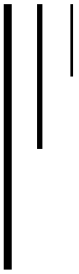


Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής

Πρόγραμμα Μεταπτυχιακών Σπουδών

«Προηγμένα Συστήματα Πληροφορικής»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Εφαρμογή Reve World Designer – Σχεδίαση εικονικών κόσμων για την πλατφόρμα REVE** **Reve World Designer – Design of virtual worlds for the REVE platform**
Ονοματεπώνυμο Φοιτητή	**Έντνα Τσελεμέγκου**
Πατρώνυμο	**Χρήστος**
Αριθμός Μητρώου	**ΜΠΣΠ/12084**
Επιβλέπων	**Θεμιστοκλής Παναγιωτόπουλος, Καθηγητής**

Ημερομηνία Παράδοσης **Ιούνιος 2015**

Τριμελής Εξεταστική Επιτροπή

Γεώργιος Τσιχριντζής Καθηγητής	Θεμιστοκλής Παναγιωτόπουλος Καθηγητής	Δημήτριος Αποστόλου Επίκουρος Καθηγητής

ΠΕΡΙΛΗΨΗ

Αντικείμενο της παρούσας μεταπτυχιακής διατριβής είναι η ανάπτυξη ενός γραφικού περιβάλλοντος σχεδιασμού εικονικών κόσμων για την πλατφόρμα REVE, η οποία επιτρέπει την ανάπτυξη ευφυών εικονικών περιβαλλόντων. Τα ευφυή εικονικά περιβάλλοντα προϋποθέτουν την σχεδίαση εικονικών κόσμων, οπότε η εν λόγω εργασία στοχεύει στο να διευκολύνει τον χρήστη σε αυτή την προσπάθεια. Ως εικονικός κόσμος νοείται ένα σύνολο από εικονικά αντικείμενα, δηλαδή αντικείμενα που αναπαριστούν με τη βοήθεια των ηλεκτρονικών υπολογιστών, φυσικά ή φανταστικά πρόσωπα ή/και πράγματα και διακρίνονται από κάποια βασικά χαρακτηριστικά που τα καθιστούν μοναδικά στον χρήστη, όπως οι διαστάσεις τους, το σχήμα τους, το χρώμα τους και η θέση τους. Για την σχεδίαση των εικονικών κόσμων χρησιμοποιείται η αναπαράσταση REVE που ορίζει έναν κόσμο ως σύνολο από items που μοντελοποιούνται σε τρία επίπεδα: στο φυσικό μοντέλο, στο σημασιολογικό μοντέλο και στο αλληλεπιδραστικό μοντέλο. Η διατριβή εστιάζει στο φυσικό μοντέλο, που αφορά στη φυσική υπόσταση των αντικειμένων του κόσμου, ενώ ταυτόχρονα γίνεται μια πρώτη προσέγγιση υλοποίησης και για τα υπόλοιπα δυο, προσφέροντας τη δυνατότητα μελλοντικής επέκτασης και βελτιστοποίησης της εφαρμογής. Η REVE χρησιμοποιεί την γλώσσα VERL για την αναπαράσταση του κόσμου, ως εκ τούτου η εφαρμογή δίνει τη δυνατότητα εισαγωγής και εξαγωγής τέτοιων αρχείων, ενώ τα αντικείμενα που αποτελούν τον κόσμο σχεδιάζονται με την γλώσσα VRML και X3D. Η τεχνολογία που χρησιμοποιήθηκε είναι η Java SE 7 και JDK 7, με εκτενή χρήση των βιβλιοθηκών swing, jaxp και xj3d, για την ανάπτυξη του γραφικού περιβάλλοντος, τη δημιουργία και επεξεργασία των verl αρχείων και την απεικόνιση των αντικειμένων και του κόσμου αντίστοιχα.

Λέξεις κλειδιά: REVE, item, εικονικός κόσμος, εικονικό περιβάλλον, φυσική αναπαράσταση, virtual model, xj3d, verl, σχεδίαση εικονικών κόσμων

ABSTRACT

The purpose of this MSc thesis is to develop a graphic environment for designing virtual worlds used in the REVE platform, which enables the development of Intelligent Virtual Environments. Intelligent Virtual Environments require the design of virtual worlds, thus, this research aims in assisting the user in that aspect. A virtual world is defined as a set of virtual objects which represent, with the use of computers, real or imaginary beings and/or things and have certain characteristics, such as dimensions, shape, colour and position, by which the user perceives them as unique. For designing the virtual worlds, we use the REVE representation, which defines a world as a combination of items that have three conceptual models: the physical model, the semantic model and the access model. In this thesis, we focus on the first model that concerns the physical representation of the objects in the world, though an approach on the other two models is also made, presenting the possibility of future enhancements. REVE uses the VERL language for representing the virtual worlds, therefore, the given application makes possible the import and export of such files, whereas the items that compose the world are designed in VRML and X3D language. The technology used is the Java SE 7 and SDK 7, with an extent use of the swing library for developing the graphic environment, of the jaxp library for processing the verl files and the xj3d library for visualizing the items and the world.

Keywords: REVE, item, virtual world, virtual environment, physical aspect, virtual model, xj3d, verl, virtual world designer

ΠΕΡΙΕΧΟΜΕΝΑ

ΕΙΣΑΓΩΓΗ

Η παρούσα μεταπτυχιακή διατριβή πραγματεύεται την ανάπτυξη ενός γραφικού περιβάλλοντος σχεδιασμού εικονικών κόσμων για την πλατφόρμα REVE. Η πλατφόρμα REVE (Representation and Execution of Virtual Environments) έχει αναπτυχθεί από το Εργαστήριο Τεχνολογίας Γνώσης του Πανεπιστημίου Πειραιώς και αποτελείται από ένα σύνολο εφαρμογών και προγραμματιστικών εργαλείων, τα οποία επιτρέπουν την ανάπτυξη εφαρμογών ευφυών εικονικών περιβαλλόντων (intelligent virtual environments, IVEs).

Ως εικονικό περιβάλλον (virtual environment) μπορούμε να ορίσουμε ένα υπολογιστικό σύστημα που επιτρέπει την αναπαράσταση εικονικών κόσμων, προσομοιώνοντας με ένα πλήθος εργαλείων και υπολογιστικών πόρων ένα φυσικό περιβάλλον, καθώς και την αλληλεπίδραση ενός ή παραπάνω χρηστών με αυτό. Το ευφυές εικονικό περιβάλλον προσθέτει δυναμικότητα στο εικονικό περιβάλλον, εισάγοντας το στοιχείο της τεχνητής νοημοσύνης. Ως εκ τούτου, σε ένα ευφυές εικονικό περιβάλλον συναντάμε οντότητες που αναπαρίστανται και δρουν αυτόνομα. Στην περίπτωση της πλατφόρμας REVE, αυτές οι οντότητες αναπαρίστανται με την μορφή ενσώματων ευφυών πρακτόρων που μέσω αισθητήρων (sensors) εκλαμβάνουν πληροφορία για τον κόσμο, ενώ μέσω επιδραστών (effectors) αλληλεπιδρούν με αυτόν, μεταβάλλοντάς τον.

Βασικό συστατικό των εικονικών περιβαλλόντων, ευφυών και μη, είναι ο εικονικός κόσμος. Ο εικονικός κόσμος αποτελείται από ένα σύνολο εικονικών αντικειμένων, όπως ένας πραγματικός κόσμος αποτελείται από ένα σύνολο αντικειμένων. Τα εικονικά αντικείμενα μπορεί να έχουν οποιαδήποτε μορφή ανάλογα το περιεχόμενο του κόσμου – άνθρωποι, ζώα, έντομα, πράγματα ακόμα και φανταστικά πλάσματα κ.ά – και διακρίνονται από κάποια βασικά χαρακτηριστικά που τα καθιστούν μοναδικά και αναγνωρίσιμα από τον χρήστη. Αυτά είναι η γεωμετρία τους, ή αλλιώς το σχήμα τους, οι διαστάσεις τους, το μέγεθός τους, το χρώμα τους, το υλικό τους, η υφή τους – όχι απαραίτητα ως απτή έννοια όσο ως η εντύπωση του υλικού ή της υφής που δίνουν στον χρήστη – και η θέση τους στο χώρο. Πέραν των φυσικών χαρακτηριστικών που οπτικοποιούν τα εικονικά αντικείμενα, αυτά μπορεί να διακρίνονται και από άλλα χαρακτηριστικά που διεγείρουν τις υπόλοιπες αισθήσεις του χρήστη, όπως την ακοή ή την αφή. Ωστόσο, δεδομένου ότι μια τέτοια αναπαράσταση των αντικειμένων προϋποθέτει πιο εξελιγμένη τεχνολογία, σε αυτή την εργασία δεν ασχολούμαστε με αυτές τις πτυχές ενός εικονικού κόσμου.

Βασικό στοιχείο ενός εικονικού κόσμου είναι η «πιστευτότητα» (believabiliy), που επηρεάζει τον βαθμό εμβύθισης ενός χρήστη σε αυτόν. Η πιστευτότητα συνεπάγεται πως ο εικονικός κόσμος πρέπει να διέπεται από ένα σύνολο νόμων και κανόνων, που είτε προσομοιώνουν τους φυσικούς νόμους – όπως η βαρύτητα ή η τριβή κατά την σύγκρουση αντικειμένων – είτε ορίζουν δικούς τους νόμους, οι οποίοι, όμως, σε κάθε περίπτωση πρέπει να πείθουν τον χρήστη για την ισχύ τους μέσα από συνοχή, καθολικότητα και συνέπεια στην εφαρμογή τους. Η πιστευτότητα σε έναν εικονικό κόσμο είναι σημαντική, γιατί επηρεάζει την αντίληψη του χρήστη κατά την αλληλεπίδρασή του με τον κόσμο, ενώ ορίζει και τον τρόπο συμπεριφοράς των ευφυών οντοτήτων σε αυτόν.

Βάσει των παραπάνω, μπορούμε να ορίσουμε τον εικονικό κόσμο ως την αναπαράσταση ενός τεχνητού κόσμου που προκύπτει ως αποτέλεσμα της επίδρασης των εικονικών περιβαλλόντων και ο οποίος αποτελείται από ένα σύνολο εικονικών αντικειμένων, προκαλώντας στον χρήστη την εντύπωση πως βρίσκεται μέσα στον κόσμο, μέσα από τη διέγερση μέρους ή όλων των αισθήσεών του. Η δημιουργία, λοιπόν, του εικονικού κόσμου είναι, ουσιαστικά, το πρώτο βήμα σε κάθε προσπάθεια ανάπτυξης μιας εφαρμογής ευφυούς εικονικού περιβάλλοντος.

Για την σχεδίαση των εικονικών κόσμων χρησιμοποιείται η αναπαράσταση REVE. Η πρωτοπορία στην REVE έγκειται, μεταξύ άλλων, στο ότι η αναπαράστασή του κόσμου αποδεσμεύεται από τις λεπτομέρειες της υλοποίησής του και ο κόσμος ορίζεται σε ένα εννοιολογικό πλαίσιο που περιλαμβάνει υψηλού επιπέδου έννοιες, όπως το γράφημα σκηνής, το αντικείμενο, η φυσική υπόσταση του αντικειμένου, η σημασιολογία του και η ικανότητα αλληλεπίδρασης ενός χρήστη με αυτό.

Η γεφύρωση μεταξύ εννοιολογικής προσέγγισης και υλοποίησης του κόσμου επιτυγχάνεται μέσα από την βασισμένη σε XML γλώσσα VERL (Virtual Environment Representation Language). Η verl ορίζει τον εικονικό κόσμο μέσα από ένα σύνολο από <items> που αποτελούν τον κόσμο, δίνοντας στοιχεία για την προέλευσή τους

και τα χαρακτηριστικά τους, εκμεταλλευόμενη την αναπαράσταση x3d και vrml αντικειμένων μέσω γραφήματος σκηνής για την επιλογή των items και την δομή της xml για την αναπαράσταση του κόσμου. Τα δεδομένα σε γλώσσα verl φορτώνονται σε επόμενο στάδιο από το σύστημα REVE Worlds, το οποίο οπτικοποιεί τον εικονικό κόσμο σε τρεις διαστάσεις, επιτρέπει την περιήγηση του χρήστη στον εικονικό κόσμο με διάφορους τρόπους, την σύνδεση εξωτερικών εφαρμογών παραγωγής συμπεριφοράς με εικονικά σώματα με στόχο την εισαγωγή ευφυών εικονικών πρακτόρων (intelligent virtual agents, IVAs) στον εικονικό κόσμο κ.ά.

Στην παρούσα διατριβή αναπτύσσουμε μια παραθυρική εφαρμογή για τον σχεδιασμό εικονικών κόσμων από μηδενική βάση, αξιοποιώντας την αναπαράσταση REVE. Κύριο αντικείμενο της εφαρμογής είναι να παράγει αρχεία που είναι συμβατά με τις υπάρχουσες υλοποιήσεις που περιλαμβάνονται στην πλατφόρμα σύμφωνα με συγκεκριμένες προδιαγραφές και να παρέχει την δυνατότητα στον χρήστη να εξάγει δεδομένα σε ένα verl αρχείο και να εισάγει δεδομένα από ένα verl αρχείο. Η δομή που ακολουθήσαμε για να επιτύχουμε τον ζητούμενο στόχο περιγράφεται παρακάτω.

Στο πρώτο κεφάλαιο εισάγουμε τον χρήστη στις βασικές έννοιες που διέπουν την ουσία της εφαρμογής μας. Στην προσπάθεια να εξηγήσουμε τι είναι εικονικός κόσμος, αναγκαστικά αναφερόμαστε και στα εικονικά περιβάλλοντα και την εικονική πραγματικότητα ως αλληλένδετες έννοιες, ενώ κάνουμε μια εκτενή αναφορά και στα ευφυή εικονικά περιβάλλοντα, εφόσον τέτοια αναπτύσσει και η πλατφόρμα REVE.

Στο δεύτερο κεφάλαιο, εξετάζουμε τους διαφορετικούς τρόπους αναπαράστασης εικονικών κόσμων, εστιάζοντας στο γράφημα σκηνής και τις γλώσσες VRML και X3D, αφού τα αντικείμενα που χρησιμοποιούμε στην εφαρμογή μας για την σύνθεση του κόσμου έχουν σχεδιαστεί βάσει αυτών και έχουν την μορφή γραφήματος σκηνής.

Στο τρίτο κεφάλαιο παρουσιάζουμε δυο πολυχρησιμοποιημένα και γνωστά εργαλεία σχεδίασης εικονικών κόσμων, τόσο για να εξετάσουμε υπάρχουσες υλοποιήσεις αντίστοιχων εργαλείων σχεδίασης όσο και για να λάβουμε υπόψη απαραίτητες λειτουργίες στην ανάλυση απαιτήσεων.

Στο τέταρτο κεφάλαιο, παρουσιάζουμε τα βασικά στοιχεία της αναπαράστασης REVE, επικεντρώνοντας στην γλώσσα verl, που θα αποτελέσει και την βάση σχεδίασης του κόσμου μας, καθώς και στο σύστημα Reve Worlds, στο οποίο είναι σκοπός να αναπαρίσταται ο τριδιάστατος εικονικός κόσμος για την δημιουργία ευφυών εικονικών περιβαλλόντων.

Στο πέμπτο κεφάλαιο προχωράμε στη σχεδίαση της εφαρμογής, όπου, αρχικά, κάνουμε μια εκτίμηση για το τι θα θέλαμε να επιτύχουμε κατά την υλοποίηση και στη συνέχεια, αναλύουμε τις τεχνολογίες που αξιοποιήσαμε τελικά για να αναπτύξουμε την εφαρμογή.

Στο έκτο κεφάλαιο κάνουμε μια αναλυτική περιγραφή της υλοποίησης της εφαρμογής, επικεντρωνόμενοι στις δυσκολίες που αντιμετωπίσαμε στην προσπάθεια αυτή. Η αναφορά των δυσκολιών έγινε τόσο για να αναδειχθεί το μέγεθος του χρόνου που απαιτούσε η τρέχουσα υλοποίηση, εμποδίζοντάς μας να εμπλουτίσουμε την εφαρμογή με επιθυμητές αλλαγές λόγω των περιορισμένων χρονικών πλαισίων, όσο και για να ληφθούν υπόψη για μελλοντικές αλλαγές και βελτιστοποίηση, ώστε να αποφευχθούν παρόμοια λάθη ή λανθασμένες προσεγγίσεις.

Στο έβδομο κεφάλαιο παρουσιάζουμε τις προδιαγραφές λειτουργίας και ένα παράδειγμα χρήσης. Στη συνέχεια, κάνουμε μια μικρή αναδρομή σε αυτά που επιτύχαμε σε σχέση με την ανάλυση απαιτήσεων και το πραγματικό αποτέλεσμα, καταθέτοντας τα συμπεράσματά μας και κάποιες ιδέες για ενδεχόμενες βελτιώσεις που μπορούν να πραγματοποιηθούν στο μέλλον.

Είναι σημαντικό να επισημάνουμε πως σκοπός της μεταπτυχιακής διατριβής δεν είναι η παροχή όλων των πτυχών της λειτουργικότητας που παρέχει η αναπαράσταση REVE για έναν εικονικό κόσμο, αλλά η επικέντρωση στην φυσική αναπαράσταση των εικονικών κόσμων. Ως εκ τούτου, εστιάζουμε στο physical aspect του κόσμου και όχι τόσο στο σημασιολογικό μοντέλο ή στο μοντέλο αλληλεπίδρασης των εικονικών αντικειμένων που συνθέτουν τον κόσμο. Άλλωστε, η διατριβή αυτή έχει ως στόχο να αποτελέσει μια βάση για επέκταση της τρέχουσας λειτουργικότητάς και ενδεχόμενη βελτιστοποίησή της.

Η εργασία αυτή πραγματοποιήθηκε σε συνεργασία με τον συνάδελφο Γεώργιο Σανιόγλου, που επικεντρώθηκε στο πρωταρχικό κομμάτι της δυνατότητας επιλογής ενός αντικειμένου για την συμμετοχή του

σε έναν εικονικό κόσμο. Η δική συνεισφορά επικεντρώθηκε στο κομμάτι της μετατροπής αυτού του αντικειμένου σε item και την σχεδίαση του εικονικού κόσμου βάσει της αναπάραστασης REVE.

ΚΕΦΑΛΑΙΟ 1 - ΒΑΣΙΚΕΣ ΕΝΝΟΙΕΣ ΚΑΙ ΟΡΙΣΜΟΙ

Για να καταφέρουμε να υλοποιήσουμε την παρούσα πτυχιακή εργασία, δεδομένου ότι θα ασχοληθούμε με την ανάπτυξη μιας εφαρμογής που σκοπό έχει να συνθέτει τρισδιάστατα αντικείμενα ώστε να δημιουργεί εικονικούς κόσμους, επιτρέποντας την αλληλεπίδραση χρηστών με αυτούς μέσω ευφυών πρακτόρων στο περιβάλλον REVE, πρέπει πρώτα από όλα να εξοικειωθούμε με τις βασικές έννοιες με τις οποίες καταπιανόμαστε. Άλλωστε, χρειαζόμαστε ορισμούς, τόσο για να διακρίνουμε τις διαφορετικές έννοιες μεταξύ τους όσο και για να διευκολύνουμε μέσω αυτών τη σχετική έρευνα που συντελείται. Συνεπώς, πρέπει να δούμε τι νοείται ως εικονικό περιβάλλον και εικονική πραγματικότητα, ως ευφυές εικονικό περιβάλλον και ως εικονικός κόσμος.

1. Εικονικό Περιβαλλον και Εικονική Πραγματικότητα

1.1.1. Το Εικονικό Περιβάλλον

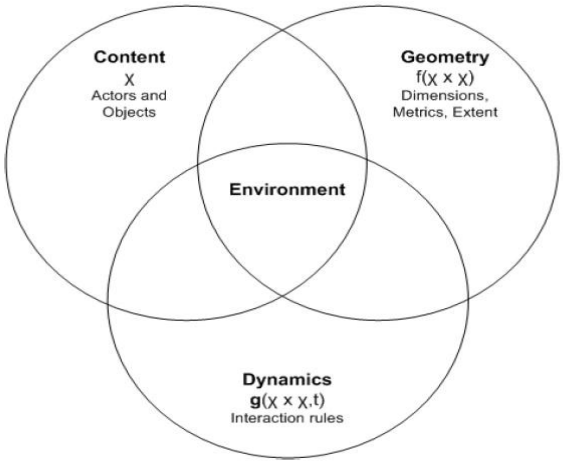
Ένα εικονικό περιβάλλον είναι ένα ετερογενές υπολογιστικό σύστημα που επιτρέπει την αναπαράσταση εικονικών κόσμων και την αλληλεπίδραση ενός ή παραπάνω χρηστών με αυτούς, προσπαθώντας να προσομοιώσει τους φυσικούς νόμους σε ένα τεχνητό κόσμο. Πρόκειται, δηλαδή, για «*μια αναπαράσταση που παράγεται από υπολογιστή, η οποία επιτρέπει ή προκαλεί στον χρήστη την αίσθηση πως είναι σε ένα περιβάλλον διαφορετικό από εκεί που πραγματικά βρίσκεται και την ανάγκη να αλληλεπιδράσει με αυτό*».

Τα εικονικά περιβάλλοντα χρονολογούνται από την δεκαετία του 1960, με τα συστήματα προσομοίωσης οχημάτων και την ανάπτυξη της τεχνολογίας τηλεχειρισμού για τη δημιουργία των πρώτων HDMs (Head Mounted Displays), που σχεδιάστηκαν για να αντικαταστήσουν τους ογκώδεις προσομοιωτές πτήσης της εποχής. Επίσης, από τα τέλη του 1980 και χάρη στον πρωτοπόρο επιστήμονα της πληροφορικής Ivan Sutherland, τα εικονικά περιβάλλοντα συσχετίστηκαν με την ανάπτυξη συστημάτων για γραφικά υπολογιστών και, έκτοτε, έχει ακολουθήσει σειρά ερευνών για την ανάπτυξη διαδραστικών τρισδιάστατων γραφικών με στόχο την συνεχή βελτίωση της επικοινωνίας μεταξύ ανθρώπου-υπολογιστή.

Σύμφωνα με τον Stephen R. Ellis, μπορούμε να ορίσουμε τα εικονικά περιβάλλοντα ως «*διαδραστικές αναπαραστάσεις ψηφιακών εικόνων που ενισχύονται από ειδική επεξεργασία και από βοηθητικές, μη οπτικές αναπαραστάσεις, όπως ακουστικές ή απτικές, ώστε οι χρήστες να πειστούν πως βρίσκονται εμβυθισμένοι σε έναν τεχνητό, συνθετικό χώρο*». Επίσης, ο Ellis το 1993 έκανε μια πολύ εύστοχη ανάλυση ενός Εικονικού Περιβάλλοντος στα λειτουργικά στοιχεία από τα οποία αποτελείται, σύμφωνα με την οποία ένα εικονικό περιβάλλον απαρτίζεται από τα παρακάτω:

- **Περιεχόμενο**: Αποτελείται από τα αντικείμενα (objects) και τα ενεργά ή δρώντα στοιχεία (actors) τα οποία μπορούν να θεωρηθούν και αυτά σαν αντικείμενα που, όμως, έχουν την δυνατότητα να ξεκινούν από μόνα τους αλληλεπιδράσεις με άλλα αντικείμενα του Εικονικού Περιβάλλοντος. Ένα τέτοιο αντικείμενο είναι ο ίδιος ο χρήστης που αντιπροσωπεύεται στο Εικονικό Περιβάλλον από τη δική του οπτική άποψη (viewpoint) του περιβάλλοντος.
- **Γεωμετρία**: Η περιγραφή του πεδίου όπου εξελίσσεται η αλληλεπίδραση.
- **Δυναμικές**: Οι κανόνες της αλληλεπίδρασης ανάμεσα στα συστατικά του περιβάλλοντος, οι οποίοι περιγράφουν την συμπεριφορά των συστατικών αυτών καθώς ανταλλάσσουν ενέργεια ή πληροφορία.

Η συσχέτιση αυτών των παραπάνω λειτουργικών στοιχείων γίνεται πιο εμφανής στην παρακάτω εικόνα.



Εικόνα 1.1: Λειτουργικά στοιχεία ενός ΕΠ

Τα Εικονικό Περιβάλλον είναι ο όρος που έχει επικρατήσει για να περιγράψει τον πιο δημοφιλή, αλλά και πιο αντιφατικό, όρο της Εικονικής Πραγματικότητας(Virtual Reality), συνεπώς η ιστορία και η εξέλιξη της εικονικής πραγματικότητας είναι ιστορία και εξέλιξη των εικονικών περιβαλλόντων.

1.1.2. Εικονική Πραγματικότητα

Η καταγωγή της Εικονικής Πραγματικότητας, όπως και του εικονικού περιβάλλοντος, χρονολογείται από το 1965 με το δημοφιλές άρθρο της εποχής 'The Ultimate Display' του επιστήμονα Ivan Sutherland. Στο εν λόγω άρθρο, ο Sutherland εισήγαγε τις βασικές έννοιες της εμβύθισης σε έναν προσομοιωμένο κόσμο όπου υπάρχουν απόλυτες αισθητηριακές είσοδοι και έξοδοι. Όπως το έθεσε ο ίδιος: "*The screen is a window through which one sees a virtual world. The challenge is to make that world look real, act real, sound real, feel real*."(Gobbetti E., Scateni R., 1998). Σε ελεύθερη μετάφραση, «*η οθόνη είναι ένα παράθυρο μέσα από το οποίο κάποιος βλέπει έναν εικονικό κόσμο. Η πρόκληση είναι να κάνεις αυτόν τον κόσμο να μοιάζει πραγματικός, να δρα σαν πραγματικός, να ακούγεται πραγματικός, να αισθάνεται πραγματικός*».

Ωστόσο, ο όρος «Εικονική Πραγματικότητα» χρησιμοποιήθηκε πρώτη φορά το 1989 από τον Jaron Larnier, ιδρυτή της εταιρίας VPL (Virtual Programming Languages) Research, που ήταν πρωτοπόρα στα συστήματα εικονικής πραγματικότητας. Ο ορισμός που έδωσε ήταν ο εξής: «*Η Εικονική Πραγματικότητα είναι ένα αλληλεπιδραστικό, τρισδιάστατο περιβάλλον, φτιαγμένο από υπολογιστή, στο οποίο μπορεί κάποιος να εμβυθιστεί*». Έκτοτε, έχουν ακολουθήσει διάφοροι ορισμοί για να περιγράψουν την εικονική πραγματικότητα, εκ των οποίων αξίζει να αναφέρουμε τους παρακάτω:

«*Η Εικονική Πραγματικότητα, αποτελεί έναν όρο που έχει γίνει πρόσφατα γνωστός, αλλά και από τους πλέον διαδεδομένους στο χώρο των υπολογιστών, ο οποίος μεταφέρει το χρήστη ή τους χρήστες, σε ένα συνθετικό, τεχνητό, εικονικό και φτιαγμένο από υπολογιστή περιβάλλον*». - M.Krueger (1991)

«*Αλληλεπιδραστικά γραφικά πραγματικού χρόνου (real-time) με τρισδιάστατα μοντέλα, συνδυασμένα με μια τεχνολογία απεικόνισης η οποία δίνει τη δυνατότητα στο χρήστη για εμβύθιση στον μοντελοποιημένο κόσμο και τη δυνατότητα για απευθείας χειρισμό*».- Fuchs H. Bishop etal. (1992)

«*Η Εικονική Πραγματικότητα αναφέρεται σε αλληλεπιδραστικά, πολυ-αισθητικά, βασισμένα στην όραση, τρισδιάστατα, περιβάλλοντα εμβύθισης, δημιουργημένα από υπολογιστή, καθώς και ο συνδυασμός των τεχνολογιών που απαιτούνται για την ανάπτυξη τέτοιων περιβαλλόντων*».- Cruz-Neira C. (1993)

«*Μπορεί να οριστεί σαν ένας νέος τρόπος επικοινωνίας μεταξύ ανθρώπου και μηχανής. Ένα από τα χαρακτηριστικά του είναι η υιοθέτηση συσκευών απεικόνισης και αλληλεπίδρασης των ανθρώπινων αισθήσεων. Στερεοσκοπικά συστήματα απεικόνισης δίνουν τη εντύπωση πραγματικής χωρικής αντίληψης των τρισδιάστατων εικόνων οι οποίες*

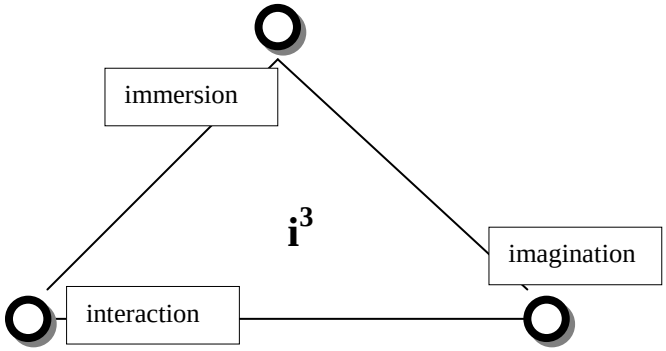
παράγονται από τον υπολογιστή. Επιπλέον, η αίσθηση του ότι είσαι εμβυθισμένος σε ένα εικονικό περιβάλλον, δυναμώνει με τη χρήση συσκευών όπως το γάντι (dataglove), το οποίο επιτρέπει πιο φυσική και ενστικτώδη απευθείας αλληλεπίδραση» - Ellis S. R. (1994)

«Ένα υπολογιστικό σύστημα το οποίο χρησιμοποιείται για τη δημιουργία εικονικών κόσμων, στους οποίους ο χρήστης έχει την εντύπωση της ύπαρξης του σε αυτούς και επιπλέον έχει την ικανότητα να πλοηγηθεί και να χειριστεί τα αντικείμενά τους».- C.Manetta & Blade R. (1995)

«Ένα μέσο το οποίο αποτελείται από αλληλεπιδραστικές εξομοιώσεις με υπολογιστή, οι οποίες 'αισθάνονται' την θέση και τις ενέργειες του χρήστη, και αντικαθιστούν ή επαυξάνουν την ανάδραση σε μία ή παραπάνω αισθήσεις, δίνοντας το αίσθημα της πνευματικής εμβύθισης ή παρουσίας στην εξομοίωση (ένας εικονικός κόσμος)». - Sherman, W. R., Craig, A., B. (2003)

1.2 Τεχνολογίες Αναπαράστασης Εικονικών Περιβαλλόντων

Για την δημιουργία εικονικών περιβαλλόντων ή εικονικής πραγματικότητας απαιτείται συνδυασμός τεχνολογιών και υπολογιστικών πόρων. Η τεχνολογία αναπαράστασής τους λειτουργεί αναπτύσσοντας μια εξατομικευμένη, σε πραγματικό χρόνο, διαδραστική προσομοίωση του περιεχομένου, της γεωμετρίας και της δυναμικής ενός τεχνητού περιβάλλοντος. Συνήθως η εικονική πραγματικότητα περιγράφεται με τα τρία I – Immersion (εμβύθιση), Interaction (αλληλεπίδραση), Imagination (φαντασία) και ορίζεται ως ένα υπολογιστικό σύστημα, με τη βασική διάκρισή από τα συμβατικά πως θέτει τον άνθρωπο στο κέντρο του και οργανώνεται γύρω από τις αισθήσεις του, δίνοντας προτεραιότητα στην όραση και δευτερευόντως, στην ακοή και στην αφή. Μπορεί, λοιπόν, να περιγραφεί ως ένα τρίγωνο με κορυφές του τα τρια I, όπως φαίνεται στην παρακάτω εικόνα:



Εικόνα 1.2.1: Το Εικονικό Περιβάλλον ως 3i

Η τεχνολογία αναπαράστασης ενός εικονικού περιβάλλοντος έρχεται αντιμέτωπη με τρία διαφορετικά ζητήματα: α) την αντίληψη για το σχήμα και την κίνηση των χρηστών και των αντικειμένων, β) την αλληλεπίδραση αυτών μεταξύ τους και με το εικονικό περιβάλλον βάσει φυσικών κανόνων (όπως είναι η υπακοή στους νόμους του Νεύτωνα) και γ) την έκταση και το χαρακτήρα του αναπτυσσόμενου περιβάλλοντος.

Μια επιτυχημένη περιβαλλοντική προσομοίωση πρέπει να παρέχει κατάλληλα κανάλια επικοινωνίας για να ανταποκρίνεται σε αυτές τις λειτουργίες. Για την αντιμετώπιση αυτών των ζητημάτων μπορούν να συνδυαστούν τρία διαφορετικά είδη hardware: α) αισθητήρες για να ανιχνεύουν την κίνηση του χρήστη, β) επιδραστές, για να εφαρμόζουν τις ενέργειες του χειριστή και γ) ειδικού σκοπού υλικό που συνδέει τους αισθητήρες και τους επιδραστές για να παράγει αισθητηριακές εμπειρίες παρόμοιες με αυτές σε ένα φυσικό περιβάλλον. Η σύνδεση αυτή στο εικονικό περιβάλλον επιτυγχάνεται μέσω ενός υπολογιστή προσομοίωσης. Σύμφωνα με τον [5],τέσσερις τεχνολογίες είναι ουσιαστικές για τα συστήματα εικονικής πραγματικότητας:

Πρώτον, οι **οπτικές (και ακουστικές και απτικές) οθόνες αναπαράστασης** που εμβυθίζουν το χρήστη σε έναν εικονικό κόσμο και αποκόβουν αντικρουόμενες αισθητηριακές εμπειρίες από τον πραγματικό κόσμο. Αυτές αποτελούν συσκευές εξόδου και μπορούν να διακριθούν στις παρακάτω κατηγορίες:

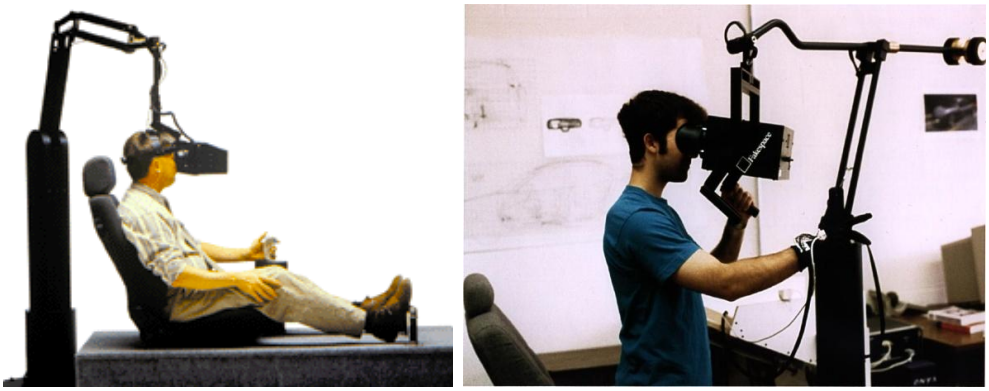
- Τα **Head Mounted Displays** (HMDs): Πρόκειται για κράνη που ο χρήστης φοράει στο κεφάλι και απομονώνουν την οπτική επαφή με τον πραγματικό κόσμο. Τα κράνη αυτά διαθέτουν δυο μικροσκοπικές στερεοσκοπικές οθόνες (μια για κάθε μάτι), που προβάλλουν τις κινούμενες εικόνες του εικονικού κόσμου. Στα κράνη συνδέονται αισθητήρες κίνησης (motion trackers) που συλλέγουν τις κινήσεις του χρήστη και σε πραγματικό χρόνο προσαρμόζουν την απεικόνιση των οθονών. Έτσι, ο χρήστης μπορεί να εξερευνήσει τον κόσμο, αλλάζοντας οπτικές γωνίες, βασισμένες στην περιστροφή του κεφαλιού.

Εικόνα 1.2: ΠαραδείγματαHead Mounted Displays

- Το **Computer Assisted Virtual Environment** (CAVE): Πρόκειται για ένα κλειστό κυβικό δωμάτιο, όπου στους τοίχους και το δάπεδο προβάλλονται στερεοσκοπικές εικόνες. Το πάτωμα μπορεί να είναι μια οθόνη προβολής προς τα κάτω, μια επιδαπέδια επιφάνεια προβολής ή μια επίπεδη οθόνη. Τα συστήματα προβολής είναι υψηλής ανάλυσης χάρη στην μικρή απόσταση θέασης, η οποία απαιτεί λίγα εικονοστοιχεία για να διατηρήσει τη ψευδαίσθηση της πραγματικότητας. Ο χρήστης (ή χρήστες) φοράει τρισδιάστατα γυαλιά μέσα στο δωμάτιο αυτό για να δει τα 3D γραφικά που δημιουργούνται από το CAVE και μπορεί να εξερευνήσει τα αντικείμενα που υπάρχουν στο περιβάλλον και να περιπλανηθεί στο χώρο, ενώ αισθητήρες κίνησης συνεχώς αναπροσαρμόζουν τη στερεοσκοπική προβολή του διευθύνοντος ατόμου.

Εικόνα 1.2.3: Παράδειγμα Cave

- Η **πανκατευθυντική διοπτρική οθόνη** (Binocular Omni-directional monitor - BOOM): Οι οθόνες και το οπτικό σύστημα βρίσκονται σε ένα κουτί το οποίο τοποθετείται σε ένα βραχίονα πολλαπλών συνδέσμων. Ο χρήστης βλέπει τον εικονικό κόσμο κοιτώντας μέσα στο κουτί και μπορεί να καθοδηγήσει το κουτί σε οποιαδήποτε θέση μέσα στον όγκο λειτουργίας της συσκευής. Οι αισθητήρες κίνησης βρίσκονται στους συνδέσμους του βραχίονα που κρατάει το κουτί.



Εικόνα 1.5: Παραδείγματα χρήσης του BOOM

- Οι **πανοραμικές οθόνες** (panoramic displays): Μια ή περισσότερες οθόνες τοποθετούνται εναλλάξ στον ίδιο χώρο σε πανοραμική θέση, όπου ένας χρήστης καθοδηγεί την οπτική γωνία. Χρησιμεύει κυρίως όταν υπάρχουν πολλαπλοί χρήστες στον ίδιο χώρο.

Εικόνα 1.6: Παραδείγματα πανοραμικών οθονών

- Η **Virtu-Sphere:** Η ομώνυμη εταιρία έχει δημιουργήσει μια μπάλα στο μέγεθος ενός ανθρώπου και συνδυάζει αισθητήρες κίνησης και HDM για μια μοναδική εμπειρία εμβύθισης. Ο χρήστης εισέρχεται στη σφαίρα η οποία βρίσκεται στερεωμένη στο έδαφος και φοράει ένα HDM, όπου προβάλλεται ο εικονικός κόσμος. Ο χρήστης μπορεί να προχωρήσει στη σφαίρα. Ενώ ο χρήστης παραμένει στατικός, η σφαίρα περιστρέφεται αντίστοιχα στις κινήσεις του χρήστη, με δυνατότητα περιστροφής 360^{o}. Αισθητήρες στους τροχούς ενημερώνουν τη CPU για το πώς κινείται ο χρήστης. Αντίστοιχα ελέγχεται η κίνηση και για την ανανέωση του κόσμου μέσω του HDM.

Εικόνα 1.7: Παράδειγμα VirtuSphere

Δεύτερον, το **σύστημα απόδοσης γραφικών** (rendering) που παράγει διαρκώς εναλλασσόμενες εικόνες σε ρυθμό από 20 ως 120 πλαίσια το δευτερόλεπτο.

Τρίτον, το **σύστημα ανίχνευσης** (tracking), που αναφέρει διαρκώς τη θέση του κεφαλιού και των άκρων του χειριστή. Για το σύστημα ανίχνευσης χρησιμοποιούνται διάφορες συσκευές εισόδου που έχουν αισθητήρες για να αντιλαμβάνονται την κίνηση του χρήστη και να στέλνουν την απαραίτητη πληροφορία στον υπολογιστή που είναι υπεύθυνος για την εξομοίωση της αλληλεπίδρασης.Τέτοιες συσκευές εισόδου είναι οι παρακάτω:

- **Ανιχνευτές** (trackers): Πρόκειται για αισθητήρες που τοποθετούνται σε συγκεκριμένα σημεία του σώματος (πχ χέρια, πόδια, κεφάλι) ή σε όλο το σώμα και ανιχνεύουν τη θέση του χρήστη, τον προσανατολισμό του ή και τα δυο. Εφαρμόζονται για ανίχνευση κίνησης κεφαλιού, ματιών, προσώπου, χεριών ή και όλου του σώματος, όπως φαίνεται στις παρακάτω εικόνες:

Εικόνα 1.8: Ανιχνευτές χεριού και σώματος

Οι ανιχνευτές μπορεί να είναι μαγνητικοί που είναι και οι πιο συνηθισμένοι, οι οποίοι τοποθετούνται στο αντικείμενο που θέλουμε να παρακολουθήσουμε την κίνησή του, οπτικοί, που κάνουν χρήση στερεοσκοπικών καμερών ή ακουστικοί, όπου οι συσκευές εισόδου είναι μικρόφωνα.

- **Γάντια δεδομένων** (data gloves):Πρόκειται για μια ηλεκτρομηχανική συσκευή που θυμίζει ένα κλασικό γάντι χεριού και φοριέται ως τέτοιο. Χρησιμοποιείται για απτικές εφαρμογές, έχοντας αισθητήρες που επιτρέπουν την ανίχνευσης της κίνησης του χεριού και των δακτύλων και αντιλαμβάνονται την πίεση, τη στροφορμή, τη γραμική δύναμη, τη θερμοκρασία και την υφή μιας επιφάνειας, επιτρέποντας έτσι την αλληλεπίδραση με εικονικά αντικείμενα.

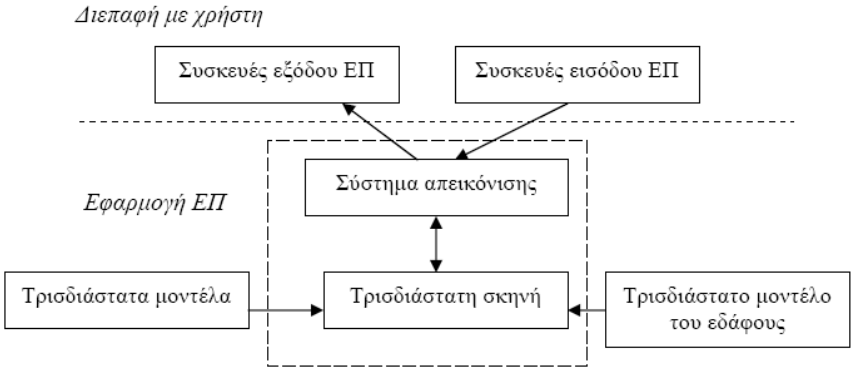
Εικόνα 1.9: Γάντια δεδομένων

- Άλλες συσκευές εισόδου που επιτρέπουν αλληλεπίδραση με το εικονικό περιβάλλον είναι το **τρισδιάστατο ποντίκι** (3D mouse), η **μπάλα ανίχνευσης** (trackball), η **μπίλια** (spaceball) και το **ηλεκτρονικό χειριστήριο** (joystick).

Εικόνα 1.10: Παραδείγματα trackball (αριστερά), space-ball (κέντρο) και joystick (δεξιά)

Τέταρτον, η **δομή της βάσης δεδομένων** και το σύστημα συντήρησης για την παραγωγή και τη διατήρηση λεπτομερών και ρεαλιστικών μοντέλων του εικονικού κόσμου.

