιποιούν τον πηγαίο κώδικα, ακόμα και να φτιάχνουν τους δικούς τους viewers. Τέτοιοι είναι οι Nicholaz Edition και Phoenix, που μετεξελίχθηκε στον Firestorm viewer.

4.4.2 Το λογισμικό - εξυπηρετητής (server)

Κάθε εξυπηρετητής (ισχυρά φυσικά υπολογιστικά συστήματα με επεξεργαστές πολλαπλών πυρήνων) του Second Life αναλαμβάνει την διαχείριση ενός grid. Το κάθε grid είναι μια ολοκληρωμένη περιοχή (διαστάσεων 256X256 μέτρων), ενώ υπάρχουν τα λεγόμενα Homestead regions, όπου έχουμε 3 regions ανά core και τα Openspace Regions, με 4 regions ανά core. Κάθε εξυπηρετητής εκτελεί το λογισμικό εξυπηρετητή και το λειτουργικό σύστημα είναι η διανομή Debian του Linux.

Οι εξυπηρετητές εκτελούν σενάρια πάνω στις περιοχές και παρέχουν επικοινωνία μεταξύ των avatars και όλων των αντικειμένων που βρίσκονται στην συγκεκριμένη περιοχή. Κάθε στοιχείο του εικονικού κόσμου ονομάζεται με την γενική ονομασία «asset». Σε αυτά συμπεριλαμβάνονται τρισδιάστατα αντικείμενα (primitives), ψηφιακές εικόνες που διακοσμούν τα primitives (textures), αρχεία ήχου, LSL scripts και πολλά άλλα. Κάθε asset ταυτοποιείται με ένα μοναδικό αναγνωριστή (unique identifier ή UUID).



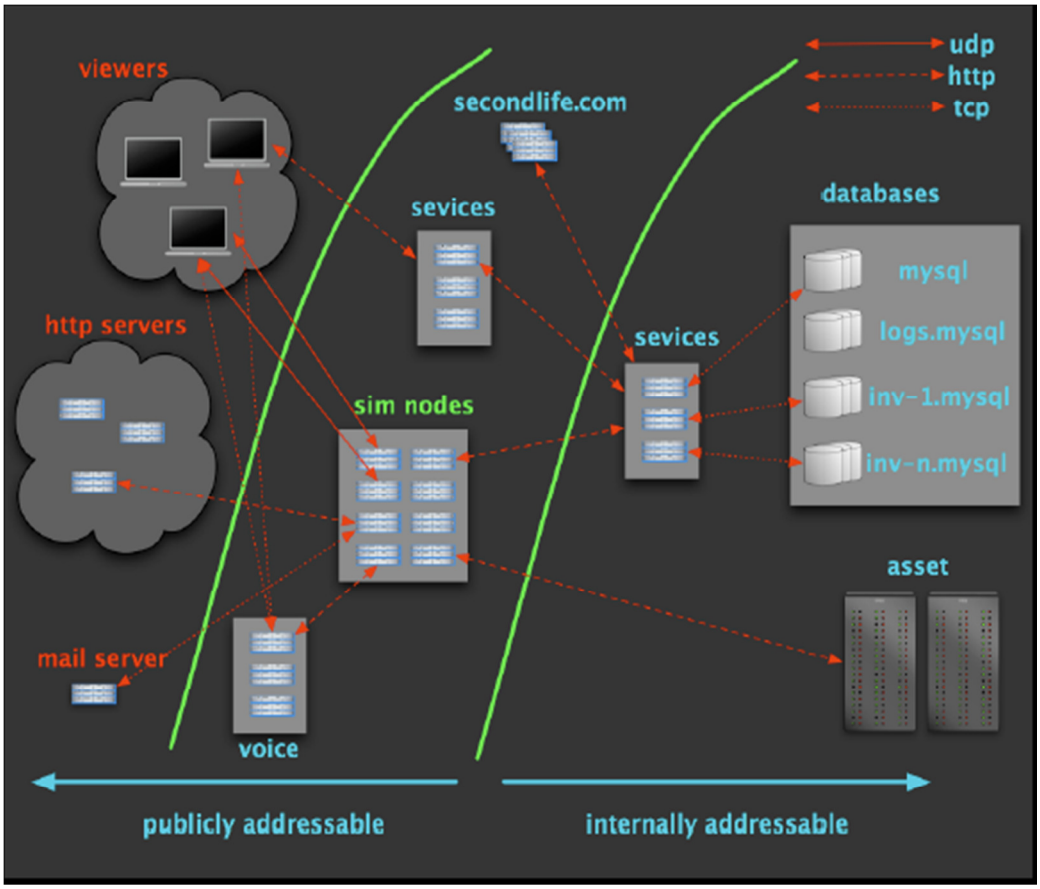
Εικόνα 14 – Η αρχιτεκτονική του δικτύου.

Τον Δεκέμβρη του 2007, ο συνολικός αποθηκευτικός χώρος που κατανάλωνε το δίκτυο του SL ξεπερνούσε τα 100TB. Κάθε server instance εκτελεί προσομοιώσεις φυσικής για να αναπαραστήσει όλες της φυσικές αλληλεπιδράσεις μεταξύ των αντικειμένων στην περιοχή ευθύνης του. Τα αντικείμενα μπορεί να είναι μη-φυσικά και ακίνητα ή φυσικά και κινητά. Κάθε σύνθετο αντικείμενο μπορεί να αποτελείται από 255 ξεχωριστά primitives, συνδεδεμένα μεταξύ τους.

Το avatar κάθε χρήστη θεωρείται φυσικό αντικείμενο και επομένως μπορεί να αλληλεπιδρά με άλλα φυσικά αντικείμενα. Για την προσομοίωση των φυσικών νόμων, το Second Life χρησιμοποιεί την μηχανή φυσικής Havok 4. Γενικά, η Linden Lab υποστηρίζει όσο το δυνατόν περισσότερο την χρήση ανοικτών τεχνολογιών και προγραμματιστικών εργαλείων, μεταξύ των οποίων ξεχωρίζουν τα Apache, MySQL, Squid και το λειτουργικό σύστημα Linux.

4.4.3 Εικονική Τεχνολογία

Τα γραφικά, η γλώσσα Linden Scripting Language και η μηχανή Havok επιτρέπουν την προσομοίωση διάφορων πραγματικών ή φανταστικών μηχανών, συσκευών και γενικότερα αντικειμένων. Έτσι, για παράδειγμα, μπορεί να συναντήσει κανείς φάρους, με αληθοφανή συμπεριφορά, αερόπλοια, ακόμα και οπλικά συστήματα. Ένα μεγάλο κομμάτι του επίσημου οδηγού της Linden Scripting Language περιγράφει την μοντελοποίηση αυτοκινούμενων οχημάτων. Ένα πρόβλημα που αντιμετωπίστηκε σε κάποιο βάθος χρόνου είναι η μετακίνηση από ένα region σε ένα άλλο, με όσο το δυνατόν μικρότερες ασυνέχειες. Η Φυσική στο SL συνίσταται κυρίως στην αποφυγή φαινομένων όπου κάποιο avatar ή άλλο αντικείμενο φαίνεται να διαπερνά κάποιο άλλο.



Εικόνα 15 – Αρχιτεκτονική του λογισμικού Second Life.

4.5 Εμπειρία Χρήσης (Gameplay)

Το Second Life δεν χρεώνει τους χρήστες του για την δημιουργία λογαριασμού και την παρουσία του μέσα στον εικονικό κόσμο. Ωστόσο, η Linden Lab διατηρεί το δικαίωμα να επιβάλλει χρεώσεις στους χρήστες που δημιουργούν και εισάγουν στο σύστημα πολύ μεγάλο αριθμό από αντικείμενα ή που διατηρούν πολλαπλούς λογαριασμούς. Αυτό το δικαίωμα δεν το έχει εξασκήσει ακόμα.

Μια εγγραφή στο σύστημα τύπου Premium κοστίζει $9.95/μήνα, $22.50/τετράμηνο και $72/Χρόνο. Μια premium εγγραφή εξασφαλίζει τεχνική υποστήριξη και δίνει στο avatar χρηματικές μονάδες για χρήση εντός του εικονικού κόσμου. Ωστόσο, η πλειοψηφία των χρηστών παραμένει στον δωρεάν, βασικό τύπο λογαριασμού. Τα avatars έχουν την μορφή που καθορίζει ο χρήστης, που μπορεί να είναι οποιαδήποτε, αν και συνηθίζεται να έχουν την μορφή που ο χρήστης τους έχει στην πραγματική ζωή.

Η κουλτούρα και γενικότερα η κοινωνική ζωή του Second Life περιλαμβάνει ό,τι και η πραγματική. Κάθε resident μπορεί να χρησιμοποιεί μόνο ένα avatar τη φορά, αν και δύναται να αλλάζει την μορφή του όποτε και όσο συχνά επιθυμεί. Οι μορφές των avatar, όπως και οτιδήποτε άλλο στο SL, μπορεί να δημιουργηθεί από τον χρήστη ή να αγοραστεί έτοιμο.

Τα avatars μπορούν να μετακινούνται περπατώντας, τρέχοντας, πετώντας ή με τηλεμεταφορά. Η τηλεμεταφορά είναι ιδιαίτερα διαδεδομένη ανάμεσα στους χρήστες του λογισμικού, μιας που ο εικονικός κόσμος είναι μεγάλης έκτασης. Τα avatars επικοινωνούν με μεθόδους chat, group chat, global instant messaging και φωνή. Το chatting χρησιμοποιείται για δημόσιες συνομιλίες ανάμεσα σε 2 ή περισσότερα avatars, και είναι φανερό σε avatars που βρίσκονται μέσα σε κάποια συγκεκριμένη απόσταση. Αντίθετα, το global instant messaging χρησιμοποιείται για ιδιωτικές συζητήσεις.

Ένα από τα ποιο σημαντικά χαρακτηριστικά του Second Life είναι η οικονομική δραστηριότητα και το ειδικό νόμισμα με το οποίο εκφράζεται (το Linden dollar που συμβολίζεται ως L$). Τα L$ χρησιμοποιούνται για αγορές, πωλήσεις, ενοικιάσεις ή εμπορία εικονικών αγαθών (κτίρια, οχήματα, ρούχα κλπ) και υπηρεσιών (ψυχαγωγία, δημιουργία περιεχομένου κλπ) με άλλους χρήστες.

Κάθε χρήστης μπορεί να αποκτήσει L$ καταβάλλοντας πραγματικά χρήματα στην υπηρεσία LindeX που διατηρεί η Linden Lab. Τα χρήματα που αποκτιούνται μέσω της οικονομικής δραστηριότητας στο Second Life, συνήθως επαρκούν για την πληρωμή μιας καλύτερης συνδρομής. Έχουν αναφερθεί, ωστόσο, και ολιγάριθμες περιπτώσεις απόκτησης μεγάλων χρηματικών ποσών.

Κεφάλαιο 5 Σενάριο, σχεδιασμός, υλοποίηση εφαρμογής

Στο παρόν κεφάλαιο, παρουσιάζεται βήμα-προς-βήμα ο σχεδιασμός και η ανάπτυξη ενός δικού μας συστήματος εικονικής διαβούλευσης, από την ανάλυση απαιτήσεων έως και την τελική υλοποίηση.

5.1 Σενάριο

Η εφαρμογή εικονικής διαβούλευσης πραγματοποιείται σε έναν εικονικό, τρισδιάστατο χώρο, στον οποίον προσέρχονται οι χρήστες σε προκαθορισμένες ημέρες και ώρες στις οποίες έχουν προγραμματιστεί οι εικονικές διαβουλεύσεις (debating events). Κάθε χρήστης επιτελεί έναν ρόλο που συμπεριλαμβάνει μια βαθμίδα συμμετοχής στην διαδικασία της διαβούλευσης και εκτείνεται από απλή παρακολούθηση μέχρι διαχείριση της διαδικασίας.

Τα debates λαμβάνουν χώρα σε προκαθορισμένες μέρες/ώρες που θα ορίζουν οι admins και με θέμα που επίσης εκείνοι θα ορίζουν. Οι χρήστες (με τα avatars τους) προσέρχονται στο δωμάτιο, θα κάθονται στις θέσεις τους και η διαβούλευση θα ξεκινάει. Θα υπάρχουν δύο πλευρές σε κάθε debate (υπέρ-κατά) και ένας προκαθορισμένος αριθμός κύκλων τοποθετήσεων. Οι τοποθετήσεις θα γίνονται εναλλάξ, για κάθε πλευρά. Οι τοποθετήσεις γίνονται μέσω του συνηθισμένου chat του second life. Στο τέλος της διαδικασίας, οι θεατές ψηφίζουν για να αναδείξουν την νικήτρια πλευρά.