Συνοπτικά, τα παραπάνω συστατικά ενός συστήματος εικονικής πραγματικότητας συγκεντρώνονται στην παρακάτω εικόνα:

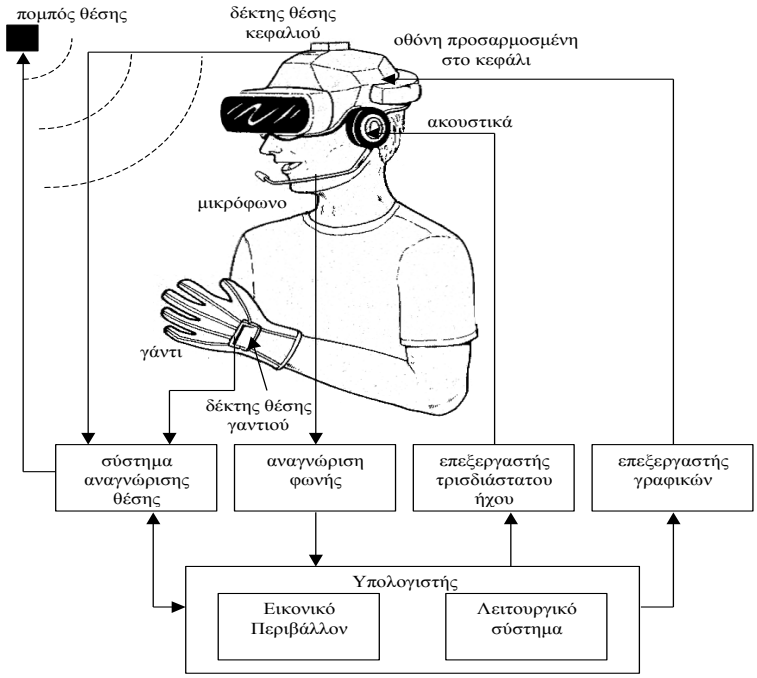


Εικόνα 1.11: Συστατικά Εικονικής Πραγματικότητας

Το σύστημα απεικόνισης (viewer) και η τρισδιάστατη σκηνή είναι δυο στοιχεία που συνδέονται στενά, αφού η επιλογή του τρισδιάστατου περιβάλλοντος απεικόνισης (3D viewer) υποδηλώνει μια τρισδιάστατη υλοποίηση της σκηνής (3D scene). Η σκηνή λαμβάνει πληροφορίες από ένα τρισδιάστατο μοντέλο του εδάφους

και από τρισδιάστατες απεικονίσεις των αντικειμένων του πραγματικού κόσμου. Αυτά τα δυο μαζί αποτελούν την τρισδιάστατη μηχανή απεικόνισης (3D player engine).

Το μοντέλο εδάφους είναι μια γεωγραφική βάση δεδομένων του εδάφους σε τρισδιάστατη μορφή και μαζί με τα τρισδιάστατα μοντέλα του πραγματικού κόσμου ανιστοιχούν στην βάση δεδομένων ενός συστήματος εικονικής πραγματικότητας, όπως περιγράφηκε παραπάνω, ενώ οι συσκευές εισόδου αντιστοιχούν στο σύστημα ανίχνευσης και οι συσκευές εξόδου ή απεικόνισης στο σύστημα απόδοσης γραφικών.



Εικόνα 1.12: Στοιχεία Εισόδου/Εξόδου σε ΕΠ

1.3 Κατηγορίες Εικονικών Περιβαλλόντων

Ανάλογα το εξειδικευμένο υλικό που χρησιμοποιείται για την επικοινωνία με το χρήστη, πράγμα που καθορίζει και τον τρόπο αλληλεπίδρασής του με το εικονικό περιβάλλον, μπορούμε να διακρίνουμε διάφορες κατηγορίες εικονικών περιβαλλόντων. Για παράδειγμα, τα Περιβάλλοντα Εμβύθισης (immersive environments) είναι εικονικά περιβάλλοντα στα οποία οι χρήστες είναι εφοδιασμένοι με κατάλληλο υλικό, πχ. χρήση HDMs, ώστε να δέχονται ερεθίσματα μόνο από το συνθετικό περιβάλλον και όχι από τον πραγματικό κόσμο. Στα Ενισχυμένα Εικονικά Περιβάλλοντα (augmented environment) η διαφορά τους έγκειται στο ότι τα εικονικά αντικείμενα προστίθενται πάνω στην άποψη του πραγματικού κόσμου, εμπλουτίζοντας στην ουσία το πραγματικό περιβάλλον του χρήστη με το εικονικό.

Από την άλλη, όταν ο τεχνητός κόσμος προβάλλεται ολόκληρος πάνω σε μια ή περισσότερες επιφάνειες του πραγματικού κόσμου (π.χ. τοίχος, τραπέζι ή δωμάτιο) ή χρησιμοποιούνται πολλαπλές οθόνες που κυκλώνουν τον χρήστη, όπως στην περίπτωση του CAVE, τότε πρόκειται για Περιβάλλοντα Προβολής (projected environments). Ακόμη, υπάρχει και η πιο συνήθης και πολυχρησιμοποιημένη περίπτωση εικονικού περιβάλλοντος που είναι το Περιβάλλον Οθόνης (desktop environment), όπου ένας τρισδιάστατος εικονικός κόσμος απεικονίζεται στην δισδιάστατη οθόνη του υπολογιστή. Σε αυτή την περίπτωση μπορεί να χρησιμοποιούνται περιφερειακές συσκευές πλοήγησης στον τρισδιάστατο εικονικό χώρο και να γίνεται χρήση στερεοσκοπικών γυαλιών ή κράνους.

Ανάλογα το πλήθος των χρηστών που πλοηγούνται σε ένα εικονικό περιβάλλον, μπορούμε να τα διακρίνουμε σε συστήματα για έναν χρήστη (single-user VEs) ή σε πολυχρηστικά κατανεμημένα εικονικά

περιβάλλοντα (multi-user, collaborative, distributed VEs), που επιτρέπουν σε μια ομάδα διασκορπισμένων χρηστών να αλληλεπιδρούν σε πραγματικό χρόνο.

1.4 Εφαρμογές Εικονικών Περιβαλλόντων

Πλέον, οι εφαρμογές και η χρήση των εικονικών περιβαλλόντων είναι ευρεία και επεκτείνεται σε πληθώρα επιστημονικών κλάδων, γεγονός που αναδεικνύει και τη σημασία τους ως σύγχρονου μέσου αλληλεπίδρασης ανθρώπου-υπολογιστή. Ένας από τους τομείς που βρίσκουν δημοφιλή εφαρμογή είναι η εκπαίδευση, όπου η δυνατότητα παρατήρησης των δεδομένων από διαφορετικές σκοπιές και η δυνατότητα πειραματισμού με αυτά, χωρίς τις πραγματικές συνέπειες που θα είχαν λάθη στον φυσικό κόσμο, ενισχύει τη διαδικασία της μάθησης και της κατανόησης, σε σχέση με τη χρήση στατικού κειμένου και συμβατικών εικόνων.

Για παράδειγμα, μέσα από το πρόγραμμα Newton Worlds (Dede et al., 1994) χρησιμοποιούνται εικονικά περιβάλλοντα για την εκμάθηση της φυσικής ή της χημείας μέσω αλληλεπίδρασης με χημικές δομές. Ακόμη και στον στρατό τα εικονικά περιβάλλοντα χρησιμεύουν στην στρατιωτική εκπαίδευση σε ένα εικονικό πεδίο μάχης. Ένα παράδειγμα είναι το Javeline Virtual Combat System που χρησιμοποιείται στην Αμερική.

Εικόνα 1.13: VEsγια στρατιωτική εκπαίδευση

Επίσης, η δυνατότητα προσομοίωσης ρεαλιστικών κόσμων βρίσκει μεγάλη χρησιμότητα στον τηλεχειρισμό (teleoperation), ιδιαίτερα σε περιπτώσεις που οι αντίστοιχες ενέργειες σε πραγματικές συνθήκες μπορεί να έχουν μεγάλο κόστος ή και να είναι επικίνδυνες, όπως στην ναυπηγική, σε προσομοιώσεις πτήσης και σε εικονικές χειρουργικές επεμβάσεις.

Εικόνα 1.14: Προσομοίωση χειρουργικής επέμβασης

Τα εικονικά περιβάλλοντα βρίσκουν εφαρμογή και στις επιχειρήσεις, καθώς επιτρέπουν την αναπαράσταση και το χειρισμό ενός μοντέλου, με σκοπό τον έλεγχο και τη βελτίωση της λειτουργικότητάς του πριν ακόμα βγει στο στάδιο της παραγωγής. Κατά κόρον αυτό συμβαίνει στον κλάδο της αρχιτεκτονικής. Με τη βοήθεια ειδικού λογισμικού CAD (Computer Aided Design) ή 3D, οι αρχιτέκτονες σχεδιάζουν τρισδιάστατα μοντέλα των κτιρίων στον υπολογιστή, στα οποία μπορεί κανείς να περιηγηθεί εικονικά μέσω του υπολογιστή και να τα εξετάσει τόσο όσον αφορά το εσωτερικό τους όσο και σε σχέση με τον περιβάλλοντα εξωτερικό χώρο.

Εικονικά περιβάλλοντα χρησιμοποιούνται και στην NASA για την απομακρυσμένη εξερεύνηση πλανητικών επιφανειών, όπου προσφέρουν τεχνικές επίλυσης προβλημάτων ελέγχου που δημιουργούνται από τη χρονοκαθυστέρηση. Αντίστοιχα, στην ρομποτική είναι άκρως χρήσιμα στην ανάλυση και στην οπτικοποίηση επιστημονικών δεδομένων. Τέλος, βρίσκουν εφαρμογή και στην τέχνη, με την εξέλιξη της διαδραστικής τέχνης και βέβαια, στην ψυχαγωγία, όπου τα τελευταία χρόνια πρωταγωνιστικό ρόλο έχει η βιομηχανία παραγωγής βιντεο-παιχνιδιών.

Βάσει των παραπάνω και δεδομένου ότι τα εικονικά περιβάλλοντα προσφέρουν δυνατότητες στον χρήστη που υπερβαίνουν τα όρια του φυσικού κόσμου, θα μπορούσαμε να διακρίνουμε σε δυο κατηγορίες τους ανθρώπους που ασχολούνται ή ενδιαφέρονται για τα εικονικά περιβάλλοντα: σε αυτούς οι οποίοι επιθυμούν να χρησιμοποιήσουν την εν λόγω τεχνολογία για να εξελίξουν το επάγγελμα ή το ενδιαφέρον τους πάνω σε ένα συγκεκριμένο αντικείμενο με τη βοήθεια των εικονικών περιβαλλόντων, καθώς και σε εκείνους που επιθυμούν να αναπτύξουν και να βελτιώσουν την ίδια την τεχνολογία ανάπτυξης των εικονικών περιβαλλόντων. Η εν λόγω μεταπτυχιακή διατριβή μας κατατάσσει στη δεύτερη κατηγορία.

Παρά τη χρησιμότητα των εικονικών περιβαλλόντων σε διάφορους τομείς και την πληθώρα έρευνας που έχει πραγματοποιηθεί γύρω από αυτά τις τελευταίες δεκαετίες, ο χώρος παρουσιάζει ακόμα σημαντικές ελλείψεις και επιδέχεται βελτιώσεων. Το πρωτεύον ζήτημα είναι πως το εξειδικευμένο υλικό που χρησιμοποιείται είναι, αφενός, υπερβολικά ακριβό για τον μέσο χρήστη, αφετέρου, υπάρχουν σημαντικοί τεχνικοί περιορισμοί όσον αφορά την επεξεργαστική ισχύ, την ανάλυση εικόνας και το εύρος ζώνης για την επικοινωνία. Κατά συνέπεια, ένα μεγάλο μέρος της έρευνας επικεντρώνεται στα εικονικά περιβάλλοντα οθόνης, που φαίνεται να είναι η πιο οικονομική και προσιτή λύση για το μέσο χρήστη.

Επίσης, φαίνεται να υπάρχει έλλειψη γενικού τύπου αρχιτεκτονικών και εργαλείων ανάπτυξης για εικονικά περιβάλλοντα, με αποτέλεσμα να είναι ιδιαίτερα επίπονη και χρονοβόρα η διαδικασία κατασκευής τους. Επιπρόσθετα, απαιτείται και σημαντικό χρονικό διάστημα για τον έλεγχο και την αποσφαλμάτωσή, ώστε να παρέχεται στο χρήστη ένα αξιόπιστο και φιλικό περιβάλλον αλληλεπίδρασης. Παραταύτα, οι υπέρμαχοι τηςτεχνολογίας υποστηρίζουν πως οι ανωτέρω περιορισμοί θα ξεπεραστούν, καθώς οι τεχνολογίες επικοινωνίας, απεικόνισης και επεξεργασίας γίνονται πιο ισχυρές και αποδοτικές με την πάροδο του χρόνου, όπως έχει δείξει η ραγδαία ανάπτυξητης πληροφορικής τις τελευταίες δεκαετίες.

1.5 Ευφυές Εικονικό Περιβάλλον

Ένα εικονικό περιβάλλον, ενώ είναι εντυπωσιακό και ελκυστικό, αν είναι στατικό και δεν υφίσταται αλλαγές, η εμπειρία εμβύθισης που προκαλεί έχει περιορισμένο ενδιαφέρον. Επομένως, ήταν απαραίτητο να δημιουργηθεί ένας αυτοματοποιημένος τρόπος παραγωγής συμπεριφοράς στα εικονικά περιβάλλοντα, ώστε να είναι πιο δυναμικά και ενδιαφέροντα, επεκτείνοντας κατά αυτόν τον τρόπο τις δυνατότητες εφαρμογής τους. Η ευκαιρία δόθηκε κυρίαρχα με την αύξηση της επεξεργαστικής ισχύος που χρησιμοποιείται για το rendering, καθιστώντας εφικτό ένα κομμάτι αυτής της ισχύος να αφιερωθεί σε θέματα πέραν της ρεαλιστικής απεικόνισης, εγκαινιάζονταςένα νέο επιστημονικό πεδίο.

«*Η αυξανόμενη επικάλυψη της τεχνολογίας που αφορά τα τρισδιάστατα διαδραστικά γραφικά περιβάλλοντα σε πραγματικό χρόνο και της τεχνολογίας της Τεχνητής Νοημοσύνης οδήγησε στην ανάπτυξη ενός νέου επιστημονικού τομέα, γνωστού με τον όρο Ευφυή Εικονικά Περιβάλλοντα».* Με άλλα λόγια, αποτελεί*«συνδυασμό έξυπνων τεχνικών και εργαλείων, που ενσωματώθηκαν σε αυτόνομες οντότητες (πράκτορες) μαζί με αποδοτικά μέσα για την γραφική αναπαράστασή τους και την δυνατότητα αλληλεπίδρασής τους*» (Aylett R., Luck M., 2000).

Πιο απλά, με τον όρο Ευφυή Εικονικά Περιβάλλοντα (IntelligentVirtualEnvironments) ορίζεται η σύγκλιση δυο διαφορετικών επιστημονικών πεδίων, της Τεχνητής Νοημοσύνης και της Εικονικής Πραγματικότητας, επιτρέποντας ευφυείς οντότητες να αναπαρίστανται και να δρουν αυτόνομα σε ένα εικονικό περιβάλλον. Μπορούμε να συναντήσουμε τον όρο και ως Ευφυή Συστήματα Εικονικής Πραγματικότητας (Intelligent Virtual Reality Systems), καθώς η νοημοσύνη ενσωματώνεται στην ίδια την αρχιτεκτονική του συστήματος, υλοποιώντας αλγορίθμους Τεχνητής Νοημοσύνης σε ένα σύστημα εικονικής πραγματικότητας.

Υπάρχουν διάφοροι παράγοντες που οδήγησαν στην διασύνδεση αυτών των δυο πεδίων, εκ των οποίων διακρίνουμε τους εξής τρεις: πρώτον, η συνεχόμενη αύξηση του μεγέθους υπολογιστικής ισχύος που μπορεί να σηκώσει ένας επιτραπέζιος υπολογιστής, όχι μόνο υποστηρίζει έναν υψηλότερο βαθμό ρεαλιστικής απεικόνισης, αλλά αφήνει διαθέσιμη υπολογιστική ισχύ που μπορεί να χρησιμοποιηθεί και για την πρόσθεση νοημοσύνης. Ο δεύτερος παράγοντας σχετίζεται με την εξέλιξη και την διαδεδομένη διαθεσιμότητα λογισμικού για 3D γραφικά και την ανάπτυξη προτύπων για 3D γραφικά, όπως η VRML '97. Τρίτον, οι τεχνολογίες Τεχνητής Νοημοσύνης, όπως η επεξεργασία φυσικής γλώσσας, έχουν επίσης αναπτυχθεί στο βαθμό που μπορούν να χρησιμοποιηθούν ως μέσο διάδρασης εντός ενός εικονικού περιβάλλοντος.

1.6 Εικονικοι Πρακτορες

Όπως αναφέρθηκε, στα ευφυή εικονικά περιβάλλοντα λειτουργούν οντότητες, γνωστές ως εικονικοί πράκτορες (virtual agents), που μπορούν να επιδράσουν με το περιβάλλον και να προκαλέσουν κάποιου είδους μεταβολή σε αυτό. Πιο συγκεκριμένα, ως εικονικοί πράκτορες ορίζονται οι αυτόνομες οντότητες σε ένα εικονικό περιβάλλον, οι οποίες έχουν συνήθως τη μορφή συνθετικών χαρακτήρων και αλληλεπιδρούν με το περιβάλλον μέσω αισθητήρων (sensors) και επιδραστών (effectors), λαμβάνοντας αποφάσεις με βάση κάποιο μοντέλο συμπεριφοράς. Οι χαρακτήρες αυτοί μπορούν να αναπαριστούν ανθρώπους, ζώα ή ακόμα και φανταστικά πλάσματα, ενώ σε κάποιες περιπτώσεις μπορούν να έχουν και πιο «άψυχη» μορφή, όπως αυτοκίνητα, ρομπότ κ.ά., ανάλογα τον κόσμο στον οποίο ανήκουν και τον σκοπό που πρέπει να εξυπηρετήσουν.

Οι εικονικοί πράκτορες μπορούν να εντοπιστούν σε δυο άκρες του ίδιου φάσματος. Από τη μια έχουμε τους φυσικούς πράκτορες (physical agents) και από την άλλη τους γνωσιακούς πράκτορες (cognitive agents). Βασικό κριτήριο και στις δυο περιπτώσεις είναι η πιστευτότητα (believability) που νοείται ως «*ο βαθμός στον οποίον ο πράκτορας υποστηρίζει ή υπονομεύει την καθολική αίσθηση παρουσίας του χρήστη στο περιβάλλον*» (Aylett R., Cavazza M., 2001). Η πιστευτότητα προκύπτει από την εφαρμογή φυσικών νόμων που ισχύουν στον πραγματικό κόσμο και παιζει τεράστιο ρόλο στην αίσθηση της παρουσίας. Σε αντίθεση με την αληθοφάνεια ή ρεαλισμό, που αφορά το βαθμό ομοιότητας ενός εικονικού κόσμου με τον πραγματικό, η πιστευτότητα σχετίζεται περισσότερο με την συμπεριφορά, παρά με την οπτική αναπαράσταση.

1.5.1. Φυσικοί Πράκτορες

Στους φυσικούς πράκτορες δίνεται έμφαση στην πιστευτή φυσική συμπεριφορά τους σε ένα εικονικό περιβάλλον. Επίσης, πρέπει να διέπονται από ρεαλιστική κινησιολογία και δυνατότητα φυσικής αλληλεπίδρασης με τον εικονικό κόσμο, εμπλουτιζόμενοι με σώμα και δυνατότητα κίνησης του, ενδεχομένως και με εκφράσεις προσώπου. Συνήθως, η επίδραση πάνω στο εικονικό περιβάλλον γίνεται με εικονικούς αισθητήρες που δε λειτουργούν σε συμβολικό επίπεδο αλλά προσομοιώνουν το φυσικό. Για παράδειγμα, ένας απλός εικονικός αισθητήρας μπορεί να αποτελείται από μια ευθεία που προβάλλεται από τα μάτια του εικονικού πράκτορα και επιστρέφει πληροφορία για οποιοδήποτε αντικείμενο του εικονικού κόσμου με το οποίο τέμνεται, αντλώντας τις δομές δεδομένων που το αναπαριστούν για την ταυτότητα και τις ιδιότητές του.

Επομένως, οι φυσικοί πράκτορες εγείρουν διάφορα ζητήματα που πρέπει να αντιμετωπιστούν. Το πρωτεύον είναι η δομή του σώματος του πράκτορα και η ικανότητα κίνησής και μετακίνησής του. Από τη στιγμή που ο πράκτορας αποκτά δυνατότητα κίνησης, προκύπτει το ζήτημα της αποφυγής σύγκρουσεώς του με άλλα αντικείμενα στο περιβάλλον, συνεπώς η ικανότητα αντίληψης και αναγνώρισης αυτών των αντικειμένων ως συμπαγή. Τέτοια αντικείμενα μπορεί να είναι στατικά, όπως ένα κτίριο ή ένα δέντρο, ή εν κινήσει, όπως άλλοι πράκτορες στο περιβάλλον. Τρίτον, δοσμένης μιας ρεαλιστικής φυσικής αναπαράστασης ενός ενσώματου πράκτορα και μιας γκάμας συμπεριφορών, προκύπτουν ζητήματα αυτονομίας, όσον αφορά τον έλεγχο του σώματος και της επίδρασής του στο περιβάλλον, ώστε η κίνηση να κατευθύνεται από εσωτερική ενεργοποίηση του σώματος του πράκτορα.

Πιο συγκεκριμένα, τα ζητήματα ελέγχου αφορούν το βαθμό και τον τρόπο με τον οποίο ένας πράκτορας μπορεί να επιδράσει πάνω στο περιβάλλον, δηλαδή αν η δυνατότητα αυτή περιορίζεται μηχανικά στο επίπεδο ενσωμάτωσης μυών ή επεκτείνεται στο επίπεδο συμπεριφορικών πράξεων, όπως το περπάτημα και το πιάσιμο ενός αντικειμένου, η χειραψία με κάποιον άλλον πράκτορα ή και στα δυο επίπεδα (Aylett R., Luck M., 2000). Τέτοια ζητήματα έχουν απασχολήσει τους επιστήμονες και στον κλάδο της ρομποτικής. Η κυρίαρχη διαφορά, όμως, είναι πως υπάρχει μεγαλύτερος βαθμός ελευθερίας ως προς την αναπαράσταση του ενσώματου πράκτορα στους εικονικούς κόσμους και οι εικονικοί πράκτορες γνωρίζουν διαρκώς τη θέση τους στο χώρο σε αντίθεση με ένα μηχανικό ρομπότ. Επιπλέον, οι εικονικοί αισθητήρες δε χρειάζεται να πάσχουν από τα προβλήματα που υπάρχουν στους αληθινούς. Από την άλλη, οι φυσικοί νόμοι που ισχύουν στον πραγματικό κόσμο, όπως η βαρύτητα, η αδράνεια, η τριβή κ.ά., πρέπει να προστεθούν αναλυτικά από τον σχεδιαστή στον εικονικό κόσμο.

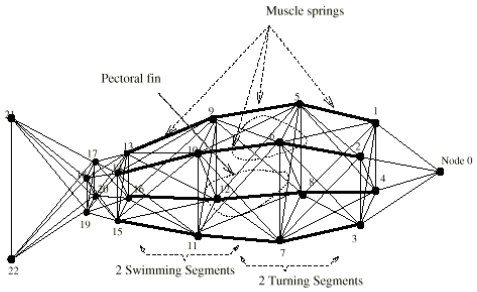
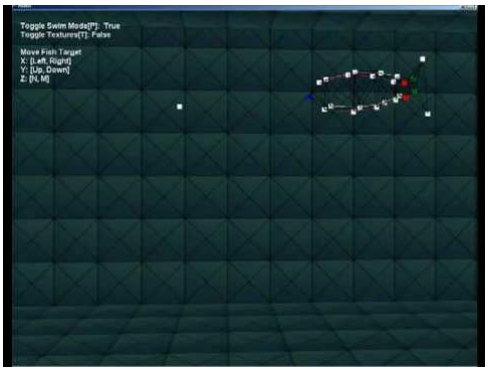


Figure 3: Dynamic fish model. Nodes are lumped masses. Lines are springs (shown at their natural lengths). Bold lines are muscle springs.



Εικόνα 1.15: Terzopoulo's fish

Ένα δημοφιλές παράδειγμα φυσικού πράκτορα είναι το ψάρι του Τερζόπουλου (Terzopoulos' fish, 1994), που έχει ως βάση δομής έναν σκελετό για το σώμα του ψαριού και συνδυάζει διάφορα μαθηματικά μοντέλα για να προσομοιώσει την κίνηση του. Όπως ένα πραγματικό ψάρι, όταν κουνάει την ουρά του επηρεάζει και το νερό που το περιβάλλει, το οποίο κινείται ανάλογα, δίνοντας ώθηση στο ψάρι για να μετακινηθεί. Σε μετέπειτα βελτίωση του μοντέλου, δίνεται έμφαση και στην αντίληψη του ψαριού, χρησιμοποιώντας ένα βιολογικά ακριβές σύστημα αντίληψης για εικονικούς πράκτορες. Σύμφωνα με αυτό, το πεδίο όρασης του πράκτορα

προβάλλεται σε έναν προσομοιωμένο αμφιβληστροειδή και χρησιμοποιούνται αλγόριθμοι από την τεχνητή όραση για να επεξεργαστούν τα pixels και να μετατραπούν σε μια αναγνωρίσιμη μορφή.

1.5.2. Γνωσιακοί Πράκτορες

Στους γνωσιακούς πράκτορες η έμφαση δίνεται στην προσομοίωση της ανθρώπινης γνωσιακής συμπεριφοράς και στη γνωσιακή αλληλεπίδραση με τον χρήστη του συστήματος. Τα ζητήματα που πρέπει να αντιμετωπιστούν αφορούν τη φυσική γλώσσα και γνωσιακές διαδικασίες, όπως ο σχεδιασμός ενεργειών (planning). Τέτοιοι πράκτορες συχνά λαμβάνουν συμβολική πληροφορία κατευθείαν από το εικονικό περιβάλλον, καθιστώντας λιγότερο διακριτό κατά πόσο η διαδικασία αίσθησής τους είναι πραγματικά αυτόνομη.

Αντίστοιχα με τους φυσικούς πράκτορες, πρέπει και εδώ να αντιμετωπιστούν ορισμένα ζητήματα που αφορούν την ανάπτυξη και την κατασκευή των γνωσιακών πρακτόρων. Πρώτον, πρέπει να υπάρχει ένα γενικό συστατικό στοιχείο της αρχιτεκτονικής του πράκτορα, υπεύθυνο για τις κριτικές γνωσιακές ικανότητές του, όπως η λογική, η ικανότητα λήψης απόφασης, ο σχεδιασμός ενεργειών, η δυνατότητα μάθησης κτλ., ανεξάρτητα από το εικονικό περιβάλλον στο οποίο μπορεί να τοποθετηθεί ο πράκτορας. Αυτό το πρόβλημα μπορεί να θεωρηθεί ως η βάση πάνω στην οποία στηρίζονται τα υπόλοιπα τμήματα της αρχιτεκτονικής ενός γνωσιακού πράκτορα και απασχολεί γενικότερα τη σχεδίαση ευφυών πρακτόρων. Δεύτερον, πρέπει ο πράκτορας να είναι αληθοφανής στο περιβάλλον στο οποίο βρίσκεται, όσον αφορά την συμπεριφορά του, όμως,και όχι την εικονική αναπαράστασή του.

Τα ευφυή εικονικά περιβάλλοντα θα πρέπει να είναι αληθοφανή τόσο σε ό,τι αφορά τις ενέργειες των πρακτόρων όσο και στις αλληλεπιδράσεις τους με άλλους πράκτορες ή χρήστες. Αυτό πρακτικά σημαίνει ότι το γνωσιακό τμήμα δεν πρέπει να είναι αποκομμένο από τα στοιχεία που επηρεάζουν τις αποφάσεις, δηλαδή κίνητρα, συναισθήματα, προσωπικότητα κλπ., τα οποία θα πρέπει να είναι σε θέση να εκφραστούν στο εικονικό περιβάλλον.Το τρίτο ζήτημα σχετίζεται με την εικονική αναπαράσταση του πράκτορα και των ενεργειών του, η οποία είναι σημαντική και μπορεί να είναι αποδοτική, μόνο εφόσον τα μοντέλα πρακτόρων παρέχουν επαρκείς λεπτομερείς πληροφορίες.

Για παράδειγμα, οι Badler et al. (1997) περιγράφουν ένα απλό μοντέλο στο οποίο η προσωπικότητα μπορεί να επηρεάσει τις ενέργειες του γνωσιακού πράκτορα, όπως π.χ. η περιέργεια ή η κούραση μπορεί να συσχετιστεί με διάφορες παραμέτρους της κίνησης ενός εικονικού πράκτορα, π.χ. ταχύτητα και πρόβλεψη. Ενώ το ίδιο το μοντέλο είναι σχετικά απλό, αποτελεί ένα ενδεικτικό παράδειγμα σύνδεσης των υψηλού επιπέδου νοητικών τμημάτων του πράκτορα με τη χαμηλού επιπέδου φυσική έκφρασή τους.

Σχετικά με τις παραπάνω κατηγορίες, αναφερόμαστε σε φάσμα και όχι σε διακριτές κατηγορίες πρακτόρων, καθώς οι πράκτορες αυτοί στην πραγματικότητα αλληλοδιαπλέκονται. Οι γνωσιακοί πράκτορες απαιτούν κάποιο βαθμό φυσικής αλληλεπίδρασης με ένα εικονικό περιβάλλον, όπως οι φυσικοί, ενώ οι φυσικοί πράκτορες συχνά απαιτούν κάποιου είδους έλεγχο σε γνωσιακό επίπεδο, όπως οι γνωσιακοί. Κατά αυτόν τον τρόπο, κάποιος θα μπορούσε να χαρακτηρίσει τη γνωσιακή άκρη του φάσματος ως δουλειά από τη συλλογιστική προς τα έξω, ενώ την φυσική άκρη του φάσματος ως δουλειά από το σώμα προς τα μέσα (Aylett R., Luck M., 2000). Ιδανικά, οι εικονικοί πράκτορες πρέπει να συνδυάζουν την αληθοφανή κίνηση και τη φυσική αλληλεπίδραση με γνωσιακές ικανότητες που αντιστοιχούν σε αυτές ενός ανθρώπινου όντος.

1.5.3. Ευφυείς Πράκτορες

Ένας εικονικός πράκτορας που δεν ακολουθεί προκαθορισμένα βήματα, αλλά μπορεί να παίρνει αποφάσεις από μόνος του και χρησιμοποιεί τεχνικές από το χώρο της Τεχνητής Νοημοσύνης για να επιτυγχάνει τους στόχους του, ονομάζεται ευφυής εικονικός πράκτορας (intelligent virtual agent). Είναι αρκετά δύσκολο να ορίσουμε τι ακριβώς είναι αυτό που κάνει έναν πράκτορα «ευφυή». Ωστόσο, μπορούμε γενικά να πούμε ότι ένας πράκτορας είναι ευφυής όταν έχει την ικανότητα να επιτελεί τους στόχους και τα καθήκοντα που έχει επιφορτιστεί. Έτσι, σε ένα ελάχιστο επίπεδο νοημοσύνης, μπορεί να δίνονται στον πράκτορα εντολές με τη μορφή κανόνων και αυτός να ενεργεί με τη βοήθεια κάποιου μηχανισμού εξαγωγής συμπερασμάτων. Σε ένα

ανώτερο επίπεδο, ο πράκτορας είναι ικανός να μαθαίνει και να προσαρμόζεται αυτόματα στο περιβάλλον έτσι, ώστε να πετυχαίνει τους στόχους του.

Αναφορικά με τα παραπάνω, ένας ευφυής εικονικός πράκτορας μπορεί να έχει σώμα και μάλιστα, το σώμα του πρέπει να κινείται με έναν ρεαλιστικό τρόπο για να πειστικό και πιστευτό. Ως εκ τούτου, το σώμα πρέπει να έχει δομή, επιφάνεια και γεωμετρία, ισοδύναμη με αυτή ενός σκελετού,άρα το σώμα να καταλαμβάνει όγκο στον εικονικό χώρο και να έχει θέση και προσανατολισμό. Επίσης, ο πράκτορας πρέπει να εκδηλώνει κάποια συμπεριφορά για τη χρήση αυτού του σώματος και πρέπει να υπάρχει αντιστοιχία ανάμεσα στη συμπεριφορά του πράκτορα και την κατάσταση του εικονικού περιβάλλοντος.

Ο όρος συμπεριφορά στο πεδίο της Τεχνητής Νοημοσύνης αναφέρεται σε ένα σύστημα ελέγχου που βασίζεται στις αισθήσεις (Brooks, 1986; Brooks, 1991), στο οποίο τα εισερχόμενα ερεθίσματα αντιστοιχίζονται σε εξερχόμενες αντιδράσεις. Η συμπεριφορά αυτή μπορεί να διαφέρει ανάλογα την εφαρμογή. Για παράδειγμα, μπορεί σε ορισμένες περιπτώσεις να εστιάζει περισσότερο στη φυσική αλληλεπίδραση με το περιβάλλον, ενώ σε άλλες να εστιάζει στη γνωσιακή συμπεριφορά, εκφραζόμενη μέσω ομιλίας ή φυσικής γλώσσας γενικά.

Επομένως, ο πράκτορας πρέπει να έχει δυνατότητα αντίληψης του περιβάλλοντος και δυνατότητα δράσης πάνω σε αυτό, πράγμα που επιτυγχάνεται με τη χρήση εικονικών αισθητήρων και επιδραστών (μηχανισμοί εφαρμογής ενεργειών στον εικονικό κόσμο) αντίστοιχα. Τέλος, η συμπεριφορά του πράκτορα πρέπει να κατευθύνεται μέσω μιας αρχιτεκτονικής ελέγχου, ώστε να διέπεται από κάποιον βαθμό αυτονομίας, όσον αφορά την δράση στο εικονικό περιβάλλον (Aylett R., Cavazza M., 2001). Συνεπώς, ο πράκτορας θα πρέπει να είναι σε θέση να παίρνει αποφάσεις μέσω μιας συλλογιστικής διαδικασίας και να κρίνει πού, πότε, πώς και αν πρέπει να εκτελέσει μια ενέργεια, χωρίς να ελέγχεται από τον ανθρώπινο παράγοντα.

Ένα δημοφιλές παράδειγμα ευφυούς πράκτορα είναι ο STEVE (Soar Training Expert for Virtual Environments). Ο STEVE (Rickel and Johnson, 1997) είναι ένας παιδαγωγικός πράκτορας που κατοικεί σε ένα εικονικό περιβάλλον, το οποίο επιβλέπει και ελέγχει μέσω ενεργειών με τη χρήση εικονικών κινητήρων. Υλοποιήθηκε στο λογισμικό VET (Virtual Environments for Training) που επιτρέπει στο περιβάλλον, τον πράκτορα και τον μαθητή να τρέχουν ως ξεχωριστές διαδικασίες και χρησιμοποιεί έναν μηχανισμό προώθησης μηνυμάτων μεταξύ των συστατικών για να αναπαραστήσει τη ροή των γεγονότων στον κόσμο. Οι άνθρωποι αλληλεπιδρούν με τον εικονικό κόσμο μέσω HDM, 3D ποντικιού ή διαδραστικού γαντιού. Ο STEVE μπορεί να αναπαρασταθεί είτε ως μέρος ενός σώματος (ως εικονικό χέρι που μπορεί να δείχνει και να πιάνει αντικείμενα) ή ως ενσώματος πράκτορας (εμφανιζόμενος ως ανθρώπινη φιγούρα με κεφάλι-κορμό-χέρια) χωρίς να επηρεάζεται το γνωσιακό επίπεδο.

Εικόνα 1.16: Ο STEVE δίνει οδηγίες στο χρήστη (αριστερά) και σε άλλον πράκτορα (δεξιά)

1.5.4. Άλλες κατηγορίες πρακτόρων

Εκτός από τους εικονικούς πράκτορες υπάρχουν και άλλες, παραπλήσιες κατηγορίες συνθετικών χαρακτήρων σε εικονικά περιβάλλοντα που συναντώνται στη διεθνή βιβλιογραφία, καθώς η έρευνα είναι σε πολλά σημεία κοινή. Οι κατηγορίες αυτές είναι οι παρακάτω:

• εικονικός ηθοποιός (virtual actor) (Shawver, 1997; Wavish and Connah, 1997): Διαφέρει από έναν εικονικό πράκτορα στο ότι δεν έχει αυτονομία. Όλες οι ενέργειές του είναι προκαθορισμένες από κάποιο σενάριο ή βασίζονται σε σχετικές εντολές του χρήστη (Cavazza et al., 1998).

• ενσάρκωση (avatar): Έχει οριστεί από τον Peterson (2005) ως εκδήλωσητου εαυτού σε έναν εικονικό κόσμο και έχει σχεδιαστεί για να ενισχύσει την αλληλεπίδρασησε σε έναν εικονικό κόσμο. Τα avatars επιτρέπουν στον χρήστη να λάβει μια ορατή μορφή στον εικονικό κόσμο, παρέχοντάς του την ευκαιρία να συμμετάσχει σε σουρεαλιστικές και φανταστικές εμπειρίες που υπερβαίνουν τον πραγματικό κόσμο στον οποίο βρίσκεται (Faloon G., 2010) Με λίγα λόγια, εξυπηρετεί ως εικονικός εκπρόσωπος ενός πραγματικού ανθρώπου σε έναν εικονικό κόσμο και οι ενέργειές του ελέγχονται (σε πραγματικό χρόνο) από τον χρήστη. Συνεπώς, εκτός από έλλειψη αυτονομίας, χαρακτηρίζεται και από έλλειψη αισθητήρων, καθώς η αντίληψη του χώρου γίνεται από τον ίδιο τοχρήστη.

• εικονικός άνθρωπος (virtual human) (Magnetat Thalmann and Thalmann, 1998; Gratch et al., 2000): Είναι μια ειδική περίπτωση εικονικού πράκτορα στην οποία έχει δοθεί έμφαση στην λεπτομερή μοντελοποίηση και προσομοίωση της ανθρώπινης κίνησης και αλληλεπίδρασης. Συνήθως οι χαρακτήρες αυτοί έχουν ιδιαίτερα περίπλοκους αισθητήρες και επιδραστές αλλά χαμηλό επίπεδο αυτονομίας.

1.7 Εικονικός Κόσμος

Ένας κόσμος νοείται ως η αναπαράσταση ενός συνόλου αντικειμένων, οπότε ένας εικονικός κόσμος ορίζεται ως ένα σύνολο εικονικών αντικειμένων. Τα εικονικά αντικείμενα μπορεί να αναπαριστούν οτιδήποτε, είτε ζωντανές οντότητες, όπως άνθρωποι, ζώα, έντομα κ.ά είτε άψυχα πράγματα, όπως καρέκλες, κτίρια, οχήματα κ.ά. Ακόμα, αντικείμενα ενός κόσμου αποτελούν και οι εικονικοί πράκτορες που μπορεί να δρουν σε αυτόν ή η αναπαράσταση του ίδιου του ανθρώπου-χρήστη στον κόσμο (avatar).

Η μορφή και ο χαρακτήρας των εικονικών αντικειμένων σχετίζεται με το περιεχόμενο που θέλουμε να προσδώσουμε σε έναν εικονικό κόσμο. Σε κάθε περίπτωση, όμως, τα εικονικά αντικείμενα διακρίνονται από συγκεκριμένα χαρακτηριστικά που τα καθιστούν αναγνωρίσιμα από τον χρήστη και του δίνουν την εντύπωση της αληθοφάνειας. Τέτοια χαρακτηριστικά είναι οι διαστάσεις των αντικειμένων, το μέγεθός τους, το σχήμα τους, το χρώμα τους, η υφή τους, το υλικό από το οποίο αποτελούνται και η συγκεκριμένη θέση τους στον τριδιάστατο εικονικό χώρο.

Η αναπαράσταση αποτελεί μια διαδικασία μέσω της οποίας δημιουργούμε ένα αντικείμενο ή σύνολο αντικειμένων που διεγείρει τις αισθήσεις του ανθρώπου με τέτοιο τρόπο, ώστε να παράγεται στον νου του η εικόνα αυτών των αντικειμένων. Η έννοια δεν αφορά μόνο τον φυσικό κόσμο, αλλά οποιονδήποτε κόσμο (μυθολογικό, ιστορικό, φανταστικό κτλ) και παίρνει διάφορες μορφές. Η αναπαράσταση χρονολογείται από την προϊστορική περίοδο, με τις πρώτες σπηλαιογραφίες όπου οι άνθρωποι απεικόνιζαν την καθημερινότητά τους. Επίσης, από την αρχαιότητα, η ανάπτυξη του θεάτρου, των τεχνών και των γραμμάτων οδήγησε στην αναπαράσταση κόσμων και αντικειμένων, με το ανέβασμα θεατρικών παραστάσεων, την καταγραφή μύθων ακόμα και την τέλεση θρησκευτικών τελετουργιών, μέσω των οποίων οι άνθρωποι προσπαθούσαν να έρθουν σε επαφή με έναν διαφορετικό κόσμο από αυτόν που ζούσαν. Επιπρόσθετα, με τη ζωγραφική και τη γλυπτική έγινε δυνατή μια διαφορετική οπτική αποτύπωση ανθρώπων, ζώων, τοπίων και καταστάσεων. Αιχμή στην οπτική αναπαράσταση αποτέλεσε η ανακάλυψη της φωτογραφίας και βέβαια, η επακόλουθη δημιουργία της κινούμενης εικόνας (video). Τέλος, το πιο σύγχρονο μέσο αναπαράστασης είναι τα τρισδιάστατα γραφικά σε υπολογιστές.

Η αναπαράσταση εικονικών κόσμων, ωστόσο, δεν στοχεύει μόνο στην οπτική απεικόνιση, αλλά προσπαθεί να διεγείρει όλες τις αισθήσεις. Όταν αυτό δεν είναι εφικτό, έμφαση δίνεται στην αίσθηση της όρασης και στην αληθοφανή οπτικοποίηση των αντικειμένων. Όπως αναφέρθηκε στην αρχή του κεφαλαίου, υπάρχουν

διάφορα σύγχρονα μέσα και εργαλεία εικονικής πραγματικότητας που στοχεύουν στη διέγερση των αισθήσεων του χρήστη, όπως τα HDMs και τα απτικά γάντια, εξού και τα αντικείμενα στον κόσμο πρέπει να γίνονται αντιληπτά από άλλες οντότητες, είτε αυτές είναι αυτόνομοι πράκτορες που κινούνται στον εικονικό κόσμο είτε ο ίδιος ο ανθρώπινος παράγοντας μέσω του avatar του. Ακόμα, κάποια αντικείμενα μπορεί να προσφέρουν και τη δυνατότητα αλληλεπίδρασης με έναν πράκτορα ή χρήστη.