5.2 Σχεδιασμός

Όπως είδαμε και σε προηγούμενο κεφάλαιο, υπάρχουν πολλές καθιερωμένες μορφές δημόσιας διαβούλευσης, και ακόμα περισσότερες ειδικευμένες παραλλαγές τους. Στην παρούσα εργασία επιλέχθηκε η απλή μορφή της κλασσικής διαβούλευσης, όπου εστιάζει στην κατάθεση λογικών επιχειρημάτων και στην πειθώ πραγματικού κοινού. Ξεκινάει με την από κοινού επιλογή ενός θέματος, στο οποίο η μία πλευρά συμφωνεί και η άλλη διαφωνεί. Στην συνέχεια, πραγματοποιείται ανταλλαγή επιχειρημάτων σε γύρους τοποθέτησης, με προκαθορισμένη χρονική διάρκεια.

5.3 Χρήστες

Το περιβάλλον εικονικής διαβούλευσης διακρίνει τους συμμετέχοντες σε κατηγορίες χρηστών, ανάλογα με τον ρόλο που επιτελούν στην διεξαγωγή της διαβούλευσης. Συγκεκριμένα, διακρίνονται οι παρακάτω κατηγορίες χρηστών:

Διαχειριστής (administrator). Πρόκειται για την ομάδα χρηστών στην οποία ανήκει η εφαρμογή, με την έννοια του κατόχου των πηγαίων και εκτελέσιμων αρχείων της εφαρμογής καθώς και του/των λογαριασμού/ών που έχουν απεριόριστα δικαιώματα επί του εικονικού χώρου.

Συντονιστής (moderator). Πρόκειται για την ομάδα χρηστών που έχει το ρόλο του συντονιστή μιας διαδικασίας διαβούλευσης που πραγματοποιείται εντός του εικονικού χώρου διαβούλευσης.

Συμμετέχων (participant). Πρόκειται για την ομάδα χρηστών που αποκτούν δικαίωμα συμμετοχής σε κάποια διαδικασία (event) διαβούλευσης.

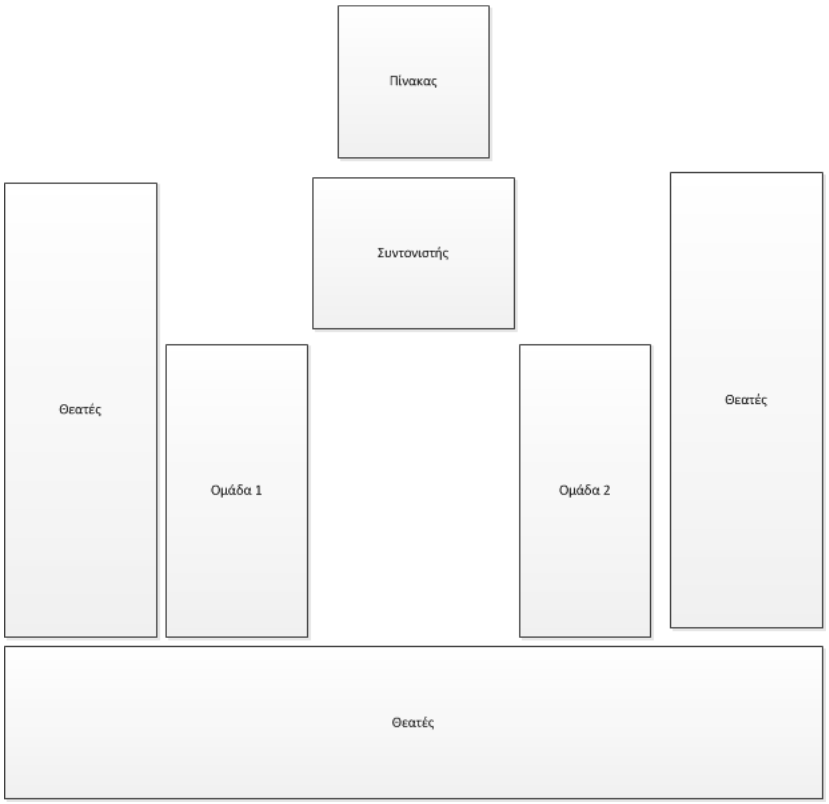
Θεατής (spectator). Πρόκειται για την ομάδα χρηστών που περιλαμβάνει όλους εκείνους τους χρήστες του λογισμικού που έχουν δικαίωμα να παρακολουθήσουν ένα event.

Εικόνα 16 - Άποψη του χώρου εικονικής διαβούλευσης.

5.4 Διαρρύθμιση χώρου

Στην παραπάνω εικόνα φαίνεται η διαρρύθμιση της αίθουσας διαβουλεύσεων, όπως αυτή έγινε render στο λογισμικό δημιουργίας των τρισδιάστατων μοντέλων, όπως θα δούμε παρακάτω. Η διαρρύθμιση του χώρου των εικονικών debates εξομοιώνει τους χώρους που χρησιμοποιούνται για διαβουλεύσεις που πραγματοποιούνται στον φυσικό κόσμο και δομείται όπως στο διάγραμμα της παρακάτω εικόνας.

Στην άκρη της αίθουσας υπάρχουν οι οριζόντιες σειρές θέσεων στις οποίες κάθονται οι απλοί θεατές τις διαβούλευσης, που έχουν δικαίωμα παρακολούθησης αλλά όχι συμμετοχής στην διαβούλευση. Επίσης, προβλέπονται θέσεις θεατών και στα πλάγια της αίθουσας πίσω από το τραπέζι της κάθε ομάδας διαβούλευσης. Στην κορυφή της αίθουσας είναι η θέση του διαχειριστή του χώρου (και συντονιστή της διαβούλευσης).



Εικόνα 17 – Διάγραμμα της αίθουσας διαβουλεύσεων.

5.5 Διαδικασίες

Οι χρήστες που επιθυμούν να παρακολουθήσουν ή να συμμετέχουν στην δημόσια διαβούλευση εγγράφονται σε σχετικό group που έχει δημιουργηθεί. Οι διαχειριστές αποστέλλουν μαζικά ενημερωτικά μηνύματα (newsletters) στα μέλη του group προκειμένου να τους ενημερώσουν για τις ημέρες, τις ώρες και τα θέματα των διαβουλεύσεων. Οι χρήστες απαντάνε με προσωπικά μηνύματα στους διαχειριστές εάν επιθυμούν να συμμετάσχουν στην συζήτηση μέχρι κάποιο προκαθορισμένο χρονικό σημείο. Εάν το υπερβούν μπορούν μόνο να παρακολουθήσουν.

Στην συνέχεια, οι διαχειριστές καταρτίζουν τις λίστες με τις ομάδες των συμμετεχόντων και κοινοποιούνται επίσης με μαζικά μηνύματα στο group. Η προσέλευση των ομάδων διαβούλευσης και των θεατών γίνεται μερικά λεπτά πριν την προκαθορισμένη έναρξη της διαβούλευσης. Όταν ολοκληρωθεί η προσέλευση, ο προεδρεύων διαχειριστής κηρύττει την έναρξη της διαβούλευσης.

5.6 Κανόνες

Θα υπάρχουν δύο πλευρές σε κάθε debate (υπέρ-κατά) και ένας προκαθορισμένος αριθμός κύκλων τοποθετήσεων. Οι τοποθετήσεις θα γίνονται εναλλάξ, για κάθε πλευρά. Οι τοποθετήσεις γίνονται μέσω του συνηθισμένου chat του second life. Στο τέλος της διαδικασίας, οι θεατές ψηφίζουν για να αναδείξουν την νικήτρια πλευρά.

5.7 Λογισμικό υλοποίησης

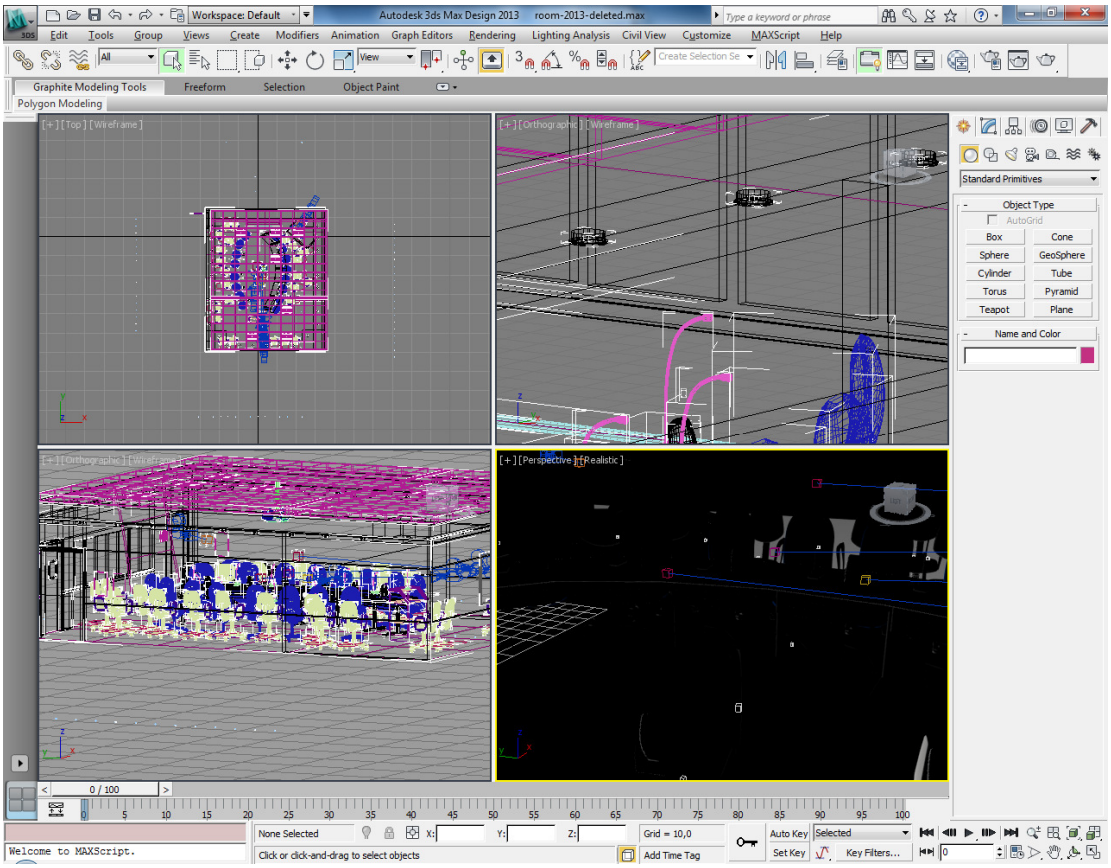
Στην ενότητα αυτή παρουσιάζουμε την χρήση των λογισμικών που αξιοποιήθηκαν για την δημιουργία του τρισδιάστατου περιβάλλοντος εικονικών διαβουλεύσεων.

5.7.1 Autodesk 3DS Max Design

Το λογισμικό 3ds Max (παλιότερα γνωστό ως 3D Studio Max), αναπτύσσεται, συντηρείται και διανέμεται από την εταιρεία Autodesk Media and Entertainment και πρόκειται για ένα λογισμικό ανάπτυξης τρισδιάστατων γραφικών (3D computer graphics - CG) για την δημιουργία animation, μοντέλων και εικόνων.

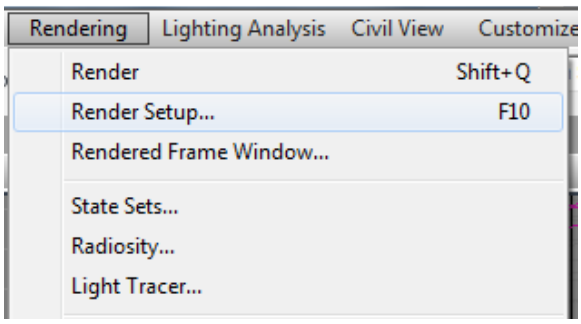
Στα βασικά του χαρακτηριστικά συμπεριλαμβάνονται οι δυνατότητες modeling, η επέκταση της λειτουργικότητας του με πρόσθετα (plugin), ενώ η χρήση του προορίζεται σε υπολογιστικά συστήματα με λειτουργικό σύστημα Microsoft Windows. Χρησιμοποιείται συχνά από σχεδιαστές ηλεκτρονικών παιχνιδιών, τηλεοπτικά στούντιο και επαγγελματίες που στηρίζονται στην δημιουργία γραφικών, όπως αρχιτέκτονες, πολιτικούς μηχανικούς και άλλους.

Επίσης, χρησιμοποιείται για οπτικά εφέ σε κινηματογραφικές ταινίες. Εκτός από τη μοντελοποίηση και τα εργαλεία animation, η τελευταία έκδοση του 3ds Max διαθέτει επίσης shaders, δυναμική προσομοίωση, συστήματα σωματιδίων, ένα προσαρμόσιμο περιβάλλον εργασίας χρήστη και τη δική της γλώσσα scripting, που ονομάζεται MaxScript. Στα πλαίσια της παρούσας εργασίας αναπτύξαμε έναν πλήρη τρισδιάστατο, εικονικό χώρο. Στην παρακάτω εικόνα βλέπουμε στιγμιότυπα από την επεξεργασία του μοντέλου.



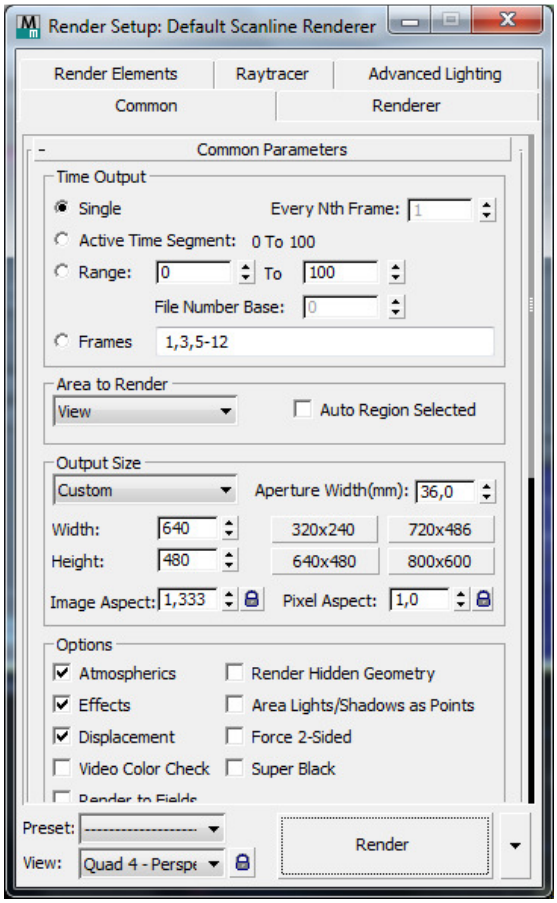
Εικόνα 18 – Το περιβάλλον εργασίας του Autodesk 3DS Max 2013.

Τα αποτελέσματα της δουλειάς μπορούν να τίθενται σε προεπισκόπηση, προκειμένου να αντιλαμβανόμαστε την πορεία της ανάπτυξης του μοντέλου. Αυτό επιτυγχάνεται από την διαδρομή Rendering →Render Setup…

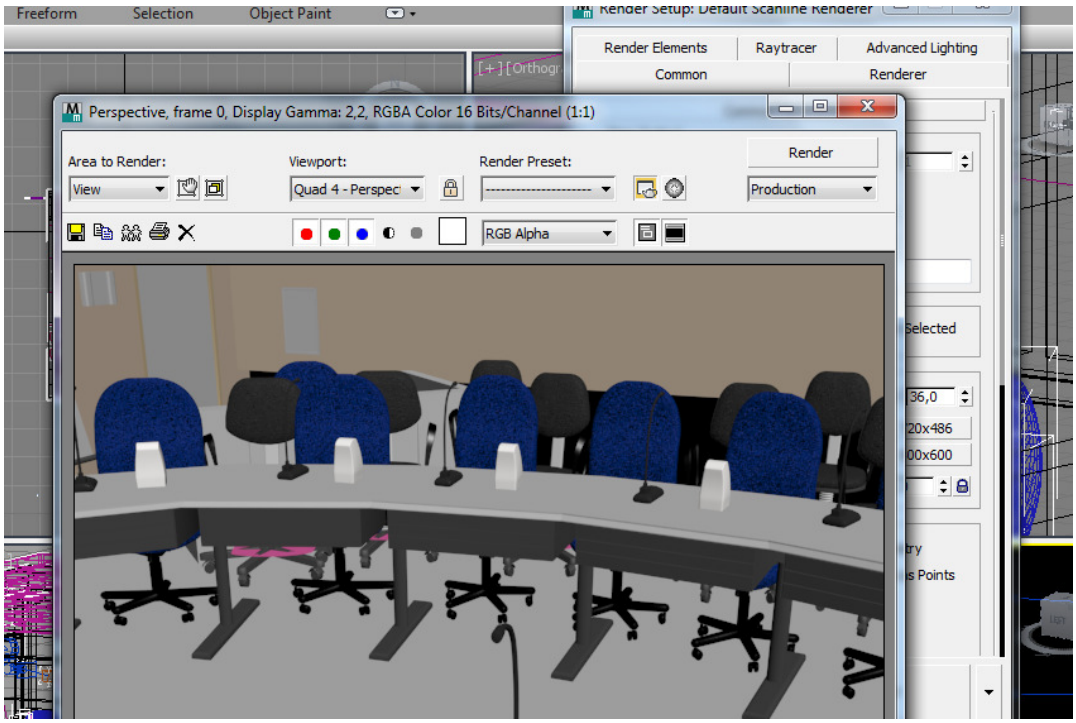


Εικόνα 19 – Διαδικασία rendering του μοντέλου (Βήμα 1ο).

Εμφανίζεται το παράθυρο ρύθμισης των παραμέτρων του rendering. Στην συνέχεια, πατάμε render και εμφανίζεται το αποτέλεσμα στην επόμενη εικόνα.



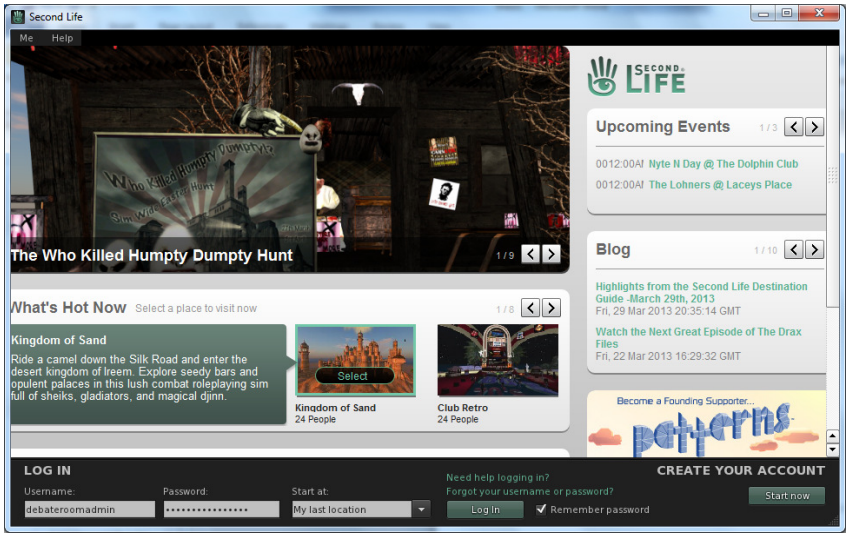
Εικόνα 20 – Διαδικασία rendering του μοντέλου (Βήμα 2ο).



Εικόνα 21 – Rendering του μοντέλου.

5.7.2 Second Life

Όπως παρουσιάστηκε και στο σχετικό κεφάλαιο, το λογισμικό Second Life (συντομογραφικά SL) είναι ένα λογισμικό εικονικού κόσμου που λειτουργεί σε δικτυακό περιβάλλον. Διανέμει μια δωρεάν εφαρμογή πρόσβασης στον εικονικό κόσμο (viewer).



Εικόνα 22 – Εισαγωγή στο παιχνίδι από την επίσημη εφαρμογή viewer.

Η εφαρμογή αυτή εγκαθίσταται σε λειτουργικά συστήματα Microsoft Windows, Linux κλπ. Και στην παρακάτω εικόνα μπορούμε να δούμε ένα στιγμιότυπο από την έναρξη του, όπου ο χρήστης πρέπει να εισάγει το username και τον κωδικό πρόσβασης του.

Κεφάλαιο 6 Υλοποίηση του Εικονικού Περιβάλλοντος

Στο έκτο κεφάλαιο, παρουσιάζεται η λειτουργία και τα χαρακτηριστικά του συστήματος που αναπτύχθηκε.

6.1 Γενική Περιγραφή

Όπως περιγράφηκε στο προηγούμενο κεφάλαιο, η εφαρμογή εικονικής διαβούλευσης πραγματοποιείται σε έναν εικονικό, τρισδιάστατο χώρο, στον οποίον προσέρχονται οι χρήστες σε προκαθορισμένες ημέρες και ώρες στις οποίες έχουν προγραμματιστεί οι εικονικές διαβουλεύσεις (debating events). Κάθε χρήστης επιτελεί έναν ρόλο που συμπεριλαμβάνει μια βαθμίδα συμμετοχής στην διαδικασία της διαβούλευσης και εκτείνεται από απλή παρακολούθηση μέχρι διαχείριση της διαδικασίας.

Τα debates λαμβάνουν χώρα σε προκαθορισμένες μέρες/ώρες που θα ορίζουν οι admins και με θέμα που επίσης εκείνοι θα ορίζουν. Οι χρήστες (με τα avatars τους) προσέρχονται στο δωμάτιο, θα κάθονται στις θέσεις τους και η διαβούλευση θα ξεκινάει. Θα υπάρχουν δύο πλευρές σε κάθε debate (υπέρ-κατά) και ένας προκαθορισμένος αριθμός κύκλων τοποθετήσεων.

Οι τοποθετήσεις θα γίνονται εναλλάξ, για κάθε πλευρά, με χρονική διάρκεια (τα σχετικά Linden scripts παρατίθενται στα παραρτήματα Β2 και Β3). Οι τοποθετήσεις γίνονται μέσω του συνηθισμένου chat του second life. Στο τέλος της διαδικασίας, οι θεατές ψηφίζουν για να αναδείξουν την νικήτρια πλευρά.

6.2 Τεχνικά Χαρακτηριστικά

Στην παρούσα εργασία δημιουργήθηκε ένα εικονικό περιβάλλον όπου μπορούν να πραγματοποιούνται εικονικές διαβουλεύσεις. Τα λογισμικά που χρησιμοποιήθηκαν ήταν:

- Second Life.
- Autodesk 3DS Max.

6.3 Εγκατάσταση

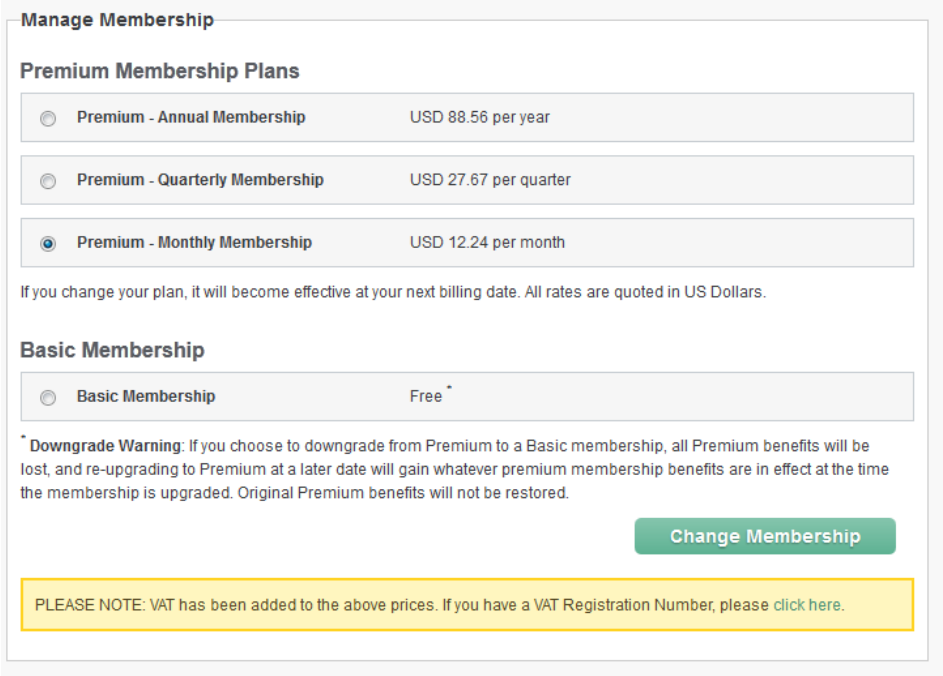
Το πρώτο βήμα για την δημιουργία του εικονικού περιβάλλοντος είναι η δημιουργία ενός χρήστη με διαχειριστικά δικαιώματα. Αυτός ο χρήστης είναι και ο πρώτος χρήστης της κατηγορίας των διαχειριστών. Η εγγραφή του νέου χρήστη, μια διαδικασία που ουσιαστικά ταυτίζεται με την δημιουργία του πρώτου avatar γίνεται δωρεάν στον διαδικτυακό τόπο https://join.secondlife.com/ .

Εικόνα 23 – Επιλογή avatar.