Συμπυκνώνοντας τα παραπάνω, ο εικονικός κόσμος προκύπτει ως αποτέλεσμα της λειτουργίας του εικονικού περιβάλλοντος ή της εικονικής πραγματικότητας και αποτελεί την αναπαράσταση ενός κόσμου με τη χρήση ηλεκτρονικών υπολογιστών. Επομένως, μπορούμε να ορίσουμε τον εικονικό κόσμο ως την αναπαράσταση ενός τεχνητού κόσμου, ο οποίος αποτελείται από ένα σύνολο εικονικών αντικειμένων και δημιουργεί στον χρήστη την εντύπωση πως βρίσκεται μέσα στον κόσμο, διεγείροντας όλες ή μέρος των αισθήσεών του.

Πρωτεύοντα χαρακτηριστικά, λοιπόν, ενός εικονικού κόσμου είναι η πιστευτότητα (believability), η εμβύθιση (immersion) και ο ρεαλισμός (realism). Η πιστευτότητα αφορά στην εφαρμογή φυσικών νόμων που ισχύουν και στον πραγματικό κόσμο και στην αληθοφανή αλληλεπίδραση μιας οντότητας – αυτόνομου ευφυούς πράκτορα ή avatar – με τον κόσμο, ώστε η συμπεριφορά εντός του κόσμου να είναι αξιόπιστη. Επομένως, όταν ένας χρήστης αλληλεπιδρά με τον εικονικό κόσμο αναμένει κάποια πράγματα να συμβούν με συγκεκριμένο τρόπο, για παράδειγμα η βαρύτητα πρέπει να προκαλεί τα αντικείμενα να πέφτουν και ο ήχος να εξασθενίζει όσο απομακρύνεται κανείς από την πηγή ή να δυναμώνει στην αντίθετη περίπτωση, προσομοιώνοντας τους φυσικούς νόμους του πραγματικού κόσμου στον εικονικό.

Όσο πιο πιστευτός είναι ένας εικονικός κόσμος, δηλαδή όσο πιο πιστός στους φυσικούς κανόνες του πραγματικού κόσμου, τόσο περισσότερο αυξάνεται και η αίσθηση παρουσίας του χρήστη σε αυτόν, η εμβύθιση, δηλαδή η αποκοπή των αισθήσεων του από τον πραγματικό κόσμο και ο προσανατολισμός των αισθήσεων και της προσοχής του στον εικονικό κόσμο.

Τέλος, ο ρεαλισμός σχετίζεται με την οπτική αναπαράσταση του κόσμου, δηλαδή την απεικόνιση, και αφορά το βαθμό ομοιότητας του εικονικού κόσμου με τον πραγματικό. Έρευνες έχουν δείξει πως η δυνατότητα να αγγίξεις εικονικά αντικείμενα, κάνει τα αντικείμενα αυτά και τον εικονικό κόσμο πολύ πιο αληθοφανή. Σύμφωνα με τον Hoffman[1998], όταν αναμνήσεις ανάμεικτης πραγματικότητας προσεγγίζουν πραγματικές αναμνήσεις, οι άνθρωποι είναι πιο πιθανό να μπερδεύουν πραγματικά και εικονικά αντικείμενα, ανάλογα με τον βαθμό ρεαλισμού.

Μια ενδιαφέρουσα προσέγγιση για τον ορισμό του εικονικού κόσμου είναι του Schroeder, ο οποίος υποστηρίζει πως η διαφορά μεταξύ εικονικής πραγματικότητας/εικονικών περιβαλλόντων και εικονικών κόσμων είναι πως ο όρος «εικονικός κόσμος» έχει καθιερωθεί για τους διαρκείς διαδικτυακούς κοινωνικούς χώρους, δηλαδή για τα εικονικά περιβάλλοντα που είναι πολυπληθή, οι άνθρωποι τα αντιλαμβάνονται ως συνεχή στο χρόνο και τα βιώνουν με άλλους ανθρώπους σαν έναν χώρο κοινωνικής αλληλεπίδρασης. Έτσι λοιπόν, οι εικονικοί κόσμοι μπορούν να διακριθούν από τα διαδικτυακά παιχνίδια και τα MMORPGs (Massively Multiplayer Online Roleplaying Games) στο ότι αποτελούν έναν τρίτο χώρο, έναν διαδικτυακό χώρο για κοινωνικοποίηση. Ειδωμένο από την αντίστροφη, τα διαδικτυακά παιχνίδια είναι ένα υποσύνολο εικονικών κόσμων, αυτών στους οποίους η δραστηριότητά τους περιστρέφεται γύρω από το παιχνίδι (gaming).

Οι εικονικοί κόσμοι έχουν αρκετά κοινά χαρακτηριστικά και λειτουργίες που τους κάνουν ελκυστικούς στους χρήστες. Συνήθως λειτουργούν ως περιβάλλοντα εμβύθισης που θυμίζουν ηλεκτρονικά παιχνίδια, όπου οι συμετέχοντες χρησιμοποιούν μια ενσώματη αναπαράσταση τους εντός του κόσμου (avatar) για να αφοσιωθούν σε μια ποικιλία δραστηριοτήτων σε ένα κοινόχρηστο χώρο. Το κοινόχρηστο περιβάλλον είναι ένα θεμελιώδες χαρακτηριστικό των εικονικών κόσμων και παρέχει τη βάση κατανόησης για ποιο λόγο δημιουργούν ένα ελκυστικό περιβάλλον στους χρήστες. «*Ως οντότητες που κατοικούμε σε εναν τρισδιαστατο φυσικό κόσμο, ο εγκέφαλός μας έχει σχεδιαστεί για να αναμένει και να αποδέχεται ερεθίσματα που μοιάζουν με αυτά του πραγματικού κόσμου. Με άλλα λόγια, ανταποκρινόμαστε στη χωρική φύση των εικονικών κόσμων κατά κύριο λόγο επειδή μας θυμίζουν τον πραγματικό*».(Mannecke B., McNeill D., Roche E., Bray D., Townsend A., Lester J., 2008)

Υπάρχουν διάφορα παραδείγματα εικονικών κόσμων τα οποία μπορούμε να αναφέρουμε, όπως το World of Warcraft, Second Life, There, Active Worlds, Kaneva, Ever Quest, Final Fantasy, Star Wars Universe κ.ά., γνωστά και ως πολυ-χρηστικά εικονικά περιβάλλοντα (MUVEs – MultiUser Virtual Environments), καθως επιτρέπουν την ταυτόχρονη ύπαρξη και αλληλεπίδραση πολλών χρηστών σε αυτούς. Όλοι αυτοί οι αναφερόμενοι κόσμοι έχουν τη δυναμική να αλλάζουν τον τρόπο με τον οποίο οι άνθρωποι αλληλεπιδρούν, πλοηγούνται στο διαδίκτυο και διευθύνουν τις επιχειρήσεις τους.

Η ραγδαία αναπτυξή τους οφείλεται,από τη μια, στα εξελιγμένα τριδιάστατα περιβάλλοντα που παρέχουν στους χρήστες εκπληκτικά γραφικά και, από την άλλη, στη δυνατότητα που παρέχουν στους χρήστες να υποκρίνονται φανταστικούς ρόλους και να δημιουργούν κοινότητες. Επίσης, η αλληλεπίδραση που προσφέρουν με άλλους ανθρώπους – φίλους ή γνωστούς, ακόμα και αγνώστους - έχει αυξήσει την δημοτικότητα τους. Όσο αυτά τα περιβάλλοντα γίνονται πιο διεισδυτικά,τόσο περισσότερη έρευνα απαιτείται για την καλύτερη κατανόηση αυτών των τρισδιάστατων χώρων. Ήδη πληροφοριακοί επιστήμονες εξερευνούν αυτό το πεδίο και μπορούν να μας διαφωτίσουν πάνω στο σχεδιασμό, την ανάπτυξη, τη διαχείριση και τη χρήση αυτών των εικονικών κόσμων.(Mannecke B., McNeill D., Roche E., Bray D., Townsend A., Lester J., 2008)

1.6.1. Ο Εικονικός Κόσμος Second Life

Σε αυτό το σημείο θα κάνουμε μία μικρή αναφορά στον εικονικό κόσμο Second Life, ένα ιδιαίτερο τέτοιο πολυχρηστικό περιβάλλον, που αποτελεί αιχμή στην εξέλιξη των εικονικών κόσμων και είναι από τα πιο δημοφιλή, καθώς έχει ξεπεράσει τους 25 εκατομμύρια χρήστες παγκοσμίως.

Το Second Life (SL) είναι ένα διαδικτυακό παιχνίδι υπολογιστή που αναπαριστά ένα εικονικό, τρισδιάστατο περιβάλλον, του οποίου η λειτουργία διαμορφώνεται εξ' ολοκλήρου από τους κατοίκους του. Το SL δόθηκε στο κοινό το 2003 από την εταιρεία Linden Lab, που εδρεύει στο San Francisco και ιδρύθηκε το 1999 από τον Philip Rosedale, προηγούμενο υπεύθυνο τεχνολογίας της εταιρίας πληροφορικής Real Networks. Η έμπνευση προήλθε από την περιγραφή ενός πανταχού, τρισδιάστατου, εικονικού κόσμου (metaverse) στο μυθιστόρημα Snow Crash (1992) και ο Rosedale οραματίστηκε έναν εικονικό κόσμο όπου οι χρήστες μπορούν να ψηφιοποιήσουν τα πάντα και να συνεργαστούν σε ένα τρισδιάστατο περιβάλλον που θα φτιαχνόταν από τους ίδιους.

Στο SL, που απέχει πολύ από τα συνηθισμένα πολυσυμμετοχικά διαδικτυακά παιχνίδια, δεν υπάρχουν προκαθορισμένοι κανόνες, οδηγίες και στόχοι που πρέπει να εκπληρωθούν. Το παιχνίδι δίνει τη δυνατότητα στους παίκτες του να δημιουργήσουν τον ψηφιακό τους εαυτό - avatar - και να ζήσουν μια παράλληλη πραγματικότητα μέσα από την οθόνη του υπολογιστή τους, χωρίς να υπάρχουν νικητές ή χαμένοι και πόντοι που υπάρχουν συνήθως σε άλλα ηλεκτρονικά παιχνίδια. Αναπαριστά - στο μέτρο του δυνατού - μία νέα, εναλλακτική πραγματικότητα, όπου ο καθένας μπορεί να παρουσιάσει τον εαυτό του όπως το επιθυμεί, απαλλαγμένος από τις συμβάσεις και τους περιορισμούς της πραγματικής του ζωής, ενώ έχουν δημιουργηθεί όλα αυτά που συνιστούν και τον πραγματικό κόσμο: σπίτια, νοσοκομεία, δρόμοι, πόλεις, μεταφορικά μέσα, επιχειρήσεις, θέατρα, κινηματογράφοι, οτιδήποτε μπορεί να συναντήσει κανείς και στην κανονική ζωή.

Στον κόσμο του SL μπορείς να δεις καθηγητές Πανεπιστημίου που διδάσκουν τους σπουδαστές τους, ωθώντας τούς να ανοίξουν εικονικά καταστήματα, μεσίτες ακινήτων που συμπληρώνουν το εισόδημα τους πουλώντας εικονικά οικόπεδα, ακόμα και μουσικούς ή άλλους καλλιτέχνες οι οποίοι παρουσιάζουν - σχεδόν ανέξοδα - τα έργα τους στα εκατομμύρια των κατοίκων του SL.

Εικόνα 1.717: SecondLifeόπως στην κανονική ζωή

Ωστόσο, η απουσία δηλωμένων στόχων είναι η ιδιαιτερότητα και το προτέρημα του SL, καθώς οι ψηφιακές απεικονίσεις των χρηστών (avatars) είναι σε θέση να κάνουν νέες κοινωνικές γνωριμίες και να εργάζονται κανονικά, είτε ως υπάλληλοι σε εικονικές εταιρίες, είτε προωθώντας τις δικές τους επιχειρηματικές ιδέες και τελικά να αποκτούν χρήματα (στην αρχή εικονικά και μετέπειτα πραγματικά), μέσω μιας πλήρως αυτόνομης οικονομίας που ανταμείβει την καινοτομία, τη δεξιοτεχνία και το ρίσκο. Επομένως, το παιχνίδι δίνει τη δυνατότητα στους χρήστες να εργαστούν και να επενδύσουν σε αυτό, αφού λειτουργεί με τα επιχειρηματικά κριτήρια της αγοράς που υπάρχουν και στην πραγματικότητα. Μάλιστα, μεγάλες πολυεθνικές εταιρείες δραστηριοποιούνται στον ψηφιακό χώρο του παιχνιδιού, βρίσκοντας έτσι πρόσφορο έδαφος για την προώθηση των προϊόντων τους, όπως οι Toyota, Mercedes, Nissan, Cisco, AOL, Philips, Sun, Intel, Sony, ING, Vodafone, Reuters κ.ά.

Εικόνα 1.718: Second Life στο Βερολίνο

Το παιχνίδι έχει ακόμα και το δικό του νόμισμα, το δολάριο Linden, που μπορεί ο οποιοσδήποτε παίχτης να το εξαργυρώσει με κανονικά χρήματα. Οι κάτοικοι μπορούν να αγοράσουν απευθείας Linden ή να κάνουν ανταλλαγές συναλλάγματος μεταξύ Linden και αμερικανικού δολαρίου, το οποίο και αποταμιεύεται στον πραγματικό τραπεζικό λογαριασμό τους, μέσω του συστήματος LindeX. Η ισοτιμία του δολαρίου Η.Π.Α. και Linden μεταβάλλεται συνεχώς, καθώς οι κάτοικοι του SL, μέσω των επιχειρηματικών και οικονομικών συναλλαγών τους, καθορίζουν την τιμή αγοράς και πώλησης των δολαρίων Linden. Αξίζει να σημειωθεί ότι τον Ιανουάριο του 2008 η αξία ενός αμερικανικού δολαρίου ήταν 267 L$, ενώ το συνολικό ΑΕΠ του SL άγγιξε τα 65 εκατομμύρια αμερικανικά δολάρια.

Βέβαια, καθώς πίσω από το Second Life κρύβονται πραγματικοί άνθρωποι, στον εικονικό κόσμο του παιχνιδιού έχουν παρατηρηθεί ορισμένα από τα φαινόμενα που υπάρχουν και στον πραγματικό κόσμο, όπως η ξενοφοβία, οι λεκτικές και σωματικές επιθέσεις, η βία, οι κοινωνικές αντιδράσεις και η κοινωνική απομόνωση.

Το βέβαιο είναι πως εικονικοί κόσμοι όπως το SL, αποτελούν μια νέα δύναμη στο χώρο της ηλεκτρονικής εκπαίδευσης. Σε σύγκριση με το game-based learning που αναφέρεται σε μικρή ομάδα ατόμων που εργάζονται απομονωμένα με περιορισμένες πηγές, η εταιρία Linden Lab εκμεταλλεύτηκε βασικές έννοιες του Web 2.0, όπως την συνεργατική δημιουργία (creative commons) και το λογισμικό ανοικτού κώδικα (open source), και δημιούργησε έναν εικονικό κόσμο στον οποίο εκπαιδευτικές εξομοιώσεις και προγράμματα μάθησης μπορούν σχετικά εύκολα και οικονομικά να εφαρμοστούν.

1.6.2. Εφαρμογές των Εικονικων Κοσμων

Οι εικονικοί κόσμοι είναι ιδιαίτερα χρήσιμοι και αποτελούν ισχυρά εκπαιδευτικά εργαλεία καθώς, πρώτον, δίνουν τη δυνατότητα στους χρήστες τους να εκτελούν καθήκοντα που μπορεί να είναι δύσκολα για αυτούς στον πραγματικό κόσμο εξαιτίας διάφορων περιορισμών (πχ κόστος, χρονοπρογραμματισμός, τοποθεσία). Δεύτερον, η διάρκεια (στο χρόνο) των εικονικών κόσμων επιτρέπει τη διατήρηση και την καλλιέργεια κοινωνικών σχέσεων που μπορούν να λειτουργήσουν ως βάση για τη συμμετοχιική εκπαίδευση και τρίτον, οι εικονικοί κόσμοι μπορούν να επεκταθούν και να προσαρμοστούν στις ανάγκες του χρήστη.

Οι Antonacci et al (2008) αναφέρουν πως η συσχέτιση με εικονικούς κόσμους επιτρέπει στους μαθητές να βιώσουν μαθησιακές ευκαιρίες που δε θα ήταν εύκολα προσιτές, όπως το παιχνίδι ρόλων, ο χειρισμός προσομοιωμένου εξοπλισμού, η σχεδίαση και η δημιουργία πραγμάτων και η δημιουργία προσομοιώσεων για φυσικές διαδικασίες. Μέσα από αυτές τις δραστηριότητες οι μαθητές καλλιεργούν τη γνωστική τους λειτουργία, όπως είναι η ερμηνεία, η κατανόηση, η ανάλυση, η ανακάλυψη, η σύγκριση, ο υπολογισμός και η επίλυση προβλημάτων (Falloon G., 2010)

Εικόνα 1.19: AvatarsστοWorldofWarcraft

Οι εικονικοί κόσμοι, ωστόσο, διαφέρουν μεταξύ τους. Υπάρχουν κατηγορίες όπως το SecondLife που εντρυφούν περισσότερο στην στην κοινωνικοποίηση και «*τη δυνατότητα να ζήσεις μια ζωή όπως πραγματικά την ήθελες*» (Mannecke B., McNeill D., Roche E., Bray D., Townsend A., Lester J., 2008). Από την άλλη, κόσμοι όπως το World of Warcraft έχουν περισσότερο χαρακτήρα ηλεκτρονικού παιχνιδιού με το χρήστη να υποδύεται ρόλους. Άλλοι κόσμοι, όπως το Virtual Battlefield System One, έχουν σχεδιαστεί για τον στρατό και την προσομοίωση ασκήσεων μάχης και διάσωσης πολιτών.

Εικόνα 1.20: VirtualBattlefieldSystemOne

Άλλοι απευθύνονται σε ανθρώπους μεγάλης ηλικίας ή βετεράνους πολέμου, για να τους βοηθήσουν να ξεπεράσουν τυχόν ψυχικά τραύματα. Τέλος, υπάρχουν και κόσμοι που απευθύνονται σε νεαρές ηλικίες, μόλις 8-13 ετών και εστιάζουν στη διαδικασία της μάθησης (Mannecke B., McNeill D., Roche E., Bray D., Townsend A., Lester J., 2008).

Στα πλαίσια αυτής της μεταπτυχιακής διατριβής, ο εικονικός κόσμος απεικονίζεται σε ηλεκτρονική μορφή στη δισδιάστατη οθόνη ενός υπολογιστή, αναπαρίσταται στην xml-based γλώσσα verl και αποτελεί τη σύνθεση ενός συνόλου τρισδιάστατων αντικειμένων, σχεδιασμένων σε γλώσσα VRML/X3D.

ΚΕΦΑΛΑΙΟ 2 – ΑΝΑΠΑΡΑΣΤΑΣΗ ΕΙΚΟΝΙΚΩΝ ΚΟΣΜΩΝ

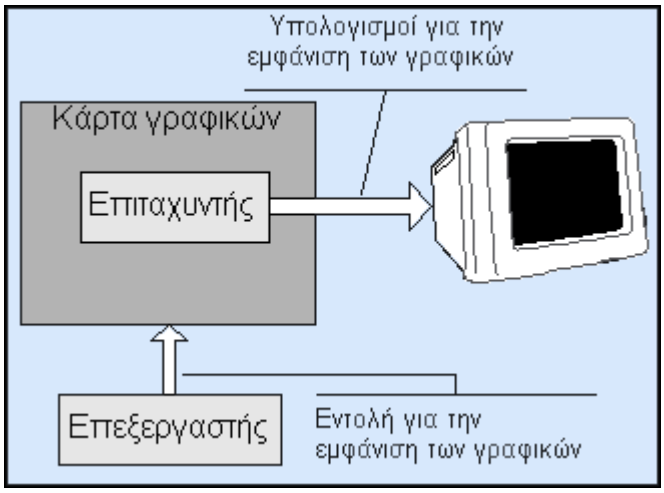
Υπάρχουν διάφορες γλώσσες που χρησιμοποιούνται για την αναπαράσταση εικονικών κόσμων, η καθεμία με τα δικά της ιδιαίτερα χαρακτηριστικά. Σε αυτό το κεφάλαιο θα κάνουμε μια αναδρομή στην εξέλιξη των γραφικών υπολογιστών, θα δούμε πώς ορίζεται ένα γράφημα σκηνής, θα περιγράψουμε συνοπτικά τη γλώσσα χαμηλού επιπέδου OpenGL και θα εστιάσουμε περισσότερο στις γλώσσες υψηλού επιπέδου που χρησιμοποιούν γραφήματα σκηνής: VRML, X3D και Java 3D.

2.1 Λίγα λόγια για τα γραφικά Η/Υ

Ως γνωστόν, η οθόνη του υπολογιστή είναι διδιάστατη, συνεπώς, κάθετι που ο υπολογιστής απεικονίζει, πρέπει να αναπαρίσταται σε δύο διαστάσεις. Για να δουλέψουμε με 3D αντικείμενα πρέπει να τα μετατρέψουμε σε 2D εικόνες. Αυτό απαιτεί ειδική επεξεργασία και μεγάλη υπολογιστική ισχύ που πριν από λίγα χρόνια δεν ήταν διαθέσιμη. Αυτή την μετατροπή την επιτελούν οι νέες κάρτες γραφικών με δυνατότητα επεξεργασίας 3D, που εξαπλώθηκαν γρήγορα στην αγορά λόγω της ζήτησης για αυξημένη ρεαλιστικότητα, για πιο λεπτομερή γραφικά και για μεγαλύτερες ταχύτητες επεξεργασίας σε προγράμματα, όπως τα παιχνίδια δράσης, οι εξομοιωτές και τα σχεδιαστικά πακέτα, που επιτρέπουν στους χρήστες να κινούνται μέσα σε ένα εικονικό κόσμο.

Στις αρχές της δεκαετίας του '90, όταν τα γραφικά λειτουργικά συστήματα έγιναν δημοφιλή, οι κάρτες γραφικών δεν είχαν καθόλου λειτουργίες επιτάχυνσης. Η κάρτα γραφικών, άλλωστε, είναι μόνο ένα μέρος του συνόλου που καθορίζει τι θα δούμε στην οθόνη. Είναι, κατά κάποιον τρόπο, ο "διαμεσολαβητής" μεταξύ του επεξεργαστή και της οθόνης. Η οθόνη είναι αυτή που, στην πράξη, παρέχει αυτό που βλέπουμε, ενώ ο επεξεργαστής είναι αυτός που υπολογίζει και αποφασίζει τι πρόκειται να δούμε. Μια συμβατική κάρτα γραφικών κάνει τη δουλειά της μετατροπής αυτών που παράγει ο επεξεργαστής σε μορφή που μπορεί η οθόνη να τα εμφανίσει. Μάλιστα, παλαιότερα, το λειτουργικό σύστημα ανάγκαζε τον επεξεργαστή να κάνει όλη τη δουλειά της εμφάνισης των γραφικών στην οθόνη, γεγονός που καθυστερούσε δραματικά την απόδοση του συστήματος.

Για να λυθεί αυτό το πρόβλημα και να ελαττωθεί το υπολογιστικό βάρος που επωμιζόταν ο επεξεργαστής του συστήματος, σχεδιάστηκαν επιταχυντές γραφικών που έκαναν τη δουλειά της μετατροπής μιας 3D εικόνας σε 2D με τη χρήση ειδικού hardware. Πρόκειται για κάρτες γραφικών, στις οποίες έχει προστεθεί η δυνατότητα να κάνουν το μεγαλύτερο μέρος των υπολογισμών για την τελική εμφάνιση μιας τριδιάστατατης απεικόνισης, δουλειά που προηγουμένως γινόταν από τον επεξεργαστή. Με έναν επιταχυντή, όταν το σύστημα χρειάζεται να εμφανίσει ένα παράθυρο στην οθόνη, δεν υπολογίζει ποιά ακριβώς εικονοστοιχεία πρέπει να φωτιστούν και με τι χρώμα, απλά στέλνει μια εντολή στην κάρτα γραφικών για να σχεδιάσει ένα παράθυρο σε κάποια περιοχή και η κάρτα γραφικών το υλοποιεί. Ο επιταχυντής μπορεί να είναι προσαρμοσμένος και εξειδικευμένος σε αυτή τη δουλειά και συνεπώς, μπορεί να είναι πιο αποδοτικός σε σχέση με τον επεξεργαστή. Από την άλλη, με αυτόν τον τρόπο, ο επεξεργαστής μπορεί να συνεχίσει να κάνει πιο χρήσιμους υπολογισμούς.



Εικόνα 2.1: Λειτουργία κάρτας γραφικών

Σήμερα, λοιπόν, είναι αναγκαίο να υπάρχουν 3D κάρτες γραφικών για να δημιουργούν τα 3D γραφικά, αλλά το μέγεθος των υπολογισμών που απαιτούνται για να μετατραπούν οι 3D εικόνες σε 2D με ρεαλιστικό τρόπο γίνεται με ειδικό hardware. Χρησιμοποιώντας τους 3D επιταχυντές, επιτρέπουμε στα προγράμματα να εμφανίζουν 3D εικονικούς κόσμους με ένα επίπεδο λεπτομέρειας και χρώματος που με τις τυπικές 2D κάρτες γραφικών ήταν αδύνατο, ενώ παράλληλα απελευθερώνουμε τον επεξεργαστή του συστήματος από την εκτέλεση μιας επιπλέον πολύπλοκης και χρονοβόρας διεργασίας.

Όσον αφορά τις ίδιες τις τρισδιάστατες εικόνες, οφείλουμε να πούμε πως είναι πολύ πιο πολύπλοκες από τις διδιάστατες, λόγω των πολύ μεγαλύτερων ποσών πληροφορίας που χρησιμοποιούνται για να δημιουργηθεί ένας ρεαλιστικός 3D κόσμος. Επιπρόσθετα, πολλές μαθηματικές λειτουργίες πρέπει να χρησιμοποιηθούν για να μετατρέψουν τον 3D κόσμο σε εικόνα που μπορεί να εμφανιστεί στην οθόνη του υπολογιστή. Όταν κοιτάμε τον πραγματικό κόσμο, τα μάτια και ο εγκέφαλος μας εκτελούν αυτή την ενέργεια αυτόματα και οι περισσότερες από τις λειτουργίες που μας επιτρέπουν να αντιλαμβανόμαστε έναν 3D κόσμο συμβαίνουν τόσο γρήγορα που δεν αντιλαμβανόμαστε ότι γίνονται. Το πώς αντιλαμβανόμαστε τον κόσμο, είναι ένα αποτέλεσμα της σύνθετης αλληλεπίδρασης κάποιων οπτικών εφέ, όπως το επίπεδο του φωτός, η σκίαση, η σχετική κίνηση και ο τρόπος που γνωρίζουμε ότι λειτουργεί ο κόσμος. Δουλειά της μηχανής γραφικών είναι να εξομοιώνει αυτή τη φυσική λειτουργία όσο το δυνατόν καλύτερα, έτσι ώστε ό,τι βλέπουμε στην οθόνη να μας να φαίνεται ρεαλιστικό.

Οι 3D εικόνες επεξεργάζονται στον υπολογιστή χρησιμοποιώντας τη μοντελοποίηση (modeling). Τα 3D μοντέλα αντιπροσωπεύουν ένα 3D αντικείμενο χρησιμοποιώντας μια συλλογή σημείων και άλλων πληροφοριών στον τρισδιάστατο χώρο, τα οποία συνδέονται μεταξύ τους με διάφορες γεωμετρικές οντότητες όπως τρίγωνα, ευθύγραμμα τμήματα, καμπύλες. Τα μοντέλα μπορούν να δημιουργηθούν είτε χειροκίνητα είτε με αλγοριθμικές διαδικασίες (procedural modeling) ή μέσω σάρωσης (model scanning). Δυο είδη 3D μοντέλων είναι τα παρακάτω:

- **Στερεά** (Solid): Χρησιμοποιούνται κυρίως για μη γραφικές προσωμοιώσεις όπως για παράδειγμα ιατρικές ή μηχανικές, για CAD και εξειδικευμένες οπτικές εφαρμογές, όπως η ανίχνευση ακτινών και η εποικοδομητική στερεά γεωμετρία (constructive solid geometry). Τα μοντέλα αυτά καθορίζουν τον όγκο του αντικειμένου που αντιπροσωπεύουν.
- **Όρια** (Shell/Boundary): Είναι ευκολότερα στη χρήση από τα στερεά μοντέλα. Καθορίζουν την επιφάνεια, π.χ. το όριο του αντικειμένου, και όχι τον όγκο του. Σχεδόν όλα τα οπτικά υποδείγματα που χρησιμοποιούνται για παιχνίδια και ταινίες είναι shell models.

Οι διαδικασίες μοντελοποίησης (modeling) είναι οι παρακάτω:

- **Polygonal Modeling**: Σημεία, κορυφές στον τριδιάστατο χώρο συνδέονται με γραμμικά τμήματα σχηματίζοντας πολύγωνα. Η φιλοσοφία των περισσότερων 3D μοντέλων σήμερα είναι χτισμένη πάνω σε πολυγωνικά μοντέλα, επειδή είναι ευέλικτα και επειδή οι υπολογιστές μπορούν να τα επεξεργαστούν σε πολύ μικρό χρόνο. Επειδή βέβαια, τα πολύγωνα είναι επίπεδες επιφάνειες, οι σύνθετες κυρτές επιφάνειες μοντελοποιούνται μόνο κατά προσέγγιση με τη χρήση πολλών πολυγώνων.
- **NURBS Modeling**: Επιφάνειες NURBS ορίζονται από spline καμπύλες οι οποίες επηρεάζονται από σταθμισμένα σημεία ελέγχου (weighted control points). Η αύξηση του βάρους για ένα σημείο θα τραβήξει την καμπύλη πιο κοντά στο σημείο αυτό. Τα NURBS είναι πραγματικά λείες επιφάνειες και όχι απλές προσεγγίσεις που χρησιμοποιούν μικρές επίπεδες επιφάνειες και έτσι, είναι ιδιαίτερα κατάλληλες για οργανικές μοντελοποιήσεις.
- **Primitives Modeling**: Αυτή η διαδικασία θεωρεί τα πρωτογενή γεωμετρικά σχήματα (όπως μπάλες, κυλίνδροι, κώνοι, κύβοι) ως δομικά στοιχεία για πιο πολύπλοκα μοντέλα. Εδώ οι μορφές ορίζονται με μαθηματικό τρόπο, με αποτέλεσμα να είναι απόλυτα ακριβείς. Επίσης, η γλώσσα ορισμού τους μπορεί να είναι πολύ απλούστερη. Γενικότερα, αποτελεί μια εύκολη και γρήγορη κατασκευή μοντέλων κατάλληλη για τεχνικές εφαρμογές.
- **Splines & Patches Modeling**: Εξαρτώνται από καμπυλωτές γραμμές για να καθορίσουν την ορατή επιφάνεια. Σε ό,τι αφορά στην ευελιξία και στην ευκολία χρήσης βρίσκονται κάπου μεταξύ καμπυλών NURBS και polygonal.
- **Απόδοση Σχεδιοκίνησης** (animation): Διακρίνεται σε τρεις βασικές μεθόδους:

- Η μέθοδος key frames (σημαντικών καρέ): Χρησιμοποιείται στα περισσότερα προγράμματα κατασκευής 3D. Τα μοντέλα τοποθετούνται σε σημαντικά χρονικά σημεία σε συγκεκριμένες θέσεις του κόσμου και το πρόγραμμα αναλαμβάνει να συμπληρώσει τα ενδιάμεσα καρέ βάσει της τροχιάς της κίνησης που έχει οριστεί.
- Η μέθοδος παραμετρικών key frames: Έχει την ίδια λογική με την προηγούμενη μέθοδο μόνο που εδώ η κάθε οντότητα (αντικείμενο, κάμερα, φως) χαρακτηρίζεται από παραμέτρους.
- Η μέθοδος του διαδικαστικού (procedural) animation: Είναι μια αλγοριθμική μέθοδος στην οποία χρησιμοποιούνται χωρικές και χρονικές μετατροπές (περιστροφή, μετακίνηση κλπ), οι οποίες καθορίζονται από παραμέτρους (π.χ. γωνία περιστροφής) οι οποίες μπορούν να αλλάξουν κατά τη διάρκεια του animation.

Γενικότερα έχουν αναπτυχθεί διάφορες τεχνικές τριδιάστατης σχεδιοκίνησης όπως:

- η κινηματική (kinimatics) η οποία αφορά τις ιδιότητες των αντικειμένων, όπως η θέση, η ταχύτητα και η επιτάχυνση. Σε περιπτώσεις όπου το αντικείμενο είναι τεμαχισμένο σε περισσότερα κομμάτια τότε τα κομμάτια αυτά συνδέονται μεταξύ τους δημιουργώντας μια δενδρική ιεραρχία.
- η δυναμική (dynamic), η οποία είναι αυτή που θα δώσει στο αντικείμενο τις φυσικές του ιδιότητες, λαμβάνοντας υπόψη τους νόμους της φυσικής και προσθέτοντας στην κίνηση του αντικειμένου

χαρακτηριστικά ρεαλιστικότητας. Εδώ χρησιμοποιούνται στοιχεία όπως το υλικό, το βάρος, το μέγεθος, η πυκνότητα.

Ουσιαστικά, κάθε 3D αντικείμενο συντίθεται από εκατοντάδες ή και χιλιάδες μικρά τρίγωνα (ή άλλα πολύγωνα) που περιγράφουν τη δομή του. Όταν το σύστημα θέλει να μετακινήσει ένα αντικείμενο, μπορούμε να πούμε, γενικά και απλουστευμένα, πως αυτό που κάνει είναι ότι αλλάζει τις γωνίες αυτών των τριγώνων για να δημιουργήσει κίνηση. Η μεγαλύτερη υπολογιστική δουλειά σχετίζεται με τη μετατροπή αυτών των καμπύλων τριγώνων σε μια συμπαγή επιφάνεια. Φυσικά, τα πραγματικά αντικείμενα δεν είναι δημιουργημένα με αυτό τον τρόπο, αλλά αυτή η τεχνική είναι αναγκαία για να πετύχουμε προσομοίωση της κίνησης. Επίσης, στον πραγματικό κόσμο, τα αντικείμενα δεν είναι απομονωμένα, αλλά αλληλεπιδρούν μεταξύ τους. Καλύπτει το ένα το άλλο, προκαλούν σκιές, αντανακλούν το φως και εμφανίζονται πιο θαμπά όταν είναι σε απόσταση.

Υπάρχουν πολύ σύνθετες μαθηματικές συναρτήσεις που χρησιμοποιούνται για το κατά πόσο ένα αντικείμενο θα είναι ορατό και τι χρώμα πρέπει να έχει. Αν παίζουμε ένα 3D παιχνίδι και θέλουμε πιστή απεικόνιση, αυτοί οι υπολογισμοί πρέπει να γίνουν είκοσι και πλέον φορές το δευτερόλεπτο. Γι' αυτό χρησιμοποιούνται οι 3D επιταχυντές, οι οποίοι είναι προσαρμοσμένοι έτσι ώστε να βελτιώνουν την απόδοση αυτών των υπολογισμών.

Κάθε φορά που η εικόνα υπολογίζεται ξανά (για παράδειγμα, εξαιτίας μιας κίνησης σε ένα ηλεκτρονικό παιχνίδι), είναι αναγκαίο να υπολογιστεί ξανά το χρώμα και η ένταση κάθε εικονοστοιχείου πάνω στη 2D οθόνη. Υπάρχουν διάφοροι τύποι υπολογισμών που χρησιμοποιούνται στην 3D επεξεργασία. Κάποιες κάρτες υποστηρίζουν περισσότερους υπολογισμούς σε σχέση με άλλες και κάποιες είναι πιο αποδοτικές σε συγκεκριμένους υπολογισμούς σε σχέση με άλλες. Μερικές από τις πιο συνηθισμένες 3D λειτουργίες είναι οι παρακάτω:

- **Φωτοσκίαση**: Πρόκειται για έναν αλγόριθμο που χρησιμοποιείται για να δώσει ρεαλιστική σκίαση σε 3D επιφάνειες. Αυτό προσθέτει βάθος στο αντικείμενο που εμφανίζεται και βοηθά στο να προσδιορίζεται καλύτερα το σχήμα του.
- **Αποκοπή**: Αυτή η λειτουργία καθορίζει ποιό μέρος του αντικειμένου εμφανίζεται στην οθόνη και αποκόπτει τα μέρη που ο χρήστης δεν μπορεί να δει. Εξοικονομεί χρόνο, καθώς τα μέρη που δεν φαίνονται απλά αγνοούνται.
- **Φωτισμός**: Τα αντικείμενα στον πραγματικό κόσμο οφείλουν τη διαμόρφωση της εμφάνισής τους στις πηγές του φωτός του χώρου. Ο φωτισμός, λοιπόν, προκαλεί την αντανάκλαση, τη σκίαση και άλλα εφέ που προστίθενται στο αντικείμενο και εμφανίζονται σε σχέση με τη θέση του αντικειμένου και την πηγή του φωτός στο χώρο.
- **Διαφάνεια**: Κάποια αντικείμενα στον πραγματικό κόσμο είναι διάφανα ή ημι-διάφανα. Ειδικοί υπολογισμοί απαιτούνται, για να καθορίσουν τη διαφάνεια ενός αντικειμένου, για παράδειγμα ποιά αντικείμενα θα είναι ορατά πίσω από ένα τζάμι.
- **Διαμόρφωση υφής**: Σε ρεαλιστικά αντικείμενα, είναι αναγκαίο να επικαλυφθούν κάποιες εικόνες πάνω τους, έτσι ώστε τα αντικείμενα αυτά να αποκτήσουν μια υφή. Η διαμόρφωση υφής επιτρέπει στα αντικείμενα να εμφανίζονται σαν να έχουν υπόσταση. Στην πραγματικότητα, υπάρχουν αρκετοί διαφορετικοί τύποι για διαμόρφωση υφής που χρησιμοποιούνται από το λογισμικό και το hardware.
- **Συσκότιση**: Πρόκειται για εφέ που χρησιμοποιείται για την αναπαράσταση εξωτερικών χώρων, θαμπώνοντας τα αντικείμενα που βρίσκονται σε απόσταση. Εξυπηρετεί δυο σκοπούς: πρώτον, βοηθάει να γίνει πιο ρεαλιστική η απεικόνιση και δεύτερον, βοηθάει τη 3D επεξεργασία να ολοκληρωθεί πιο γρήγορα, καθώς τα αντικείμενα που βρίσκονται σε απόσταση, μπορούν να υπολογιστούν πολύ γρήγορα, αφού δε χρειάζονται πολλές λεπτομέρειες.
- **Φιλτράρισμα**: Υπάρχουν διάφοροι τύποι φιλτραρίσματος που μπορούν να εφαρμοστούν σε μια εικόνα. Χρησιμοποιούνται για να "καθαρίσουν" την εικόνα και να μαλακώσουν την υφή και τα σχήματα.
- Buffering: Δεν πρόκειται πραγματικά για μια 3D λειτουργία, όπως οι άλλες που αναφέρθηκαν παραπάνω, καθώς δεν είναι κάτι που έχει να κάνει με τα δεδομένα. Ωστόσο, οι εξελιγμένες κάρτες

γραφικών περιλαμβάνουν καταχωρητές μνήμης (memory buffers), που χρησιμοποιούνται για διάφορες λειτουργίες κατά τη διάρκεια των πολύπλοκων υπολογισμών. Όσους περισσότερους καταχωρητές έχει μια κάρτα, τόση μεγαλύτερη ευελιξία έχει στο να κάνει κάποιες συγκεκριμένες λειτουργίες. Αυτό οφείλεται στο γεγονός ότι οι 3D κάρτες χρειάζονται συνήθως περισσότερη μνήμη από όση θα χρειάζονταν για να κρατούν απλώς την εικόνα. Τα καινούρια AGP (Accelerated Graphics Port) συστήματα μπορούν να χρησιμοποιήσουν τη μνήμη του συστήματος για αυτό το σκοπό.

2.1.1. Το rendering

Το rendering είναι μια διαδικασία των γραφικών του υπολογιστή που κάνει αυτόματα τη μετατροπή 3D μοντέλων σε διδιάστατες εικόνες, με ή χωρίς τρισδιάστατα φωτορεαλιστικά εφέ.Στην ουσία, πρόκειται για το τελευταίο στάδιο δημιουργίας μιας διδιάστατης εικόνας ή ενός animation από μια ψηφιακή σκηνή που έχει στηθεί. Μπορεί να παραλληλιστεί στον φυσικό κόσμο με το τράβηγμα μιας φωτογραφίας ή με την κινηματογράφηση μιας σκηνής, αφού πρώτα έχουν στηθεί όλα τα σκηνικά. Το rendering μπορεί να χρειαστεί κλάσματα του δευτερολέπτου ή μέρες ολόκληρες για να αποδόσει μια μόνο εικόνα/πλαίσιο. Σε γενικές γραμμές, οι διαφορετικές τεχνικές που έχουν αναπτυχθεί εφαρμόζονται, ανάλογα τις ανάγκες, είτε για φωτο-ρεαλιστική απόδοση είτε για απόδοση σε πραγματικό χρόνο (real-timerendering), όπως συμβαίνει στην περίπτωση των εικονικών περιβαλλόντων.

Στο real-timerendering ο στόχος είναι να μεταδοθεί όση περισσότερη πληροφορία δύναται να επεξεργαστεί το ανθρώπινο μάτι σε ένα κλάσμα του δευτερολέπτου (ή αλλιώς σε ένα πλαίσιο). Ο πρωτεύον στόχος είναι να επιτευχθεί ο πιο υψηλός βαθμός φωτορεαλισμού στην ελάχιστη επιτρεπτή ταχύτητα rendering, που συνήθως αντιστοιχεί σε 24 πλαίσια/sec, καθώς είναι ο ελάχιστος ρυθμός που μπορεί να ξεγελάσει το ανθρώπινο μάτι, δημιουργώντας την ψευδαίσθηση της κίνησης.

Το real-time rendering αφορά πολυγωνικά αντικείμενα και ενισχύεται από τη Μονάδα Επεξεργασίας Γραφικών/GPU (Graphical Process Unit) του υπολογιστή. Η αύξηση της επεξεργαστικής ισχύος των υπολογιστών τα τελευταία χρόνια έχει συνεισφέρει στην αύξηση της ταχύτητας του rendering, ενώ παράλληλα έχει μειωθεί το αντίστοιχο οικονομικό κόστος.

2.2 Γράφημα σκηνής

Ένα γράφημα σκηνής είναι ένας τρόπος να ταξινομείς ιεραρχικά τα δεδομένα μιας σκηνής, δηλαδή είναι μια δομή δεδομένων με κόμβους-γονείς και κόμβους-παιδιά, όπου η σχέση γονέα-παιδιού είναι θεμελιώδης και η σειρά των κόμβων είναι σημαντική, καθώς τα παιδιά κληρονομούν ιδιότητες των γονέων. Το γράφημα σκηνής χρησιμοποιείται συχνά σε σύγχρονα βιντεοπαιχνίδια και σε εφαρμογές επεξεργασίας γραφικών που χρησιμοποιούν διανύσματα (vector-based graphics), όπως Acrobat 3D, Autodesk 3D Studio Max, Adobe Illustrator, AutoCAD, Corel DRAW, Open Scene Graph, Open SG, VRML97 και X3D.

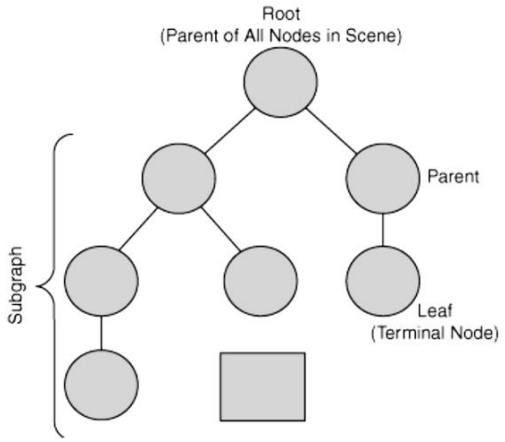
Το γράφημα σκηνής είναι μια δομή που ορίζει τη λογική και συνήθως και την χωρική αναπαράσταση μιας σκηνής γραφικών. Αν και ο ορισμός του γραφήματος σκηνής ποικίλει, λόγω του διαφορετικού τρόπου εφαρμογής της από τους προγραμματιστές σε διαφορετικές εφαρμογές, εντούτοις θα περιγράφαμε το γράφημα σκηνής ως μια συλλογή κόμβων σε μια δομή γραφήματος ή δέντρου. Ένα δεντρο μπορεί να έχει πολλά παιδιά, αλλά συνήθως έχει έναν μοναδικό γονέα, με τις ιδιότητές του να εφαρμόζονται σε όλους τους κόμβους-παιδιά, οπότε, το αποτέλεσμα μιας λειτουργίας που εφαρμόζεται σε μια ομάδα αυτόματα μεταδίδεται σε όλα τα μέλη της ομάδας. Για παράδειγμα, μια χαρακτηριστική ιδιότητα είναι η ικανότητα ομαδοποίησης σχετικών σχημάτων/αντικειμένων για τη δημιουργία ενός ενιαίου, συμπαγούς αντικειμένου που μπορεί να μετατοπιστεί, να μετασχηματιστεί και να επιλεχθεί σαν να επρόκειτο για ένα ξεχωριστό, αυτόνομο αντικείμενο.

Το γράφημα σκηνής αποτελεί μια χρήσιμη αφαιρετική μέθοδο, τόσο για τα δεδομένα της σκηνής όσο και για τη συμπεριφορά της, καθώς παλαιότερα, αυτά τα δύο ορίζονταν διαδικαστικά. Ο κώδικας που όριζε τη σκηνή ήταν διεσπαρμένος μαζί με τον κώδικα που όριζε τη λειτουργικότητα της σκηνής. Κατά αυτόν τον τρόπο, αλλαγές στη σκηνή ή στη συμπεριφορά συνεπάγονταν αλλαγές στον κώδικα και το αποτέλεσμα ήταν ένας

πολύπλοκος και ανελαστικός κώδικας, δύσκολος στη δημιουργία, την τροποποίηση και τη συντήρηση. Χωρίζοντας τη σκηνή από τις λειτουργίες που εφαρμόζονται σε αυτήν, το προγραμματιστικό μοντέλο του γραφήματος σκηνής εγκαθιδρύει ένα καθαρό όριο ανάμεσα στην αναπαράσταση της σκηνής και την απόδοση της στην κάρτα γραφικών (rendering). Συνεπώς, η σκηνή μπορεί να συντεθεί και να συντηρηθεί ανεξάρτητα από τις μεθόδους που λειτουργούν επ' αυτής, δίνοντας έμφαση στο οπτικό περιεχόμενο της σκηνής και στην όποια συμπεριφορά και συσχέτιση υπάρχει μεταξύ των αντικειμένων σε αυτήν και όχι στο πώς αυτό το περιεχόμενο υφίσταται επεξεργασία σε χαμηλότερο επίπεδο. Η υποβόσκουσα υλοποίηση και οι λεπτομέρειες του rendering υπεξαιρούνται από το προγραμματιστικό μοντέλο του γραφήματος σκηνής, πχ. στην VRML είναι το plug-in του browser υπεύθυνο για αυτήν την χαμηλού επιπέδου αναπαράσταση.

Ένα γράφημα σκηνής αποτελείται από κόμβους (που αναπαριστούν τα αντικείμενα στη σκηνή) και ακμές (που συνδέουν τους κόμβους και ορίζουν συσχετίσεις μεταξύ τους). Συνδυαστικά, οι κόμβοι και οι ακμές παραγούν ένα γράφημα που οργανώνει τα αντικείμενα σε μια ιεραρχική δομή, βάσει της χωρικής τους θέσης στη σκηνή. Με την εξαίρεση του κόμβου-ρίζα, που ορίζει το σημείο έναρξης του γραφήματος σκηνής, κάθε κόμβος έχει έναν γονέα. Οι κόμβοι που εμπεριέχουν άλλους κόμβους είναι κόμβοι-γονείς, ενώ οι κόμβοι που εμπεριέχονται είναι κόμβοι-παιδιά. Κόμβοι που περιέχουν παιδιά είναι κόμβοι ομαδοποίησης, ενώ οι τελευταίοι κόμβοι στην ιεραρχία ορίζονται ως κόμβοι-φύλλα. Λειτουργίες επί της σκηνής μπορούν να εφαρμοστούν σε όλους τους κόμβους ή μπορούν να περιοριστούν σε ένα συγκεκριμένο υποσύνολο του γραφήματος, συνεπώς μια σκηνή μπορεί να συντίθεται από ξεχωριστούς κόμβους ή από ολόκληρα υποσύνολα που μπορούν να ενσωματωθούν ή να αποσπαστούν από τη σκηνή, σύμφωνα με τις εκάστοτε ανάγκες.

Ειδωμένο οπτικά, ένα γράφημα σκηνής θυμίζει μια δενδροειδή δομή. Η δομή δεδομένων DAG (Directed Acyclic Graph – Κατευθυνόμενος Ακυκλικός Γράφος) χρησιμοποιείται συχνά, επειδή υποστηρίζει διαμοίρασμα των κόμβων (node sharing) σε υψηλό επίπεδο στον γράφο. Οι κόμβοι στη δομή DAG μπορούν να έχουν πάνω από ένα γονέα, εις βάρος, ωστόσο, επιπλέον πολυπλοκότητας στον κώδικα. Στο DAG όλοι οι κόμβοι πρέπει να έχουν μια ορισμένη κατεύθυνση σχέσης γονέα–παιδιού στην οποία δεν επιτρέπονται κύκλοι/επαναλήψεις.

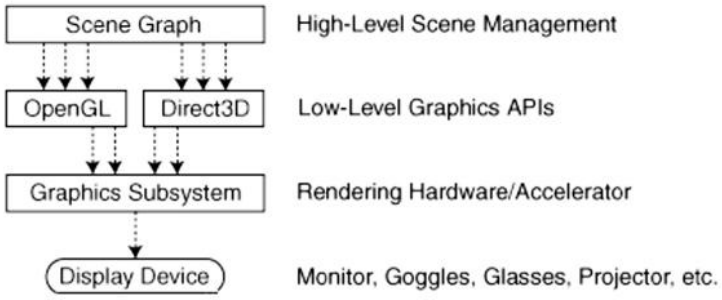


Εικόνα 2.2: Δομή Γραφήματος Σκηνής

Τα γραφήματα σκηνής που χρησιμοποιούνται για τριδιάστατο περιεχόμενο, συνήθως υποστηρίζουν κόμβους που αναπαριστούν πρωτογενή τριδιάστατα γεωμετρικά σχήματα (προδηλωμένα τετράγωνα, κώνους, σφαίρες κ.ά), πολύπλοκα πολυγωνικά σχήματα, φώτα, υλικό, ήχο κ.οκ. Επίσης, υπάρχουν κόμβοι που μπορεί να υποστηρίζουν οπτικοακουστικό περιεχόμενο, συγχρονισμό, ειδικά εφέ και άλλες λειτουργικότητες για τη σύνθεση πολυμέσων (multimedia).

Το γράφημα σκηνής υποστηρίζει διάφορες πράξεις, μέσω διάσχισης της δομής του γράφου με κατεύθυνση από τη ρίζα προς τα φύλλα. Η διάσχιση του γράφου είναι απαραίτητη για την εκτέλεση ενός πλήθους λειτουργιών που αφορούν ενέργειες του rendering και σχετίζονται με μετασχηματισμούς, αποκοπές, φωτισμό και άλλες λειτουργίες, πιο διαδραστικές, όπως η ανίχνευση συγκρούσεων και η επιλογή αντικειμένων. Οι

κόμβοι που επηρεάζονται από μια δοσμένη λειτουργία διατρέχονται κατά τη διάρκεια μιας διάσχισης. Φτάνοντας στον κόμβο, η εσωτερική του κατάσταση μπορεί να μεταβληθεί (εφόσον υποστηρίζεται κάτι τέτοιο από την αντίστοιχη γλώσσα), ώστε να ανακλά την κατάσταση της συγκεκριμένης λειτουργίας εκείνη τη στιγμή. Οι διασχίσεις κατά το rendering συμβαίνουν διαρκώς με τα διαδραστικά γραφικά, επειδή η κατάσταση των πραγμάτων αλλάζει όσο συχνά αλλάζει και η οπτική γωνία του χρήστη, οδηγώντας σε συνεχή ερωτήματα (queries) στο γράφημα σκηνής και ανανεώσεις σε μια διαρκώς εναλλασσόμενη προοπτική. Τέλος, ένα γράφημα σκηνής μπορεί να θεωρηθεί ως μια υψηλού επιπέδου περιγραφή των API γραφικών χαμηλού επιπέδου, όπως η OpenGL και η DirectX.



Εικόνα 2.3: Ροή υλοποίησης Γραφήματος Σκηνής

2.3 Γλώσσες Αναπαράστασης Εικονικών Κόσμων

2.3.1. Η γλώσσα τριδιάστατων γραφικών OpenGL

Η γλώσσα OpenGL (Open Graphics Library) χρησιμοποιείται για σχεδίαση 2D και 3D γραφικών χαμηλού επίπεδου, με την έννοια ότι μιλάει απευθείας στην κάρτα γραφικών. Με τον όρο OpenGL δεν αναφερόμαστε σε μια συγκεκριμένη βιβλιοθήκη, αλλά σε ένα πρότυπο υλοποίησης βιβλιοθηκών σχεδίασης γραφικών. Το πρότυπο καθορίζει μια προγραμματιστική διεπιφάνεια (Application Programming Interface). Εμπεριέχει, δηλαδή, το σύνολο των συναρτήσεων που πρέπει να υλοποιεί μία βιβλιοθήκη γραφικών προκειμένου να είναι συμβατή με αυτό. Το API χρησιμοποιείται για να αλληλεπιδράσει με την GPU (Graphics Processing Unit) και να εκτελέσει το rendering με τη βοήθεια των επιταχυντών, όπως είδαμε παραπάνω.

Η OpenGL αναπτύχθηκε από την εταιρία Silicon Graphics Inc. στις αρχές του 1990 και έκτοτε χρησιμοποιείται ευρέως στις εφαρμογές CAD, στην εικονική πραγματικότητα, σε προσομοιώσεις πτήσεις, σε επιστημονικές εφαρμογές για 3D οπτικοποίηση δεδομένων και σε βιντεοπαιχνίδια. Την OpenGL σήμερα τη διαχειρίζεται ένας μη κερδοσκοπικός συνεταιρισμός ονόματι KhronosGroup.

Εφόσον με τον όρο OpenGL δεν αναφερόμαστε σε μια συγκεκριμένη βιβλιοθήκη αλλά σε ένα πρότυπο που ορίζει τη λειτουργικότητα μιας βιβλιοθήκης σχεδίασης, μπορούμε να ακολουθήσουμε τις ίδιες συμβάσεις σε όλες τις υλοποιήσεις του προτύπου (τις ίδιες εντολές). Αυτό σημαίνει δυο πράγματα: α) δεν υπάρχει περιορισμός ως προς τη γλώσσα προγραμματισμού στην οποία θα υλοποιηθεί το πρότυπο της OpenGL. Ενδεικτικά, μπορεί να χρησιμοποιηθούν βιβλιοθήκες της σε γλώσσες προγραμματισμού Fortran, C, C++, Java, Python κ.ά. και β) ο κώδικας που συντάσσουμε είναι ανεξάρτητος πλατφόρμας (platform independent) και μπορει να εκτελεστεί σε ευρεία γκάμα λειτουργικών συστημάτων (Windows, Linux, MacOS) χωρίς ριζική τροποποίηση της δομής του. Οι βιβλιοθήκες των περισσότερων νέων μεταγλωττιστών εμπεριέχουν (ή υπάρχει η δυνατότητα να ενσωματωθεί σε αυτούς) μια υλοποίηση της OpenGL. Αυτά είναι και τα δυο μεγαλύτερα πλεονέκτηματα που προσφέρει η OpenGL σε σύγκριση με άλλα API.

Η αρχιτεκτονική της OpenGL δουλεύει κυρίως με καταστάσεις (states), δηλαδή ο προγραμματιστής ορίζει μια κατάσταση και η OpenGL κάνει render με βάση αυτη την κατάσταση μέχρι να γίνει αλλαγή της τρέχουσας

κατάστασης. Για παράδειγμα, εαν θέλω να κάνω render με κόκκινο χρώμα, θέτω το χρώμα ως κόκκινο και κάνω render τα αντικειμένα που θέλω με αυτό το χρώμα.

Οι κατηγορίες βιβλιοθηκών που συναντά κανείς σε υλοποιήσεις της OpenGL είναι οι:

- OpenGL Core Library
- OpenGL Utility Library (GLU)
- OpenGL Utility Toolkit (GLUT)

Η βασική (core) βιβλιοθήκη έχει σχεδιαστεί ως μια βελτιωμένη διεπαφή ανεξάρτητη από το hardware και είναι η:

- OpenGL32 για Windows
- GL για Unix / Linux συστήματα

Η GLU παρέχει πολλά από τα χαρακτηριστικά μοντέλων, όπως quadric επιφάνειες και NURBS καμπύλες και επιφάνειες. Παρέχει λειτουργικότητα στην βασική OpenGL αλλά δεν χρειάζεται να ξαναγράψουμε κώδικα. Η GLUT συνδέει την OpenGL με το παραθυρικό σύστημα:

- GLX για X συστήματα
- WGL για Windows
- AGL για Macintosh

2.3.2. VRML

Η VRML (Virtual Reality Modeling Language) είναι μια γλώσσα προγραμματισμού για τη δημιουργία τρισδιάστατων (3D) γραφικών αλληλεπίδρασης. Η VRML οφείλει την προέλευσή της στο πρώτο παγκόσμιο συνέδριο δικτύου το Μάρτιο του 1994, αναπτύχθηκε από το Web3D Consortium και προτυποποιήθηκε από την ISO/IEC. Η VRML χρησιμοποιείται κατά κόρον για τριδιάστατα γραφικά στο διαδίκτυο, οπότε οι προγραμματιστές μπορούν να δημιουργήσουν ιστοσελίδες με ακολουθίες εικόνων, έτσι ώστε οι επισκέπτες να μπορούν να αλληλεπιδράσουν με αυτές, να τις περιστρέφουν ή να περιηγούνταισε αυτές.

Ουσιαστικά, η γλώσσα VRML είναι το πρότυπο 3D γραφικών για τον παγκόσμιο ιστό (www) και δημιουργεί ένα συγκεκριμένο τύπο αρχείου, το οποίο περιέχει μια ASCII περιγραφή τρισδιάστατων σκηνών (γράφημα σκηνής). Η συγγραφή προγραμμάτων σε VRML απαιτεί την ύπαρξη ενός οποιουδήποτε text editor και την αποθήκευση των αρχείων με την κατάληξη .wrl (από το world=κόσμος). Οι χαρακτήρες που επιτρέπονται καθορίζονται από το UTF-8 format, του οποίου υποσύνολο είναι οι ASCII χαρακτήρες. Τα προγράμματα παρουσιάζονται σε οποιονδήποτε browser, με την προϋπόθεση ότι έχει εγκατασταθεί το απαιτούμενο plugin για VRML 2 στον υπολογιστή. Με την VRML μπορούμε να σχεδιάσουμε τριδιάστατα γεωμετρικά αντικείμενα από τα οποία αποτελείται ένας εικονικός κόσμος και να συνθέσουμε εξολοκλήρου έναν εικονικό κόσμο. Επιπλέον, επιτρέπει τον ορισμό διαφόρων σημείων-συνδέσμων, τα οποία οδηγούν σε κάποιο συγκεκριμένο URL (πιθανώς κάποια άλλη σκηνή ή κάποιο τυπικό έγγραφο www).

Η γενική δομή της γλώσσας έχει ως εξής: Τα VRML αρχεία αποτελούνται από την επικεφαλίδα (header), τα σχόλια (comments) του προγραμματιστή, τους ορισμούς κόμβων (nodes), που μάλιστα είναι το θεμελιώδες δομικό στοιχείο της γλώσσας και φυσικά, από τα ονόματα των κόμβων, τα πεδία (fields) τους και τις αντίστοιχες τιμές τους. Με μια πιο αφαιρετική ματιά, τα VRML αρχεία αποτελούνται από σύνολα κόμβων που είναι ενταγμένα σε ιεραρχίες υποσυνόλων και αλληλεπιδρούν μεταξύ τους, δημιουργώντας ένα γράφημα σκηνής. Τα σύνολά αυτά αποτελούν τον κόσμο (world). Οι κόμβοι χρησιμοποιούνται για να εκφράσουν τα σχήματα και τις αντίστοιχες ιδιότητες τους, ενώ κάποια γεωμετρικά σχήματα προσφέρονται ως έτοιμα αντικείμενα (πχ Box, Sphere, Cylinder κ.ά). Κατά τον ορισμό των κόμβων απαιτείται να καθορίζεται το όνομα του κόμβου, το είδος του, καθώς και ένας αριθμός προαιρετικών πεδίων με τις αντίστοιχες τιμές τους. Ως τιμές των πεδίων μπορεί να είναι μεταξύ άλλων και κάποιος άλλος κόμβος. Επίσης, τα πεδία μπορεί να ορίζουν πώς

κόμβοι του ίδιου τύπου διαφέρουν μεταξύ τους, καθώς και να ορίζουν μια σειρά γεγονότων (events) που κάποιος κόμβος μπορεί να αποστείλει ή να εκλάβει. Όταν ένας κόμβος λαμβάνει ένα γεγονός, αντιδρά μεταβάλλοντας την κατάστασή του, η οποία μπορεί να πυροδοτήσει επιπρόσθετα γεγονότα. Οι κόμβοι μπορούν να αλλάξουν την κατάσταση των αντικειμένων στη σκηνή με τη χρήση γεγονότων. Επίσης, η υλοποίηση ενός κόμβου ορίζει πώς αντιδρά σε γεγονότα, πότε μπορεί να παράγει και να αποστείλει γεγονότα, καθώς και οποιαδήποτε οπτική ή ακουστική αναπαράσταση μπορεί να έχει στη σκηνή.

Ο αρχικός κόμβος ονομάζεται γονέας (parent), ενώ όλοι οι άλλοι κόμβοι που περιέχονται σε αυτόν αποκαλούνται παιδιά του (children). Προφανώς ο κόμβος- γονέας ή αλλιώς πρόγονος των υπολοίπων μπορεί να είναι παιδί σε κάποιον άλλο γονέα. Κατ' αυτόν τον τρόπο φτάνουμε σε έναν αρχικό κόμβο από τον οποίο, τελικά, προκύπτουν όλοι οι υπόλοιποι, τον κόμβο ρίζα (root node). Ένας κόμβος μπορεί να οριστεί μια φορά και να χρησιμοποιηθεί περισσότερες χωρίς να οριστεί, με τη χρήση των λέξεων DEF και USE, όπου η πρώτη καθορίζει το όνομα του κόμβου που θα χρησιμοποιηθεί ξανά και η δεύτερη τον αναφέρει όταν χρησιμοποιείται. Επίσης, υπάρχουν κόμβοι που χρησιμοποιούνται για να ομαδοποιούν αντικείμενα, όπως ο κόμβος Group. Μπορούμε να κατηγοριοποιήσουμε γενικά τους κόμβους που ορίζει το πρότυπο VRML ως εξής:

- Γεωμετρικοί κόμβοι που ορίζουν το σχήμα ή τη μορφή ενός αντικειμένου (πχ Shape)
- Κόμβοι γεωμετρικών ιδιοτήτων που χρησιμοποιούνται για να ορίσουν συγκεκριμένες όψεις των γεωμετρικών κόμβων
- Κόμβοι εμφάνισης (appearance) που ορίζουν το υλικό της γεωμετρίας και ιδιότητες υφής για το αντικείμενο
- Κόμβοι ομαδοποίησης που ορίζουν ένα χώρο συντεταγμένων για κόμβους-παιδιά που μπορεί να εμπεριέχουν. Για παράδειγμα, ένας κόμβος ομαδοποίησης που περιέχει έναν ή περισσότερους κόμβους-παιδιά είναι ο Transform. Κάθε Transform κόμβος έχει το δικό του τοπικό σύστημα συντεταγμένων πάνω στο οποίο τοποθετούνται οι κόμβοι-παιδιά, που μπορεί να είναι άλλοι Transform κόμβοι, Group κόμβοι ή Shape κόμβοι. Επίσης, ο κόμβος Transform υποστηρίζει πράξεις μετασχηματισμού των αντικειμένων που αφορούν τη θέση στο χώρο, την κλίμακα και το μέγεθος και εφαρμόζονται σε όλα τα παιδιά του κόμβου.
- Κόμβοι για πηγές φωτός (light-source) που φωτίζουν αντικείμενα στη σκηνή
- Κόμβοι αισθητήρων (sensor nodes) που αντιδρούν σε αλλαγές του περιβάλλοντος και ενέργειες του χρήστη
- Interpolator κόμβοι που ορίζουν λειτουργίες για animations
- Κόμβοι χρονο-εξάρτησης που ενεργοποιούνται και απενεργοποιούνται σε συγκεκριμένα διαστήματα του χρόνου
- Δεσμευόμενοι (bindable) κόμβοι-παιδιά που είναι μοναδικοί, γιατί μόνο ένας κόμβος κάθε είδους μπορεί να δεσμευτεί ή να επηρεάσει την εμπειρία του χρήστη μια συγκεκριμένη χρονική στιγμή
- Script κόμβοι που διευκολύνουν τη δυναμική συμπεριφορά ενός εικονικού περιβάλλοντος, επιτρέποντας τη χρήση προγραμματιστικών γλωσσών, όπως η ECMAScript, η JavaScript, και η Java. Οι Script κόμβοι χρησιμοποιούνται κυρίως για να υποδηλώσουν μια αλλαγή στη σκηνή ή κάποιου είδους ενέργεια του χρήστη, για να εκλάβουν γεγονότα (events) από άλλους κόμβους, για να ενθυλακώσουν modules του προγράμματος που παράγουν υπολογισμούς ή για να πυροδοτήσουν αλλού στη σκηνή αλλαγές με αποστολή γεγονότων.

Ο εξωτερικός προγραμματιστικός έλεγχος πάνω στο γράφημα σκηνής της VRML γίνεται μέσω του External Authoring Interface (EAI), που είναι ένα μοντέλο διασύνδεσης της διεπαφής μεταξύ των κόσμων της VRML και εξωτερικών περιβαλλόντων.

Η γλώσσα VRML σχεδιάστηκε για ικανοποιεί τις παρακάτω ανάγκες και απαιτήσεις:

- **Authorability** (συγγραφικότητα): Η VRML πρέπει να καθιστά δυνατή την ανάπτυξη συστημάτων εφαρμογών και επεξεργαστών κειμένου (editor), καθώς και την εισαγωγή δεδομένων από άλλους τύπους αρχείων (format).

- **Completeness** (ολοκλήρωση): Να υπάρχουν όλες οι πληροφορίες που χρειάζονται για τον χειρισμό και να υπάρχει ένα ολοκληρωμένο σύνολο χαρακτηριστικών.
- **Composability** (συνθεσιμότητα): Η δυνατότητα να συνδυάζονται τα στοιχεία της VRML και να καθίστανται επαναχρησιμοποιήσιμα.
- **Extensibility** (επεκτασιμότητα): Η δυνατότητα να προστίθενται νέα στοιχεία.
- **Implementability** (υλοποιησιμότητα):Οι προδιαγραφές της VRML πρέπει να παρέχουν επαρκείς πληροφορίες για την υλοποίηση χωρίς να υπάρχουν αμφισημίες.
- **Ικανότητα Multi –User**: Να μην αποκλείει την υλοποίηση σε πολυχρηστικά περιβάλλοντα.
- **Orthogonality** (ορθογωνικότητα): Τα στοιχεία VRML πρέπει να είναι ανεξάρτητα μεταξύ τους, αλλά στην περίπτωση που υπάρχουν εξαρτήσεις, αυτές πρέπει να είναι καλά ορισμένες και δομημένες.
- **Performance** (απόδοση): Τα στοιχεία πρέπει να σχεδιάζονται με έμφαση στη διαδραστική απόδοση πάνω σε διαφορετικές πλατφόρμες υπολογιστών.
- **Scalability**: Τα στοιχεία της VRML είναι σχεδιασμένα για μεγάλες συνθέσεις, αλλά πρέπει να έχουν περιορισμένο πεδίο εφαρμογής.
- **Well - structured** (καλά δομημένη): Τα στοιχεία πρέπει να έχουν μια καλά ορισμένη διεπαφή και έναν απλά ορισμένο σκοπό. Στοιχεία πολλαπλών σκοπών και παρενέργειες πρέπει να αποφεύγονται.

Εικόνα 2.4: Το τσαγιερό της Utah. Πρωτοκατασκευάστηκε στοπανεπιστήμιο της Γιούτα το 1975 από τον M.Newell. Το πολύ-γωνικό πλέγμα τσαγιερού έχει γίνει ένα σημαντικό στοιχείο στα κομπιούτερ γρφικών. Αυτή η εκδοχή του VRML περιέχει 1976κορυφέςκαι 3751 πολύγωνα.

Το εύρος και το πεδίο εφαρμογής της VRMLπεριλαμβάνουν:

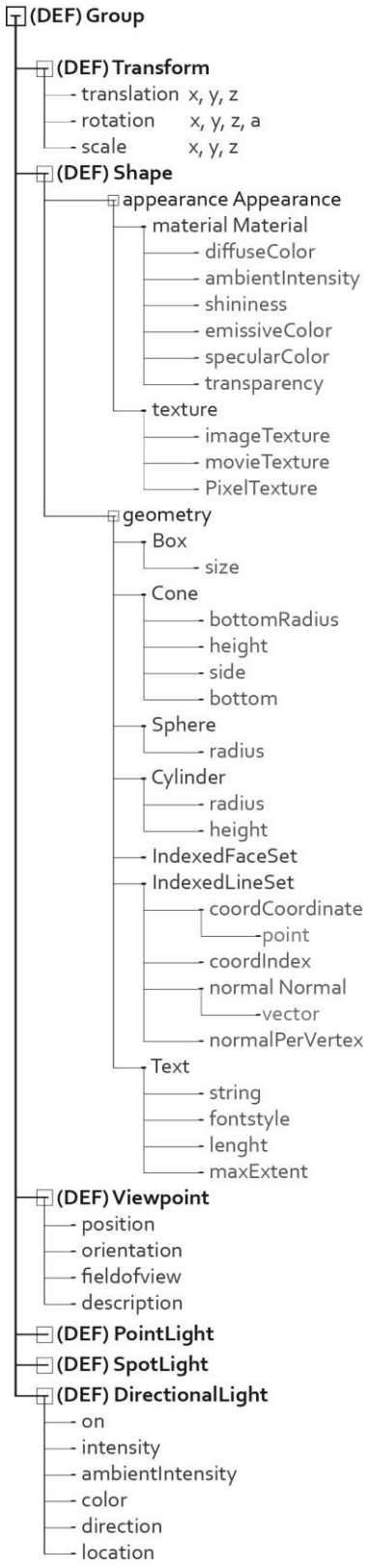
- Έναν μηχανισμό για αποθήκευση και μεταφορά διδιάστατων και τριδιάστατων δεδομένων
- Στοιχεία αναπαράστασης διδιάστατης και τριδιάστατης ακατέργαστης πληροφορίας
- Στοιχεία για τον ορισμό των ιδιοτήτων αυτής της πληροφορίας και στοιχεία για την όψη και την μοντελοποίηση διδιάστατης και τριδιάστατης πληροφορίας
- Έναν μηχανισμό ενσωμάτωσης δεδομένων από άλλους τύπους meta-αρχείου
- Έναν μηχανισμό ορισμού νέων στοιχείων που επεκτείνουν τις δυνατότητες του meta-αρχείου ώστε να υποστηρίξει επιπρόσθετους τύπους και είδη πληροφορίας.

Το μεγάλο πλεονέκτημα της VRML έγκειται στο γεγονός ότι μια περιγραφή φτάνει (μέσω του Internet) στον browser του χρήστη ως ένα απλό ASCII αρχείο και στη συνέχεια ο browser αναλαμβάνει να κάνει το λεγόμενο rendering, δημιουργεί δηλαδή μια όψη (view) της σκηνής με βάση την ASCII περιγραφή. Αυτό σημαίνει ότι συνήθως χρειάζονται λιγότερα bytes για να περιγραφεί μια σκηνή. Το σημαντικότερο,όμως, είναι ότι ο browser μπορεί να αναλάβει να δείξει την ίδια σκηνή και από μια διαφορετική οπτική γωνία. Ο χρήστης, κατά αυτόν τον τρόπο, έχει τη δυνατότητα ουσιαστικά να περιηγηθεί μέσα στη σκηνή.

Τυπικά, ο χρήστης έχει την επιλογή να περπατήσει ή να πετάξει μέσα στον χώρο οποιαδήποτε στιγμή, χρησιμοποιώντας το ποντίκι ή τους πίνακες ελέγχου πάνω στον browser. Ένας χρήστης μπορεί να κοιτάξει τριγύρω ή πάνω από αντικείμενα, να περπατήσει ή να πετάξει μέσα στο περιβάλλον, να εξετάσει αντικείμενα, ακόμη και να αλληλεπιδράσει με αυτά.

Η τρέχουσα και πλήρης λειτουργική έκδοση της VRML αντιστοιχεί στο πρότυπο ISO/IEC 14772-1:1997 και είναι γνωστή ως VRML97. Μια σύγχρονη γλώσσα που διαδέχτηκε την VRML είναι η γλώσσα X3D (ISO/IEC 19775-1), η οποία προσφέρει σημαντικές μειώσεις στις απαιτήσεις ανάπτυξης ενώ εξελίσσει το χώρο των γραφικών 3D τόσο στο διαδίκτυο όσο και πέρα από αυτό.

Παρακάτω παρουσιάζουμε τη βασική δομή των κόμβων της VRML, όπου διακρίνεται η ιεραρχία τους σε γράφημα σκηνής. Το DEF προηγείται ενός κόμβου για να του προσδώσει κάποιο όνομα, πχ αν θέλουμε να ονομάσουμε έναν Transform κόμβο ως transform, θα χρησιμοποιήσουμε την εντολή DEF transform Transform.


(DEF) Group
(DEF) Transform
translation x, y, z
rotation x, y, z, a
scale x, y, z
(DEF) Shape
appearance Appearance
material Material
diffuseColor
ambientIntensity
shininess
emissiveColor
specularColor
transparency
texture
imageTexture
movieTexture
PixelTexture
geometry
Box
size
Cone
bottomRadius
height
side
bottom
Sphere
radius
Cylinder
radius
height
IndexedFaceSet
IndexedLineSet
coordCoordinate
point
coordIndex
normal Normal
vector
normalPerVertex
Text
string
fontstyle
lenght
maxExtent
(DEF) Viewpoint
position
orientation
fieldofview
description
(DEF) PointLight
(DEF) SpotLight
(DEF) DirectionalLight
on
intensity
ambientIntensity
color
direction
location

2.3.3. X3D

Το X3D (eXtensible 3D) είναι μια αρχιτεκτονική που αναπαριστά τρισδιάστατα γραφικά χρησιμοποιώντας την εννοιολογική μοντελοποίηση μέσω ενός γραφήματος σκηνής. Αποτελεί την διαδοχή της VRML και μας δίνει την δυνατότητα να περιγράψουμε μία τρισδιάστατη σκηνή (γράφημα σκηνής) με ορισμούς αντικειμένων είτε σε VRML97 είτε σε XML.

Το X3D σχεδιάστηκε για μια πληθώρα συστημάτων – για επιστημονικά workstations, επιτραπέζιους και φορητούς υπολογιστές, PDAs, tablets, κινητά τηλέφωνα και συσκευές που δεν έχουν την απαραίτητη υπολογιστική ισχύ που απαιτείται για την VRML. Το X3D δίνει επίσης τη δυνατότητα της ενσωμάτωσης υψηλής απόδοσης 3D σε μετάδοση (broadcast) και ενσωματωμένες συσκευές, ενώ αποτελεί και τον ακρογωνιαίο λίθο των αρχικών δυνατοτήτων MPEG-4 για 3D. Η χρήση του έχει ευρύ πεδίο εφαρμογών – από τη μηχανική και την απεικόνιση επιστημονικών δεδομένων εως τα πολυμέσα, τη ψυχαγωγία, την εκπαίδευση, ακόμα και σε πολυχρηστικά περιβάλλοντα. Όπως και η VRML, το X3D έχει σχεδιαστεί ως ένα διεθνές πρότυπο για ενσωματωμένα 3D γραφικά και πολυμέσα. Ωστόσο, υποστηρίζει πολλαπλές κωδικοποιήσεις (encodings), όπως η κωδικοποίηση XML, καθιστώντας το πιο σύγχρονο και δημοφιλές πρότυπο από την VRML.

Η αρχιτεκτονική του X3D αποτελείται από αρθρωτά συστατικά στοιχεία λειτουργικότητας, γνωστά ως components, που είναι εύκολα κατανοητά και υλοποιήσιμα από τους προγραμματιστές των εφαρμογών. Γενικά, θα περιγράφαμε ένα X3D component ως ένα σύνολο από συσχετισμένες λειτουργίες, αποτελούμενες από διάφορα αντικείμενα και υπηρεσίες (services). Τυπικά πρόκειται για μια συλλογή κόμβων. Βέβαια, ένα component μπορεί να ενέχει και κωδικοποιήσεις, υπηρεσίες API, ή άλλες ιδιότητες του X3D. Η πιο σημαντική διαφορά σε σχέση με την VRMLείναι πως, σπάζοντας την λειτουργικότητα σε διακριτά συστατικά στοιχεία που φορτώνονται κατά το χρόνο εκτέλεσης, μειώνεται σημαντικά η πολυπλοκότητα της εφαρμογής.

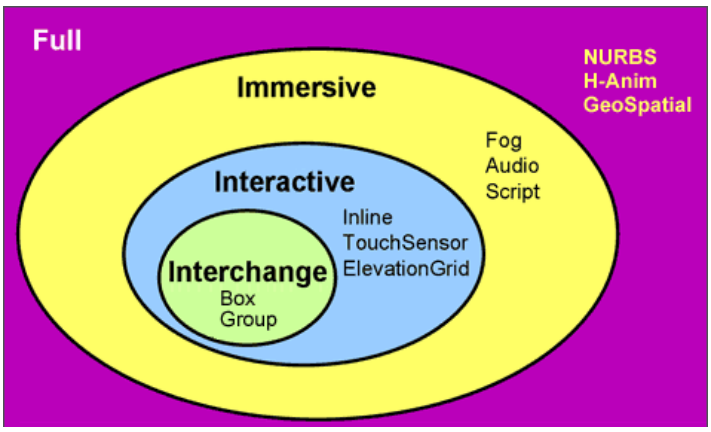
Το πρότυπο X3D προδιαγράφει ένα σύνολο από components που περιλαμβάνουν ένα Core component, το οποίο ορίζει την κύρια λειτουργικότητα κατά το χρόνο εκτέλεσης για το σύστημα X3D, αφηρημένους (abstract) τύπους κόμβων, πεδία (fields), δρομολόγηση (routing) και μοντέλα γεγονότων (event model).

Βασικά components που υποστηρίζει από μόνο του το πρότυπο είναι: Time (κόμβοι που παρέχουν χρονική λειτουργικότητα), Aggregation και Transformation (κόμβοι οργάνωσης και ομαδοποίησης που υποστηρίζουν την ιεραρχία σε ένα γράφημα σκηνής), Geometry (κόμβοι αναπαράστασης γεωμετρικών σχημάτων), Geometric Properties (κόμβοι που ορίζουν τις βασικές ιδιότητες των γεωμετρικών κόμβων), Appearance (κόμβοι που περιγράφουν τις ιδιότητες εμφάνισης μιας γεωμετρίας και της σκηνής), Lighting (κόμβοι που φωτίζουν αντικείμενα στη σκηνή) και ένα σωρό άλλα χαρακτηριστικά που περιλαμβάνουν συστατικά όπως οι: Navigation (πλοήγηση), Interpolation, Text (κείμενο), Sound (ήχος), Pointing Device Sensorκαι Environmental Sensor (κόμβοι αισθητήρες), Texturing (υφή), Prototyping (προτυποποίηση) και Scripting.

Η αρχιτεκτονική X3D επιτρέπει, επίσης, διαστρωματώσεις από «προφίλ» που μπορούν να παρέχουν αυξημένη λειτουργικότητα για περιβάλλοντα εμβύθισης και μπορούν να υποστηριχθούν ανεξάρτητα. Συνεπώς, τα components μπορούν να επεκταθούν ή να τροποποιηθούν ανεξάρτητα, με την πρόσθεση νέων επιπέδων ή νέων συστατικών στοιχείων που μπορούν να προσφέρουν καινούρια χαρακτηριστικά, όπως το streaming. Το γεγονός αυτό καθιστά την X3D μια εύκολα επεκτάσιμη αρχιτεκτονική, όπου η εξέλιξη ενός συστατικού στοιχείου δεν καθυστερεί το σύνολο. Τα βασικά προφίλ που παρέχει το X3D, ανάλογα με το σκοπό που θέλει να πετύχει ο προγραμματιστής και τις ιδιότητες που τον ενδιαφέρουν να χρησιμοποιήσει, είναι τα παρακάτω :

- **Interchange**: Είναι το βασικό προφίλ για την επικοινωνία μεταξύ των εφαρμογών. Υποστηρίζει γεωμετρικά σχήματα, υφές, βασικό φωτισμό και κίνηση. Δεν υπάρχει μοντέλο για rendering στο χρόνο εκτέλεσης, καθιστώντας το πολύ εύκολο να χρησιμοποιηθεί και να ενσωματωθεί σε οποιαδήποτε εφαρμογή.
- **Interactive**: Επιτρέπει βασική αλληλεπίδραση με το τρισδιάστατο περιβάλλον, προσθέτοντας διάφορους κόμβους-αισθητήρες που χρησιμοποιούνται για την περιήγηση και την αλληλεπίδραση του χρήστη με τον κόσμο (πχ. PlanseSensor, TouchSensor κ.ά.), για τη βελτίωση του συγχρονισμού ή για επιπρόσθετο φωτισμό (πχ. Spotlight, PointLight).

- **Immersive**: Παρέχει τη δυνατότητα για πλήρη τρισδιάστατα γραφικά και πλήρη αλληλεπίδραση με τον χρήστη μέσω αισθητήρων, συμπεριλαμβάνοντας την ενσωμάτωση ήχου, την ανίχνευση συγκρούσεων 3D αντικειμένων, ομίχλης, και τη δυνατότητα scripting (μικρά κομμάτια κώδικα σε JavaScript) .
- **Full**: Συμπεριλαμβάνει όλους τους δηλωμένους κόμβους (nodes) που υπάρχουν στο πρότυπο, όπως επίσης και τις NURBS (Non-Uniform B-Splines) καμπύλες και επιφάνειες, το H-Anim (humanoid animation) πρότυπο και γεοχωρικά συστατικά (geospatial components).



Εικόνα 2.5: Πολυεπίπεδα προφίλ του X3D

Μερικά από τα πλεονεκτήματα της χρήσης του X3D είναι: α) το μικρό μέγεθος των αρχείων που το καθιστά ιδανικό για δικτυακές εφαρμογές, β) η υποστήριξη που προσφέρει για διαδικτυακές εφαρμογές, ώστε να μπορούν οι χρήστες να μοιράζονται αντικείμενα σε πραγματικό χρόνο, γ) η δυνατότητα λειτουργίας του σε όλες τις πλατφόρμες ανεξάρτητα από το λειτουργικό σύστημα που το φιλοξενεί, αφού είναι προσανατολισμένο στην εννοιολογική περιγραφή της σκηνής, δ) η δυναμικότητα που προσφέρει για την επέκταση του σε συνδυασμό με την διαπεραστικότητα που μπορεί να έχει με τον χρήστη και ε) το γεγονός ότι είναι open-source αρχιτεκτονική και διατίθεται δωρεάν. Επίσης, μπορεί εκτός από την απεικόνιση τρισδιάστατων αντικειμένων να αναπαράγει και αρχεία βίντεο και ήχου.

Ο εικονικός κόσμος που δημιουργείται με το X3D δεν έχει τίποτα να ζηλέψει από αντίστοιχους κόσμους που δημιουργούνται με επαγγελματικά προγράμματα. Όπως διαπιστώσαμε παραπάνω, μπορούμε και εδώ να έχουμε κάμερες, αισθητήρες εξωτερικών συσκευών, αλλά και αισθητήρες κίνησης μέσα στον κόσμο. Με κατάλληλα scripts μπορούμε να εκμεταλλευτούμε τις εισόδους και εξόδους των αισθητήρων και να δημιουργήσουμε μια πολύ φιλική προς το χρήστη τρισδιάστατη εφαρμογή.

2.3.4. JAVA 3D

Η Java 3D είναι μια δυνατότητα που προσφέρει η Java για την αναπαράσταση τριδιάστατων γραφικών. Πρόκειται για ένα γράφημα σκηνής βασισμένο σε ένα 3D API (Application Programming Interface) που τρέχει στην πλατφόρμα της Java. Τα προγράμματα που είναι γραμμένα σε Java 3D μπορούν να τρέχουν σε διαφορετικού είδους υπολογιστές, ακόμα και στο διαδίκτυο.

Η βιβλιοθήκη κλάσεων της Java 3D παρέχει μια πιο απλή διεπαφή από τις περισσότερες βιβλιοθήκες γραφικών και ενθυλακώνει τον προγραμματισμό γραφικών, χρησιμοποιώντας μια πραγματικά αντικειμενοστραφής προσέγγιση και προσφέροντας αρκετές δυνατότητες για να παράγει καλά παιχνίδια και animations. Μια σκηνή δομείται μέσω γραφήματος σκηνής, που αποτελεί μια αναπαράσταση των αντικειμένων που πρέπει να εμφανίζονται. Αυτό το γράφημα σκηνής έχει μια δενδροειδή ιεραρχία που εμπεριέχει αρκετά στοιχεία που είναι απαραίτητα για την εμφάνιση των αντικειμένων.

Η J3D χρησιμοποιεί προγραμματιστικό μοντέλο ενός DAG γραφήματος σκηνής, παρόμοια με της VRML και του X3D. Οφείλουμε, όμως, να σημειώσουμε πως τα γραφήματα σκηνής της Java 3D είναι πιο δύσκολα να

φτιαχτούν, δεδομένης της ίδιας της πολυπλοκότητας της Java ως γλώσσα, καθώς για κάθε αντικείμενο της σκηνής, κόμβο μετασχηματισμού ή συμπεριφορά, πρέπει να δημιουργείται ένα καινούριο στιγμιοτύπο αντικειμένου, χρησιμοποιώντας τις αντίστοιχες κλάσεις της Java, να οριστούν και να αρχικοποιηθούν τα πεδία του αντικειμένου και έπειτα να προστεθούν τα αντικείμενα στη σκηνή.