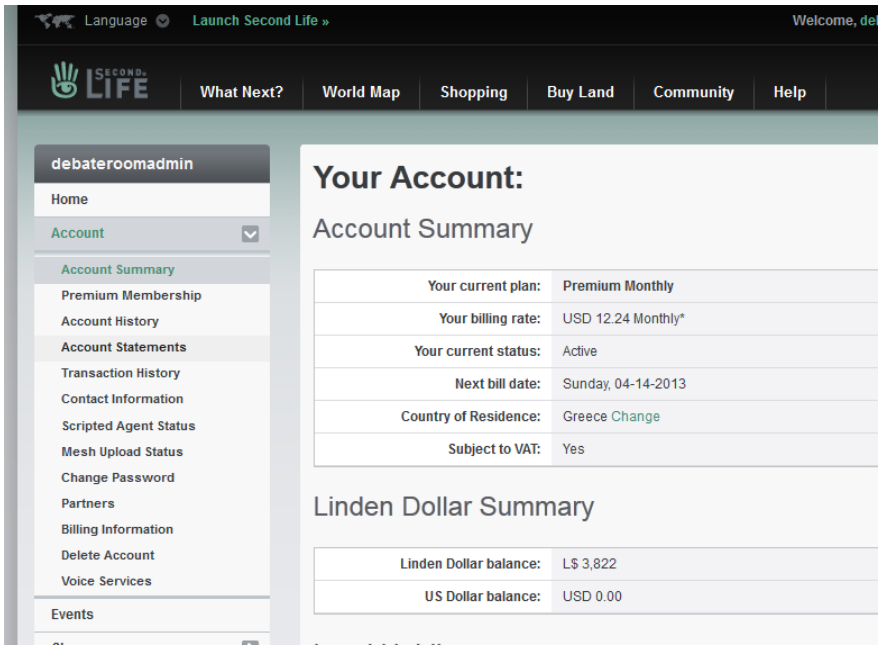
Ουσιαστικά συνίσταται στην συμπλήρωση πληροφοριών για το avatar, που πρακτικά συνάδουν και με τα προσωπικά στοιχεία του χρήστη, όπως για παράδειγμα το username, του οποίου η φόρμα εισαγωγής φαίνεται στην παρακάτω εικόνα.

Εικόνα 24 - Επιλογή ονόματος χρήστη.

Στην συνέχεια, συμπληρώνονται τα υπόλοιπα προσωπικά στοιχεία, ο τύπος του λογαριασμού. Σημειώνεται, πως για να υπάρχει η δυνατότητα δημιουργίας αντικειμένων και ανάρτησης τους στο εικονικό περιβάλλον απαιτείται ο premium λογαριασμός, που έχει κάποια, μικρή χρηματική χρέωση.

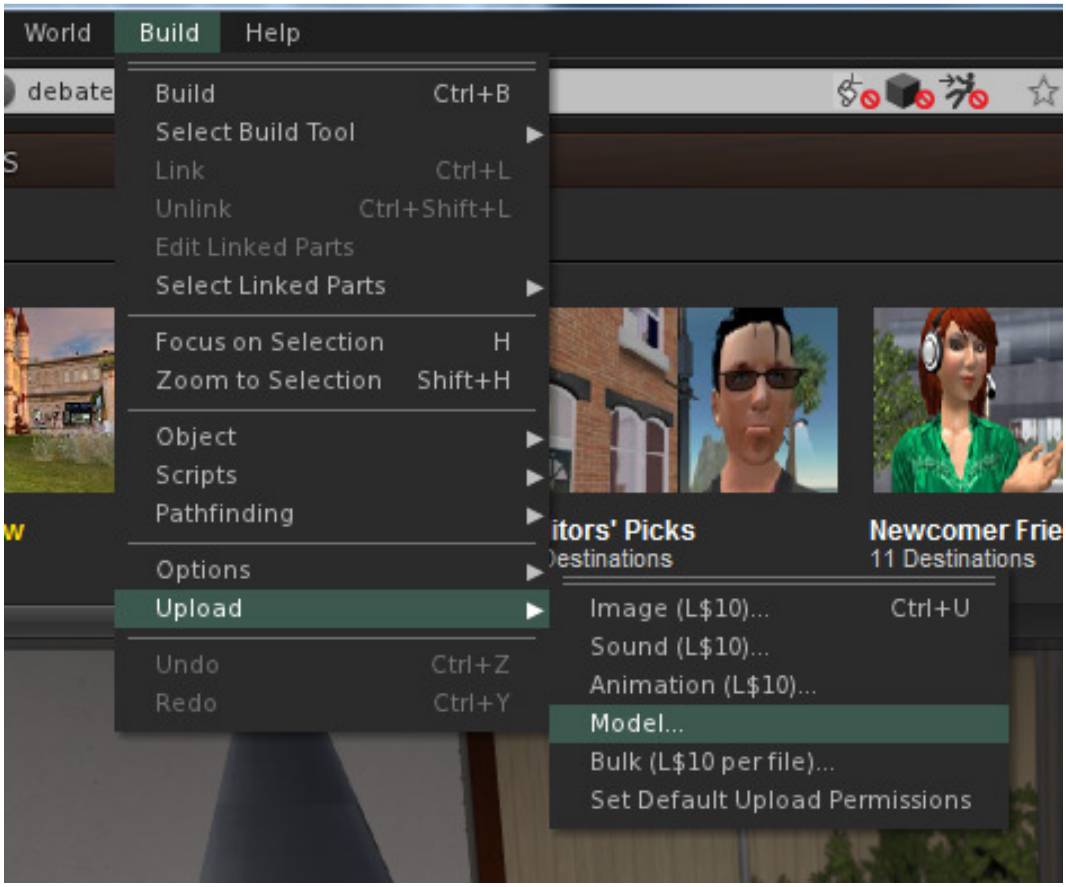


Εικόνα 25 – Επιλογή premium λογαριασμού και χρέωσης.



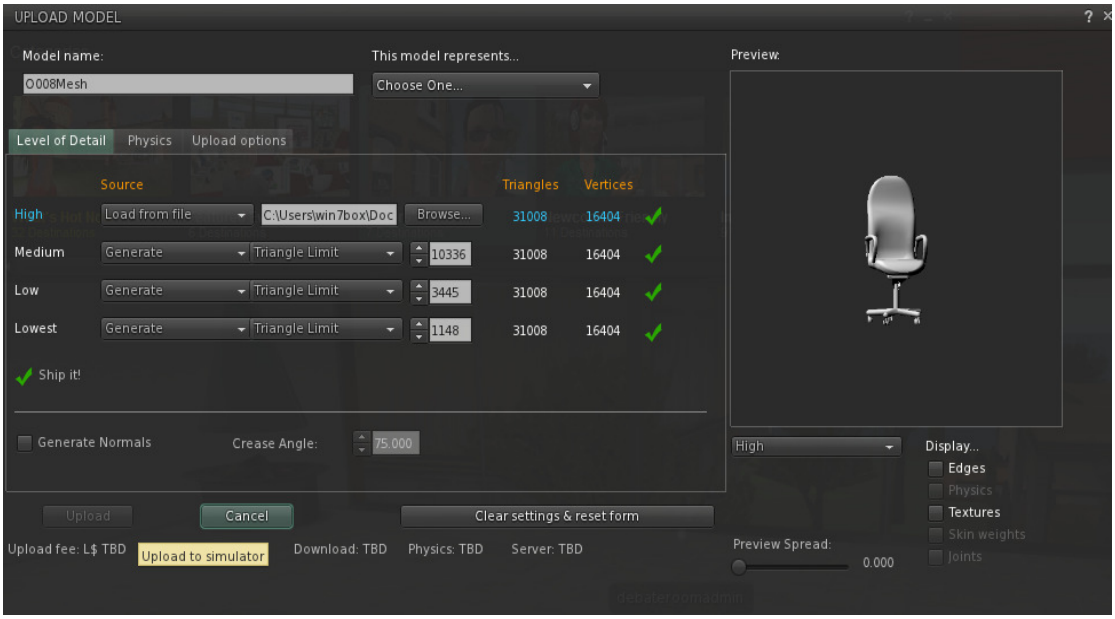
Εικόνα 26 – Δικαιώματα λογαριασμού.

Μετά την εγγραφή και την επιβεβαίωση της, ο χρήστης μπορεί να χρησιμοποιήσει την εφαρμογή viewer για να εισέλθει στον εικονικό κόσμο. Η επιλογή του premium account προσφέρει ένα αρχικό κομμάτι εικονικής γής (parcel) στον χρήστη, επί του οποίου έχει δικαίωμα να χτίζει και να τοποθετεί αντικείμενα, όπως κτίρια, έπιπλα και άλλα Στην παρακάτω εικόνα φαίνεται η διαδικασία ανάρτησης ενός τρισδιάστατου μοντέλου (build → upload →model). Εκτός από μοντέλα, ο χρήστης μπορεί να αναρτήσει και textures, από την διαδρομή build → upload →model.



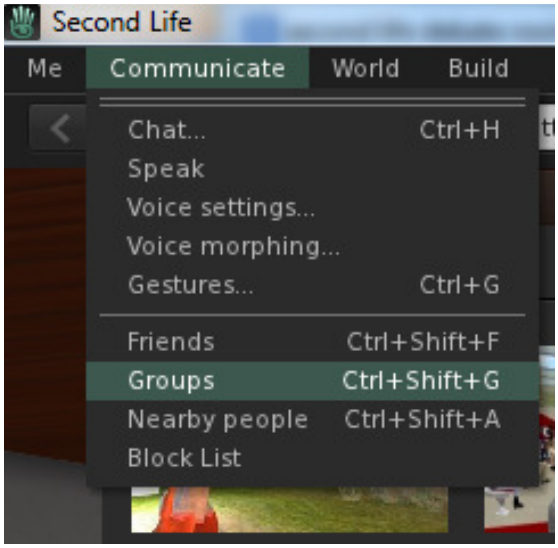
Εικόνα 27 – Ανάρτηση τρισδιάστατου μοντέλου (Βήμα 1^{ο}).

Στην παρακάτω εικόνα φαίνεται το πλαίσιο διαλόγου όπου ο χρήστης μπορεί να τροποποιήσει παραμέτρους του μοντέλου, πριν αναρτηθεί στον εικονικό κόσμο και να εκτιμήσει το χρηματικό (σε linden dollars) κόστος.



Εικόνα 28 – Ανάρτηση τρισδιάστατου μοντέλου (Βήμα 2ο).

Σημειώνεται πως η ανάρτηση αντικειμένων μπορεί να είναι πολύ ακριβή για τον χρήστη εάν το μοντέλο είναι μεγάλο και πολύπλοκό όπως στην προκειμένη εργασία. Για αυτό το λόγο, χρησιμοποιήθηκε το MaxScript του παραρτήματος Β1, για τον τεμαχισμό του συνολικού μοντέλου σε επιμέρους, πολλά και μικρά σε μέγεθος μοντέλα. Στην συνέχεια δημιουργήθηκε ένα group χρηστών το οποίο θα είναι το σύνολο χρηστών το οποίο θα συμμετέχει στις διαβουλεύσεις που θα διοργανώνονται. Αυτό επιτυγχάνεται από την επιλογή Communicate →Groups.

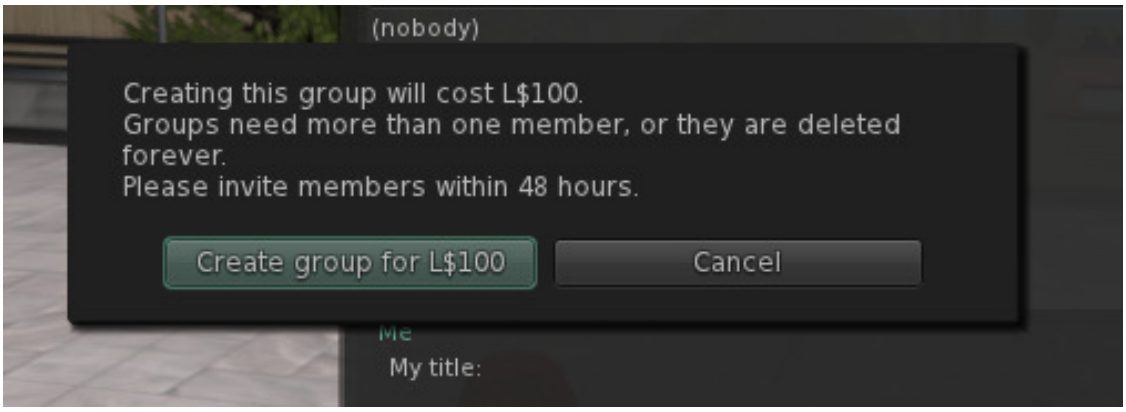


Εικόνα 29 – Δημιουργία ομάδας εικονικών διαβουλεύσεων (Βήμα 1ο).

Στο παρακάτω πλαίσιο διαλόγου επιλέγουμε «New Group».

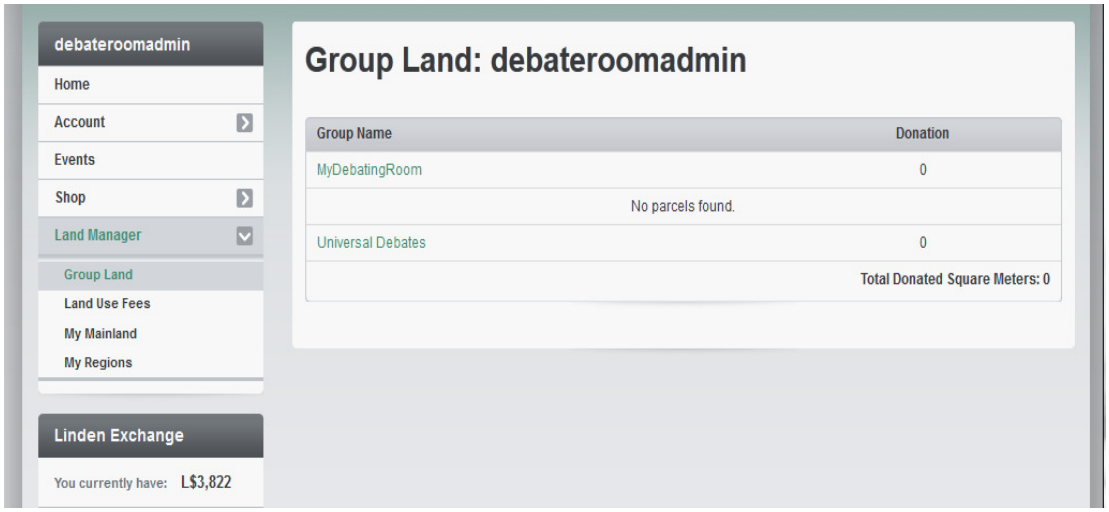
Εικόνα 30 - Δημιουργία ομάδας εικονικών διαβουλεύσεων (Βήμα 2ο).