Η Java 3D βασίζεται σε υπάρχουσα τεχνολογία, όπως η DirectX και η OpenGL, γεγονός που βοηθά σε ταχύτερη εκτέλεση των προγραμμάτων. Επίσης, η Java 3D μπορεί να ενσωματώσει αντικείμενα που έχουν δημιουργηθεί από άλλα πακέτα μοντελοποίησης, όπως αντικείμενα TrueSpace ή VRML. Κάποια από τα πιο βασικά χαρακτηριστικά της είναι τα εξής:

- Πολυνηματική (Multithreaded) δομή γραφήματος σκηνής
- Ανεξαρτησία από το λειτουργικό σύστημα, καθώς τρέχει σε διαφορετικές πλατφόρμες
- Generic API πραγματικού χρόνου, που μπορούν να χρησιμοποιηθούν τόσο για απεικόνιση (visualization) όσο και για gaming
- Παροχή hardware-accelerated JOGL, OpenGL και Direct 3D renderers (αναλόγως την πλατφόρμα)
- Παροχή εξελιγμένων μοντέλων εικονικής πραγματικότητας, βασισμένα στην όψη (viewmodels) και υποστήριξη στερεοσκοπικής απόδοσης και πολύπλοκων συνθέσεων multi-display.
- Εγγενής υποστήριξη για HDMs
- Υποστήριξη για CAVE (multiple screen projectors)
- 3D χωρικό ήχο κ.ά.

Από την άλλη, η J3D δεν ορίζει κάποιο συγκεκριμένο μορφότυπο αρχείου, όπως άλλα προγραμματιστικά μοντέλα που χρησιμοποιούν γράφημα σκηνής. Αντ' αυτού, χρησιμοποιεί run-time loaders που φορτώνουν και υποστηρίζουν ένα εύρος 3D αρχείων, όπως VRML, X3D, Wavefront (OBJ), AutoCAD Drawing Interchange File (DXF), Caligari true Space (COB), Lightwave Scene Format (LSF), Lightwave Object Format (LOF), 3D-Studio (3DS) και άλλα.

ΚΕΦΑΛΑΙΟ 3 – Εργαλεία Σχεδίασης Εικονικών Κόσμων

Προκειμένου να προχωρήσουμε στην ανάλυση απαιτήσεων και μετέπειτα στη σχεδίαση της εφαρμογής μας, οφείλουμε να δούμε, πέρα από τις υπάρχουσες δυνατότητες που προσφέρονται από άποψη προγραμματιστικών γλωσσών για την αναπαράσταση ενός εικονικού κόσμου, τι δυνατότητες προσφέρονται από άποψη εργαλείων σχεδίασης εικονικών κόσμων, εφόσον ένα τέτοιο εργαλείο σκοπεύουμε να φτιάξουμε. Ως εκ τούτου, μελετάμε τη δομή δυο δημοφιλών και αξιόπιστων εφαρμογών σχεδίασης, που χρησιμοποιούν γράφημα σκηνής για την αναπαράσταση του κόσμου, και αναφέρουμε κάποιες από τις βασικές επιλογές που επιτρέπονται σε αυτες, , καθώς πιστεύουμε ότι μπορούν να εμπνεύσουν και την δική μας εφαρμογή.

3.1 Η εφαρμογή Unity

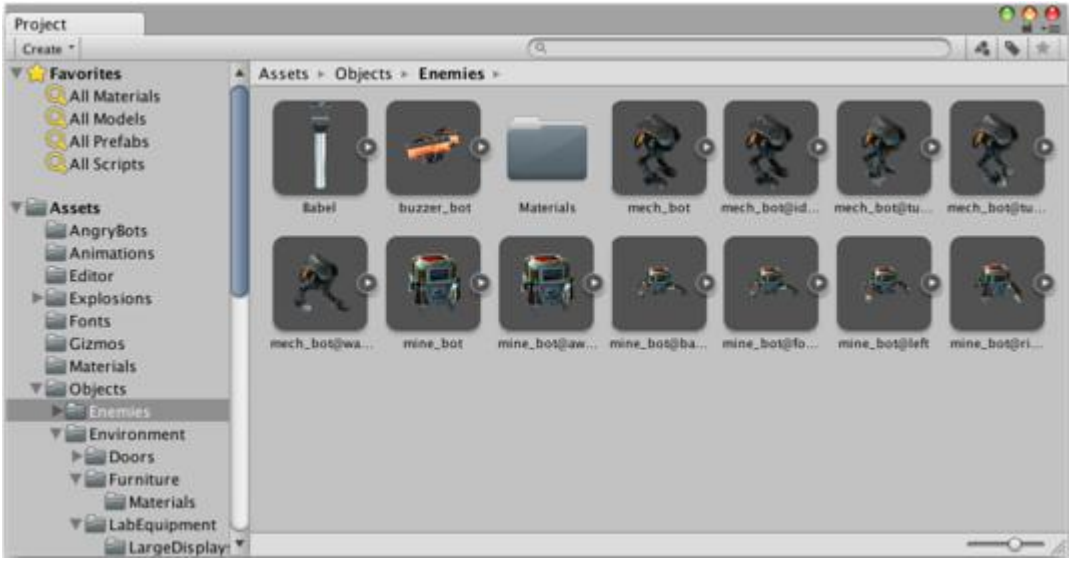
Η εφαρμογή Unity είναι ένα λογισμικό ανάπτυξης εικονικών κόσμων και εικονικών πρακτόρων για τη δημιουργία βιντεοπαιχνιδιών. Έχει αναπτυχθεί με κώδικα C# και C++ και διαθέτει ένα πλούσιο σε δυνατότητες GUI που προσφέρει στο χρήστη μια ποικιλία έτοιμων αντικειμένων και εργαλείων για να αναπτύξει διαδραστικά μοντέλα σκηνής 2 ή 3ων διαστάσεων. Διακρίνεται από την ποιότητα των υπηρεσιών που παρέχει όσον αφορά τα οπτικοακουστικά εφέ, ενώ διαθέτειπαράλληλα και ένα προγραμματιστικό περιβάλλον στο χρήστη για την πρόσθεση scripts στη σκηνή καθώς και δυνατότητες testing. Επίσης, είναι ιδανικό για multi-player εφαρμογές, καθώς υποστηρίζει την ταυτόχρονη εκτέλεση σε διαφορετικές πλατφόρμες.Παρακάτω, παρουσιάζουμε κάποια από τα βασικά χαρακτηριστικά του προγράμματος, μελετώντα το interfacetou.

Αρχικά, το περιβάλλον ανάπτυξης διακρίνεται σε ένα μενού εργαλείων (Toolbar), όπου προσφέρονται οι βασικές λειτουργίες που χρησιμοποιούνται ευρέως στο πρόγραμμα και τεσσερα βασικά παράθυρα (views) που

συνυπάρχουν και παρουσιάζουν διαφορετικά χαρακτηριστικά της εφαρμογής και είναι τα: Scene, Inspector, Hierarchy και Project.

Εικόνα 3.1: Το GUI του Unity

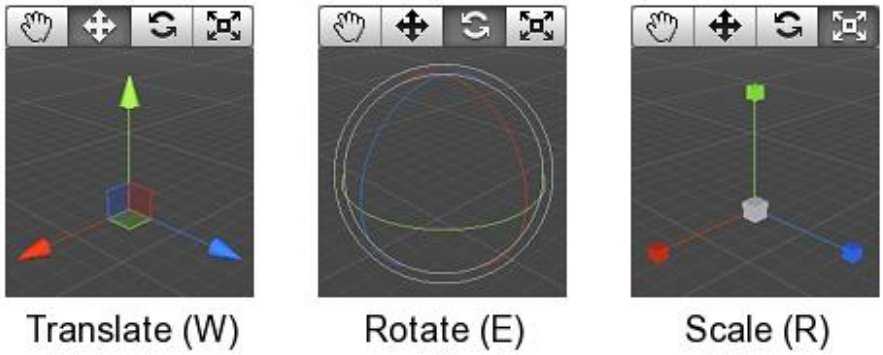
Το Project view επιτρέπει πρόσβαση σε αντικείμενα που ανήκουν στον φάκελο εργασίας της εφαρμογής, εμφανίζοντας τη δομή των φακέλων που αποτελούν το project σαν μια λίστα ιεραρχίας. Επιλέγοντας ένα φάκελο, εμφανίζονται δεξιά τα χαρακτηριστικά του μαζί με εικόνες των αντικειμένων που εμπεριέχει ο φάκελος σε μικρογραφία. Το παράθυρο προσφέρει και μια γραμμή αναζήτησης, όπου ο χρήστης μπορεί να πληκτρολογήσει το όνομα ενός αντικειμένου που θέλει να αναζητήσει για να διαχειριστεί. Επίσης, ο χρήστης μπορεί να σύρει (drag and drop) ένα αντικείμενο από το Project viewστο Hierarchy ή Scene View, ώστε αυτό να εμφανιστεί τη σκηνή.



Εικόνα 3.2: Project tab

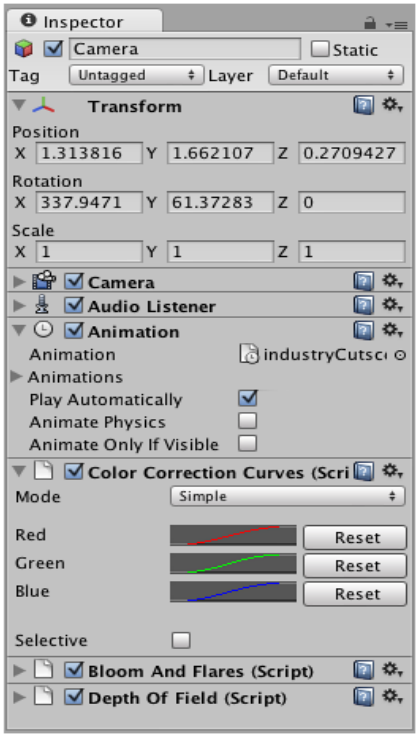
Το Hierarchy εμφανίζει την ιεραρχία κάθε των αντικειμένων που υπάρχουν στην σκηνή σε μορφή ιεραρχίας, ανάλογα τη σύνδεση που υπάρχει μεταξύ τους. Μέσα σε αυτό το παράθυρο, ο χρήστης μπορεί να επιλέξει ένα αντικείμενο και να το σύρει σε κάποιο άλλο, μετατρέποντάς το σε κόμβο-παιδί αυτού του αντικειμένου. Ο κόμβος-παιδί κληρονομεί κάθε κίνηση και περιστροφή που γίνεται στον κόμβο-γονέα. Κατά αυτόν τον τρόπο αξιοποιούνται οι δυνατότητες που προσφέρει το γράφημα σκηνής.

Το Scene view είναι το περιβάλλον όπου απεικονίζεται η σκηνή και τοποθετούνται τα αντικείμενα. Για την επιλογή των αντικειμένων και την πλοήγηση στη σκηνή χρησιμοποιούνται επιλογές από το Toolbar ή συνδυαστικά το ποντίκι με κάποιο keystroke (από το πληκτρολόγιο). Από το Toolbar δίνεται η δυνατότητα Transform που επιτρέπει την μετακίνηση, την περιστροφή και την μεγέθυνση/σμίκρυνση ενός αντικειμένου. Επίσης, δίνεται η δυνατότητα περιστροφής και μετατόπισης της κάμερας λήψης πανω στους άξονες xyz, αλλάζοντας την οπτική γωνία παρατήρησης του κόσμου.



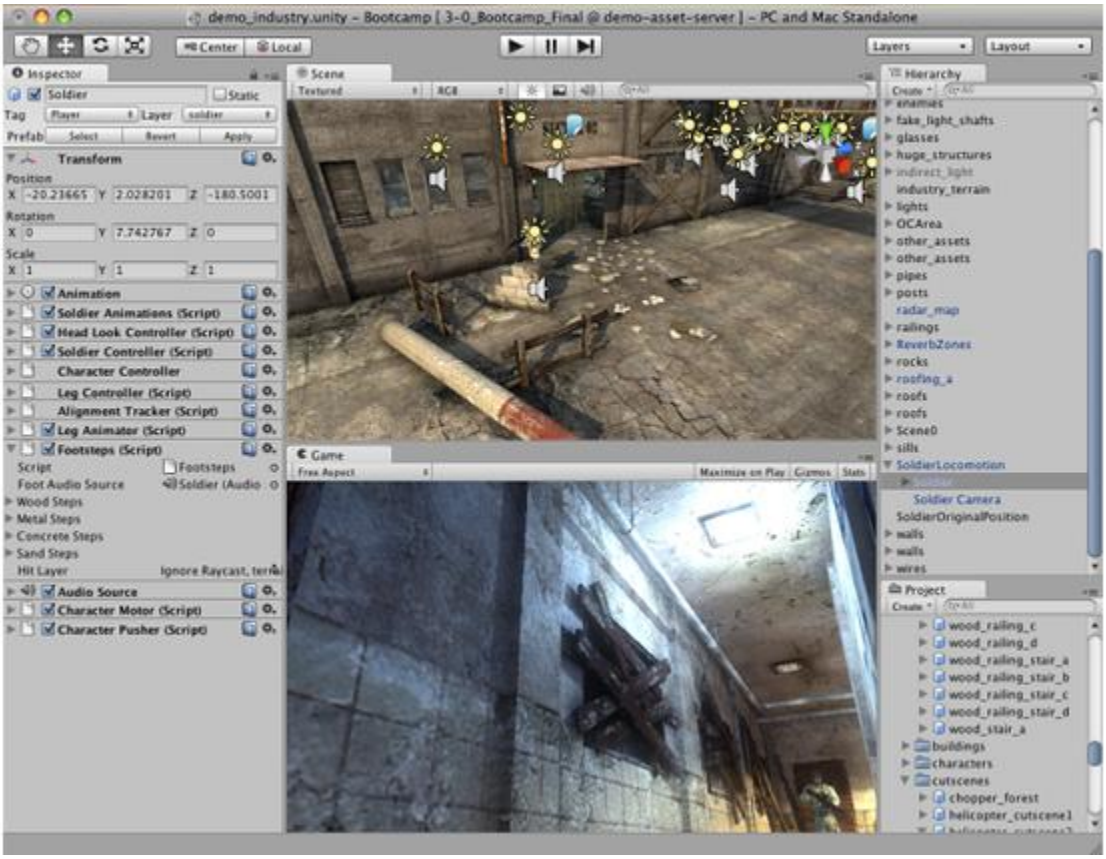
Εικόνα 3.3: Trasform επιλογές

Το Inspector view προβάλλει λεπτομερείς πληροφορίες για το αντικείμενο της σκηνής που διαλέγει ο χρήστης, συμπεριλαμβανομένου των ιδιοτήτων του και των συστατικών από τα οποία μπορεί να συντίθεται. Σε αυτό το παράθυρο, ο χρήστης μπορεί να μεταβάλλει τη λειτουργικότητα αντικειμένων στην σκηνή, προσθέτοντας συστατικά όπως είναι ο φωτισμός ή τα Scripts, αλλάζοντας παραμέτρους και ελέγχοντας τη συμπεριφορά του κόσμου κατά την εκτέλεση.



Εικόνα 3.4: Inspector view

Τέλος, ο χρήστης μπορεί να προσαρμόσει το περιβάλλον εργασίας του με βάση τις ανάγκες που έχει, εμφανίζοντας κάθε view σε διαφορετικό παράθυρο ή προσαρμόζοντας την οθόνη με εναλλακτικό τρόπο και αποθηκεύοντας αυτές τις αλλαγές σε νεό layout, όπως φαίνεται παρακάτω, για να μπορέσει να το επαναχρησιμοποιήσει:



Εικόνα 3.5: Μεταβολή των viewsστο περιβάλλον εφαρμογής

3.2 Η εφαρμογη 3D Studio Max

Μια επίσης δημοφιλής εφαρμογή για την παραγωγή τριδιάστατων γραφικών σε υπολογιστή, με τη δημιουργία τριδιάστατων μοντέλων, εικόνων, animations και βιντεοπαιχνιδιών είναι το πρόγραμμα της Autodesk, 3D Studio Max. Χρησιμοποιείται κατά κόρον για αρχιτεκτονικό σχεδιασμό, ανάπτυξη εικονικών κόσμων για βιντεοπαιχνίδια και ταινίες animation και υποστηρίζεται από το λειτουργικό σύστημα των Windows. Παρότι δεν προσφέρεται δωρεάν και υπάρχει μια απόκλιση από τις δυνατότες που μας ενδιαφέρουν και τις λειτουργίες που προορίζεται, αξίζει να δούμε κάποιες από τις βασικές του λειτουργίες, μελετώντας το interface του.

Το περιβάλλον ανάπτυξης διακρίνεται στα παρακάτω:

- QuickAccessToolbar: Πρόκειται για ένα μενού εργαλείων στην πρώτη γραμμή του περιβάλλοντος ανάπτυξης που παρέχει τις πιο βασικές εντολές που αφορούν τη διαχείριση αρχείων (άνοιγμα αρχείου, άνοιγμα φακέλου, αποθήκευση), εντολές αναίρεσης ή επανεκτέλεσης καθώς και μια drop-down λίστα για εναλλαγή μεταξύ των διαφορετικών παραθύρων εργασίας.

Εικόνα 3.6: Quick Access Toolbar

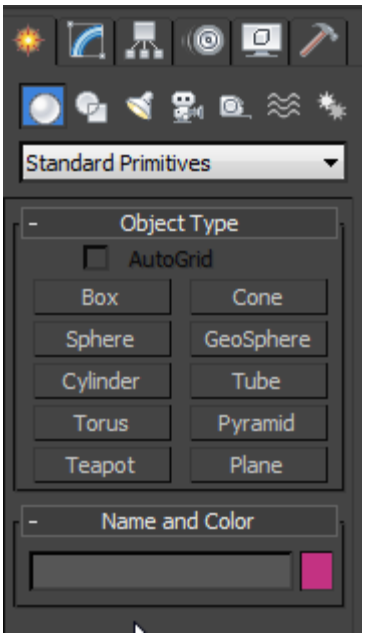
- Menu: Το μενού με σύνοψη όλων των διαθέσιμων επιλογών που προσφέρει η εφαρμογή, ανά κατηγορία ενέργειας.

Εικόνα 3.7: Menu

- Main Toolbar: Πρόκειται για ένα μενού επιλογών που παρέχει γρήγορη πρόσβαση σε εργαλεία και παράθυρα που χρησιμοποιούνται ευρέως για τις πιο βασικές ενέργειες που εκτελούνται στο πρόγραμμα.

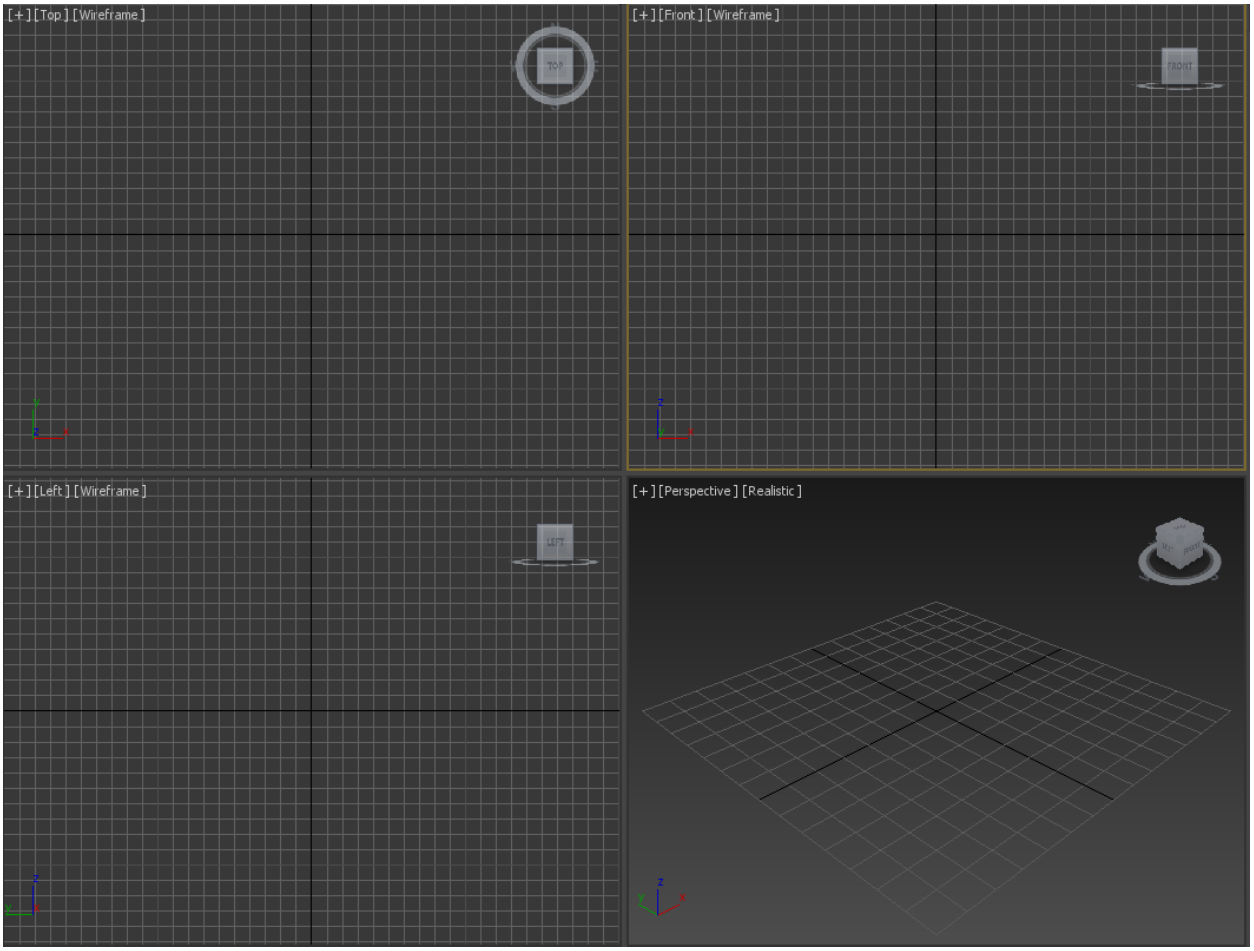
Εικόνα 3.8: Main Toolbar

- Graphite Modeling Tools: Πρόκειται για μια περιοχή εργαλείων στο πάνω μέρος της οθόνης που χρησιμοποιείται για τη διευκόλυνση της σχεδίασης αντικειμένων (μοντελοποίηση, χρωματισμός σκηνής κ.α).
- Command Panel: Αποτελείται από έξι UI panels, στα οποία μπορεί να πλοηγηθεί εναλλάξ ο χρήστης. Αυτά δίνουν πρόσβαση σε εργαλεία που αφορούν τη δημιουργία ή την επεξεργασία της γεωμετρίας των αντικειμένων, την προσθήκη πολυπλοκότητας σε μια γεωμετρία, την προσθαφαίρεση φωτισμού κ.α., που αφορούν την μοντελοποίηση, τα animations, τις επιλογές εμφάνισης μιας σκηνής ή ενός αντικειμένου και άλλες δυνατότητες.



Εικόνα 3.9: Command Panel

- Viewport: Πρόκειται για 4 διαφορετικά παράθυρα όπου εμφανίζεται η σκηνή υπό διαφορετικές οπτικές γωνίες. Επιπλέον, ο χρήστης μπορεί να επιλέξει εργαλεία για να ρυθμίσει κάποιο από τα παράθυρα να εμφανίζει με διαφορετικό τρόπο τη σκηνή, πχ με πλέγματα (wireframe) ή σκιασμένη (shaded).



Εικόνα 3.10: Viewports

- Viewport Label Menus: Είναι μενού που επιτρέπει στο χρήστη να αλλάξει τι εμφανίζει κάθε viewport, συμπεριλαμβανομένου την οπτική γωνία παρατήρησης και το στυλ της σκίασης.
- Viewport Navigation controls: Κουμπιά που επιτρέπουν την πλοήγηση στη σκηνή στο τρεχον viewport.

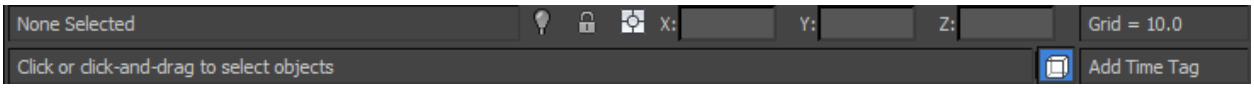
Εικόνα 3.11: Navigation Controls

- Slate MaterialEditor: Προσφέρει ρυθμίσεις για την επεξεργασία των υλικών και των maps των αντικειμένων, ώστε να επιτυγχάνεται μεγαλύτερος ρεαλισμός.
- TimeSlider: Αφορά τα animations και επιτρέπει τη χρονική πλοήγηση σε αυτά και την μετάβαση σε διαφορετικό frame στο πέρασμα του χρόνου.

Εικόνα 3.12: Time Slider

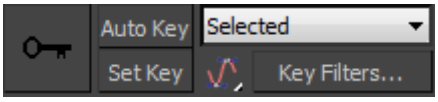
- Status Bar Controls: Πρόκειται για μια περιοχή στο τέλος του παραθύρου όπου παρέχονται πληροφορίες για την κατάσταση της σκηνής. Επίσης, στα δεξιά εμφανίζονται στοιχεία για τις συντεταγμένες, όπου μπορεί χειροκίνητα ο χρήστης να μεταβάλλει τις τιμές για να μετατοπίσει ένα αντικείμενο. Στα αριστερά, προσφέρεται η δυνατότητα εισαγωγής μιας γραμμής script βάσει του MAX Script listener που χρησιμοποιεί η εφαρμογή. Η MAXScript είναι μια built-inscripting γλώσσα που χρησιμοποιείται για να αυτοματοποιήσει ο χρήστης επαναλαμβανόμενες εργασίες, για να συνδυάσει

υπάρχουσες λειτουργικότητες για διαφορετικό σκοπό, για να αναπτύξει νέα εργαλεία και interfaces κ.ά.



Εικόνα 3.13: Status Bar Controls

- Animation & Time Controls: Πρόκειται για επιλογές που μπορούν να μεταβάλλουν ένα animation στη διάρκεια του χρόνου.

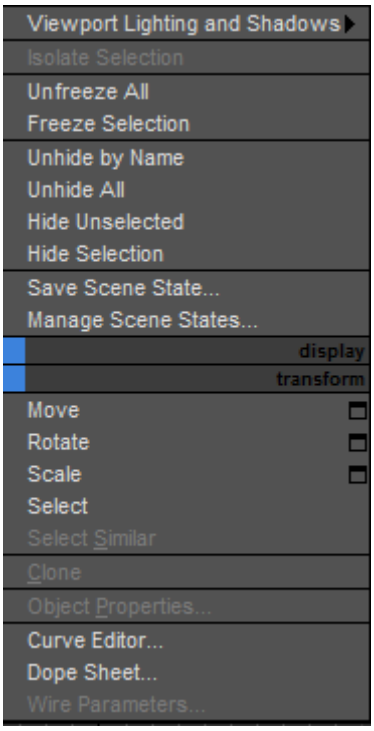


Εικόνα 3.14: Animation & Time controls

- Animation Playback Controls: Κουμπιά που επιτρέπουν στο χρήστη να δει την σκηνή σε κίνηση στη διάρκεια του χρόνου.

Εικόνα 3.15: Playback controls

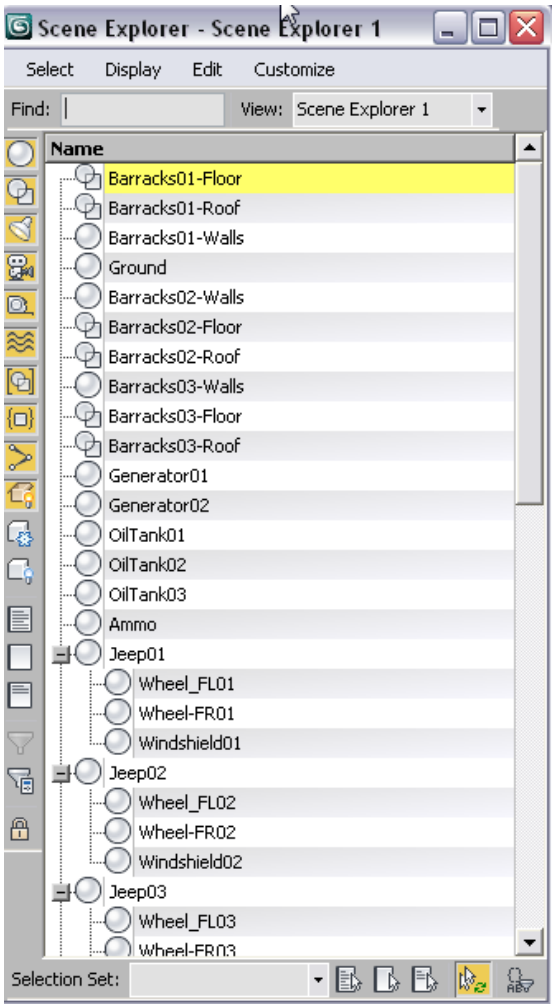
- Rendered Frame Window: Σε αυτό το παράθυρο γίνεται rendering της σκηνής με τις κατάλληλες ρυθμίσεις που θα ορίσει ο χρήστης.
- Quad Menu: Πρόκειται για ένα μενού που εμφανίζεται όταν ο χρήστης πατάει δεξί κλικ σε μια περιοχή του viewport. Οι επιλογές που εμπεριέχει εξαρτώνται από το αντικείμενο που επιλέγεται.



Εικόνα 3.16: QuadMenuσε άδεια σκηνή

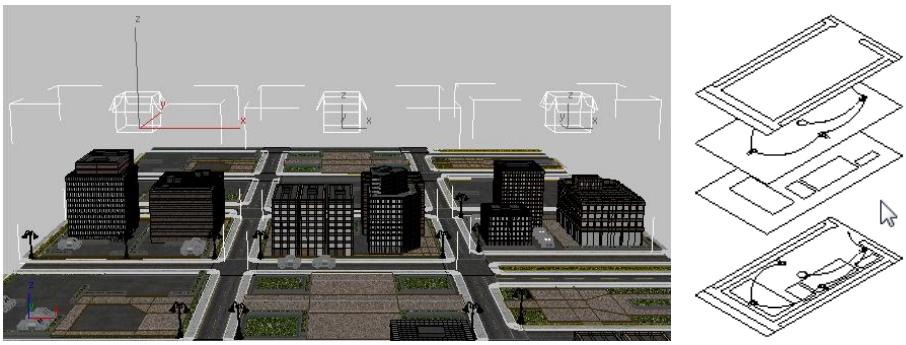
Για τη διαχείριση των σκηνών, των αρχείων και των φακέλων/projects προσφέρονται τα παρακάτω εργαλεία:

- Scene Explorer: Είναι ένα δυνατό εργαλείο για να βλέπει ο χρήστης τις ιδιότητες του αντικειμένου που μπορούν να μεταβληθούν, για να επιλέγει αντικείμενα βάσει διάφορων κριτηρίων και για να δημιουργεί και να τροποποιεί ιεραρχίες αντικειμένων.



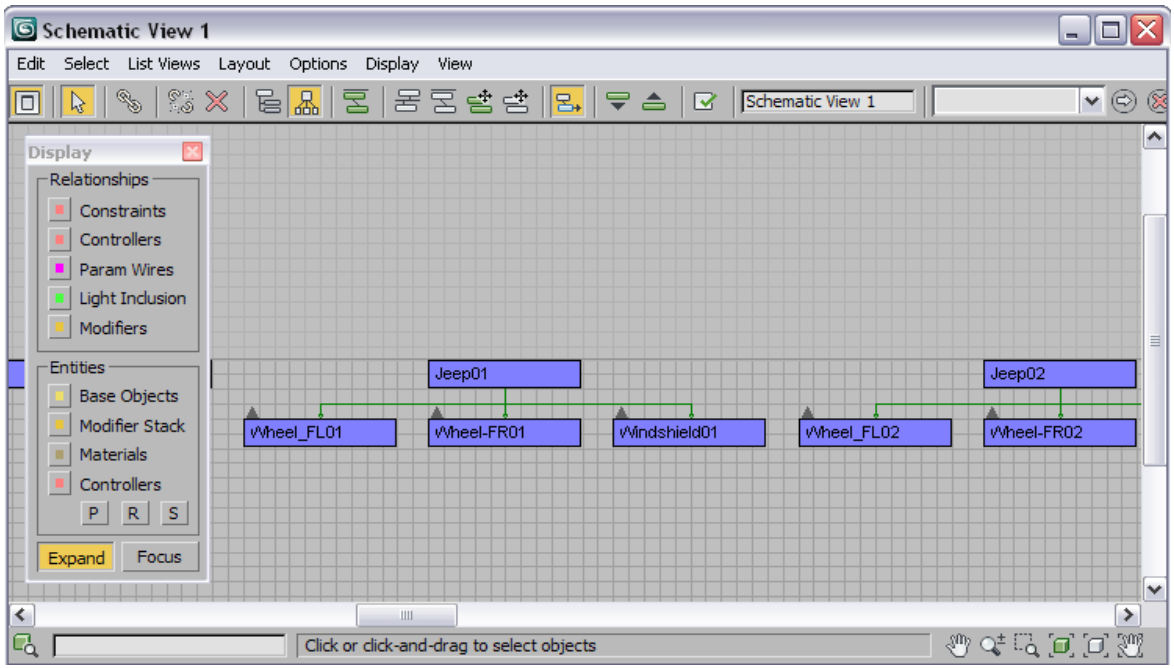
Εικόνα 3.17: Scene Explorer

- Organizational Tools: Εδώ υπάγονται τα Container, Group και Layer. Τα δυο πρώτα χρησιμεύουν για να οργανώνουν τα αντικείμενα βάσει της λογικής τους συσχέτισης, έτσι ώστε μια τέτοια ομάδα αντικειμένων να μπορεί να τη διαχειριστεί ο χρήστης σαν ένα αντικείμενο. Το Group προσφέρει προσθήκη βασικής λειτουργικότητας, ενώ το Container παρέχει προχωρημένες ιδιότητες, όπως η κοινή χρήση αρχείων και κανόνες κληρονομικότητας.Το Layer είναι σαν μια διάφανη επίστρωση πάνω στην οποία ο χρήστης μπορεί να οργανώσει και να ομαδοποιήσει διαφορετικά είδη πληροφορίας για τη σκηνή. Αντικείμενα στο ίδιο layer μπορουν να μοιράζονται ίδιο χρώμα, ρυθμίσεις rendering και ρυθμίσεις εμφάνισης.



Εικόνα 3.18: Container (αριστερά) και Layer (δεξιά)

- State Sets και Scene Sets: Προσφέρουν γρήγορους τρόπους αποθήκευσης διαφορετικών καταστάσεων της σκηνής με διαφορετικές ιδιότητες στις οποίες μπορεί να ανατρέξει ο χρήστης όποτε θέλει, για να παράγει διαφορετικές ερμηνείες rendering ενός μοντέλου.
- Schematic View: Εμφανίζει τη σκηνή σαν ένα γραφικό σχήμα, αντί για γεωμετρία. Προσφέρει έναν εναλλακτικό τρόπο στο χρήστη να επιλέξει ή να μετονομάσει αντικείμενα στη σκηνή και να πλοηγηθεί μεταξύ των τροποποιήσεων (modifiers). Είναι εξαιρετικά χρήσιμο για τηνιεραρχική απεικόνιση των αντικειμένων.

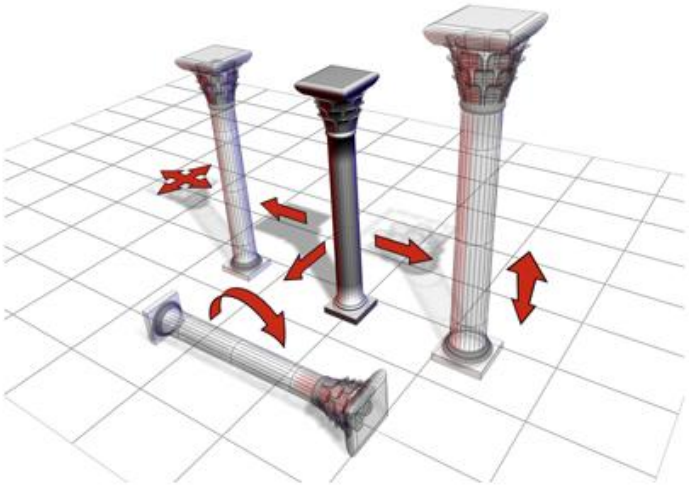


Εικόνα 3.19: Schematic View

Στο 3D Studio Max λοιπόν, ο χρήστης μπορεί να σχεδιάσει αντικείμενα τριδιάστατης γεωμετρίας ή διδιάστατα σχήματα και στη συνέχεια να τα επεξεργαστεί, μέσω των σχετικών εργαλείων που προσφέρονται. Επίσης, το πρόγραμμα προσφέρει ένα σύνολο έτοιμων αντικειμένων για χρήση. Οι αλλαγές που υφίσταται ένα αντικείμενο αποθηκεύονται σε μια στοίβα, δίνοντας τη δυνατότητα στο χρήστη να αλλάξει την επίδραση που είχε μια τροποποίηση, αφαιρώντας την προκείμενη επεξεργασία από τη στοίβα.

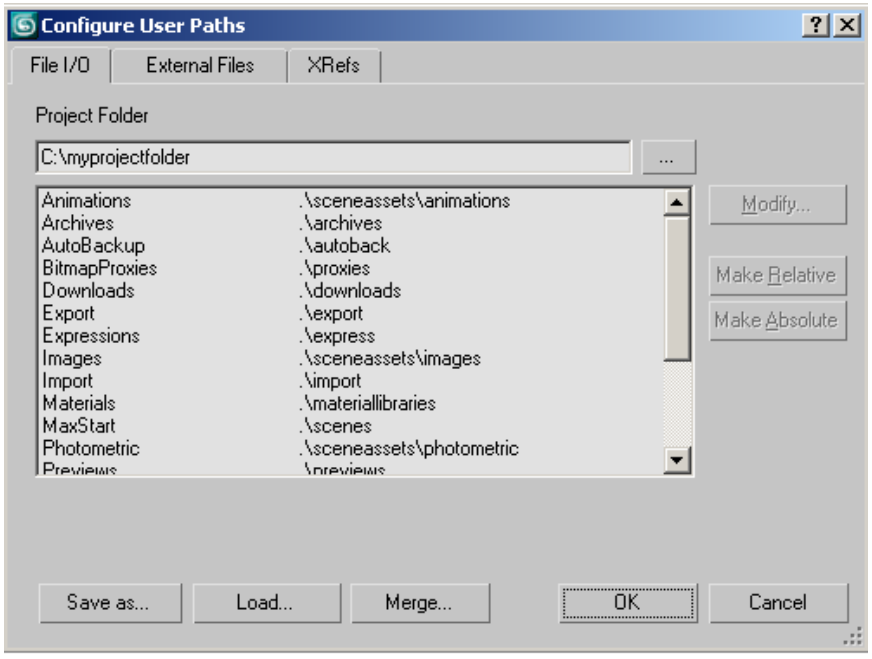
Η επιλογή ενός αντικειμένου είναι ουσιαστική για την εκτέλεση κάποιας ενέργειας επί αυτού και γίνεται με διάφορους τρόπους. Οι πιο κοινοί είναι με το ποντίκι ή το πληκτρολόγιο ή σύροντας μια περιοχή τριγύρω από το αντικείμενο. Ακόμα, η επιλογή μπορεί να γίνει βάση του ονόματος του αντικειμένου ή άλλων ιδιοτήτων του,

όπως το χρώμα ή η κατηγορία στην οποία υπάγεται. Επιπλέον, η επιλογή μπορεί να γίνει μέσω ενός schematic view, δηλαδή ενός παραθύρου που εμφανίζει τα αντικείμενα της τρέχουσας σκηνής σε ιεραρχία. Τέλος, μετά την επιλογή μπορεί να γίνει η τοποθέτηση των αντικειμένων στην κατάλληλη θέση, χρησιμοποιώντας εργαλεία transform για μετατόπιση (Move), για περιστροφή (Rotate) και για κλιμάκωση (Scale) και να γίνει ευθυγράμμιση (Align) των αντικειμένων βάσει ενός άξονα. Ακόμη, ο χρήστης μπορεί να κλειδώσει ή να κρύψει κάποια αντικείμενα ή μια κατηγορία αντικειμένων για να αποφύγει μη-επιθυμητή τροποποίησή τους, ενώ μπορεί μετά να τα ξεκλειδώσει ή να τα επανεμφανίσει.



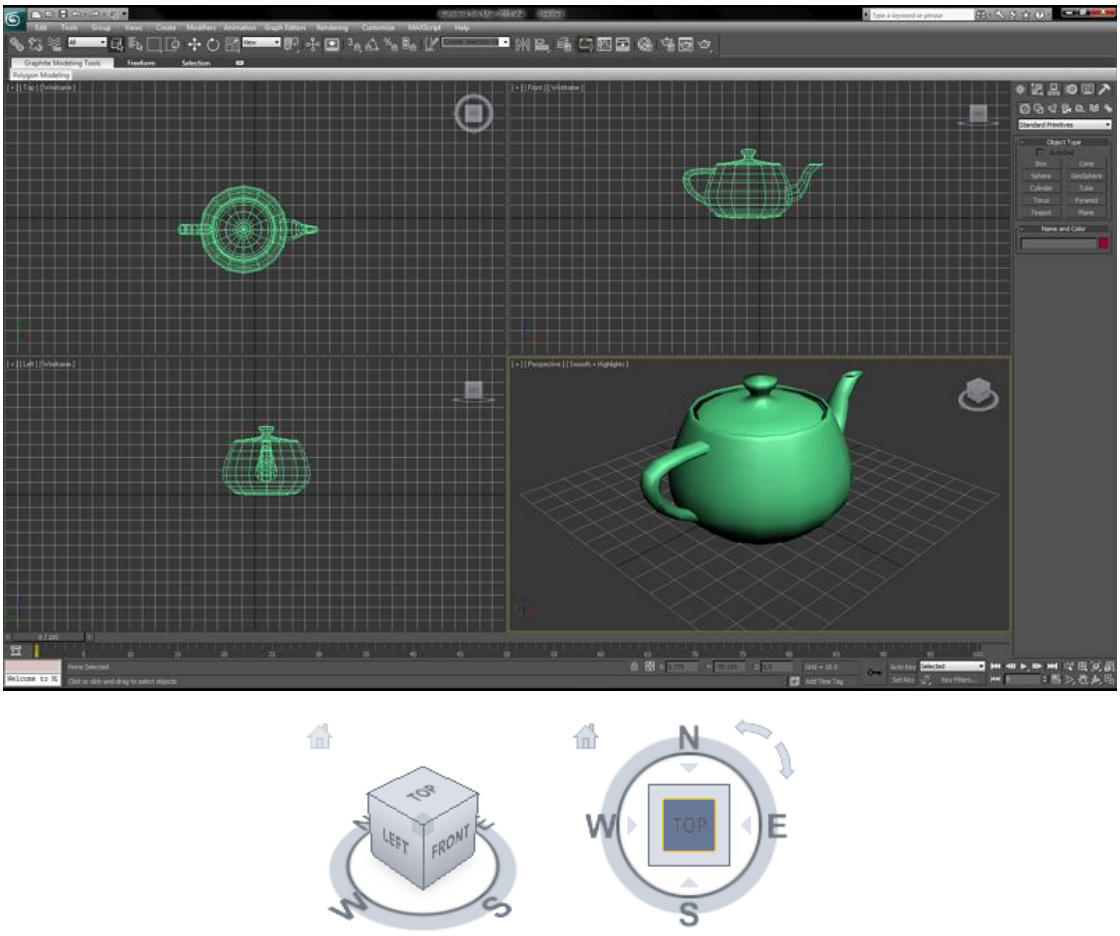
Εικόνα 3.20: Η κολώνα σε μετατόπιση, κλιμάκωση και περιστροφή

Η εφαρμογή επιτρέπει την εισαγωγή (Import) και επεξεργασία πλήθους τύπων αρχείου, πέρα από τα .3ds αρχεία που παράγει το ίδιο το πρόγραμμα. Επίσης, όταν επιλέγεται το άνοιγμα ενός αρχείου, η εφαρμογή θυμάται το τελευταίο μονοπάτι που χρησιμοποιήθηκε προς διευκόλυνση του χρήστη. Ωστόσο, προσφέρει και δυνατότητες ρύθμισης των μονοπατιών που χρησιμοποιούνται για διαφορετικά είδη αρχείων.



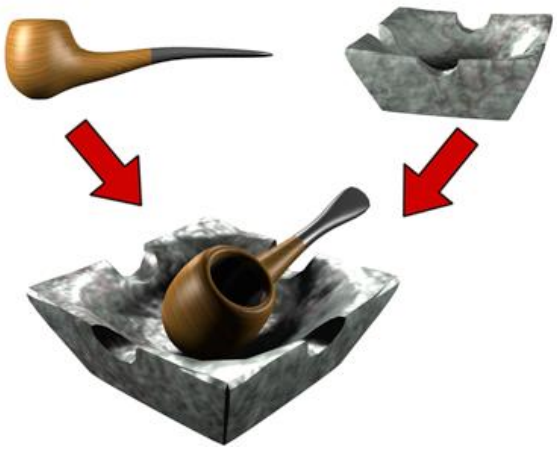
Εικόνα 3.21: Ρύθμιση μονοπατιών χρήστη

Κάθε αντικείμενο στη σκηνή τοποθετείται σε ένα τριδιάστατο κόσμο, τον οποίο ο χρήστης παρακολουθεί μέσα από ένα ή παραπάνω viewports. Υπάρχει μια ποικιλία επιλογών για την οπτικοποίηση αυτού του χώρου, από την πλήρη απλούστευση ως την εξαιρετική λεπτομέρεια που μπορεί να εμφανίσει. Κάθε viewport μπορεί να εμφανισει δυο οπτικές για το αντικείμενο: την αξονομετρική ή την προοπτική. Η περιήγηση στον τριδιάστατο χώρο γίνεται με την προσαρμογή της θέσης, της περιστροφής και του μεγέθους των views. Επίσης, ο προσανατολισμός ενός view μπορεί να αλλάξει με τη βοήθεια του εργαλείου View Cube, που εμφανίζεται σε κάθε view και πρόκειται για έναν τριδιάστατο κύβος που μας φανερώνει ποια πλευρά του viewport βλέπουμε και ποιος ο προσανατολισμός του.



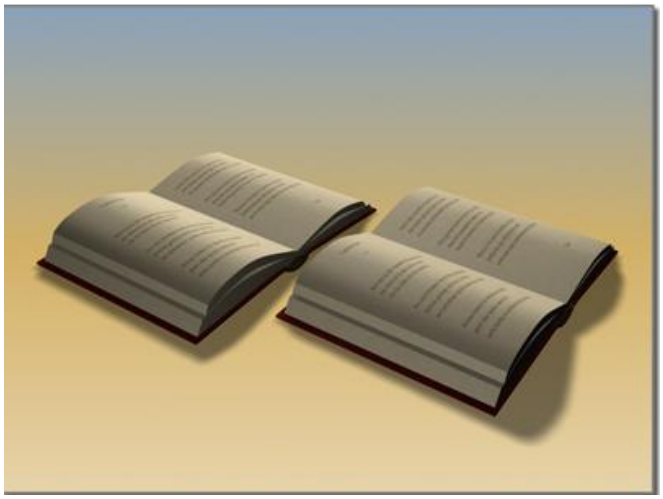
Εικόνα 3.22: ΤeapotστηνviewportsκαιοCubeView

Επίσης, μια ενδιαφέρουσα επιλογή είναι το Merge όπου διαφορετικές σκηνές συνδυάζονται για την παραγωγή μιας νέας σκηνής. Κατά την σύντμηση ενός αρχείου, ο χρήστης μπορεί να επιλέξει ποια αντικείμενα να συγχωνεύσει. Ακόμη, προσφέρεται η δυνατότητα της αντικατάστασης (Replace) ενός αντικειμένου με ένα άλλο, με τη προϋπόθεση ότι έχουν το ίδιο όνομα. Αυτό γίνεται για απλοποίηση της σκηνής, χρησιμοποιώντας πιο ελαφριά αντικείμενα και αντικαθιστώντας τα με τα τελικά μοντέλα πριν το rendering.



Εικόνα 3.23: Συγχώνευση αντικειμένων

Τέλος, ο χρήστης μπορεί να δημιουργήσει διαφορετικά είδη αντιγράφων ενός αντικειμένου: copies, instances ή references. Το copy δημιουργεί ένα πιστό αντίγραφο του αντικειμένου, αλλαγές στο οποίο δεν επηρεάζουν το αρχικό. Αντίθετα, το instance είναι αντίγραφο του πρωτότυπου, αλλαγές στο οποίο αντικατοπτρίζονται και στο πρωτότυπο. Το reference, από την άλλη, είναι ένας συνδυασμός των δυο, με την έννοια ότι αλλαγές στις παραμέτρους που είχε και το πρωτότυπο όταν δημιουργήθηκε το αντίγραφο, θα επηρεάσει και τα δυο αντικείμενα, ενώ τροποποιήσεις σε άλλες παραμέτρους επιδρούν μόνο στο αντίγραφο.



Εικόνα 3.24: Δημιουργία αντιγράφου

Οι ρυθμίσεις και οι επιλογές που προσφέρει η εφαρμογή 3D Studio Max είναι πραγματικά πολλές και καλύπτουν ένα εύρος περιπτώσεων, καθιστώντας το πρόγραμμα δύσκολο στον χειρισμό για έναν αρχάριο, αλλά παράλληλα προσφέροντας φοβερές δυνατότητες για τη δημιουργία ρεαλιστικών σκηνών. Σε αυτό το κεφάλαιο, κάναμε μια σύντομη αναφορά στη λειτουργικότητα του, μελετώντας το user interface του και εστιάζοντας μονάχα σε ένα μικρό υποσύνολο των εργαλείων του, βάσει των αναγκών που θέλουμε να εξυπηρετήσει και η δική μας εφαρμογή.

ΚΕΦΑΛΑΙΟ 4 – Η ΑΝΑΠΑΡΑΣΤΑΣΗ REVE

Σε αυτό το κεφάλαιο γίνεται μια προσέγγιση της αναπαράστασης REVE (Representation and Execution of Virtual Environments), η οποία αποτελεί μια υψηλού επιπέδου αναπαράσταση εικονικών κόσμων που χρησιμοποιείται στην πλατφόρμα REVE. Η πλατφόρμα REVE έχει αναπτυχθεί από το Εργαστήριο Τεχνολογίας Γνώσης του Πανεπιστημίου Πειραιώς και αποτελείται από ένα σύνολο εφαρμογών και προγραμματιστικών εργαλείων τα οποία επιτρέπουν την ανάπτυξη ευφυών εικονικών περιβαλλόντων.

Η αναπαράσταση REVE αναπαριστά έναν εικονικό κόσμο ως μια δομή δεδομένων που συντίθεται από αυτόνομα και ανεξάρτητα μεταξύ τους αντικείμενα (items). Κάθε αντικείμενο, όπως και στον πραγματικό κόσμο, έχει μια φυσική αναπαράσταση (physical aspect) που αφορά την υπόσταση του αντικειμένου στον χώρο (σχήμα, μέγεθος, χρώμα, υφή, υλικό, θέση), μια σημασιολογική αναπαράσταση (semantic aspect) που αφορά στην πληροφορία που μπορούν να εκλάβουν οι χρήστες ενός εικονικού κόσμου για το αντικείμενο, όπως οι ευφυείς εικονικοί πράκτορες, και μια αναπαράσταση λειτουργικότητας (access aspect) που αφορά τη δυνατότητα αλληλεπίδρασης ενός χρήστη ή πράκτορα με το αντικείμενο.

Η αναπαράσταση REVE υλοποιείται στη γλώσσα αναπαράστασης VERL, που αναπτύχθηκε για αυτό τον σκοπό, ενώ η οπτικοποίηση του τριδιάστατου κόσμου και η αλληλεπίδραση με τους πράκτορες επιτυγχάνεται μέσω του συστήματος Reve Worlds.

4.1 Η γλωσσα verl

Η γλώσσα VERL (Virtual Environment Representation Language) αναπτύχθηκε το 2010 από τον Γ. Αναστασάκη, αναπληρωτή καθηγητή του Πανεπιστημίου Πειραιώς, για την αναπαράσταση εικονικών κόσμων. Η VERL είναι μια γλώσσα βασισμένη στη μεταγλώσσα XML και χρησιμοποιεί μοντέλα 3D αντικειμένων που έχουν συνταχθεί σε VRML97 ή X3D για να ορίσει έναν εικονικό κόσμο για το σύστημα REVE Worlds (Representation and Execution of Virtual Environments).

Η γλώσσα VERL λειτουργεί ως ένα προτυποποιημένο και διαφανές σημείο διασύνδεσης μεταξύ του εννοιολογικού επιπέδου και του επιπέδου υλοποίησης ενός εικονικού κόσμου, επιτυγχάνοντας ξεκάθαρη διάκριση μεταξύ αυτών των δυο πτυχών του κόσμου. Στην ουσία, επιτυγχάνει να γεφυρώσει την εννοιολογική, υψηλού επιπέδου περιγραφή ενός εικονικού κόσμου με την τεχνική, χαμηλού επιπέδου υλοποίησή του, στα πλαίσια ενός συγκεκριμένου εικονικού περιβάλλοντος. Η ανεξαρτησία του εννοιολογικού επιπέδου επιτυγχάνεται μέσω της XML σύνταξης, ενώ το επίπεδο υλοποίησης αντιστοιχεί στις τιμές συγκεκριμένων ορισμάτων του κόσμου.

Η δομή της VERL ορίζεται από ενα DTD (Document Type Definition). Κάθε .verl αρχείο εμπεριέχει τα αντικείμενα (items) που συνθέτουν τον εικονικό κόσμο. Ως items ορίζονται τα «*ξεχωριστά συστατικά ενός εικονικού κόσμου που διακρίνονται από όσο το δυνατόν απλούστερη γεωμετρική δομή, μέγιστη σημασιολογική και αλληλεπιδραστική αυτονομία, καθώς και γενικότερα, ως προς το ρόλο τους στον εικονικό κόσμο*» (Anastassakis G. 2010). Αν τα αντικείμενα αυτά βρίσκονται στην ίδια τοποθεσία προέλευσης, δηλαδή στο ίδιο VRML/X3D αρχείο, τότε υπάγονται και στο ίδιο itemGroup, ενώ σε διαφορετική περίπτωση υπάγονται σε διαφορετικά itemGroup.

Κάθε <itemGroup> έχει 3 attributes:

- name: το όνομα του itemGroup, όπως το ορίζει ο προγραμματιστής
- location: η τοποθεσία του αρχείου VRML ή X3D στο οποίο υπάγονται τα αντικείμενα που μας ενδιαφέρουν
- behaviours: παράμτετρος που παίρνει τιμές true/false ανάλογα αν θα οριστεί συμπεριφορά για αυτό το itemGroup ή όχι.

Κάθε item της VERL αντιστοιχεί σε έναν κόμβο κάποιου VRML ή X3D αρχείου, το οποίο ορίζεται και διευκρινίζεται στο location του itemGroup.

Το <item> έχει επίσης 4 attributes:

- name: το όνομα του item, όπως το ορίζει ο χρήστης

- class: παράμετρος που παίρνει τιμές real ή unreal ανάλογα αν ο πράκτορας θα αντιλαμβάνεται αυτό το αντικείμενο ή όχι, επηρεάζοντας τον τρόπο αλληλεπίδρασης και την συμπεριφορά του πράκτορα
- fit: παράμετρος που παίρνει τιμές true ή false, ανάλογα αν θέλουμε το τοπικό σύστημα συντεταγμένων του αντικειμένου να ευθυγραμμίζεται με το σύστημα συντενταγμένων του κόσμου ή όχι
- fitcentre: παράμετρος που παίρνει τιμές true ή false, ανάλογα αν θέλουμε το κέντρο του bounding box του αντικειμένου να συμπίπτει με την αρχή των αξόνων του συστήματος συντεταγμένων του κόσμου.

Η δομή, επομένως, ενός itemGroup έχει την παρακάτω μορφή:

```
<itemGroup name="<itemGroupName>" location="<location>" behaviours="true">
  <item name="<itemName>" class="<itemClass>" fit="true" fitCentre="false">
  </item>
</itemGroup>
```

Για κάθε item, η VERL προσφέρει 3 επίπεδα ορισμού μοντέλων: το virtual model, το semantic model και το access model, που αντιστοιχούν στις διάφορες όψεις που έχει ένα αντικείμενο. Πιο συγκεκριμένα, το virtual model αφορά στη φυσική αναπαράσταση του αντικειμένου (physical aspect), το semantic model αφορά στην σημασιολογική άποψη του αντικειμένου (semantic aspect), που επιτυγχάνεται με την χρήση συμβόλων, ενώ το access model αφορά στις δυνατότητες δράσης του αντικειμένου και αλληλεπίδρασης ενός χρήστη με αυτό, μέσω σημείων πρόσβασης (acess points) και μεθόδων (functions) που προσδίδουν τη λειτουργικότητα.

Στο <virtual model> ορίζεται το χαρακτηρισικό source στο οποίο αναγράφεται η πηγή του αντικειμένου, δηλαδή το όνομα που έχει ο κόμβος που μας ενδιαφέρει στο αντίστοιχο VRML/X3D αρχείο. Προϋπόθεση, λοιπόν, συμμετοχής ενός κόμβου στην αναπαράσταση REVE είναι να πρόκειται για έναν ονοματισμένο κόμβο στο αντίστοιχο VRML/X3D αρχείο, συνεπώς για έναν DEF Transform κόμβο.

Επίσης, στο virtual model περιλαμβάνεται ο ορισμός του στοιχείου <transform>, όπου ορίζονται πιθανοί μετασχηματισμοί του αντικειμένου στο χώρο και πιο συγκεκριμένα περιλαμβάνει τα attributes:

- translation: μετόπιση του αντικειμένου στους άξονες x y z
- rotation: περιστροφή του αντικειμένου κατά ακτίνια (rad) βάσει κάποιου άξονα
- scale: μεταβολή του μεγέθους ενός αντικειμένου ανα άξονα.

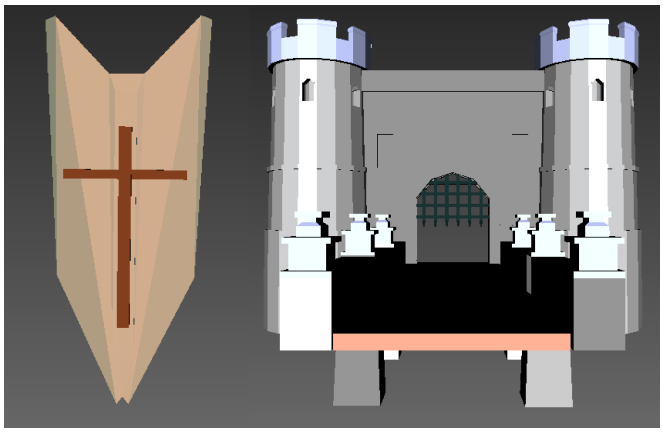
Η δομή, λοιπόν, του virtual model ενός item έχει την παρακάτω μορφή:

```
<item name="<itemName>" class="<itemClass>" fit="true" fitCentre="false">

  <virtualModel source="<virtualModelSource>">
    <transform translation="<tx ty tz>" rotation="<rx ry rz ra>" scale="<sx sy
sz>"/>
  </virtualModel>

</item>
```

Ένα παράδειγμα χρήσης του virtual model είναι το παρακάτω, όπως το χρησιμοποιήσαμε στην μεταπτυχιακή μας εργασία στο μάθημα 'Ευφυή Εικονικά Περιβάλλοντα', για την τοποθέτηση δυο ασπίδων στον εικονικό κόσμο, έτσι ώστε να συμπέσουν πάνω στις πύλες ενός κάστρου:

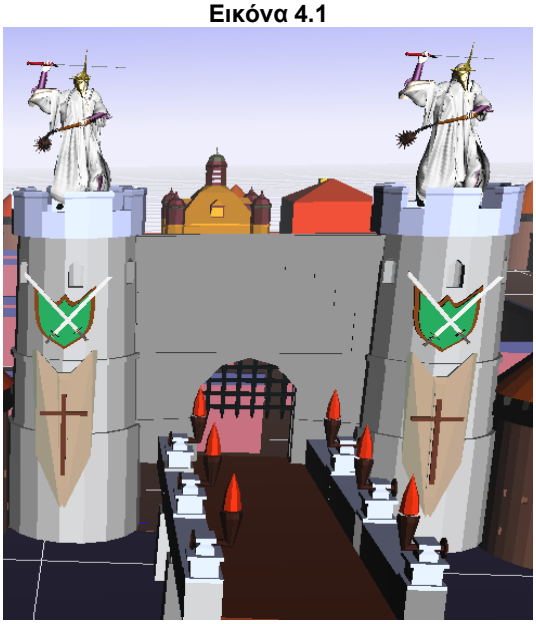
```
<itemGroup name="shield" location="knight_shield.wrl" behaviours="true">
  <item name="shield1" class="real.general" fit="true">
    <virtualModel source="Shield001">
        <transform translation="4.5 9 3.5" rotation="0.4 0 0 1.5" scale="0.2
0.2 0.2"/>
    </virtualModel>
  </item>
  <item name="shield2" class="real.general" fit="true">
    <virtualModel source="Shield001">
```

```
        <transform translation="18.5 9 3.5" rotation="0.4 0 0 1.5" scale="0.2
0.2 0.2"/>
    </virtualModel>
  </item>
</itemGroup>
```



α) ασπίδα (wrl) **β) πύλη (wrl)**

Εικόνα 4.1



γ) Πύλη με ασπίδες, μέσω verl

Εικόνα 4.2

Όπως αναφέρθηκε, η VERL δίνει τη δυνατότητα ορισμού και ενός σημασιολογικού μοντέλου (semantic model) για κάθε αντικείμενο. Πιο συγκεκριμένα, για αυτόν τον σκοπό, μέσα σε κάθε item μπορούμε να ορίσουμε ένα στοιχείο <semanticModel>, το οποίο παρέχει πληροφορίες για κάθε αντικείμενο με τη μορφή συμβόλων (symbols).

Τα σύμβολα διακρίνονται σε δυο κατηγορίες: α) αυτά που είναι προκαθορισμένα και αφορούν γενικές πληροφορίες για το αντικείμενο, όπως η κλάση του, η τοποθεσία του, ο προσανατολισμός του και οι διαστάσεις του bounding box του και β) αυτά που μπορούν να οριστούν στην εφαρμογή, δίνοντας επιπλέον δυνατότητες σε κάθε αντικείμενο. Στην ουσία, ο ορισμός συμβόλων για κάθε είδους σημασιολογία για ένα item, επιτρέπει στο χρήστη να αντιμετωπίζει όλα τα πιθανά προβλήματα ενός μοντέλου.

Κάθε στοιχείο <symbol> έχει ένα attribute name, όπου ορίζεται η ονομασία του. Κάθε item μπορεί να έχει ένα ή παραπάνω symbols και κάθε symbol στοιχεία τύπου <argument>, τα οποία λαμβάνουν τα παρακάτω 2 attributes:

- class: σε επίπεδο προδιαγραφών πρόκειται για τον σημασιολογικό ορισμό (semantics) των παραμέτρων (args) που ακολουθούν. Σε επίπεδο υλοποίησης, προσδιορίζει την java κλάση ενός αντικειμένου που θα παρέχει τιμή για την παράμετρο. Η αναπαράσταση REVE προσφέρει 2 built-in argument classes, οι οποίες είναι οι:
  - *X3DConnectorL1*: Για αυτή την κλάση, η τιμή της παραμέτρου αντιστοιχεί στην τιμή ενός συγκεκριμένου πεδίου ενός κόμβου στην ιεραρχία του virtual model. Το πεδίο πρέπει να είναι προσβάσιμο, είτε ως τύπος εισόδου είτε ως τύπος εξόδου και η τιμή του προωθείται ως έχει, χωρίς μετατροπές, παρά μόνο περικλείοντάς την σε string. Σε περίπτωση που η τιμή του πεδίου δεν είναι ορισμένη ή προκύψει κάποιο σφάλμα, η παράμετρος παίρνει την τιμή "<null>". Τέλος, κάθε φορά που απαιτείται η τιμή της παραμέτρου, η τιμή του πεδίου διαβάζεται εκ νέου, ορίζοντας μια ενεργή σχέση μεταξύ του πεδίου και της παραμέτρου. Η πλήρης ονομασία της κλάσης είναι reve.scene.item.semantic.argument.X3DConnectorL1
  - *ZeroArgumentItemMethodConnectorL1*: Η τιμή της παραμέτρου για αυτή την κλάση είναι η τιμή που επιστρέφει μια συγκεκριμένη public μέθοδος του item του οποίου το semantic model εμπεριέχει το symbol. Αντίστοιχα και εδώ, η τιμή δεν επιδέχεται μετατροπές και περνιέται ως string. Αν η τιμή επιστροφής είναι null ή σε περίπτωση σφάλματος η τιμή του αντικειμένου είναι "<null>" και ανανεώνεται κάθε φορά που ζητείται, τρέχοντας εκ νέου τη μέθοδο για την τιμή επιστροφής. Η πλήρης ονομασία της κλάσης είναι reve.scene.item.semantic.argument. ZeroArgumentItemMethodConnectorL1
- args: Πρόκειται για τις παραμέτρους του συγκεκριμένου <argument> που, αν ειναι παραπάνω από μια, διακρίνονται με κενό ή κόμματα. Η τιμή αυτού του attribute παίρνει ως string τις παραμέτρους που παράγει η μέθοδος δημιουργίας του αντικειμένου που προσδιορίζεται στο attribute class και παρέχει τιμή για την παράμετρο.

Η δομή, λοιπόν, του σημασιολογικού μοντέλου έχει την παρακάτω μορφή:

```
<item name="<itemName>" class="<itemClass>" fit="true" fitCentre="false">

  <semanticModel>
    <symbol name="<symbolName>">
      <argument class="<argumentClass>" args="<argumentInitializationArgs>" />
    </symbol>
  </semanticModel>

</item>
```

Τέλος, η VERL παρέχει και ένα μοντέλο πρόσβασης (access model) στο αντικείμενο, όπως διατυπώθηκε προηγουμένως. Ο ορισμός αυτού του μοντέλου επιτρέπει την προσθήκη λειτουργικότητας σε ένα αντικείμενο και συνεπώς, τη διάδραση ενός χρήστη ή ενός εικονικου πράκτορα με αυτό. Η συμπερίληψη του access model στον κόσμο γίνεται με τον ορισμό του στοιχείου <accessModel> εντός ενός στοιχείου <item>.

Στη συνέχεια, κάτω από το στοιχείο <accessModel> εισάγεται ένα σημείο πρόσβασης (access point), μέσα από το οποίο επιτυγχάνεται η αλληλεπίδραση με το αντικείμενο. Στην ουσία, το σημείο πρόσβασης λειτουργεί ως συνδετικός κρίκος μεταξύ του αντικειμένου και της λειτουργικότητάς του. Για παράδειγμα, ένα access point μπορεί να είναι ένα κουμπί ενεργοποίησης ή απενεργοποίησης μιας κίνησης. Κάθε σημείο πρόσβασης αναπαρίσταται και ως αντικείμενο στον κόσμο, επομένως αντίστοιχα διακρίνεται από τα χαρακτηριστικά της θέσης, του προσανατολισμού και των διαστάσεων, με την ιδιαιτερότητα πως αυτά ορίζονται πάντα σε θέση σχετική με το τοπικό σύστημα συντεταγμένων του αντικειμένου για το οποίο προορίζεται. Κατά αυτόν τον τρόπο, η μεταβολή των χαρακτηριστικών του αντικειμένου θα παρασύρει και το αντίστοιχο σημείο

πρόσβασης, οπότε π.χ. αν ένα κουμπί βρίσκεται πάνω στο αντικείμενο και το αντικείμενο μετακινηθεί, ανάλογα θα μετακινηθεί και το κουμπί, διατηρώντας τη λογική σχέση των δυο στο χώρο.

Ο ορισμός ενός σημείου πρόσβασης γίνεται με το στοιχείο <accessPoint> που παίρνει τα παρακάτω 2 attributes:

- name: το όνομα που προσδίδει ο χρήστης στο σημείο πρόσβασης
- node: το όνομα του κόμβου στο γράφημα σκηνής του αντικειμένου, με το οποίο θα συσχετιστεί το εν λόγω σημείο πρόσβασης

Τέλος, για να προστεθεί και η λειτουργικότητα που συνεπάγεται το σημείο πρόσβασης, π.χ. με το πάτημα ενός κουμπιού το αντικείμενο να μετακινηθεί 10 μέτρα στον άξονα x, δηλαδή τι ενέργεια θα εκτελεστεί στο αντικείμενο εφόσον ο πράκτορας ή ο χρηστης αλληλεπιδράσει με αυτό, ορίζουμε το στοιχείο <function> που παίρνει τα παρακάτω 3 attributes:

- name: το όνομα της μεθόδου που εκτελείται, όπως το ορίζει ο χρήστης
- class: η κλάση στην οποία ανήκει η μέθοδος που εκτελείται. Σε επίπεδο προδιαγραφών πρόκειται για τον ορισμό του τρόπου με τον οποίο θα τροποποιηθεί ένα αντικείμενο, μόλις εκτελεστεί η μέθοδος. Σε επίπεδο υλοποίησης, προσδιορίζει την java κλάση που αποτελεί την υλοποίηση της μεθόδου. Η αναπαράσταση REVE προσφέρει 3 built-in function classes για τη χρήση τους στη VERL, οι οποίες είναι οι εξής:
  - *AccessPointTranslateFunctionL1*: η κλάση αυτή χρησιμοποιείται για να μετατοπίσει ένα αντικείμενο σε σχέση με το σημείο πρόσβασής του. Η μετατόπιση μπορεί να είναι είτε απόλυτη, είτε σχετική και ο κόμβος που σχετίζεται με το access point πρέπει να υποστηρίζει μετατόπιση, συνεπώς, να είναι Transform κόμβος. Δεν υπάρχουν παράμετροι αρχικοποίησης για αυτή την κλάση, παρά μόνο παράμετροι εκτέλεσης της μορφής: '<x y z>, <relative>' που δηλώνουν αν η μετατόπιση θα είναι απόλυτη ή σχετική (τιμή false ή true αντίστοιχα) και κατά πόσο (σε μέτρα) για κάθε άξονα.

    Το πλήρες όνομα της κλάσης είναι reve.scene.item.access.function.AccessPointTranslateFunctionL1
  - *AccessPointRotateFunctionL1*: η κλάση αυτή χρησιμοποιείται για να περιστρέψει ένα αντικείμενο σε σχέση με το σημείο πρόσβασής του. Η περιστροφή μπορεί να είναι είτε απόλυτη, είτε σχετική και ο κόμβος που σχετίζεται με το accesspoint πρέπει να την υποστηρίζει, συνεπώς, να είναι Transform κόμβος. Δεν υπάρχουν παράμετροι αρχικοποίησης για αυτή την κλάση, παρά μόνο παράμετροι εκτέλεσης της μορφής: '<x y z a>, <relative>' που δηλώνουν αν η περιστροφή θα είναι απόλυτη ή σχετική (τιμή false ή true αντίστοιχα), σε ποιον άξονα θα γίνει (τιμή 1 ή 0) και κατά πόσο (σε ακτίνια) για κάθε άξονα. Το πλήρες όνομα της κλάσης είναι reve.scene.item.access.function.AccessPointRotateFunctionL1
  - *X3DFieldFunctionL1*: η κλάση χρησιμοποιείται για να γράψει μια τιμή σε ένα συγκεκριμένο πεδίο ενός συγκεκριμένου κόμβου στη δομή του αντικειμένου. Προϋπόθεση αποτελεί το πεδίο αυτό να είναι εγγράψιμο, δηλαδή ο τύπος πρόσβασής του να είναι είτε "inputOutput" είτε "inputOnly". Η μέθοδος προσπαθεί να μετατρέψει την δοσμένη τιμή στον τύπο του πεδίου. Αν η μετατροπή αποτύχει, η εκτέλεση σταματάει. Οι παράμετροι αρχικοποίησης είναι της μορφής: '<node_name>, <field_name>', ενώ οι παράμετροι εκτέλεσης εξαρτώνται από τον τύπο του πεδίου. Το πλήρες όνομα της κλάσης είναι reve.scene.item.access.function.X3DFieldFunctionL1.
- args: αφορά τις παραμέτρους αρχικοποίησης που μπορεί να υπάρχουν για τη συγκεκριμένη μέθοδο

Η δομή, λοιπόν, του μοντέλου αλληλεπίδρασης έχει την παρακάτω μορφή:

```
<item name="<itemName>" class="<itemClass>" fit="true" fitCentre="false">

 <accessModel>
  <accessPoint name="<accessPointName>" node="<nodeName>">
   <function name="<functionName>" class="<functionClass>"
args="<functionInitializationArgs>" />
  </accessPoint>
```

```
 </accessModel>

</item>
```

Ένα παράδειγμα χρήσης του access model προκύπτει από την μεταπτυχιακή μας εργασία για το μάθημα 'Ευφυή Εικονικά Περιβάλλοντα', όπου διάφοροι πράκτορες περιπλανιώνται σε έναν εικονικό κόσμο, εις αναζήτηση μιας βόμβας. Η βόμβα μας είχε έναν μοχλό που κινούταν όσο η βόμβα ήταν ενεργή. Όταν ο πράκτορας-πυροτεχνουργός έβρισκε τη βόμβα, τοποθετούσε το χέρι του πάνω σε ένα κουμπί (το access point) για να την απενεργοποιήσει, γεγονός που απεικονιζόταν με το σταμάτημα του μοχλού της βόμβας.

Για να επιτύχουμε αυτό το αποτέλεσμα, στο .x3d αρχείο μας με τον κόμβο Bomba, τοποθετήσαμε μέσα στα ομαδοποιημένα προς κίνηση αντικείμενα, δηλαδή τον μοχλό, μία συνάρτηση «SpeakerVibration-TS TimeSensor» και στο τέλος του αντικειμένου ορίζαμε ένα διακόπτη, τον «MultiSwitch-SC Script» κόμβο, για την διαδραστικότητα της απενεργοποίησης της βόμβας μέσω μίας συνάρτησης, της setActive. Η λειτουργικότητα στο συγκεκριμένο αντικείμενο ορίζεται στην verl όπως παρακάτω:

```
<abr version="1.0">
  <world version="1.1" name="key_bombTest" dimensions="100.0, 261.8">
    <itemGroup          name="bomb_example"          location="key_bomb.x3dv"
behaviours="true">
      <item name="bomb" class="real.general" fit="true">
        <virtualModel source="Bomba">
          <transform />
        </virtualModel>
        <accessModel>
          <accessPoint name="buttons" node="MultiSwitch-SC">
            <function                                              name="setActive"
              class="reve.scene.item.access.function.X3DFieldFunctionL1"
              args="setActive"/>
          </accessPoint>
        </accessModel>
      </item>
   </itemGroup>
  </world>
</abr>
```

Κατά αυτόν τον τρόπο, όταν ο πράκτορας-πυροτεχνουργός έβλεπε τη βόμβα και την πλησίαζε, μπορούσε να τοποθετήσει το χέρι του στο σημείο πρόσβασης (κουμπί), μέσω του οποίου καλούνταν η μέθοδος setActive που σταματούσε την κίνηση του μοχλού.

Εικόνα 4.3: Βόμβα με μοχλό και κουμπί απενεργοποίησης

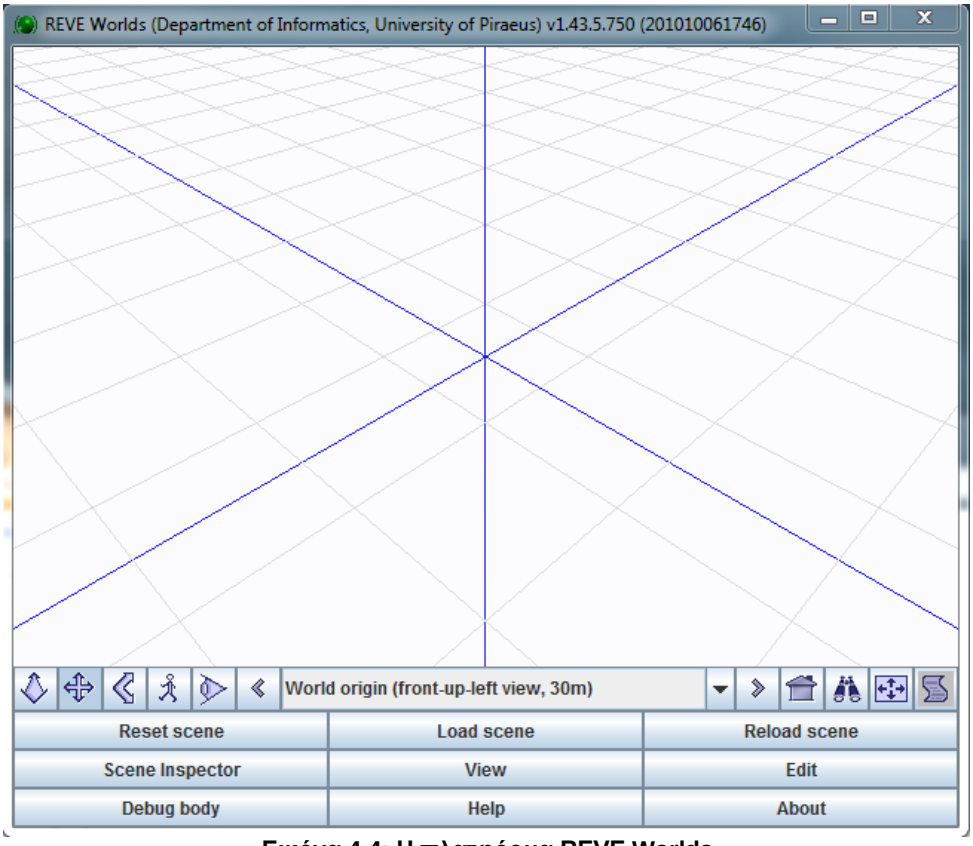
Συμπερασματικά, η VERL, δεδομένου ότι βασίζεται στην μεταγλώσσα XML, διακρίνεται και από τα αντίστοιχα πλεονεκτήματα που αυτή προσφέρει. Συνεπώς, παρέχει τη δυνατότητα υλοποίησης σε διαφορετικές πλατφόρμες, είναι ανεξάρτητη από το περιβάλλον εκτέλεσης, είναι απλή και φιλική προς το χρήστη, είναι επεκτάσιμη και αποτελεί ένα ισχυρό, εκφραστικά, εργαλείο για τη μοντελοποίηση εικονικών

κόσμων σύμφωνα με την αναπαράσταση REVE. Η VERL, ως εκ τούτου, μπορεί να γίνει αντικείμενο επεξεργασίας από πλήθος διαθέσιμων σχεδιαστικών εργαλείων και βιβλιοθηκών λεκτικής και συντακτικής επεξεργασίας, διατηρώντας ξεκάθαρη και ανεπηρέαστη, σε κάθε περίπτωση, τη σημασιολογία την οποία αναπαριστά, καθιστώντας την ιδανική για αναπαραστάσεις εικονικών κόσμων.

4.2 Το σύστημα Reve Worlds

Το σύστημα REVE Worlds είναι μια εφαρμογή που ενσαρκώνει την αναπαράσταση εικονικών κόσμων με βάση την αναπαράσταση REVE, η οποία επιτρέπει μια υψηλού επιπέδου αναπαράσταση του κόσμου, διαχωρίζοντας το εννοιολογικό πλαίσιο από την υλοποίηση του. Πιο συγκεκριμένα, προσεγγίζει γενικές, αλλά ταυτόχρονα, βασικές έννοιες όπως: αντικείμενο, σημασιολογία αντικειμένου, λειτουργικότητα, αλληλεπίδραση, κόσμος, διακρίνοντας τα συστατικά που αποτελούν έναν εικονικό κόσμο από την υλοποίησή του για μια συγκεκριμένη εφαρμογή, γεγονός που κάνει την αναπαράσταση REVE ιδιαίτερα εύχρηστη, ευέλικτη και επεκτάσιμη.

Η εφαρμογή REVE Worlds αποτελείται από μια διεπαφή (API) καιαπό ένα σύνολο συστατικών λογισμικού και βοηθημάτων. Είναι γραμμένη σε γλώσσα java και χρησιμοποιεί τη βιβλιοθήκη Xj3D, η οποία αποτελεί μια υλοποίηση του προτύπου X3D σε java. Ως εκ τούτου, υποστηρίζει αντικείμενα και μοντέλα σκηνής που βασίζονται στο πρότυπο X3D καθώς και στην VRML97, της οποίας αποτελεί εξέλιξη, προσφέροντας έτσι ένα μεγάλο εύρος τριδιάστατων μοντέλων. Από την άλλη, η γλώσσα VERL αποτελεί το συνδετικό εργαλείο ανάμεσα στην πλατφόρμα REVE Worlds και την αναπαράσταση REVE. Συνεπώς, η αναπαράσταση ενός κόσμου στην πλατφόρμα επιτυγχάνεται με το άνοιγμα ενός αρχείου .verl σε αυτήν, κατά το οποίο φορτώνεται ο κόσμος και εμφανίζεται σε ένα τριδιάστατο άξονα (πλέγμα) συντεταγμένων.

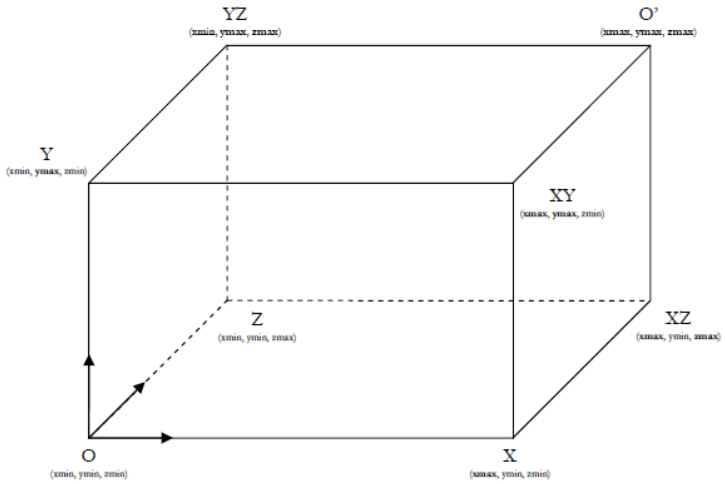


Εικόνα 4.4: Η πλατφόρμα REVE Worlds

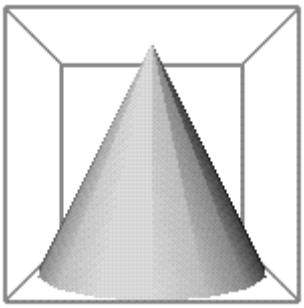
Η Reve Worlds είναι ιδιαίτερα φιλική προς τον χρήστη, προσφέροντάς του μια ποικιλία δυνατοτήτων μέσω εργαλείων του GUI. Για παράδειγμα, ο χρήστης μπορεί να περιηγηθεί στον εικονικό κόσμο και να έχει την αίσθηση πως είτε περπατάει σε αυτόν είτε τον μελετάει από ψηλά, με την εντύπωση πως πλανιέται από πάνω του. Επίσης, ο χρήστης, μπορεί να μεταφερθεί σε ένα συγκεκριμένο μέρος του χώρου όπου θα επιλέξει, μελετώντας από κοντινή απόσταση αυτό το συγκεκριμένο τμήμα του κόσμου και επιτρέποντας τη γρήγορη μετάβαση από ένα σημείο σε ένα άλλο. Ακόμα, ο χρήστης μπορεί να επιλέξει διαφορετικές οπτικές γωνίες παρατήρησης του κόσμου και να επιστρέψει οποιαδήποτε στιγμή στην πρωταρχική οπτική γωνία, δηλαδή την άποψη που είχε ο κόσμος όταν φορτώθηκε για πρώτη φορά στην πλατφόρμα. Τέλος, ο χρήστης μπορεί να εκτελέσει επιπλέον ενέργειες που του επιτρέπουν να επεξεργαστεί και να βελτιώσει την ανάπτυξη του εικονικού κόσμου, μεταξύ των οποίων είναι η επαναφόρτωση ή ο επανορισμός μιας σκηνής, η προσθαφαίρεση οπτικών γωνιών παρατήρησης και η μελέτη της δομής του γραφήματος σκηνής και των αντικειμένων/κόμβων, καθώς και των bounding boxesπου τα περικλείουν. Επίσης, μέσω εργαλείων της REVE είναι εφικτή και η αλληλεπίδραση με αντικείμενα που χαρακτηρίζονται από κάποια λειτουγικότητα, δηλαδή έχουν accesspoints και functions.

Είναι σημαντικό να σημειώσουμε πως στην εφαρμογή REVE Worlds, κάθε <item> που ορίζεται στην VERL, ενώ εμφανίζεται ως συμπαγές αντικείμενο στον τριδιάστατο χώρο, περικλείεται από ένα προσανατολισμένο ορθογώνιο παραλληλεπίπεδο, μοναδικό για κάθε αντικείμενο και γνωστό στην βιβλιογραφία ως OrientedBoundingBox (OBB). Προϋπόθεση σωστού υπολογισμού του bounding box ενός αντικειμένου είναι η εισαγωγή του στον κόσμο ως συνόλου συνδεδεμένων ευθύγραμμων τμημάτων ή επιφανειών και όχι ως συμπαγών πρωτογενών σχημάτων. Για παράδειγμα, αν έχουμε ένα αντικείμενο-σφαίρα στην VRML, αντί να το φορτώσουμε ως γεωμετρία Sphere, θα πρέπει να το μετατρέψουμε σε γεωμετρία IndexedFaceSet, πράγμα που γίνεται αυτόματα από το εργαλείο επεξεργασίας VRML αντικειμένων, VrmlPad.