Ο χρήστης προειδοποιείται ότι η δημιουργία του group θα στοιχίσει ένα χρηματικό ποσό, εκφρασμένο και αυτό σε linden dollars. Επίσης, συστήνεται η πρόσκληση χρηστών έτσι ώστε να μην θεωρηθεί ανενεργό από το Second Life και να διαγραφεί αυτόματα.



Εικόνα 31 - Δημιουργία ομάδας εικονικών διαβουλεύσεων (Βήμα 3ο).

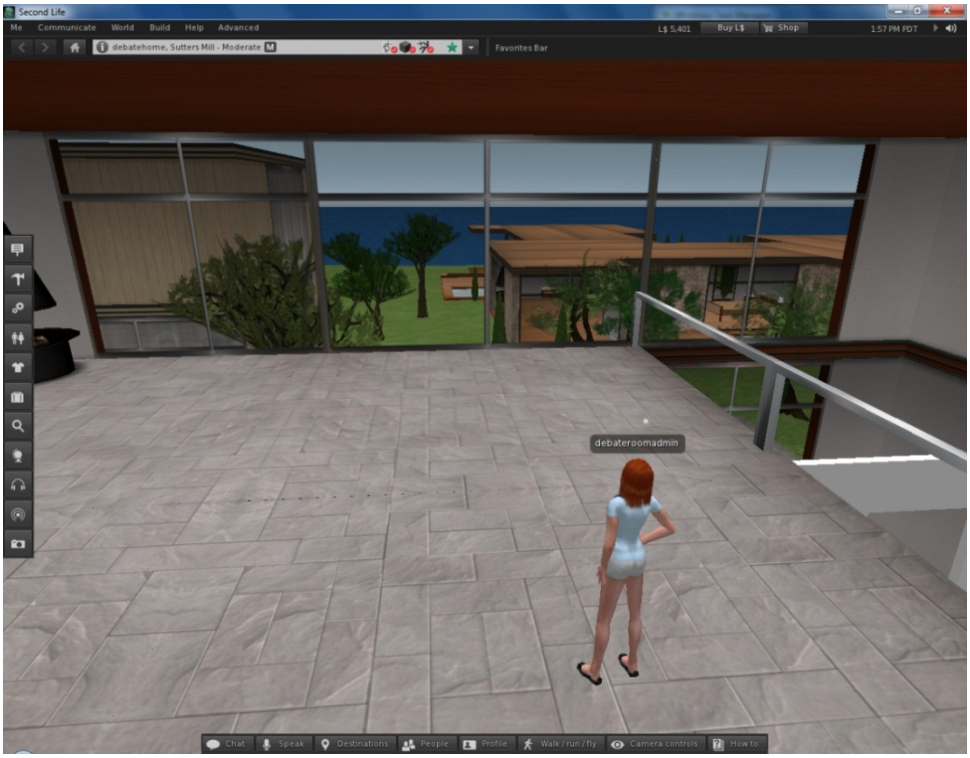
Στην παρακάτω εικόνα φαίνεται από τον λογαριασμό χρήστη η ταυτότητα του group διαβούλευσης. Εκεί φαίνονται τα μέλη, η συνεισφορά τους σε γη και τα δικαιώματα του καθενός.



Εικόνα 32 – Ταυτότητα group.

6.4 Χρήση

Για την χρήση και την διαχείριση του εικονικού περιβάλλοντος χρησιμοποιήθηκε η επίσημη εφαρμογή χρήστη (viewer) του Second Life. Στην παρακάτω εικόνα φαίνεται το avatar του χρήστη πως πλοηγείται στο κτίριο που στεγάζει τον χώρο διαβούλευσης.

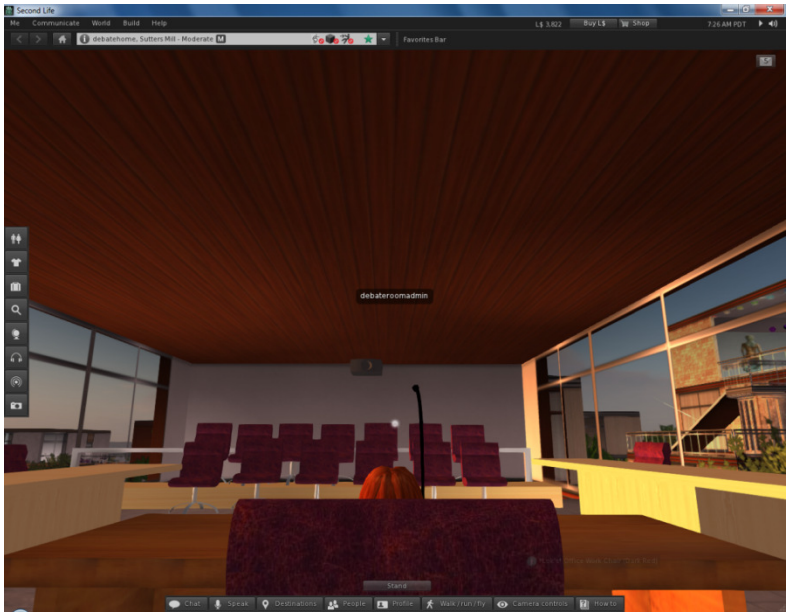


Εικόνα 33 – Το εικονικό κτίριο.

Στα παρακάτω στιγμιότυπα μπορούμε να δούμε τον εικονικό χώρο διαβούλευσης μετά την τοποθέτηση εικονικών αντικειμένων.

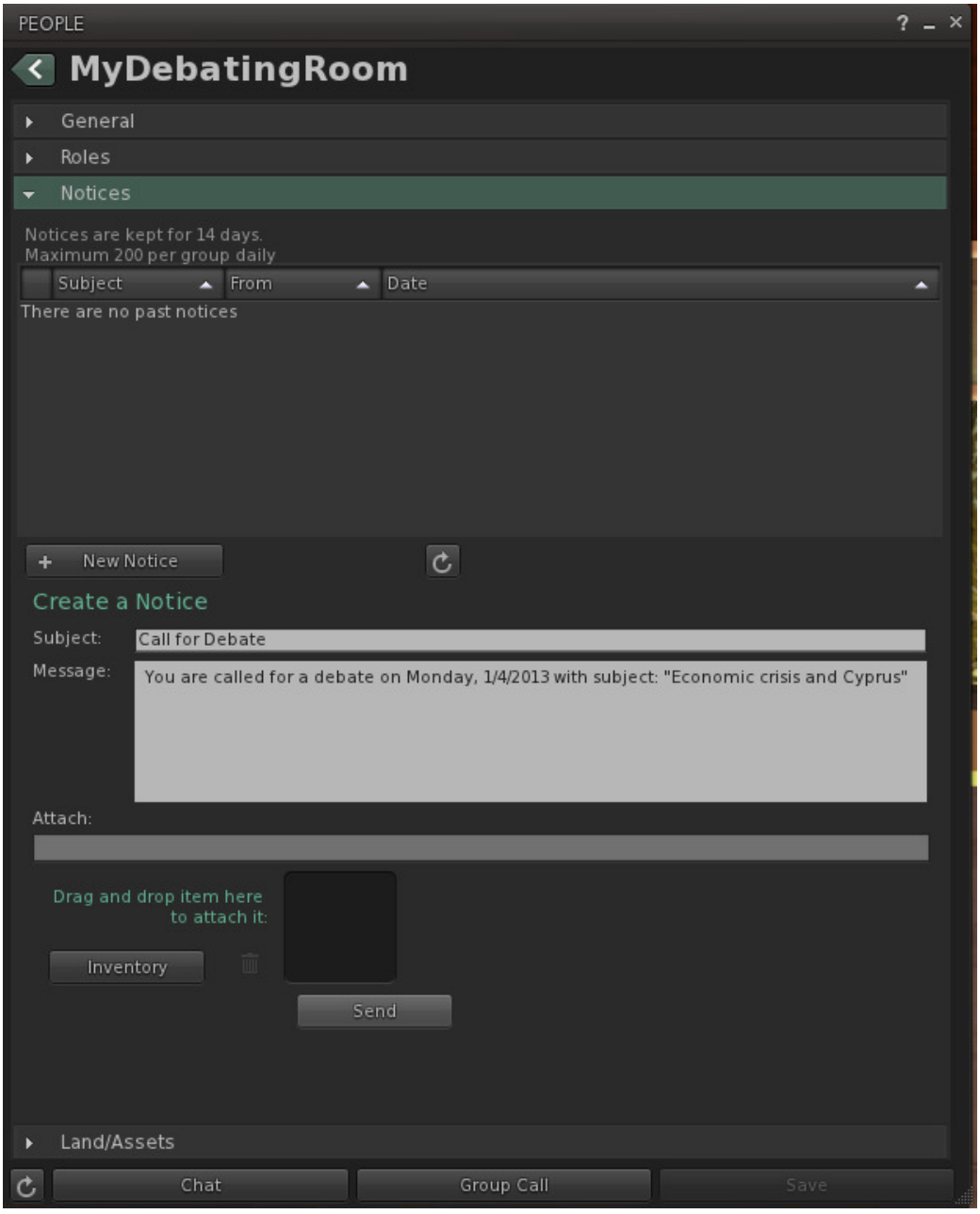
Εικόνα 34 – Αίθουσα διαβουλεύσεων (1).

Σημειώνεται πως το υψηλό κόστος ανάρτησης ολόκληρου του μοντέλου μας υποχρέωσε να αναρτήσουμε μόνο τα πιο βασικά, επιμέρους μοντέλα που ήταν απαραίτητα για την διαμόρφωση του εικονικού χώρου.



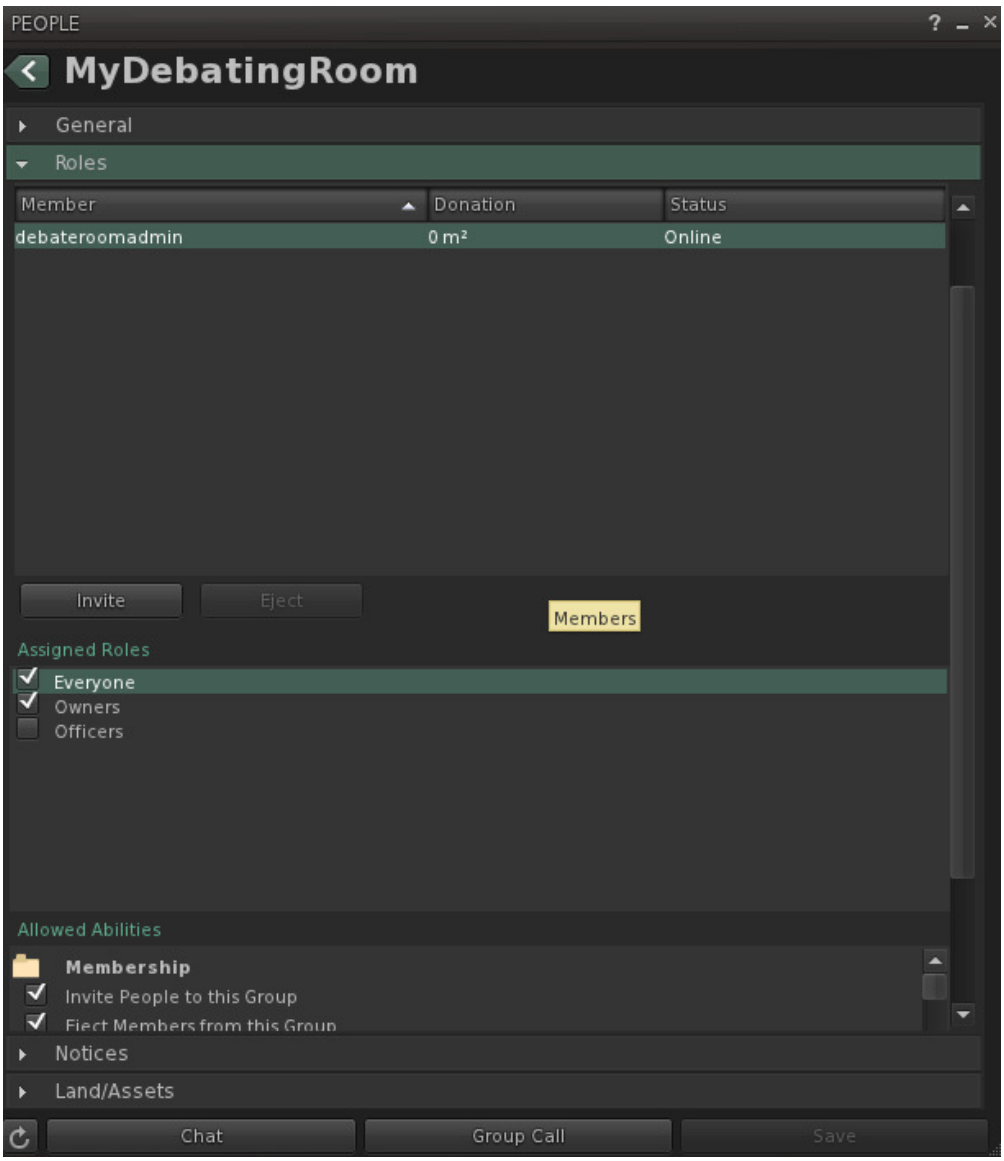
Εικόνα 35 – Αίθουσα διαβουλεύσεων (2).