Στην ουσία, το Oriented Bounding Boxείναι ένα τριδιάστατο ορθογώνιο παραλληλεπίπεδο που έχει θέση, διαστάσεις και προσανατολισμό και περιβάλλει κάθε αντικείμενο ξεχωριστά, με το μέγεθός του να είναι το ελάχιστο δυνατό ώστε να περικλείει το εν λόγω αντικείμενο. Πιο συγκεκριμένα, οι διαστάσεις του για κάθε άξονα υπολογίζονται από το ζευγάρι συντεταγμένων με την μέγιστη και ελάχιστη τιμή (ακρότατα) που εμφανίζονται στο item που περικλείεται. Έτσι, για τον άξονα x οι διαστάσεις είναι (x_{min}, x_{max}) του αντικειμένου, για τον άξονα y τα (y_{min}, y_{max}) και για τον άξονα z είναι τα (z_{min}, z_{max}), έχοντας την μορφή που φαίνεται στην παρακάτω εικόνα:



Εικόνα 4.5: Διαστάσεις OBB

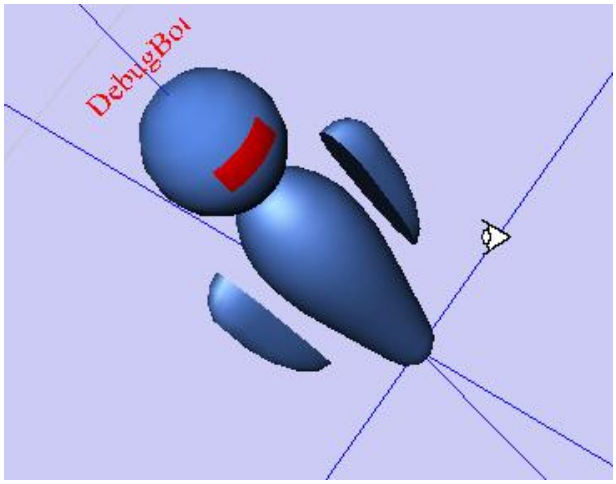


Εικόνα 4.6: Κώνος με το bounding box του

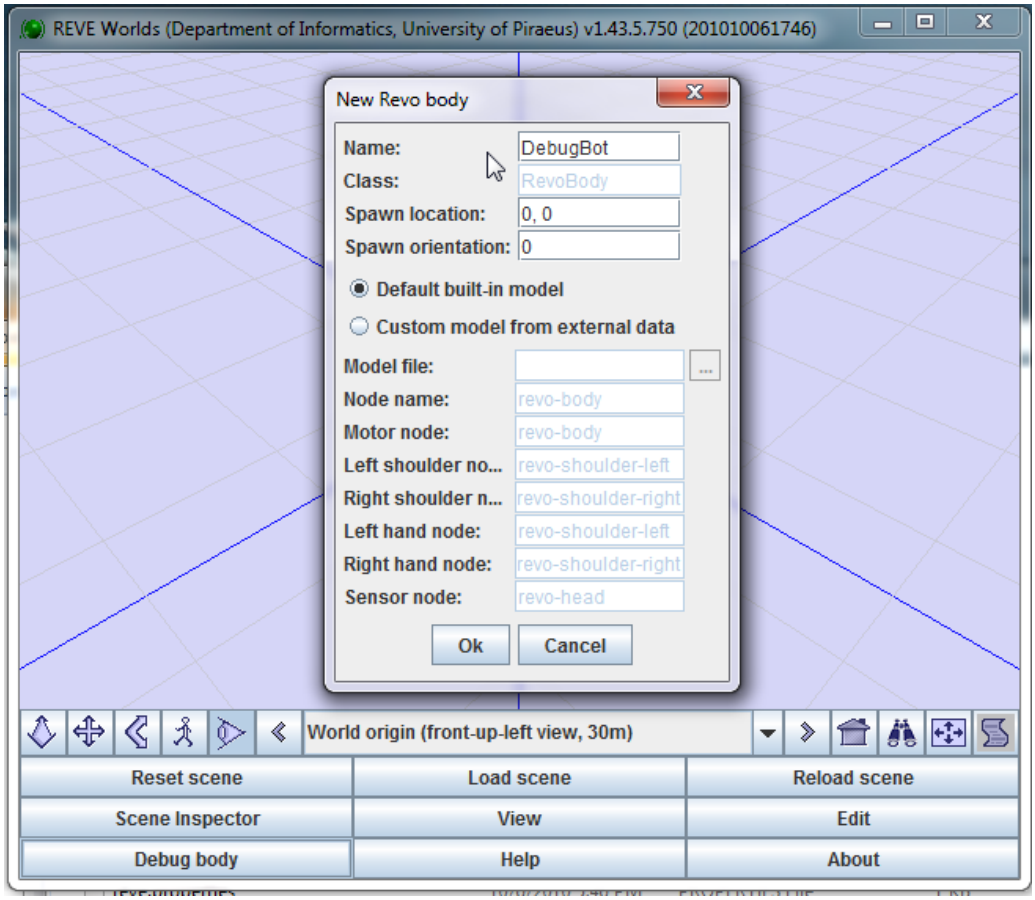
Τέλος, η πλατφόρμα REVE Worlds προσφέρει τη δυνατότητα εισαγωγής πρακτόρων στη σκηνή, οι οποίοι είναι ικανοί να αλληλεπιδράσουν με τον εικονικό κόσμο, αντιλαμβανόμενοι τα σημεία πρόσβασης (access points) των αντικειμένων και ενεργοποιώντας την όποια λειτουργία τους. Οι εικονικοί πράκτορες είναι σχεδιασμένοι ειδικά για αυτόν τον σκοπό και προσπαθούν να προσομοιώσουν το ανθρώπινο σώμα, έχοντας μάτια για να αντιλαμβάνονται το χώρο που βρίσκεται στο οπτικό τους πεδίο, χέρια για να αλληλεπιδρούν με τα αντικείμενα και σώμα για να περιπλανιούνται στον κόσμο. Η αποφυγή συγκρούσεων των πρακτόρων με τα αντικείμενα του κόσμου επιτυγχάνεται μέσω της αντίληψης των bounding boxestων αντικειμένων, εξού και η γεωμετρία IndexedFaceSet των αντικειμένων είναι σημαντική.

Η αρχικοποίηση και τοποθέτηση ενός πράκτορα στον κόσμο - με τον ορισμό του ονόματός του, της θέσης και του προσανατολισμού του ως άλλο ένα αντικείμενο στον κόσμο- μπορεί να γίνει είτε χειροκίνητα μέσα από το εργαλείο Debug body της πλατφόρμας, όπου, by default, ο πράκτορας εμφανίζεται στην αρχή τον αξόνων, είτε προγραμματιστικά, με κλάσεις java που υλοποιούν το πακέτο reve.sara.agent.* όπου μπορεί να προστεθεί

και ευφυΐα στον πράκτορα για μια ροή ενεργειών που οφείλει να εκτελέσει. Αντίστοιχα, η περιπλάνηση του πράκτορα και η διάδραση με τον κόσμο μπορεί να γίνει είτε χειροκίνητα από τον χρήστη, μέσω μιας γραμμής εντολών CLI για τον έλεγχο της κίνησης του πράκτορα, είτε προγραμματιστικά με κλάσεις java.



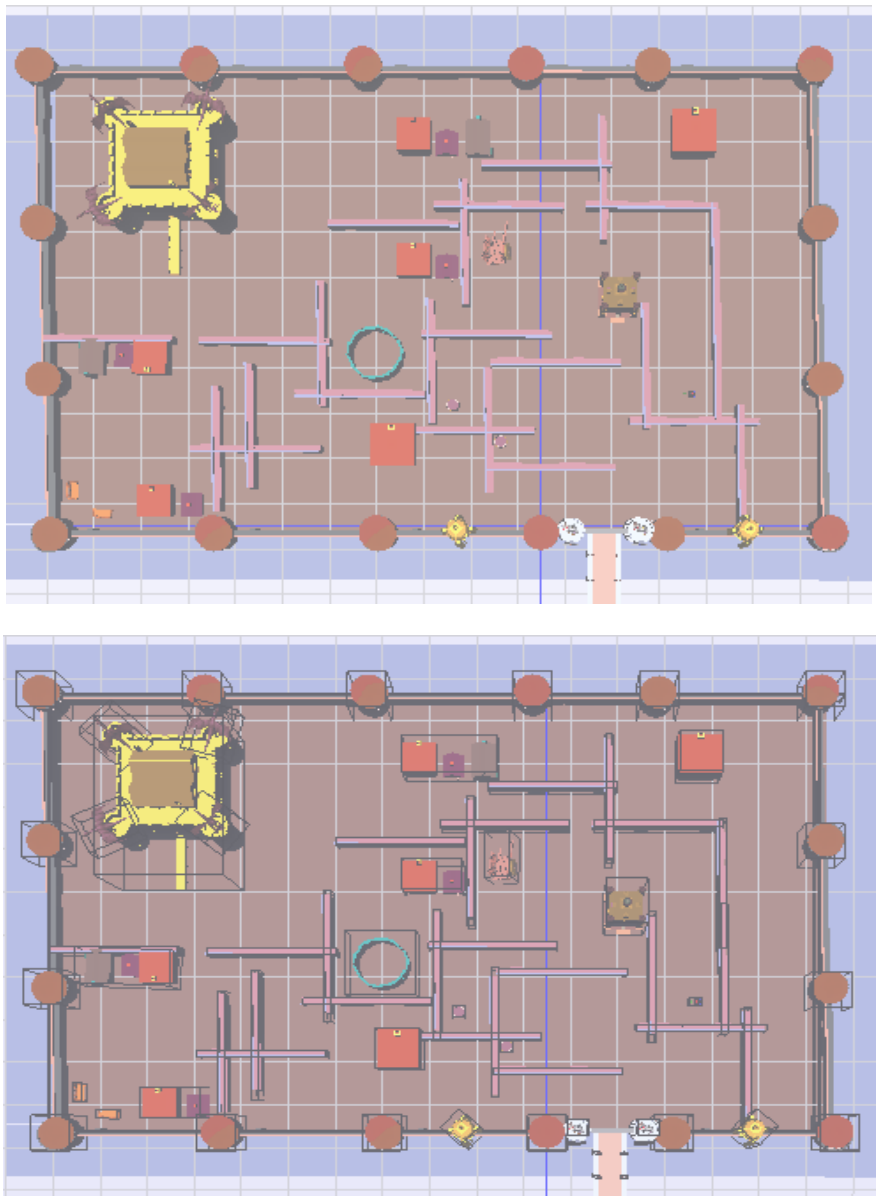
Εικόνα 4.7: Ο πράκτοραςτης REVE



Εικόνα 4.8: Το εργαλείο Debug body

Στην ουσία, η REVE περιλαμβάνει εργαλεία για πλήρη έλεγχο ενός πράκτορα, πέρα από τον ορισμό της συμπεριφοράς του, που μπορεί να γίνει προγραμματιστικά σε java. Ωστόσο, το κεφάλαιο των ευφυών εικονικών πρακτόρων της REVE δεν θα μας απασχολήσει περαιτέρω στην εν λόγω διατριβή. Οφείλουμε, όμως, να επισημάνουμε πως οι πράκτορες αντιλαμβάνονται τα bounding boxes των αντικειμένων που είναι ορατά, που δηλώνονται δηλαδή ως real αντικείμενα στη verl, και όχι τα ίδια τα αντικείμενα, γεγονός που θα μας απασχολήσει για τη σωστή σύνθεση του κόσμου στην εφαρμογή, ώστε να αποφεύγονται επικαλύψεις που μπορούν να οδηγήσουν σε σφάλματα. Για την σωστή δημιουργία των bounding boxes πρέπει να αποφεύγονται χρήσεις προκαθορισμένων σχημάτων στα αντικείμενα, όπως η σφαίρα, ο κώνος κτλ και να γίνεται μετατροπή της γεωμετρίας τους σε πολύγωνα (indexedFaceSet ή indexedLineSet).

Τέλος, παρουσιάζουμε ένα παράδειγμα εικονικού κόσμου στην REVE χωρίς και με τα περικλείοντα bounding boxes των αντικειμένων.



Εικόνα 4.9: Εικονικός κόσμος (α) με BB, (β) χωρίς BB

ΚΕΦΑΛΑΙΟ 5 – ΣΧΕΔΙΑΣΗ ΕΦΑΡΜΟΓΗΣ

Σε αυτό το κεφάλαιο μελετάμε τις απαιτήσεις που θέλουμε να πληρεί η εφαρμογή Reve World Designer για να είναι εύχρηστη και αξιοποιήσιμη από τους χρήστες, υπενθυμίζοντας πως στόχος της εφαρμογής είναι να μπορεί να συνθέσει, να αναπαραστήσει και να επεξεργαστεί έναν εικονικό κόσμο σε γλώσσα verl. Στη συνέχεια εξετάζουμε τις τεχνολογίες που χρησιμοποιήσαμε για την υλοποίηση της εφαρμογής.

5.1 Ανάλυση Απαιτήσεων

Πρώτα και κύρια, η εφαρμογή πρέπει να είναι ανεξάρτητη από το λειτουργικό σύστημα, δηλαδή να παίζει σε διαφορετικά λειτουργικά περιβάλλοντα και ταυτόχρονα να μην απαιτεί πολλούς υπολογιστικούς πόρους από άποψη κατανάλωσης μνήμης και επεξεργαστικής ισχύος, ώστε να είναι εύκολα προσβάσιμη στους χρήστες από έναν απλό προσωπικό υπολογιστή. Επίσης, θέλουμε ιδανικά η εφαρμογή μας να είναι scalable, δηλαδή να μπορεί να υποστηρίζει πολλαπλούς χρήστες ταυτόχρονα, δίνοντας τη δυνατότητα στο πλήθος των χρηστών να αυξάνεται. Αυτό συνεπάγεται την ταυτόχρονη δημιουργία ενός εικονικού κόσμου από πολλούς χρήστες. Ωστόσο,η συγκεκριμένη λειτουργία αποτελεί δευτερεύων στόχο στην παρούσα διπλωματική εργασία και δεν θα επεκταθούμε σε αυτήν.

Είναι σημαντικό να διευκρινίσουμε εδώ πως όλες οι επιθυμητές ενέργειες που στοχεύουμε να πραγματοποιεί η εφαρμογή, θέλουμε να γίνονται μέσα από ένα περιβάλλον εύχρηστο, διαδραστικό, κατανοητό στον χρήστη και φιλικό προς αυτόν, επόμενως η εφαρμογή πρέπει να διαθέτει ένα γραφικό περιβάλλον (Graphic User Interface) το οποίο θα χειρίζεται ο χρήστης για τις λειτουργίες που θέλει να εκτελέσει. Συνεπώς, όλες οι ενέργειες που περιγράφονται παρακάτω θα πρέπει να ενσωματώνονται σε αυτό.

Το GUI θα είναι ένα κεντρικό παράθυρο που θα αποτελείται από διάφορα επιμέρους παράθυρα, για να επιτυγχάνεται μεγαλύτερη αυτονομία και λειτουργικότητα μεταξύ των διαφορετικών δυνατοτήτων που θα περιλαμβάνονται σε αυτά. Ένα παράθυρο πρέπει να λειτουργεί ως μέσο αναζήτησης κι επιλογής του αντικειμένου που θέλουμε να προσθέσουμε στον κόσμο. Ένα δεύτερο παράθυρο πρέπει να λειτουργεί ως η οθόνη όπου αναπαρίσταται και συντίθεται ο εικονικός κόσμος. Σε αυτό θα εμφανίζεται ένα τριδιάστατο πλέγμα αξόνων (x,y,z) στο οποίο θα τοποθετούνται τα αντικείμενα που θέλουμε να αποτελέσουν τον κόσμο. Το πλέγμα συνίσταται να είναι παρόμοιο, όσον αφορά τις αποστάσεις, με αυτό που υπάρχει στο REVE Worlds, ώστε η μετάβαση από τη μια πλατφόρμα στην άλλη να γίνεται ομαλά, χωρίς επιπτώσεις στον κόσμο και ο χρήστης να έχει εξαρχής υπόψη του την τελική μορφή αναπαράστασης του κόσμου.

Δεδομένου ότι θέλουμε η εφαρμογή να υλοποιεί τις ενέργειες που ως τώρα επιτυγχάνονται μέσω συγγραφής κώδικα VERL, πρέπει γενικά, να επιτρέπει την αναπαράσταση ενός εικονικού κόσμου που δομείται σύμφωνα με την αναπαράσταση REVE και να μπορεί να συνθέσει έναν εικονικό κόσμο από τριδιάστατα αντικείμενα που η δομή τους ιεραρχείται σε γράφημα σκηνής. Ως εκ τούτου, η εφαρμογή πρέπει από τη μια, να μπορεί να εισάγει και να διαβάζει αρχεία τύπου X3D και VRML, ώστε να χρησιμοποιεί τέτοια αντικείμενα για τη σύνθεση του και, από την άλλη, να παράγει/εξάγει VERL αρχεία που αναπαριστούν έναν εικονικό κόσμο και μπορούν να χρησιμοποιηθούν στην πλατφόρμα REVE Worlds. Για αυτό τον σκοπό, είναι απαραίτητο η εφαρμογή να διαβάζει και να επεξεργάζεται τόσο γραφήματα σκηνής όσο και XML κώδικα, πάνω στον οποίο έχει βασιστεί η VERL.

Πέρα από το άνοιγμα των αρχείων, πρέπει η εφαρμογή να δίνει στο χρήστη τη δυνατότητα αναζήτησης φακέλων στον υπολογιστή του, που μπορεί να εμπεριέχουν αντικείμενα για τον κόσμο, εμφανίζοντάς του, μέσω μιας δενδροειδής δομής, τους διαθέσιμους φακέλους τους οποίους μπορεί να αναπτύξει για να εμφανιστούν τα ονόματα των αντικειμένων που μπορεί να χρησιμοποιήσει. Επίσης, πατώντας το όνομα ενός αντικειμένου ο χρήστης θα πρέπει να μπορεί να βλέπει σε κάποιο παράθυρο μια μικρογραφία της εικόνας του, ώστε να αποφασίσει αν το χρειάζεται στην προκειμένη περίπτωση για τον κόσμο του. Συνεπώς, αν ο χρήστης θέλει να προσθέσει ένα αντικείμενο στον κόσμο, θα πρέπει να μπορεί να το κάνει είτε, όπως αναφέρθηκε, ανοίγοντας το αρχείο ενός αντικειμένου και εισάγοντάς το στον κόσμο σε μια προκαθορισμένη θέση, όπως η αρχή των αξόνων, είτε επιλέγοντάς το αντικείμενο από το συγκεκριμένο παράθυρο και σέρνοντάς το στο παράθυρο του κόσμου στη θέση που τον ενδιαφέρει. Αντίστοιχα, μέσα από το ίδιο παράθυρο μπορεί να δίνεται

η δυνατότητα στο χρήστη να δημιουργεί αντιγράφο ενός αντικειμένου από έναν φάκελο σε έναν άλλο ή αποκοπή ενός αντικειμένου από ένα φάκελο σε έναν άλλον, προσφέροντας καλύτερη διαχείριση των projects του χρήστη.

Μια σημαντική λειτουργία, μετά την εισαγωγή ενός αντικειμένου στον κόσμο, είναι η επεξεργασία του. Με αυτό εννοούμε τις δυνατότητες που προσφέρει το virtual model της REVE, δηλαδή την περιστροφή, την μετατόπιση ή την αλλαγή της κλίμακας ενός αντικειμένου που πρέπει να επιτυγχάνει και η εφαρμογή μας. Στόχος είναι αυτές οι ενέργειες να δύνανται με δυο τρόπους. Ο πρώτος θα επιτρέπει την εκτέλεση αυτών των ενεργειών μέσα από το παράθυρο απεικόνισης του κόσμου, με τη βοήθεια του ποντικιού (πχ σύρσιμο ενός αντικειμένου παράλληλα με κάποιον άξονα για μετατόπιση) ή πλήκτρων από το πληκτρολόγιο, εφόσον έχει επιλεχθεί το αντικείμενο που μας ενδιαφέρει. Ο δεύτερος αφορά την ύπαρξη ενός ξεχωριστού παραθύρου που θα επεξεργάζεται συγκεκριμένα το virtual mode του αντικειμένου, όπου θα εμφανίζονται οι παράμετροι κάθε <transform> ενέργειας, συνεπώς ο χρήστης αλλάζοντας τις τιμές αυτών των παραμέτρων θα μπορεί να μετατοπίσει, να περιστρέψει ή να μεγενθύνει/μικρύνει το επιλεγμένο αντικείμενο αντίστοιχα.

Σε αυτό το σημείο οφείλουμε να τονίσουμε πως η επιλογή ενός αντικειμένου πρέπει να γίνεται εμφανής στον χρήστη. Αυτό σημαίνει πως κατά την επιλογή ενός αντικειμένου, πχ πατώντας ένα αντικείμενο με το ποντίκι, αυτό θα αποκτά διακριτά χαρακτηριστικά, όπως με το να γίνεται πιο έντονο το περίγραμμά του, ώστε ο χρήστης να είναι σίγουρος πως έχει επιλέξει το σωστό αντικείμενο. Επίσης, μπορεί να υπάρχει μια μικρή περιοχή στο GUI όπου θα εμφανίζεται σε μικρογραφία το επιλεγμένο αντικείμενο και θα περιστρέφεται, ώστε ο χρήστης να έχει μια περίοπτη άποψη για αυτό.

Επιπλέον, μπορεί να δίνεται η δυνατότητα στον χρήστη να χρησιμοποιήσει πολλές φορές το ίδιο αντικείμενο, δημιουργώντας αντίγραφά του στον κόσμο. Για παράδειγμα, επιλέγοντας ένα αντικείμενο και πατώντας δεξί κλικ με το ποντίκι μπορεί να εμφανίζεται μια επιλογή 'Copy' που θα φτιάχνει ένα πανομοιότυπο <item> στον κόσμο, με το ίδιο όνομα και κάποιον αριθμό που θα αντιστοιχεί στον αριθμό αντιγράφου. Αυτό θα διευκολύνει τον χρήστη από τη χειροκίνητη δημιουργία ίδιων αρχείων για να χρησιμοποιήσει πολλαπλές φορές ένα αντικείμενο στον κόσμο. Από την άλλη, πέρα από την εισαγωγή και την τροποποίηση ενός αντικειμένου, θα υπάρχει και η δυνατότητα μετονομασίας ενός αντικειμένου και διαγραφής του, με την έννοια της αφαίρεσής του ως <item> από τον συγκεκριμένο εικονικό κόσμο. Αυτό, αντίστοιχα μπορεί να γίνεται με ενέργεια 'Delete' που εμφανίζεται με δεξί κλικ κατά την επιλογή ενός αντικειμένου στην σκηνή.

Επιπρόσθετα, στον χρήστη πρέπει να δίνεται η δυνατότητα να αναιρέσει μια ενέργεια ή να επανεκτελέσει μια ενέργεια (undo-redo), συνεπώς πρέπει, τεχνικά, να αποθηκεύονται σε μια στοίβα οι τελευταίες ενέργειες που πραγματοποιεί, ώστε να ανακαλούνται. Ακόμα, θέλουμε οι πιο συχνές λειτουργίες που θα κάνει ο χρήστης (πχ. άνοιγμα αρχείου, αποθήκευση κόσμου ή αποθήκευση ως, αναίρεση, επανεκτέλεση, αντιγραφή αντικειμένου, επικόλληση αντικειμένου, διαγραφή αντικειμένου) να είναι συγκεντρωμένες και έυκολα προσβάσιμες από τον χρήστη σε ένα συγκεκριμένο χώρο της εφαρμογής, άρα, να εμπεριέχονται κατά προτίμηση σε ένα toolbar.

Επιστρέφοντας στις δυνατότητες της αναπαράστασης REVE, θέλουμε μέσα από την εφαρμογή να επιτρέπεται ο χειρισμός κάθε άποψης (aspect) του αντικειμένου, διατηρώντας την αυτονομία μεταξύ των επιπέδων αυτών. Συνεπώς, πέρα από ένα παράθυρο για το physical aspect, όπως περιγράφηκε παραπάνω, θέλουμε να έχουμε άλλα δυο παράθυρα για το semantic aspectκαι το access aspect αντίστοιχα. Σε αυτά τα παράθυρα, κατά την επιλογή ενός αντικειμένου θα εμφανίζονται οι παράμετροι που αφορούν το κάθε μοντέλο, δίνοντας τη δυνατότητα στον χρήστη να μεταβάλλει τις τιμές του. Για παράδειγμα, στο semantic model μπορεί ο χρήστης να προσθαφαιρεί symbols σε ένα αντικείμενο, ενώ στο access model μπορεί να προσθαφαιρεί access points, να τα τροποποιεί και να προσθαφαιρεί functions σε αυτά. Μάλιστα, όσον αφορά τις functions, θα μπορεί να επιλέξει μέσα από την build-in λειτουργικότητα (classes) που προσφέρει η αναπαράσταση Reve.

Άλλη μια δυνατότητα που θέλουμε να καλύπτει η εφαρμογή είναι να προσφέρει στο χρήστη διαφορετικές γωνίες παρατήρησης του κόσμου, δηλαδή να μπορεί να μεταβάλλει την κάμερα παρατήρησης, βλέποντας είτε κάτοψη (άξονες xy ή yz), είτε πρόσοψη (xz) είτε προοπτική (xyz). Επίσης, θέλουμε ο χρήστης να έχει τη δυνατότητα να προσαρμόζει το περιβάλλον εργασίας του όπως τον εξυπηρετεί, δηλαδή να μπορεί να μετακινεί

και να τοποθετεί αλλού τα διαφορετικά παράθυρα (tabs) που υπάρχουν ταυτόχρονα στην εφαρμογή ή να τα κρύβει και να τα εμφανίζει όπως επιθυμεί.

Θα πρέπει όλες οι διαθέσιμες βασικές ενέργειες να είναι προσβάσιμες από το χρήστη και από ένα κεντρικό 'Μενού' όπου θα μπορεί να τις αναζητήσει, καθώς να δίνεται και η επιλογή της 'Βοήθειας', όπου θα εμφανίζονται οδηγίες για τον τρόπο λειτουργίας της εφαρμογής. Τέλος, θέλουμε ιδανικά να μπορεί ο κόσμος να γίνεται rendered και να φορτώνεται απευθείας στην πλατφόρμα ReveWorlds για περαιτέρω επεξεργασία ή σύνδεση με τους εικονικούς πράκτορες.

5.2 Τεχνολογίες Υλοποίησης

Σε αυτό το κεφάλαιο παρουσιάζουμε τις τεχνικές επιλογές που κάναμε για να ικανοποιήσουμε στον μέγιστο βαθμό την ανάλυση απαιτήσεων.

Η εφαρμογή αναπτύχθηκε στη γλώσσα προγραμματισμού Java και πιο συγκεκριμένα χρησιμοποιήθηκε το περιβάλλον Java SE (Standard Edition) Environment 7 της Oracle και το Java Development Kit (JDK) 7. Η συγκεκριμένη έκδοση προτιμήθηκε γιατί ήταν η πιο πρόσφατη και ανανεωμένη από άποψη βιβλιοθηκών και λειτουργικότητας κατά την εκκίνηση υλοποίησης της μεταπτυχιακής μας διατριβής. Αναφορικά με την επιλογή της γλώσσας java έναντι άλλης, λάβαμε υπόψην αρκετούς παράγοντες, όπως εξηγούνται παρακάτω.

Πρώτον, η java είναι μια έγκυρη και αξιόπιστη γλώσσα προγραμματισμού που υποστηρίζεται από διαφορετικές πλατφόρμες και είναι ανεξάρτητη από αυτές χάρη στην Java Virtual Machine. Συνεπώς, η java ικανοποιεί την απαίτηση η εφαρμογή μας να υποστηρίζεται από διαφορετικά λειτουργικά συστήματα και να είναι εύκολα προσβάσιμη στους χρήστες από τον προσωπικό τους υπολογιστή.

Δεύτερον, η java είναι μια αντικειμενοστραφής γλώσσα (object-oriented) που επιτρέπει την επαναχρησιμοποίηση κώδικα και την αναπαράσταση αυτόνομων οντοτήτων με διακριτές λειτουργίες, μέσω της δημιουργίας αντικειμένων (objects), γεγονός που μας εξυπηρετεί για την αναπαράσταση των στοιχείων που αποτελούν τον κόσμο μας, όπως το item και τα διάφορα μοντέλα λειτουργίας του.

Τρίτον, η java έχει έτοιμες βιβλιοθήκες αναπαράστασης γραφικών GUI, επομένως είναι μια γλώσσα κατάλληλη για εφαρμογές που χρειάζονται γραφική διεπαφή για αλληλεπίδραση με τον χρήστη. Πιο συγκεκριμένα, προσφέρει δυο πακέτα εργαλείων (toolkit) αναπαράστασης γραφικών, το Astract WindowToolkit (AWT) και την Swing, που είναι μέρος των Java Foundation Classes (JFC).

Η AWT είναι μια φορητή βιβλιοθήκη που συνδέει μια εφαρμογή με το GUI της μηχανής και προσφέρει μια υψηλού επιπέδου αφαίρεση αναπαράστασης γραφικών, καθώς κρύβει τις λεπτομέρειες του UI πάνω στο οποίο θα τρέξει η εφαρμογή. Παρέχει μια ποικιλία έτοιμων υλοποιήσεων όσον αφορά συστατικά στοιχεία διεπαφής χρήστη και ένα ικανό μοντέλο χειρισμού γεγονότων (event handling model). Επίσης, προσφέρει εργαλεία γραφικών και απεικόνισης, όπως σχήματα, χρώματα, κλάσεις γραμματοσειρών, καθώς και διαφορετικούς τρόπους ταξινόμησης παραθύρων, ανεξαρτήτως ανάλυσης και μεγέθους οθόνης. Επιπλέον, υποστηρίζει την μεταφορά δεδομένων, ώστε να είναι δυνατή η αποκοπή και η επικόλληση ενός αντικειμένου μεταξύ διαφορετικών στοιχείων της εφαρμογής. Η AWT χρησιμοποιείται κυρίως για stand-alone εφαρμογές και applets.

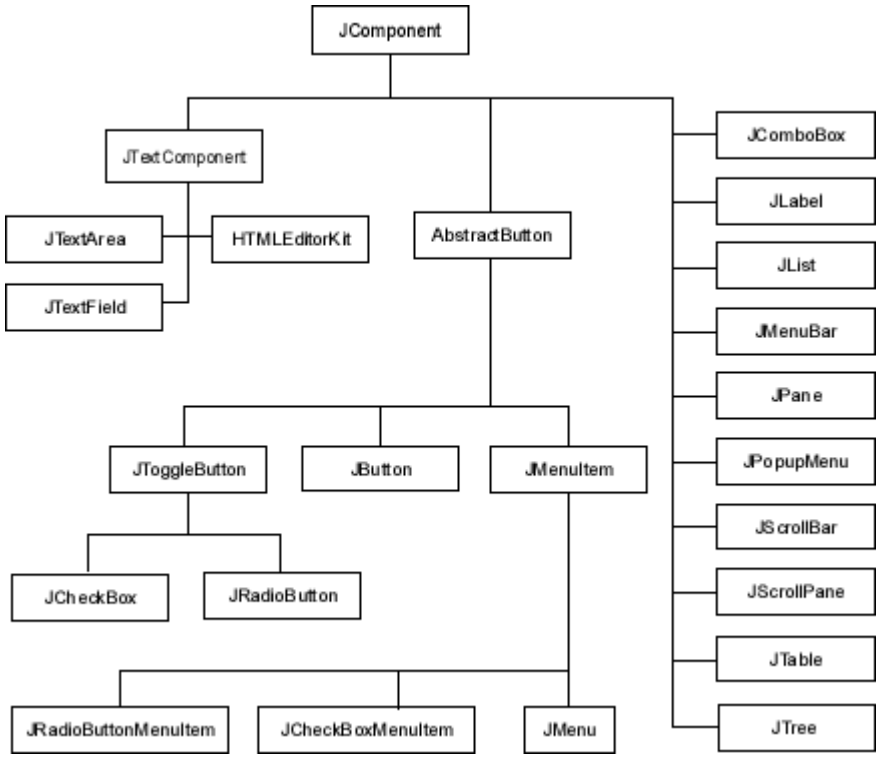
Η βιβλιοθήκη που χρησιμοποιήσαμε εμείς για την εφαρμογή μας είναι η Swing (javax.swing.*), καθώς έχει όλα τα χαρακτηριστικά της AWT προσφέροντας, ωστόσο, ακόμα περισσότερα ή βελτιστοποιημένα στοιχεία αναπαράστασης, όπως είναι η δενδρική δομή, οι πίνακες, οι λίστες, τα κουμπιά, τα πλαίσια και τα παράθυρα με εμφωλευμένες καρτέλες, ενώ υποστηρίζει και πληθώρα λειτουργικότητας επί αυτών, όπως η δυνατότητα να ταξινομήσεις τα επιμέρους στοιχεία ή να τα σύρεις στην οθόνη (drag and drop). Ακόμα, έχει έτοιμες κλάσεις που υποστηρίζουν την λειτουργικότητα για ενέργειες undo-redo. Επιπλέον, η swing υποστηρίζει διαφορετική αισθητική αναπαράσταση (look and feel) της εφαρμογήςσε αντίθεση με την AWT, πράγμα που την καθιστά προσαρμόσιμη στα γούστα του χρήστη. Για παράδειγμα, μια εφαρμογή μπορεί να έχει την default αισθητική μιας Java εφαρμογής ή την αισθητική των windows ή των mac.Ακόμα, η swing υποστηρίζει επιπλέον δυνατότητες, όπως είναι η χρήση εικονιδίων και pop-up menus που δεν τα παρέχει η AWT. Τέλος, σε αντίθεση με την AWT που τα στοιχεία της εξαρτώνται από τα αντίστοιχά τους σε κώδικα μηχανής, η Swing βιβλιοθήκη

είναι εξολοκλήρου σχεδιασμένη σε java, γεγονός που την καθιστά πλήρως αυτόνομη από τους περιορισμούς της συγκεκριμένης πλατφόρμας υλοποίησης.

SWING	AWT
Swing is a considered as light weight.	AWT is considered as heavy weight.
swing components are platform independent because they are developed purely in java.	AWT components are platform dependent.
Swing components require javax.swing package.	AWT components require java.awt package
Swing takes less memory.	AWT takes more memory.

Εικόνα 5.1: Βασικές διαφορές swing-awt

Όπως γίνεται αντιληπτό, η πληθώρα στοιχείων διεπαφής που προσφέρει η Swing παράλληλα με την υποστήριξη της λειτουργικότητας των συστατικών της AWT βιβλιοθήκης και την ανεξαρτησία της από το λειτουργικό σύστημα, την καθιστά ιδανική βιβλιοθήκη για την εφαρμογή μας, προσφέροντας τη δυνατότητα επέκτασης και εμπλουτισμού των επιλογών που προσφέρει, προς διευκόλυνση του χειρισμού της εφαρμογής από υποψήφιους χρήστες. Αυτό δεν σημαίνει ότι δεν χρησιμοποιούμε στοιχεία από την AWT – άλλωστε η Swing έχει φτιαχτεί με βάση αυτήν, επομένως κάτι τέτοιο θα ήταν σχεδιαστικά αδύνατο για ορισμένα συστατικά – αλλά σημαίνει πως εκμετταλλευόμαστε τις δυνατότητες της Swing, αποφεύγοντας κλάσεις της AWT όπου δύναται, χωρίς, όμως, να συνδυάζουμε συστατικά στοιχεία από τις δυο βιβλιοθήκες με λάθος τρόπο, πχ τοποθετώντας ένα JButton σε ένα Frame αντί για ένα Jframe που είναι το σχεδιαστικά σωστό.



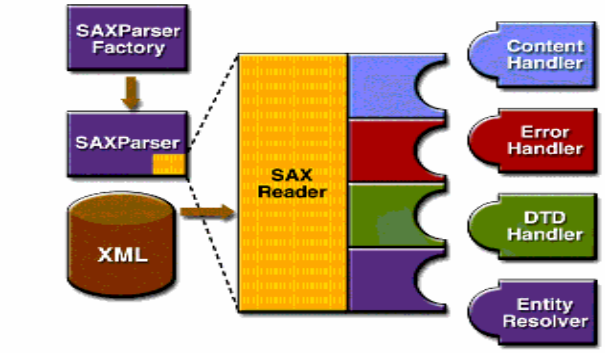
Εικόνα 5.2: Τα στοιχεία της Swing από το JComponent

Πέραν της ευκολίας αναπαράστασης γραφικών που έχουν οι παραπάνω βιβλιοθήκες της java, διαθέτουν και κλάσεις και για τον χειρισμό γεγονότων (event handling) μέσω της διεπαφής, όπως αναφέρθηκε. Η λειτουργικότητα αυτή είναι κυρίαρχη στην εφαρμογή μας, εφόσον πρέπει το πρόγραμμα να αντιλαμβάνεται τις ενέργειες του χρήστη και να αντιδρά κατάλληλα, επιτρέποντας ή όχι μια ενέργεια υπό συνθήκες. Για παράδειγμα, είναι ουσιαστική η δυνατότητα αναγνώρισης εισόδου ενός αντικειμένου στον κόσμο από τις επιλογές που έχει κάνει ο χρήστης, καθώς η σχεδίαση του εικονικού κόσμου πραγματοποιείται καθαρά από τον ίδιο και δεν αποτελεί μια λειτουργία που εκτελείται αυτόματα. Επίσης, η αναγκαιότητα των event handlerskαι των listeners γίνεται πιο εύκολα κατανοητή αν αναλογιστούμε ότι μια επιλογή μέσω της διεπαφής χρήστη, όπως είναι το πάτημα ενός κουμπιού, δεν σημαίνει τίποτα από μόνη της αν δεν υπάρχει κώδικας που να ορίζει συγκεκριμένα την λειτουργία μιας ενέργειας, καθώς και τις προϋποθέσεις υπό τις οποίες αυτή μπορεί να εκτελεστεί.

Μια επιπλέον δυνατότητα που μας προσφέρει η java και είναι απαραίτητη στην εφαρμογή μας είναι το πακέτο κλάσεων για επεξεργασία αρχείων (java.io.*), δεδομένου ότι χρειαζόμαστε αρχεία τόσο για να φορτώσουμε ένα αντικείμενο στον κόσμο όσο και για να δημιουργήσουμε την αναπαράσταση του κόσμου ή να φορτώσουμε έναν έτοιμο κόσμο για περαιτέρω επεξεργασία στην εφαρμογή. Συνδυαστικά με την input/output βιβλιοθήκη, η java μας παρέχει μια επιπλέον βιβλιοθήκη που αναγνωρίζει, διαβάζει, επεξεργάζεται και δημιουργεί αρχεία σε μορφή xml, γνωστή ως JAXP (Java API for XML Processing). Δεδομένου ότι η αναπαράσταση REVE χρησιμοποιεί την xml-based γλώσσα verl για τον εικονικό κόσμο, η εν λόγω βιβλιοθήκη της java κρίνεται απαραίτητο στοιχείο για την αναπαράσταση του εικονικού μας κόσμου.

Η JAXP βιβλιοθήκη αποτελείται από διάφορα πακέτα. Αξιοποιεί τις προδιαγραφές ενός προγράμματος ανάλυσης (parser) XML με το Simple API for XML Parsing (SAX) και το Document Object Model (DOM), επιτρέποντας κατά αυτό τον τρόπο την μεταφορά/μετάφραση των δεδομένων σαν μια ροή γεγονότων ή την αναπαράσταση τους μέσω αντικειμένου. Επιπλέον, υποστηρίζει το πρότυπο XSLT (Extensible Stylesheet Language Transformations), μέσω του οποίου δίνει τον έλεγχο στον προγραμματιστή για την αναπαράσταση των δεδομένων που του επιτρέπει να τα μετατρέψει σε άλλα xml αρχεία ή αρχεία άλλου μορφότυπου, όπως η html.

Επίσης, η JAXP επιτρέπει στον προγραμματιστή να δουλέψει με DTDs (Document Type Definition) για να αποφύγει συγκρούσεις μεταξύ ονομάτων, δεδομένου ότι το DTD είναι ένα έγγραφο που περιγράφει την εσωτερική δομή ενός αρχείου xml, ορίζοντας τα επιτρεπτα στοιχεία (elements) και χαρακτηριστικά (attributes) του. Αυτό για την δική μας περίπτωση είναι ιδιαίτερα σημαντικό καθώς και η verl χρησιμοποιεί το δικό της DTD, το οποίο μπορούμε να αξιοποιήσουμε.Επίσης, σχεδιασμένη για να είναι ευέλικτη, η JAXP επιτρέπει τη χρήση οποιουδήποτε προγράμματος ανάλυσης συμβατού με XML μέσα από την εφαρμογή. Αυτό επιτυγχάνεται με αυτό που ονομάζεται ένα «pluggability» στρώμα , που επιτρέπει τη σύνδεση του SAX ή του DOM API με την εφαρμογή. Το στρώμα αυτό επιτρέπει επίσης τη σύνδεση ενός επεξεργαστή XSL, επιτρέποντάς τον έλεγχο πάνω στον τρόπο εμφάνισης των δεδομένων XML.



Εικόνα 5.3: Δομή JAXP

Τέλος, η java παρέχει ειδικές βιβλιοθήκες για την αναπαράσταση αντικειμένων VRML και X3D. Πιο

συγκεκριμένα, στην εφαρμογή μας χρησιμοποιούμε την Xj3D, που πρόκειται για ένα έργο του Web 3D Consortium που εστιάζει στη δημιουργία ενός toolkit για την αναπαράσταση περιεχομένου σε VRMLκαι X3D και είναι εξ'ολοκλήρου υλοποιημένο σε java. Εξυπηρετεί ένα διπλό σκοπό, ο πρώτος αφορά την ύπαρξη μιας βάσης κώδικα για τον πειραματισμό σε νέες περιοχές των προδιαγραφών X3D και ο δεύτερος αφορά σε βιβλιοθήκη, που ενθαρρύνει τους προγραμματιστές να τη χρησιμοποιούν στις εφαρμογές τους για την υποστήριξη της τεχνολογίας X3D. Χρησιμοποιείται, λοιπόν, για να εισάγει κανείς vrml περιεχόμενο σε μια εφαρμογή ή για να δημιουργήσει ένα πλήρες πρόγραμμα περιήγησης (browser), που εμφανίζει τα 3d πρότυπα μοντελοποίησης για τους μορφότυπους VRML97 και X3D. Ένα κλασικό πρόγραμμα περιήγησης – εφεξής xj3d browser – έχει την μορφή που παρουσιάζεται στην Εικόνα 5.4 και επιτρέπει τις παρακάτω λειτουργίες:

Open: Εμφανίζει ξεχωριστό παράθυρο που επιτρέπει στον χρήστη να περιηγηθεί στο σύστημα φακέλων του υπολογιστή του και να επιλέξει ένα VRML97 ή X3D αρχείο, το οποίο εμφανίζεται την κεντρική οθόνη. Η γραμμή Location εμφανίζει την ακριβή τοποθεσία του αρχείου στον υπολογιστή και κρατάει ιστορικότητα των αντικειμένων που έχουν φορτωθεί. Επομένως, επιλέγοντας ένα άλλο path στο Location από τα διαθέσιμα και πατώντας Go, φορτώνεται το αντικείμενο που ορίζεται στο path.

Reload: Επαναφορτώνει το αρχείο που εμφανίζεται στην κεντρική οθόνη ή στο Location αντίστοιχα.


Fly: Επιτρέπει στο χρήστη να περιηγηθεί στο χώρο σαν να ίπταται, σαν να αιωρείται πάνω από το αντικείμενο που εξετάζει.


Pan: Επιτρέπει στον χρήστη να περιεργαστεί το αντικείμενο κατα μήκος των αξόνων x και y.


Tilt: Επιτρέπει στο χρήστη να προσκλίνει στο αντικείμενο.


Walk: Επιτρέπει στο χρήστη να περιηγηθεί ελεύθερα στο χώρο, πλησιάζοντας ένα αντικείμενο ή απομακρυνόμενος από αυτό.