Στην επόμενη εικόνα φαίνεται η αποστολή μιας ειδοποίησης (notice) στους χρήστες του group όπου ενημερώνει για την διεξαγωγή μιας νέας διαβούλευσης.



Εικόνα 36 – Αποστολή πρόσκλησης (notice) για συνεδρία διαβούλευσης.

Στην επόμενη εικόνα φαίνεται το παράθυρο επεξεργασίας των μελών του group διαβούλευσης και των δικαιωμάτων τους. Εκεί μπορούμε να χωρίσουμε τους χρήστες στις κατηγορίες που προαναφέρθηκαν.



Εικόνα 37 – Διαχείριση μελών.

Εικόνα 38 – Voice Chat.

Τελικά Συμπεράσματα

Η ανάπτυξη των Τεχνολογιών Πληροφορικής και Επικοινωνιών έδωσε ώθηση και νέες πτυχές σε κάθε επιστημονική, οικονομική, πολιτική και κοινωνική δραστηριότητα. Μια από τις κυριότερες πτυχές είναι η επικοινωνία και η συνεργατικότητα μεταξύ των εμπλεκόμενων μερών κάθε τέτοιας δραστηριότητας, αυξάνοντας έτσι και την ανάγκη για ανταλλαγή απόψεων και λήψη αποφάσεων.

Τα συστήματα ηλεκτρονικής συμμετοχής, τα οποία βασίζονται στις έννοιες του διαλόγου και της διαβούλευσης, αποτελούν μια από τις σημαντικότερες λύσεις για την γεφύρωση του χάσματος μεταξύ του πολίτη και του Κράτους, των συμμετεχόντων μεταξύ τους και γενικότερα ανάμεσα σε κάθε άλλο φυσικό ή νομικό φορέα.

Στην παρούσα εργασία, μελετήθηκαν η έννοια και τα χαρακτηριστικά του διαλόγου (debate), οι τεχνολογίες Εικονικής Πραγματικότητας (Virtual Reality), τεχνολογικά εργαλεία που είναι ήδη σε χρήση.

Σχεδιάστηκε και υλοποιήθηκε ένα σύστημα δημόσιας διαβούλευσης με την χρήση σύγχρονων τεχνολογιών ανάπτυξης και συγκεκριμένα, το Autodesk 3DS Max 2013 και του Second Life. Το πρώτο χρησιμοποιήθηκε για την δημιουργία των αντικειμένων που απαρτίζουν έναν χώρο διαβουλεύσεων και το Second Life επιλέχθηκε σαν η πλατφόρμα φιλοξενίας του εικονικού χώρου.

Μελλοντική εργασία

Ο εικονικός χώρος που αναπτύχθηκε μπορεί να επεκταθεί περαιτέρω ώστε να προσφέρει μια ακόμα πιο πλήρη και πιο οργανωμένη εμπειρία χρήσης. Πιο συγκεκριμένα, μπορεί να εμπλουτιστεί με περισσότερα τρισδιάστατα αντικείμενα και να επεκταθεί ώστε να επιτρέπει την προσέλευση και εισδοχή ακόμα περισσότερων χρηστών.

Μπορεί να διαφημιστεί ώστε να προσελκύσει χρήστες και να αποκτήσει πιο «σφιχτά χαρακτηριστικά» όσον αφορά την διαχείριση με την αυτοματοποίηση εργασιών που γίνονται χειρωνακτικά από τους διαχειριστές. Τέλος, μπορούν να ενσωματωθούν ακόμα περισσότερες παράμετροι, όπως διαφοροποιημένοι κανόνες ανά group χρηστών.

Βιβλιογραφία

[Snider, Lawrence, 2009] Alfred Snider, Edwin W. Lawrence, Nothing Virtual About the Arguments: using new media to enable debating, National Meeting, National Federation of High Schools, Buffalo, New York, 6 August 2009.

[Yuan et al., 2004] Yuan, T., Moore, D. and Grierson, A. (2004). An Assessment of Dialogue Strategies for a Human Computer Debating System, via Computational Agents. In Proceedings of ECAI'2004 Workshop on Computational Models of Natural Argument, pp.17-24 Valencia Spain.

[Cassell, Vilhjálmsson, 1999] Cassell, J. and Vilhjálmsson, H. (1999). "Fully Embodied Conversational Avatars: Making Communicative Behaviors Autonomous." *Autonomous Agents and Multi-Agent Systems* 2(1): 45-64.

[Karakapilidis, Papadias, 2001] Karacapilidis, N. I. & Papadias, D. (2001), 'Computer supported argumentation and collaborative decision making: the HERMES system.', Inf. Syst. 26 (4) , 259-277 .

[Dryzek, 1990]J. Dryzek, «Deliberative Democracy and Beyond: Liberals, Critics, Contestations», Oxford University Press, 2000 και Gutmann A. και D. Thompson, «Democracy and Disagreement. Why moral conflict cannot be avoided in politics and what should be done about it», Harvard University Press, 1990.

[Χριστοπούλου,2011] Χριστοπούλου Σωτηρία, Δημόσια διαβούλευση μέσω του Διαδικτύου. Η περίπτωση της δικτυακής πλατφόρμας opengov.gr, Τελική Εργασία, 2011.

[Pečiva,2007] Pečiva, J. 2007. Active Transactions in Collaborative Virtual Environments. PhD Thesis, Brno, Czech Republic, FIT VUT, ISBN 978-80-214-3549-0.

[Tramberend,2003] Tramberend H. Avocado : a Distributed Virtual Environment framework. Bielefeld (Germany): Bielefeld University; 2003.

Παράρτημα Α: διευθύνεις Internet

1) http://www.hitl.washington.edu/research/knowledge_base/distvr/
2) http://www.convinceme.net/
3) http://cve.sourceforge.net/

Παράρτημα Β: Κώδικας Συστήματος και προγραμμάτων εγκατάστασης

Β1. Κώδικας τεμαχισμού του 3DS MAX μοντέλου στα επιμέρους αντικείμενα

```
escapeenable = true
rollout ExportObj "OS3D.se Mass export Selected objects v1.5"
width:341 height:262
(

	global theClasses
	global outPath
	local debug = false
	edittext edtPrefixText "Prefix" pos:[25,79] width:226
height:16 readOnly:false
	button btnStartExport "Start Export" pos:[248,172] width:90
height:86
	checkbox chkAddPrefix "" pos:[7,78] width:15 height:16
checked:true toolTip:"if checked the Prefix is added to the
filename"
	dropdownList ddl2 "FileType" pos:[7,6] width:244 height:40
	checkbox chkDebug "Debug" pos:[264,105] width:56 height:16
checked:false toolTip:"This will show error messages"
	spinner spnNrOfDigits "Digits" pos:[110,104] width:57
height:16 range:[0,100,7] type:#integer
	checkbox chkAddZeros "Add Zeros" pos:[7,104] width:80
height:16 checked:true
	edittext edtFrameRange "Frame Range" pos:[8,55] width:243
height:16 readOnly:false
	spinner spnFrameByStep "By Step" pos:[285,55] width:49
height:16 range:[0,1000,1]
	label lblStatus "Info window" pos:[9,179] width:231 height:70
	GroupBox grp1 "Info" pos:[2,165] width:244 height:93
	spinner spnOffset "Offset" pos:[192,104] width:59 height:16
range:[-10000,10000,0] type:#integer
	button btn2 "Browse output" pos:[134,136] width:204 height:24
toolTip:"Output folder"
	checkbox chkSingleMesh "SingleMesh" pos:[264,80] width:72
height:16 checked:false toolTip:"This combines all objects in a
group into one single mesh... can be useful in some cases"
	button btnResetIniFile "R" pos:[312,1] width:26 height:21
toolTip:"This resets the parameters to the default values, there is
no undo for this..."
	button btnSelectVisibleObjects "Select Visible geometry"
pos:[7,136] width:120 height:24 toolTip:"This just helps you select
all geometry that are currently visible in the viewport.."
	checkbox chkAddFrame "add Frame" pos:[264,28] width:72
height:16 checked:true toolTip:"if this is checked the scripts
appends the frame to the object name, useful when exporting ranges"
	-------Start function definitioin
	fn CheckIfFloatIsInt invalue =
	(
	inval = invalue as float
	tempval = filterstring (inval as string) "."
	if tempval[2] == "0" then return true else return false
```

```
	)

	fn OS3DGetRanges OS3DRange OS3DByStep =
	(
	--usage: FrameRange = OS3DGetRanges "10-20,31,35,36" 1.0
	--OS3DByStep = 1.1
	byStep = (if CheckIfFloatIsInt (OS3DByStep as float) then
(OS3DByStep as integer) else (OS3DByStep as float))
	FrameArray = #()
	print "---start range processing---"
	inputString = filterString OS3DRange ","
splitEmptytokens:false

	for str in inputstring do
		(
		if (matchpattern str pattern:"*-*")
			then
			(
				print "Found a range"
				tmpString = filterString str "-"
splitEmptytokens:false
				OS3DStartFrame = tmpString[1] as float
				OS3DEndFrame = tmpString[2] as float

				for i = OS3DStartFrame to OS3DEndFrame by
ByStep do --collect (i)
				(
					if CheckIfFloatIsInt (OS3DByStep as
float)
					then
						(
						appendIfUnique FrameArray (i as
integer)
						)
					else
						(
						appendIfUnique FrameArray (i as
float)
						)
					)

			) --end then

			else
			(
				print "Found single frame"
				if CheckIfFloatIsInt (OS3DByStep as float)
					then
						(
						appendIfUnique FrameArray (str
as integer)
						)
					else
						(
						appendIfUnique FrameArray (str
as float)
						)

			) -- end else
```

```
		)
	--sort FrameArray
	return FrameArray
	)

	fn OS3DFileName inputPath objName UsePrefix Prefix
UseZeroPadding NrDigits FrmNr FrmOffset AddFrameNumber =
	(
		--usage: OS3DFileName $.name true "myPrefix" true 5 17.4
		NewName =  inputPath + "\\"
		FrameNr = (FrmOffset + FrmNr) as string
		FrameNameArray = FilterString FrameNr "." --replace the
"." in a float value
		if FrameNameArray.count == 2 then
		(
			FrameNr = FrameNameArray[1] + "_" +
FrameNameArray[2]
		)


		if UsePrefix then
		(
		NewName += (Prefix + "-")
		)
		NewName += objName

		if AddFrameNumber then (
		NewName += "-"
		)

		if AddFrameNumber  then (
			if UseZeroPadding then
			(
				ZeroNumber = ""
				fr = FrameNr as string
				AddNrZeros = NrDigits - fr.count
				for i = 1 to AddNrZeros do
				(
				ZeroNumber += "0"
				)
				ZeroNumber += fr
				NewName += ("Frame_"    + ZeroNumber as
string)
			)
			else
			(
			NewName += ("Frame_"    + FrameNr as string)
			)
		) --end use frame number

		NewName
	)
	fn OS3D_isGroupMasterHead item =
	(
		objArray = #()

		ab_gh = isGroupHead item
		ab_gp = item.parent == undefined
		if ab_gh and ab_gp do (
```

```
                print ("the master is:" + item.name)
                join objArray item.children
                append objArray item

                )
        return objArray
        )

              fn OS3D_item_name itemArray =
        (
              for item in itemArray do
              (
              belongToGroup = false
              ab_gh = isGroupHead item
              ab_gp = item.parent == undefined

                    if ab_gh and ab_gp do (
                    belongToGroup = true
                    return item.name
                    )

              )
              return ""


        )



        fn NewfileName inputPath objName ischecked nr=
        (

        if ischecked == true then
        (
        --set the filename to this prefix
              np =  inputPath + "\\" + objName + "_"+ nr as string --
+".obj"
              if debug then print np
        )

        else
        (
        --set the filename to the object filename
              np = inputPath + "\\" + objName --e+ ".obj"
              if debug then print np
        )
        return np
        )


fn SnapshotSelectionIntoMesh selObjs theTime =
(
                  snaphostArray = for o in selObjs where
(superclassof o == GeometryClass) collect
                  (
                        with animate on
                        (
                        --slidertime = theTime
                        oldparent = o.parent
```

```
                        o.parent = undefined
                        MeshObj = at time slidertime snapshot o
                        o.parent = oldparent
                        )
                        MeshObj
                  )

                  if snaphostArray.count > 1 then
                  (
                   for m = 2 to snaphostArray.count do (attach
snaphostArray[1] snaphostArray[m])
                  )
      return snaphostArray[1]
)

      fn OS3DDoExport SelOnly ShowPrompt fname asSingleMesh theTime=
      (
            --usage OS3DDoExport true true
            result = false
            if ShowPrompt then
            (
                  print fname
                  if asSingleMesh then (

                        selObjs = (selection as array)
                        --thetime = slidertime
                        SingleObj = SnapshotSelectionIntoMesh
selObjs theTime
                        select SingleObj
                        result = (exportFile fname
using:(theClasses[ddl2.selection]) selectedOnly: SelOnly)
                        delete SingleObj
                  )
                  else
                  (
                        result = (exportFile fname
using:(theClasses[ddl2.selection]) selectedOnly: SelOnly)
                  )
            ) --end showprompt = yes

            else --if promt = no
            (
                  print fname
                  if asSingleMesh then (
                        selObjs = (selection as array)
                        --thetime = slidertime
                        SingleObj = SnapshotSelectionIntoMesh
selObjs theTime
                        print selObjs
                        select SingleObj
                        result = (exportFile fname #noPrompt
using:(theClasses[ddl2.selection]) selectedOnly: SelOnly)
                        delete SingleObj
                  )

                  else (
                        result = (exportFile fname #noPrompt
using:(theClasses[ddl2.selection]) selectedOnly: SelOnly)
                  )
```

```
            )--- end prompt = no

            if result != true then
            (
            if chkDebug.state == true then (messageBox "OS3D debug
INFO:\nYou have a selected a filformat that doesn´t except the
selected only state..")
            result = (exportFile fname
using:(theClasses[ddl2.selection]) selectedOnly: false)
            )
      )

      fn StatusText =
      (
            try
            (
            TimeRange = OS3DGetRanges edtFrameRange.text
spnFrameByStep.value
            lblStatus.caption = ("Number of frames that will be
exported: " + TimeRange.count as string + "\n")
            objsArray = (selection as array)
            fname = OS3DFileName "" objsArray[1].name
chkAddPrefix.state edtPrefixText.text chkAddZeros.state
spnNrOfDigits.value TimeRange[TimeRange.count] spnOffset.value
chkAddFrame.state
            --fname = OS3DFileName "" "ExampleName" true  "prefix"
true 7 42.0
            lblStatus.caption += ("Example Filename for current
selection:\n" + fname)
            )
            catch(if chkDebug.state do (print "Status update did not
work"))
      )

      struct OS3D_iniFunctions (
            fn iniFileName = ("os3d_massExportObj.ini" ),

            fn OS3D_SetIniSetting iniSection iniKey iniVal = (
            --creates a new setting and stores it in the inifile
under the users local setting
            iniFolder =       getdir #plugcfg
            iniFilePath = ( inifolder +
"\\"+OS3D_iniFunctions.iniFileName())
            returnVal = setINISetting iniFilePath iniSection iniKey
(iniVal as string)
            if debug then (
                  if returnVal then (
                              format "\n\n==OS3D ini store\nValue:
\n  %\nwas written to key: \n  % \nand stored in ini file:\n  %\n"
iniVal iniKey iniFilePath
                        ) else (
                              format "FAILED TO WRITE - Value: % was
written to key: % and stored in ini file:\n%" iniVal iniKey
iniFilePath
                        )
            )
            ),

            fn OS3D_GetIniSetting iniSection iniKey = (
```

```
            --gets a  setting
            iniFolder =       getdir #plugcfg
            iniFilePath = ( inifolder +
"\\"+OS3D_iniFunctions.iniFileName())

                  if (hasINISetting iniFilePath iniSection iniKey)
then (
                        returnVal = getINISetting iniFilePath
iniSection iniKey

                        if debug then (
                              if returnVal != "" then (
                                    format "\n\n==OS3D ini
load\nValue: \n  %\nwas loaded from key: \n  % \n stored in ini
file:\n  %\n" returnVal iniKey iniFilePath
                              ) else (
                                    format "got empty value: % could
be the correct key: % that is stored in ini file:\n%\ncould also
mean that the inifile does not exist or this user has no access to
it" returnVal iniKey iniFilePath
                              )
                        )
                        returnVal
                  ) else (
                        format "ini file does not have this key: %
set.." iniKey
                        returnVal = ""
                  )

            ),

            fn OS3D_RemoveIniFile = (
                  iniFolder =       getdir #plugcfg
                  iniFilePath = ( inifolder +
"\\"+OS3D_iniFunctions.iniFileName())
                  deletefile iniFilePath
            ),

            fn OS3D_ExistIniFile = (
                  iniFolder =       getdir #plugcfg
                  iniFilePath = ( inifolder +
"\\"+OS3D_iniFunctions.iniFileName())
                  (getFiles iniFilePath).count > 0
            )
      )

      fn os3d_createDefaultSettings = (

            OS3D_iniFunctions.OS3D_SetIniSetting "Parameters"
"FrameRange" "1-10,13,16,30-35"
            OS3D_iniFunctions.OS3D_SetIniSetting "Parameters"
"PrefixText" "MyPrefixText"
            OS3D_iniFunctions.OS3D_SetIniSetting "Parameters"
"PrefixUse" "true"
            OS3D_iniFunctions.OS3D_SetIniSetting "Parameters"
"addZeros" "true"
            OS3D_iniFunctions.OS3D_SetIniSetting "Parameters"
"Digits" "7"
```

```
            OS3D_iniFunctions.OS3D_SetIniSetting "Parameters"
"Offset" "0"
            OS3D_iniFunctions.OS3D_SetIniSetting "Parameters"
"ByStep" "1"
            OS3D_iniFunctions.OS3D_SetIniSetting "Parameters"
"SingleMesh" "false"
            OS3D_iniFunctions.OS3D_SetIniSetting "Parameters"
"debug" "false"
            OS3D_iniFunctions.OS3D_SetIniSetting "Parameters"
"selectedPlugin" "1"

            OS3D_iniFunctions.OS3D_SetIniSetting "FilePaths"
"outputFolder" "c:\\temp\\"

            --"true" as booleanClass
      )

      fn OS3D_LoadIniSettings DialogWindowInstance = (
            DialogWindowInstance.edtFrameRange.text =
(OS3D_iniFunctions.OS3D_GetIniSetting "Parameters" "FrameRange")
            DialogWindowInstance.edtPrefixText.text =
(OS3D_iniFunctions.OS3D_GetIniSetting "Parameters" "PrefixText")
            DialogWindowInstance.chkAddPrefix.state =
(OS3D_iniFunctions.OS3D_GetIniSetting "Parameters" "PrefixUse") as
booleanclass
            DialogWindowInstance.ddl2.selection =
(OS3D_iniFunctions.OS3D_GetIniSetting "Parameters" "selectedPlugin")
as integer
            DialogWindowInstance.chkDebug.state =
(OS3D_iniFunctions.OS3D_GetIniSetting "Parameters" "debug") as
booleanclass
            DialogWindowInstance.spnNrOfDigits.value =
(OS3D_iniFunctions.OS3D_GetIniSetting "Parameters" "Digits") as
integer
            DialogWindowInstance.chkAddZeros.state =
(OS3D_iniFunctions.OS3D_GetIniSetting "Parameters" "addZeros") as
booleanClass
            DialogWindowInstance.spnFrameByStep.value =
(OS3D_iniFunctions.OS3D_GetIniSetting "Parameters" "ByStep") as
float
            DialogWindowInstance.spnOffset.value =
(OS3D_iniFunctions.OS3D_GetIniSetting "Parameters" "Offset") as
integer
            DialogWindowInstance.chkSingleMesh.state =
(OS3D_iniFunctions.OS3D_GetIniSetting "Parameters" "SingleMesh") as
booleanClass
            DialogWindowInstance.chkAddFrame.state =
(OS3D_iniFunctions.OS3D_GetIniSetting "Parameters" "AddFrameNumber")
as booleanClass

            DialogWindowInstance.btn2.caption = outPath =
OS3D_iniFunctions.OS3D_GetIniSetting "FilePaths" "outputFolder"

      )

      fn OS3D_SaveIniSettings DialogWindowInstance = (
            OS3D_iniFunctions.OS3D_SetIniSetting "Parameters"
"FrameRange" DialogWindowInstance.edtFrameRange.text
```

```
            OS3D_iniFunctions.OS3D_SetIniSetting "Parameters"
"PrefixText" DialogWindowInstance.edtPrefixText.text
            OS3D_iniFunctions.OS3D_SetIniSetting "Parameters"
"PrefixUse" DialogWindowInstance.chkAddPrefix.state

            OS3D_iniFunctions.OS3D_SetIniSetting "Parameters"
"selectedPlugin" DialogWindowInstance.ddl2.selection
            OS3D_iniFunctions.OS3D_SetIniSetting "Parameters"
"debug" DialogWindowInstance.chkDebug.state
            OS3D_iniFunctions.OS3D_SetIniSetting "Parameters"
"Digits" DialogWindowInstance.spnNrOfDigits.value
            OS3D_iniFunctions.OS3D_SetIniSetting "Parameters"
"addZeros" DialogWindowInstance.chkAddZeros.state
            OS3D_iniFunctions.OS3D_SetIniSetting "Parameters"
"ByStep" DialogWindowInstance.spnFrameByStep.value
            OS3D_iniFunctions.OS3D_SetIniSetting "Parameters"
"Offset" DialogWindowInstance.spnOffset.value
            OS3D_iniFunctions.OS3D_SetIniSetting "Parameters"
"SingleMesh" DialogWindowInstance.chkSingleMesh.state
            OS3D_iniFunctions.OS3D_SetIniSetting "Parameters"
"AddFrameNumber" DialogWindowInstance.chkAddFrame.state

            OS3D_iniFunctions.OS3D_SetIniSetting "FilePaths"
"outputFolder" outPath
      )



--    OS3D_iniFunctions.OS3D_SetIniSetting "Test section" "antonKey"
"tempoLama"
--    OS3D_iniFunctions.OS3D_GetIniSetting "Test section" "antonKey"

--OS3D_iniFunctions.OS3D_RemoveIniFile()
------end function definition----







      on ExportObj open do
      (
            theClasses = exporterPlugin.classes
            theClassesNames = #()
            for exptyp in theClasses do
            (
                  append theClassesNames (exptyp as string)
                  )
            ddl2.items = theClassesNames
            --Set some default values
            if (OS3D_iniFunctions.OS3D_ExistIniFile()) then (
                  try (
                        OS3D_LoadIniSettings ExportObj
                  ) catch (
                        print "could not load ini file properly,
resetting and creating a new one"
                        OS3D_iniFunctions.OS3D_RemoveIniFile()
```

```
                )
            ) else (
                OS3D_createDefaultSettings()
                OS3D_LoadIniSettings ExportObj
            )

            --edtFrameRange.text = "1-10,13,16,30-35"
            --edtPrefixText.text = "MyPrefixText"

      )
      on ExportObj close do
      (
            OS3D_SaveIniSettings ExportObj
      )
      on edtPrefixText changed val do
      (
            StatusText()
      )
      on btnStartExport pressed do
      (
            OS3D_SaveIniSettings ExportObj
            theClasses = exporterPlugin.classes
            if outPath == undefined then
            (
            outPath = ""
            outPath = getSavePath initialDir:outpath
            )

            if outPath != undefined then
            (
            selArray = selection as array
            sceneObjs =#()

            TimeRange = OS3DGetRanges edtFrameRange.text
spnFrameByStep.value

            if (animationRange.start as float) > TimeRange[1] then
            (
                  animationRange= interval TimeRange[1]
animationRange.end
            )

            if (animationRange.end ) < TimeRange[TimeRange.count]
then
            (
                  animationRange= interval animationRange.start
TimeRange[TimeRange.count]
            )
            timeDisplayMode = #frameTicks



            sceneObjs = #()

            for item in       selArray do
            (

                  if (isGroupMember item) or (isGroupHead item) then
                  (
```

```
                        temp = (OS3D_isGroupMasterHead item)
                        append sceneObjs temp

                  )
                  else --just a single obj
                  (
                        print item.name
                        append sceneObjs #(item)
                  )
            )
            sceneobjs = makeUniqueArray sceneobjs
            ic = 0
            for i = 1 to sceneObjs.count do
            (
                  --oname = (if chkAddPrefix.state then
edtPrefixText.text else sceneObjs[i].name)

                  if sceneObjs[i].count != 0 then
                  (
                        ic += 1
                        if ic == 1 then --the first object in the
selection will display the promt to set the settings...
                        (

                        showPrompt = true
                              for f in TimeRange do (
                                    if showPrompt  then
                                    (
                                          select sceneObjs[i]
                                          ExpObjName =
OS3D_item_name sceneObjs[i]
                                          if ExpobjName == "" then
ExpobjName = sceneObjs[i][1].name
                                          NewName = OS3DFileName
outPath ExpobjName chkAddPrefix.state edtPrefixText.text
chkAddZeros.state spnNrOfDigits.value f spnOffset.value
chkAddFrame.state
                                          slidertime = (f as time)
                                          OS3DDoExport true true
NewName chkSingleMesh.checked slidertime
                                          showPrompt = false

                                    )else
                                    (
                                          select sceneObjs[i]
                                          ExpObjName =
OS3D_item_name sceneObjs[i]
                                          if ExpobjName == "" then
ExpobjName = sceneObjs[i][1].name
                                          NewName = OS3DFileName
outPath ExpobjName chkAddPrefix.state edtPrefixText.text
chkAddZeros.state spnNrOfDigits.value f spnOffset.value
chkAddFrame.state
                                          slidertime = (f as time)
                                          OS3DDoExport true false
NewName chkSingleMesh.checked slidertime

                                    )
```

```
                              )
                        )

                        if ic > 1 then --for every obj exept the
first try to skip the prompt...
                        (

                              for f in TimeRange do
                              (
                              select sceneObjs[i]
                              ExpObjName = OS3D_item_name
sceneObjs[i]
                              if ExpobjName == "" then ExpobjName =
sceneObjs[i][1].name
                              NewName = OS3DFileName outPath
ExpObjName chkAddPrefix.state edtPrefixText.text chkAddZeros.state
spnNrOfDigits.value f spnOffset.value chkAddFrame.state
                                    slidertime = (f as time)
                                    OS3DDoExport true false NewName
chkSingleMesh.checked slidertime

                              )
                        )
                  )
            )

            --select sceneObjs
            timeDisplayMode = #frames
      )

      else
            (
                  messagebox "No output path selected"
            )
      )---------end export
      on chkAddPrefix changed theState do
      (
            if theState then edtPrefixText.readOnly = false else
edtPrefixText.readOnly = true
            StatusText()
            OS3D_SaveIniSettings ExportObj
      )
      on chkDebug changed state do
      (
                  print ("debug is " + state as string)
                  debug = state
            )
      on spnNrOfDigits changed val do
      (
            StatusText()
            OS3D_SaveIniSettings ExportObj
      )
      on chkAddZeros changed theState do
      (
            StatusText()
            OS3D_SaveIniSettings ExportObj
      )
      on edtFrameRange changed val do
      (
```

```
            StatusText()
            OS3D_SaveIniSettings ExportObj
      )
      on spnFrameByStep changed val do
      (
            StatusText()
            OS3D_SaveIniSettings ExportObj
      )
      on spnOffset changed val do
      (
            StatusText()
            OS3D_SaveIniSettings ExportObj
      )

      on chkAddFrame changed val do (
            StatusText()
            OS3D_SaveIniSettings ExportObj
      )

      on btn2 pressed do
      (
            if outPath == undefined then
            (
            outPath = ""
            outPath = getSavePath()
            )
            else
            (
            outPath = (getSavePath initialDir:outpath)
            )

            if outPath != undefined then (
            btn2.caption = outPath
            ) else (
            btn2.caption = "click to set output folder"
            )
            StatusText()
            OS3D_SaveIniSettings ExportObj
      )
      on btnResetIniFile pressed do
      (
                  OS3D_createDefaultSettings()
                  OS3D_LoadIniSettings ExportObj
            )
      on btnSelectVisibleObjects pressed do
      (
                  select (for o in geometry where (canConvertTo o
mesh) and o.isHidden != true collect o)
            )
)

createdialog ExportObj
```

B2. Κώδικας Timer του Second Life

```
integer timerchannelspeak = 1234;// Any valid integer can be used.
Just make sure that it matches the control message.
integer timerchannellisten = 1235;
```

```
float   timeint = 1;      // Time interval. 1 = 1 second, 60 = 60
seconds, .5 = half of a second, etc. Adjust this
                          //   as needed. Remember to adjust the
number of ticks that is passed to this module via
                          //   other scripts. I.E. if this value
is set at .1 and you need the event to be reported
                          //   every second, then the value sent
should be 10.

////////////////////////////////////////////////////////////////////
//////////////////
//Don't change anything else unless you *really* need to.

////////////////////////////////////////////////////////////////////
//////////////////
//Declarations
integer ticks;               // master timer counter

list    timername;           // Name of the timer
list    timerticks;          // Duration of the timer
list    timercycles;         // How many times the timer will be
cycled before it is removed
list    timerprim;           // Prim the timer was added from
list    timerticksremaining; // How many ticks remain before the
event is reported

string  timerstate;        // Running state of the timer. 0 =
Stopped; 1 = Running. While this will be only ones
                           //   and zeros, it should not be
converted to an integer with bitwise testing against
                           //   it due to the dynamic state of the
lists and since the lists could grow beyond
                           //   the limits of a 32-bit integer.
Originally wrote script with this as a list.
                           //   Converted it to a string to save
memory.

////////////////////////////////////////////////////////////////////
//////////////////
// User Defined Functions
add(string a, integer b, integer c, integer d, integer e, integer
f){ //adds a timer to the lists
    integer g = llGetListLength(timername);
    timername           = llListInsertList(timername,           [a],
g);
    timerticks          = llListInsertList(timerticks,          [b],
g);
    timercycles         = llListInsertList(timercycles,         [c],
g);
    timerprim           = llListInsertList(timerprim,           [d],
g);
    timerticksremaining = llListInsertList(timerticksremaining, [e],
g);
    timerstate          = llInsertString  (timerstate,           g,
(string)f);
}

remove(integer a){ //removes a timer from the lists
```

```
    timername           = llDeleteSubList(timername,
a,a);
    timerticks          = llDeleteSubList(timerticks,
a,a);
    timercycles         = llDeleteSubList(timercycles,
a,a);
    timerprim           = llDeleteSubList(timerprim,
a,a);
    timerticksremaining = llDeleteSubList(timerticksremaining,
a,a);
    timerstate          = llDeleteSubString(timerstate,
a,a);
}

clear(){ // Clears all timers and stops the master timer
    timername           = [];
    timerticks          = [];
    timercycles         = [];
    timerprim           = [];
    timerticksremaining = [];
    timerstate          = "";
    llSetTimerEvent(ticks = 0);
}

////////////////////////////////////////////////////////////////////
//////////////////
// Main Script
default
{
    state_entry(){
        llSetTimerEvent(0); //Start master timer in an idle state
    }
    on_rez(integer a){
        clear(); // Reset the lists
    }
    link_message(integer a, integer b, string c, key d){
        if(b == timerchannellisten){ // Check to make sure message
is from the correct channel
            if(c == "TIMER_STATUS_CHECK") { // send status report if
proper command is received
                    llMessageLinked(a,
timerchannelspeak,"TIMER_STATUS," +
                        (string)ticks + "," + (string)timeint + ","
+
                        (string)llGetListLength(timername) + "," +
                        (string)llGetFreeMemory() + "," +
                        (string)(llFloor((llGetFreeMemory() -
500)/130)) +
                        "," + (string)timerchannelspeak + "," +
                        (string)timerchannellisten, NULL_KEY);
            } else if (c == "TIMER_RESET") { //Reset timer module
lists if proper command is received
                clear();
            } else {
                list e = llCSV2List(c);
                if(llGetListLength(e) >= 3){ // makes sure the list
length is at minimum a valid command length
```

```
                    string  f = llToLower(llList2String(e, 0));  //
Gets the command then converts to lowercase to make the commands
case insensitive
                    string  g = (string)a +
llGetSubString(llList2String(e, 1), 0, 9);

// Gets timer name from the command string and prepends the prim
number to it to

// guarantee uniqueness from prim to prim within the same object.
Also truncates the

// the timer name to 10 characters to allow for reliable memory
usage estimations.
                    integer h = (integer)llList2String(e, 2);    //
Gets the timer's interval from the command string
                    integer i = (integer)llList2String(e, 3);    //
Gets the timer's cycle from the command string
                    integer j = llListFindList(timername, [g]);  //
Index of the called timer
                    if(f == "add"){
                        if(h < 1 || i < 1) { // Checks to make sure
both the ticks and the cycles are positive

llMessageLinked(llList2Integer(timerprim, i),
timerchannelspeak,"TIMER,ERROR,Non positive Tick or Cycle value.",
NULL_KEY);
                            jump end;
                        }
                        if(llFloor((llGetFreeMemory() - 500)/130) ==
0) { // Checks available memory, attempts to leave at least 500
bytes available.

llMessageLinked(llList2Integer(timerprim, i),
timerchannelspeak,"TIMER,ERROR,Out of memory", NULL_KEY);
                        } else {
                            if(timername == [])
                                llSetTimerEvent(timeint); // If the
master timer is not started, then start it.
                            if(j != -1) // Check to see if the timer
already exists
                                remove(j);// if it does, remove it
                            add(g, h, i, a, h, 1); // Add timer
                            if(ticks == 0)
                                llSetTimerEvent(timeint); // If the
master timer is not started, then start it.
                        }
                    } else if(f == "rem") {
                        remove(j); // Remove timer
                        if(timername == [])
                            llSetTimerEvent(ticks = 0);
                    } else if(f == "pau") {
                        if(j != -1){ // Make sure the timer already
exists. If not, do nothing.
                            timerstate =
llDeleteSubString(timerstate, j, j); // Change timer
                            timerstate = llInsertString(timerstate,
j, "0");  //              state to paused
                        }
```

```
                    } else if(f == "res") {
                        if(j != -1){ // Make sure the timer already
exists. If not, do nothing.
                            timerstate =
llDeleteSubString(timerstate, j, j); // Change timer
                            timerstate = llInsertString(timerstate,
j, "1");  //              state to run
                        }
                    }
                    @end;
                    // If the message made it this far without doing
anything, then the command must not be a timer command
                } else {
                    llMessageLinked(a,
timerchannelspeak,"TIMER_STATUS," + //This part was added solely for
debugging purposes and can be removed.

(string)ticks + "," +

(string)timeint + "," +

(string)llGetListLength(timername) + "," +

(string)llGetFreeMemory() + "," +

(string)(llFloor((llGetFreeMemory() - 500)/130)) + "," +

(string)timerchannelspeak + "," +

(string)timerchannellisten,
                                                        NULL_KEY);
                }
            }
        }
    }

    timer(){ // master timer
        ++ticks;
        integer a = llGetListLength(timername); // Determines how
many timers to update
        integer i;
        list timerstoremove = [];
        for (i = 0; i < a; ++i) { // Goes through the entire timer
list
            if(llGetSubString(timerstate, i, i) == "1"){ // Check
timer run state. If 0, then step to next timer.
                timerticksremaining =
llListReplaceList(timerticksremaining,
[llList2Integer(timerticksremaining, i) - 1], i, i); // Subtracts 1
from ticksremaining
                if(llList2Integer(timerticksremaining, i) == 0) { //
Checks to see if the timer has elapsed
                    if(llList2Integer(timercycles, i) != -1) //
Checks to see if timer is indefinite
                        timercycles = llListReplaceList(timercycles,
[llList2Integer(timercycles, i) - 1], i, i); // Subtracts 1 from
cycles
```

```
                    timerticksremaining =
llListReplaceList(timerticksremaining, [llList2Integer(timerticks,
i)], i, i); // Resets ticksremaining
                    // Send linked message reporting timer event
                    llMessageLinked(llList2Integer(timerprim, i),
timerchannelspeak,"TIMER," +

llDeleteSubString(llList2String(timername, i), 0,
llStringLength((string)llList2Integer(timerprim, i)) - 1) +
                                        "," +
(string)llList2Integer(timerticks, i) + "," +
(string)llList2Integer(timercycles, i) + "," + (string)a + "," +

(string)(llFloor((llGetFreeMemory() - 500)/130)), NULL_KEY);
                    if(llList2Integer(timercycles, i) == 0) {//
Checks to see if tiemr has expired
                        timerstoremove = (timerstoremove=[]) +
timerstoremove + [i]; // Assemble list of timers to remove
                    }
                }
            }
        }
        a = llGetListLength(timerstoremove); // Determines how many
timers to remove
        for (i = a; i > 0; --i) {
            remove(i - 1); // Removes timer
            if(timername == [])
                llSetTimerEvent(ticks = 0);
        }
    }
}
```

Β3. Παράδειγμα χρήσης του Timer στο Second Life

```
integer counter;
float   gap = 2.0;

default
{
    state_entry()
    {
        llSetTimerEvent(gap);
        llSay(0,"Πήρατε τον λόγο!");
    }

    touch_start(integer total_number)
    {
        llSay(0, "The timer stops.");
        llSetTimerEvent(0.0);
        counter = 0;
    }

    timer()
    {

        key    avatarKey  = llDetectedKey(0);
        string avatarName = llDetectedName(1);

        ++counter;
```

```
         if((counter % 10) == 0) {
        llSay(0,
            (string)counter+" ticks have passed in " +
(string)llGetTime()
            + " script seconds.\nEstimated elapsed time for
"+avatarName+":  " + (string)(counter * gap));
        }
    }
}
```