Track: Επιτρέπει στο χρήστη να περιεργαστεί το αντικείμενο (ή τον κόσμο) από όλες τις πλευρές του, περιστρέφοντας το με σημείο αναφοράς τον άξονα κατά μηκος της πλευράς που επιλέγει, αποκτώντας έτσι μια συνολική εικόνα για αυτό.


Examine: Επιτρέπει στο χρήστη να επεξεργαστεί σφαιρικά ένα αντικείμενο, περιστρέφοντας το προς κάποιον άξονα που επιλέγει με σημείο αναφοράς το κέντρο του αντικειμένου.


Previous Viewpoint: Μεταφέρει την οπτική γωνία του αντικείμενου στην επόμενη που ορίζεται, αν ορίζεται.

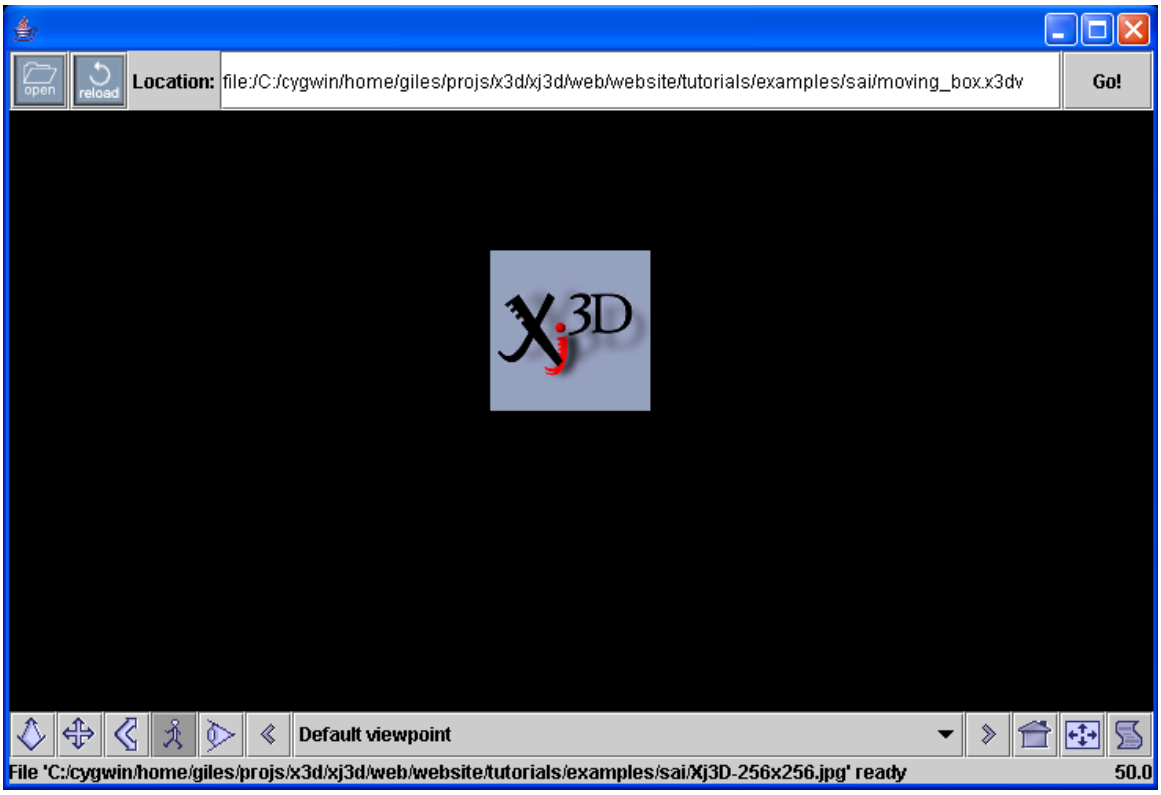

Next Viewpoint: Μεταφέρει την οπτική γωνία του αντικείμενου στην προηγούμενη που ορίζεται, αν ορίζεται.


Return to current Viewpoint: Μεταφέρει την οπτική γωνία του αντικειμένου στην προεπιλεγμένη που έχει το αντικείμενο κατά την εισαγωγή του στην σκηνή.


Look At: Επιτρέπει στο χρήστη να πατήσει σε ένα σημείο του χώρου και να μεταφερθεί σε αυτό.

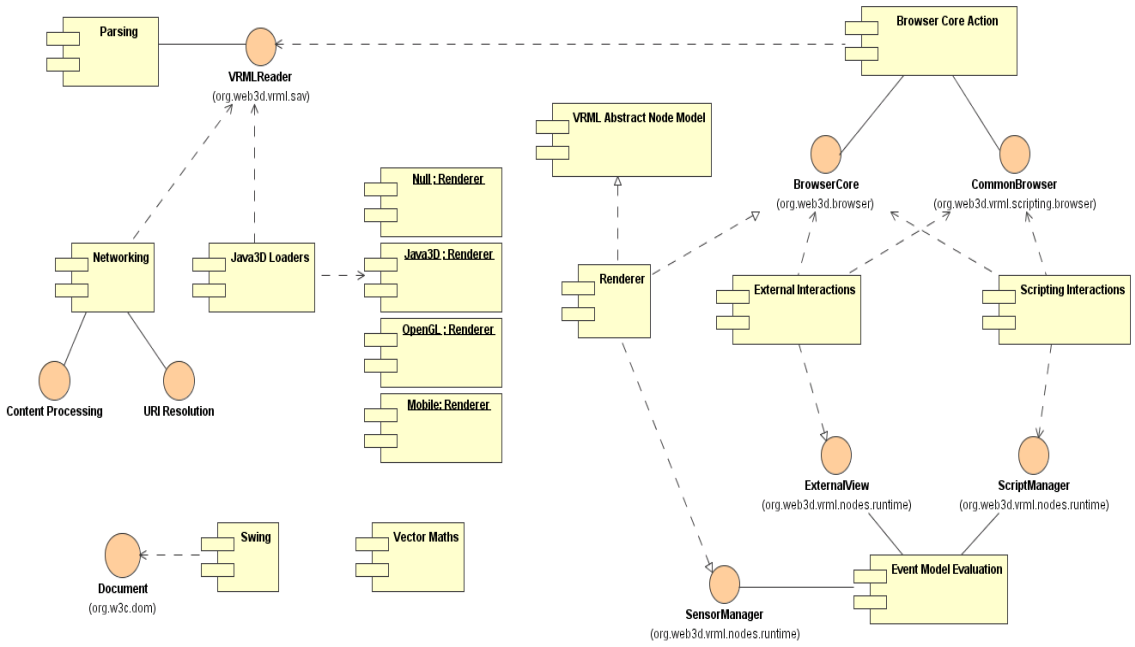
 Fit to World: Μεταφέρει το αντικείμενο στο κέντρο της οθόνης ώστε να καταλαμβάνει όλο τον επιτρεπτό χώρο που υπάρχει στη σκηνή. Κατά αυτόν τον τρόπο, δίνει την αίσθηση του ολικού zoom-in στον χρήστη.

 Show Browser Console:Ανοίγει έναπαράθυρο με πληροφορίες για τον browser. Κατά κύριο λόγο, εμφανίζει προειδοποιήσεις και πληροφορίες για το αντικείμενο που βρίσκεται στη σκηνή.



Εικόνα 5.4: Xj3D Browser

Καθίσταται σαφές ότι η Xj3Dείναι απαραίτητο εργαλείο για να εμφανίσουμε τα αντικείμενα που θέλουμε να επιλέξουμε για τον εικονικό κόσμο, για να τα εισάγουμε στον κόσμο, για να σχεδιάσουμε τον κόσμο και φυσικά για να περιηγηθούμε σε αυτόν. Στην εφαρμογή μας χρησιμοποιήσαμε την πιο πρόσφατη έκδοση της βιβλιοθήκης xj3d, που έχει αριθμό έκδοσης 2.0.



Εικόνα 5.5: Αρχιτεκτονική του Xj3D

Προς διευκόλυνση ανάπτυξης του κώδικα της εφαρμογής χρησιμοποιήσαμε το εργαλείο NetBeans IDE 7.3.1. σε περιβάλλον Windows Ultimate 7. Το NetBeans IDE (Integrated Development Environment) είναι μια open-source, ελεύθερα προσβάσιμη πλατφόρμα ανάπτυξης λογισμικού γραμμένη σε Java, που επιτρέπει στις εφαρμογές να αναπτυχθούν από ένα σύνολο αρθρωτών στοιχείων λογισμικού, γνωστά ως modules. Παρέχει πληθώρα εργαλείων που διευκολύνουν τον προγραμματιστή να συνδέσει τα επιμέρους τμήματα που συνθέτουν μια εφαρμογή και χρησιμοποιείται ευρέως για την ανάπτυξη κώδικα σε java, τόσο για ελεύθερη χρήση όσο και επαγγελματικά.

Επίσης, για την ανάγνωση και επεξεργασία verl αρχείων, που αναπαριστούν τον εικονικό κόσμο για την πλατφόρμα Reve, χρησιμοποιήσαμε την τελευταία έκδοση 6.7.8.2 του εργαλείου Notepad++. Πρόκειται για μια εφαρμογή που θυμίζει το κλασικό σημειωματάριο (Notepad) των Windows, αλλά στην ουσία είναι ένα πρόγραμμα επεξεργασίας πηγαίου κώδικα που υποστηρίζει πολλές γλώσσες, εκ των οποίων και η xml. Πέραν τούτου, για την επαλήθευση της σωστής σύνταξης της verl και της αναπαράστασης του εικονικού κόσμου χρησιμοποιήσαμε και την πλατφόρμα ReveWorlds, όπως αυτή παρουσιάστηκε στο Κεφάλαιο 4.2.

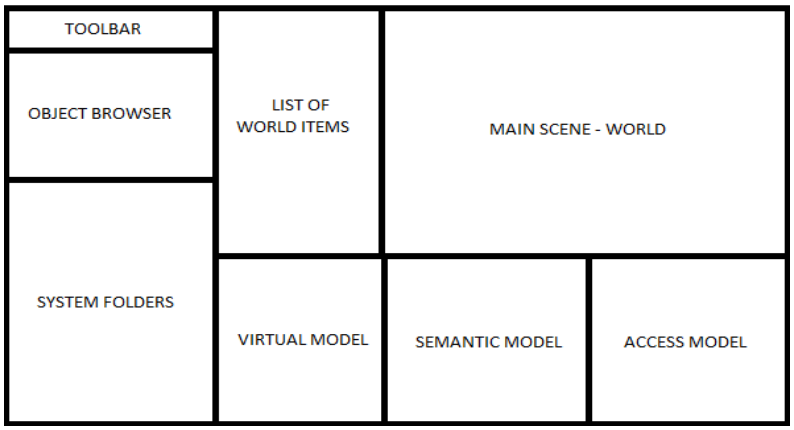
Τέλος, για την πρωταρχική επεξεργασία και την απεικόνιση των 3d αντικειμένων που είτε δημιουργούσαμε είτε τα παίρναμε έτοιμα, ώστε να πληρούν τις προδιαγραφές της αναπαράστασης REVE, χρησιμοποιήσαμε το εργαλείο Vrml Pad 3.0 και το Cortona 3D Viewer. Το VrmlPad είναι εργαλείο ανάπτυξης και επεξεργασίας κώδικα VRML, κατάλληλο για την σχεδίαση τριδιάστατων αντικειμένων. Από την άλλη, το Cortona 3D Viewer (παλαιότερα γνωστό ως Cortona VRML Client) λειτουργεί ως ένα VRML plug-in για τα πιο δημοφιλή προγράμματα περιήγησης στο Διαδίκτυο (Internet Explorer , Mozilla Firefox , Google Chrome , Opera) και εφαρμογών γραφείου (Microsoft PowerPoint , Microsoft Word). Υποστηριζόμενο από τα MS Windows το Cortona 3D Viewer είναι ελεύθερο για προσωπική και ακαδημαϊκή χρήση και δίνει την δυνατότητα φόρτωσης ενός .wrl αρχείου για την απεικόνιση του περιεχομένου του στον browser του χρήστη.

ΚΕΦΑΛΑΙΟ 6 – ΥΛΟΠΟΙΗΣΗ ΕΦΑΡΜΟΓΗΣ ΚΑΙ ΔΥΣΚΟΛΙΕΣ

Σε αυτό το κεφάλαιο παρουσιάζουμε τα επιμέρους τμήματα που συνθέτουν την εφαρμογή, τη λειτουργικότητα που έχουν και πώς αλληλοσυνδέονται μεταξύ τους, καθώς και τις δυσκολίες που αντιμετωπίσαμε κατά την υλοποίησή της.

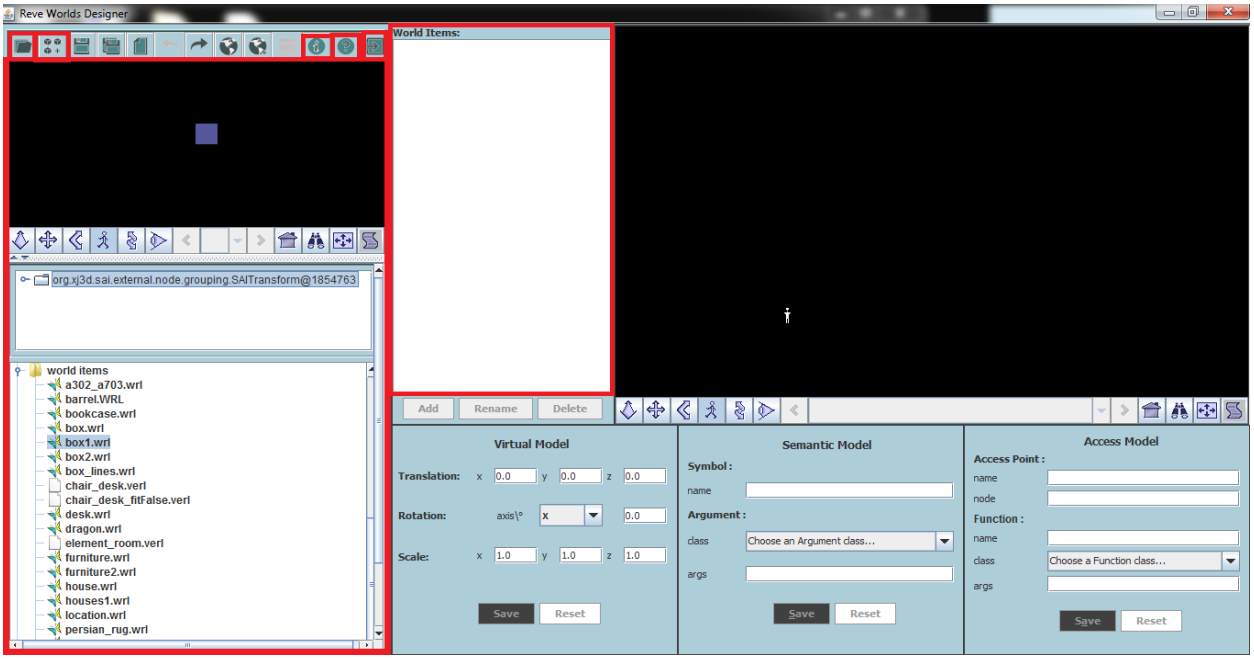
6.1 Δομή Εφαρμογής

Η εφαρμογή συμπτύχθηκε από δυο κομμάτια. Το πρώτο, που περιγράφεται αναλυτικά στην μεταπτυχιακή διατριβή του Γεώργιου Σανιόγλου με τίτλο «*Εφαρμογή Reve World Designer: Επιλογή τριδιάστατου αντικειμένου για τη σύνθεση εικονικών κόσμων*» αφορά την επιλογή αντικειμένου βάσει των προδιαγραφών που θέτει η αναπαράσταση Reveγια τη χρήση ενός αντικειμένου στον εικονικό κόσμο που θέλουμε να δημιουργήσουμε. Το δεύτερο αφορά στη σχεδίαση του εικονικού κόσμου, δηλαδή τη φυσική αναπαράσταση του, μετατρέποντας τα αντικείμενα που λαμβάνει από το πρώτο κομμάτι σε items που συνθέτουν τον εικονικό κόσμο, βάσει των προδιαγραφών της αναπαράστασης REVE.

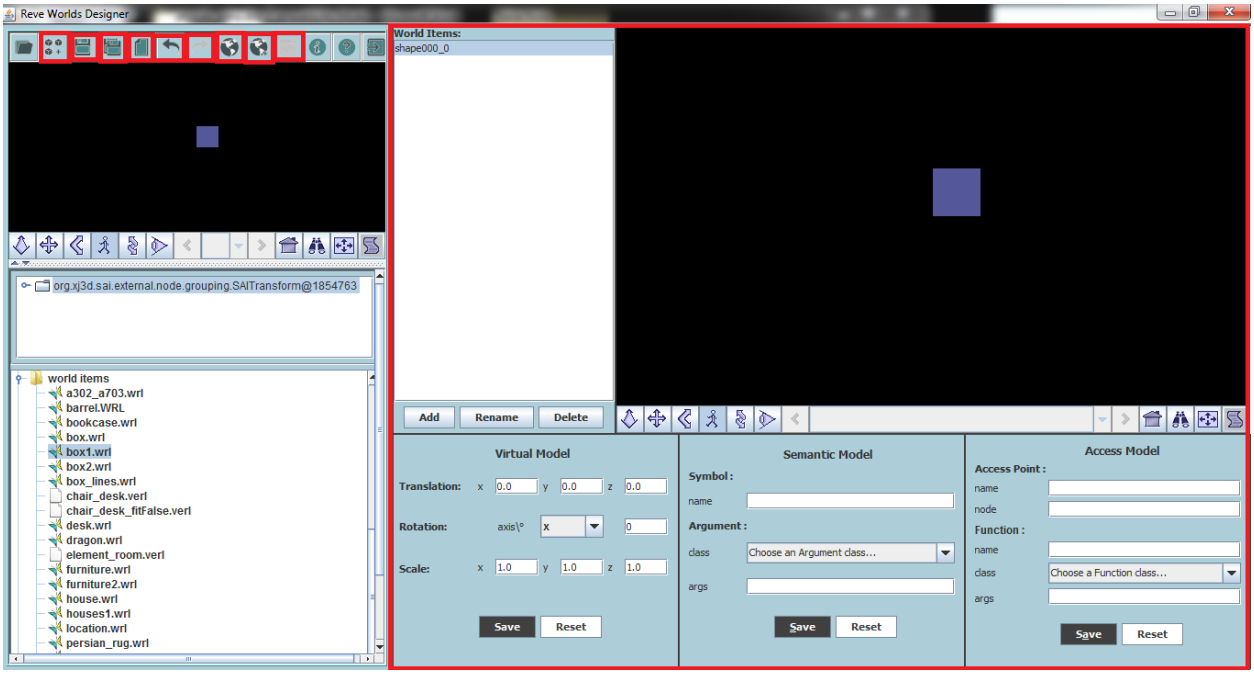


Εικόνα 6.1: Σχεδιαστική δομή παραθύρων

Αρχικά, συμφωνήθηκε να φτιαχτεί το σχέδιο της γραφικής διεπαφής ως ένα mock-up των παραθύρων από τα οποία θα αποτελείται, όπως φαίνονται στην Εικόνα 6.1. Το πρώτο κομμάτι, που χωρικά είναι και το αριστερό κομμάτι αφορά στην επιλογή του αντικειμένου ή κάποιου υπο-μέρους του για να εισαχθεί στον εικονικό κόσμο και περιλαμβάνει τα στοιχεία που εμφανίζονται στην Εικόνα 6.2 και αναλύονται και στην προαναφερθείσα μεταπτυχιακή διατριβή. Το δεύτερο κομμάτι αφορά τη σχεδίαση του εικονικού κόσμου, τροποποιώντας κάποια στοιχεία του πρώτου τμήματος και εισάγοντας καινούρια, όπως φαίνεται στην Εικόνα 6.3.



Εικόνα 6.2: Πρώτο κομμάτι – επιλογή αντικειμένου

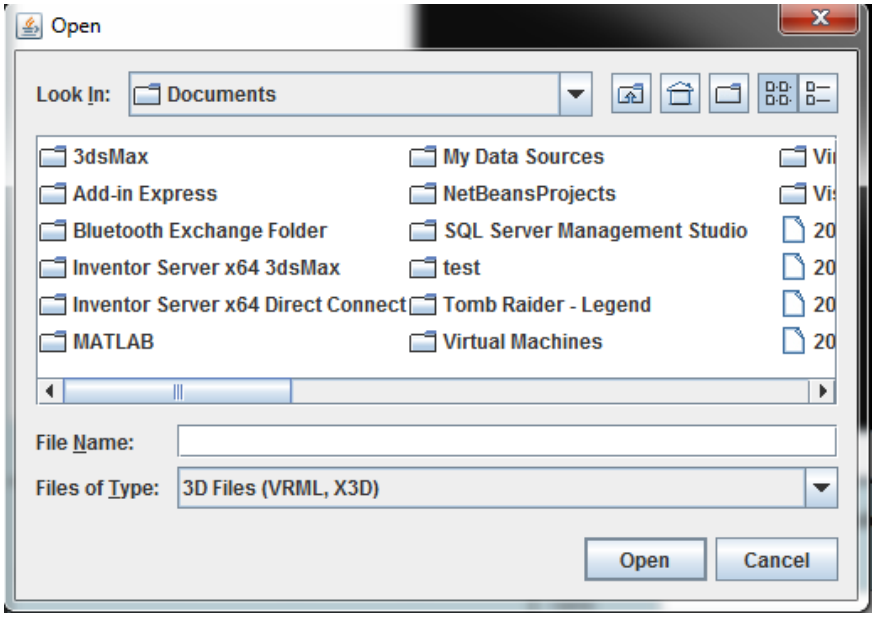


Εικόνα 6.3: Δεύτερο κομμάτι – σχεδίαση κόσμου

Παρακάτω παρουσιάζουμε τα συστατικά που αποτελούν την εφαρμογή μας,την λειτουργικότητά τους, καθώς και τις δυσκολίες που αντιμετωπίσαμε κατά την υλοποίησή τους.

6.2 Επιλογή Αντικειμένου (3D Object)

Το πρώτο βήμα για την σχεδίαση ενός εικονικού κόσμου είναι η επιλογή των αντικειμένων που θα τον αποτελούν. Η αναζήτησή τους πρέπει να είναι μια εύκολη διαδικασία για τον χρήστη, επομένως αποφασίστηκε να γίνει με δυο τρόπους. Ο πρώτος τρόπος, που είναι και ο πιο κλασικός, επιτυγχάνεται από το κεντρικό μενού, από την επιλογή Open File  , όπου αξιοποιώντας το JfileChooser component της swing, δίνουμε τη δυνατότητα στον χρήστη να περιηγηθεί στους φακέλους του υπολογιστή του, με default τον φάκελο My Documents, για να επιλέξει τα αρχεία με κατάληξη .wrl (για VRML97 αντικείμενα) ή .x3dv (για X3D αντικείμενα), όπως φαίνεται στην Εικόνα 6.4.



Εικόνα 6.4: OpenFile

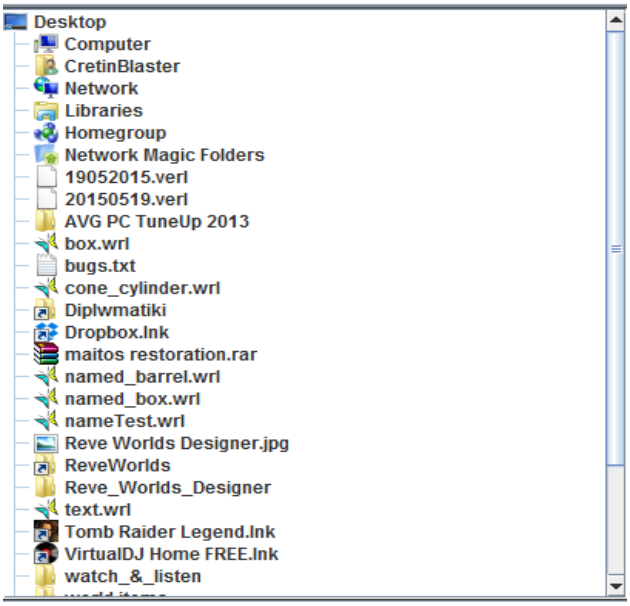
Σε περίπτωση που ο χρήστης επιλέξει να βλέπει όλα τα αρχεία (All Files)στο 'Files of Type'και διαλέξει ένα αρχείο μη συμβατό με την εφαρμογή, δηλαδή ούτε .wrl ούτε .x3dv, του εμφανίζεται σχετικό μήνυμα λάθους (Εικόνα 6.5).

Εικόνα 6.5: Προσπάθεια ανοίγματος μη συμβατού αρχείου από το OpenFile

Στην περίπτωση, όμως, που το αρχείο που διαλέξει είναι αρχείο κόσμου verl, εμφανίζεται σε ξεχωριστό παράθυρο ο κώδικας του αρχείου (Εικόνα 6.6). Η δυνατότητα αυτή δίνεται στο χρήστη στα πλαίσια αναζήτησης αντικειμένου, εφόσον μέσα από τον κώδικα της verl, μπορεί να εντοπίσει ένα αντικείμενο που τον ενδιαφέρει.

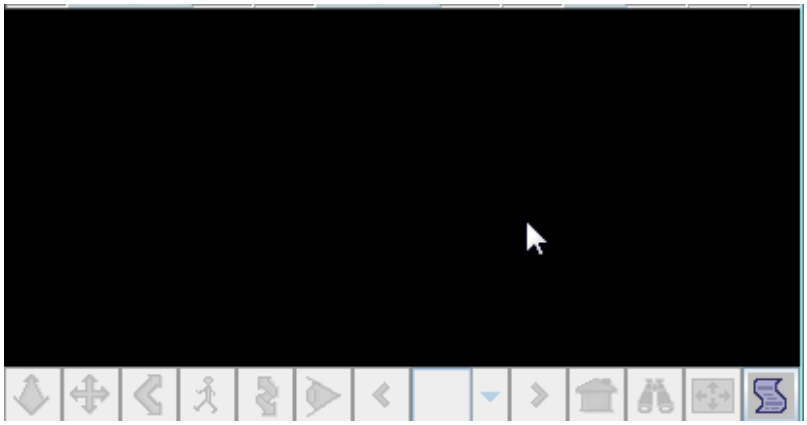
Εικόνα 6.6: Εμφάνιση κώδικα για verl αρχεία

Ο δεύτερος τρόπος δίνεται μέσα από την εμφάνιση των φακέλων συστήματος του χρήστη, μέσω του FileSystemView της swing που εμφανίζεται σε ξεχωριστό παράθυρο στην εφαρμογή (Εικόνα 6.7). Κατά αυτόν τον τρόπο, θεωρούμε πως είναι πιο άμεση και γρήγορη η πρόσβαση στα αρχεία του χρήστη, ενώ του επιτρέπεται παράλληλα η δυνατότητα να εκτελεί και άλλες ενέργειες, καθώς κλικάροντας τρεις φορές με το ποντίκι (γιατί 2 είναι οι default για το άνοιγμα φακέλου) σε ένα μη συμβατό αρχείο, αυτό ανοίγει με το προεπιλεγμένο πρόγραμμα του λειτουργικού συστήματος, πχ κλικάροντας 3 φορές ένα word αρχείο αυτό θα ανοιξει με το MS Office Word. Μόνο στην περίπτωση των verl αρχείων, η συμπεριφορά είναι αντίστοιχη με το Open File, δηλαδή εμφανίζεται σε άλλο παράθυρο ο κώδικας verl αυτού του αρχείου.



Εικόνα 6.7: Εμφάνιση φακέλων συστήματος για εύκολοτερη περιήγηση του χρήστη στα αρχεία

Ωστόσο, πέρα από την επιλογή του αρχείου, είναι απαραίτητο ο χρήστης να βλέπει ποιο αντικείμενο ακριβώς απεικονίζεται στο εν λόγω αρχείο ώστε να αποφασίσει αν θέλει πράγματι να το προσθέσει στον κόσμο. Δημιουργήσαμε, λοιπόν, πάνω από το παράθυρο εμφάνισης φακέλων συστήματος, έναν xj3d browser, τον οποίο αποκαλούμε 'Object Viewport', όπου φορτώνεται το αντικείμενο που επιλέγεται κάθε φορά (Εικόνα 6.8).Επομένως, επιλέγοντας ένα κατάλληλο (δηλαδή .wrl/.x3dv) αρχείο από το Open File ή κλικάροντας μια φορά σε ένα τέτοιο αρχείο από το παράθυρο εμφάνισης φακέλων συστήματος, το αντικείμενο φορτώνεται στον browser. Βέβαια, αυτό προϋποθέτει το αρχείο να είναι σωστά δομημένο ως vrml ή x3d κώδικας και να μην είναι corrupt, διαφορετικά, όπως και στο Cortona 3D Viewern απεικόνισή του δεν καθίσταται δυνατή.



Εικόνα 6.8: Object Viewport – Οθόνη απεικόνισης επιλεγμένου αντικειμένου

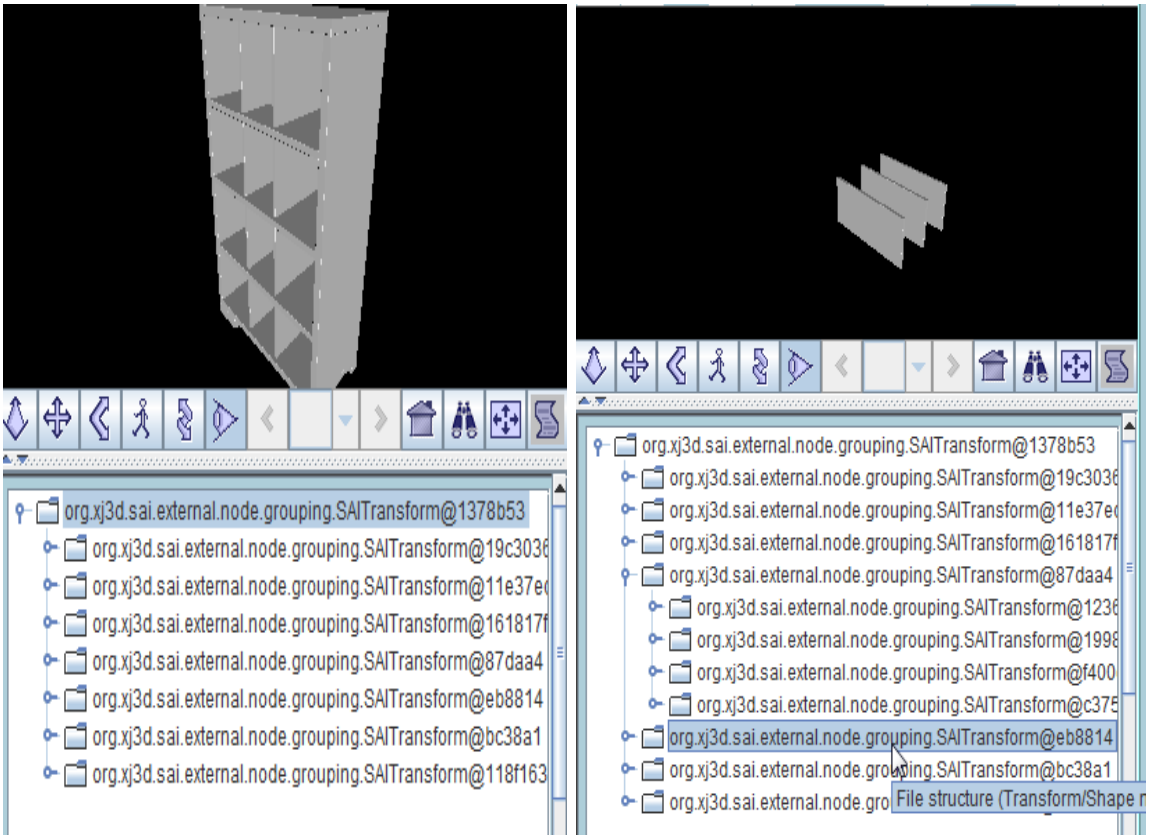
Τα ζητήματα που αντιμετωπίσαμε σε αυτό το σημείο ήταν διάφορα και μας δυσκόλεψαν αρκετά. Πρώτα από όλα, για την απεικόνιση του αντικειμένου δεν αρκεί η δημιουργία ενός αντικειμένου από την java κλάση ExternalBrowser που παρέχει το βασικό πλαίσιο περιήγησης της σκηνής, αλλά απαιτείται και η δημιουργία αντικειμένου από την java κλάση X3DScene, για την δημιουργία της σκηνής επί της οποίας τοποθετείται το τριδιάστατο αντικείμενο. Διαπιστώσαμε πως για να φορτώσουμε ένα άλλο αντικείμενο στον browser,δεν αρκούσε να αντικαταστήσουμε απλά τη σκηνή με το νέο αρχείο με τη χρήση της σχετικής μεθόδου replaceWorld() της xj3d, αλλά απαιτούσε να σετάρουμε εκ νέου όλο τον browser και τη σκηνή, μαζί με το Jpanel στο οποίο κουμπώνουν. Διαφορετικά, η μια σκηνή κρυβόταν πίσω από την άλλη δημιουργώντας inconsistency τόσο στη λειτουργικότητα όσο και στο γραφικό περιβάλλον διεπαφής. Επομένως, έπρεπε να ακολουθήσουμε διαφορετική προσέγγιση για την περίπτωση που επιλέγεται ένα αρχείο για πρώτη φορά και για την περίπτωση που έχει ήδη φορτωθεί ένα αρχείο και πάμε να φορτώσουμε άλλο.

Επιπρόσθετα, κατά την επιλογή ενός μη συμβατού αρχείου ή φακέλου, η σκηνή έπρεπε να καθαρίζει, ώστε αυτό που βλέπει ο χρήστης να ανταποκρίνεται σε αυτό που επιλέγει κάθε φορά. Άρα, έπρεπε εκ νέου να εκτελέσουμε την ίδια διαδικασία αφαίρεσης και εισαγωγής όλου του browser, καθώς η αντικατάσταση της σκηνής με null τιμή, παρότι αναφερόταν στο σχετικό tutorial, δεν δούλευε, ενώ η αφαίρεση των κόμβων-ρίζα (root nodes) του αντικειμένου από τη σκηνή – ώστε να αποφευχθεί η διαδικασία προσθαφαίρεσης του browser - απαιτούσε διαφορετική προσέγγιση για την επανενεργοποίηση της σκηνής, γεγονός που αύξανε την πολυπλοκότητα στον κώδικα.

Τέλος, υπήρχε σοβαρό ζήτημα με τα verl αρχεία, καθώς ως άγνωστος μορφότυπος στο λειτουργικό σύστημα, η επιλογή τους αφαιρούσε ολοκληρωτικά το JPanel, μαζί με το toolbar που υπήρχε από πάνω του. Αδυνατώντας να εφαρμόσουμε την ίδια λύση για τα υπόλοιπα μη-συμβατά αρχεία, αποφασίσαμε στην περίπτωση που επιλέγεται verl αρχείο από τους φακέλους συστηματος, η εικόνα του browser να διατηρείται ίδια με την προηγούμενη, επομένως πρακτικά επαναφορτώνεται το αρχείο που υπήρχε πριν ή η σκηνή παραμένει κενή αν ένα μη-συμβατό αρχείο είχε επιλεχθεί τελευταίο. Τα παραπάνω επιτεύχθηκαν με τη χρήση ενός counter που ελέγχει αν είναι η πρώτη φορά που φορτώνεται ένα αρχείο και αν το αρχείο αυτό είναι

έγκυρο, καθώς και διατηρώντας σε μεταβλητές στοιχεία για το αρχείο που είχε φορτωθεί τελευταίο στον browser,για την περίπτωση που μετά επιλεχθεί verl αρχείο.

Στη συνέχεια, έπρεπε να διασφαλίσουμε πως το αντικείμενο που επιλέγει ο χρήστης μπορεί να εισαχθεί στον κόσμο, δηλαδή πληρεί τις προδιαγραφές της verl να πρόκειται για έναν DEF Transform κόμβο. Επομένως, έπρεπε να ελέγξουμε την εσωτερική δομή του αντικειμένου, δηλαδή τους X3D κόμβους που το αποτελούν. Παράλληλα, θέλαμε να δώσουμε στο χρήστη τη δυνατότητα να περιηγηθεί στο γράφημα σκηνής του αντικειμένου και να επιλέξει επιμέρους κόμβους. Για παράδειγμα, αν το αντικείμενο είναι μια βιβλιοθήκη, θέλαμε ο χρήστης να μπορεί να επιλέξει μόνο τα ράφια που το αποτελούν για να τα προσθέσει στον κόσμο (Εικόνα 6.9).

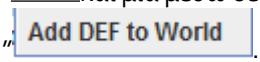


Εικόνα 6.9: Εμφάνιση βιβλιοθήκης (αριστερά) και επιλογή μόνο ραφιών (δεξιά)

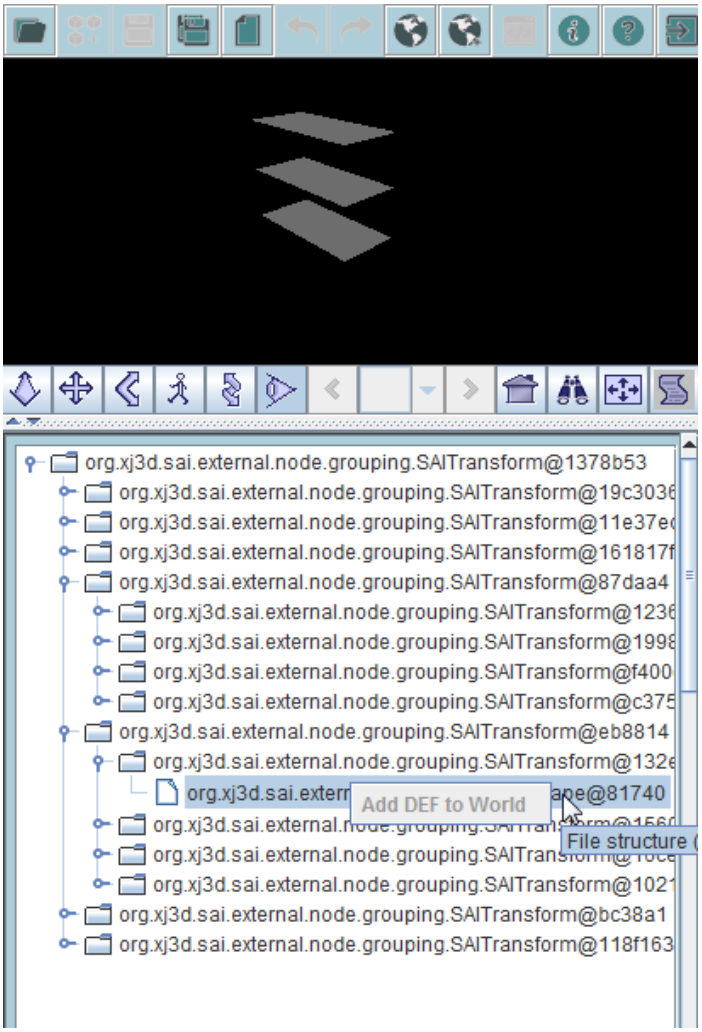
Επομένως, κατά την επιλογή του αντικειμένου, ψάχνουμε τους κόμβους-ρίζα, εφεξής root nodes, εκ των οποίων επιλέγουμε μόνο τα Transform nodes. Ελέγχουμε αν τα Transform nodes έχουν children κι άλλα Transform nodes, για να επαναλάβουμε αναδρομικά την διαδικασία. Αυτό έχει ως αποτέλεσμα να εμφανίζουμε ένα γράφημα σκηνής του αντικειμένου που λαμβάνει υπόψη την εσωτερική του δομή για Transform και Shape nodes. Κατά την επιλογή των εσωτερικών κόμβων του γραφήματος σκηνής, ο browser αδειάζει από το root node που απεικονίζει, με την μέθοδο removeRootNode() της xj3dκαι επαναγεμίζει με τον επιλεγμένο κόμβο, καθιστώντας αυτόν ως root node με τη μέθοδο addRootNode() της xj3d.

Σε αυτό το σημείο, οφείλουμε να πούμε πως έπρεπε να τοποθετήσουμε το γράφημα σκηνής του αντικειμένου σε ένα σημείο χωρικά κοντά με την αναπαράστασή του, χωρίς ωστόσο να καταλαμβάνει ένα καθορισμένο μέγεθος κάθε φορά στην οθόνη, δεδομένου ότι ο χώρος ήταν περιορισμένος και δε θέλαμε να τον υπερφορτώσουμε με πληροφορία. Επομένως, αποφασίσαμε να προσθέσουμε το γράφημα σκηνής σε ένα JPanel το οποίο εμφανίζεται μόνο κατά την επιλογή ενός αντικειμένου και καταλαμβάνει ένα χώρο σχετικό με

το μέγεθος του γραφήματος σκηνής του. Αυτό, βέβαια, είχε ως αποτέλεσμα να πρέπει να καθαρίζουμε το γράφημα σκηνής κατά την επιλογή διαφορετικού αντικειμένου ή να το αφαιρούμε τελείως κατά την επιλογή μη-συμβατού αρχείου.

Τέλος, έπρεπε να βρεθεί τροπος να εισαχθεί ο επιλεγμένος κόμβος στη σκηνή του κόσμου. Δόθηκαν και εδώ δυο επιλογές, μια μέσω του κεντρικού μενού με το κουμπί "Add Node to World"  και μια μέσω δεξιού κλικ στον κόμβο που θέλουμε να επιλέξουμε με την ονομασία "Add DEF to World" 

. Και στις δυο περιπτώσεις, ελέγχεται αν ο κόμβος που επιλέγει ο χρήστης είναι DEF Transform και μόνο σε αυτή την περίπτωση ενεργοποιείται η δυνατότητα προσθήκης του (Εικόνα 6.10).



Εικόνα 6.10: Οι επιλογές προσθήκης είναι γκριζαρισμένες για Shape κόμβους

Σε αυτό το σημείο αντιμετωπίσαμε δυο ζητήματα. Το πρώτο αφορούσε την προσθήκη του κόμβου μέσα από το κεντρικό μενού, καθώς αν ο χρήστης περιηγούνταν στο γράφημα σκηνής επιλέγοντας διαφορετικούς κόμβους, οι κόμβοι αυτοί προστίθονταν όλοι – και όχι μόνο ο τελευταίος - κατά το πάτημα του 'Add Node to World' κουμπιού, γεγονός που δεν συνέβαινε με το δεξί κλικ. Διαπιστώσαμε πως αυτό οφειλόταν στο γεγονός πως από το κουμπί του κεντρικού μενού δεν ήταν διακριτός ποιος κόμβος έχει επιλεχθεί, με αποτέλεσμα σε κάθε επιλογή κόμβου να συσσωρεύονται στη μνήμη και να προστίθενται όλοι. Αυτό λύθηκε με τη χρήση μιας

static μεταβλητής που κρατάει πληροφορία για την τελευταία επιλογή και καθαρίζει τα προηγούμενα δεδομένα κάθε φορά, οπότε ο κόμβος που τελικά προστίθεται είναι μόνο ο τελευταίος.

Το δεύτερο ζήτημα αφορούσε την προσπάθεια να επιτρέψουμε οποιονδήποτε Transform κόμβο του γραφήματος σκηνής να προστεθεί στον κόσμο, αυξάνοντας τις επιλογές του χρήστη. Αυτό προϋπέθετε να μετατρέψουμε εμείς τον κόμβο σε DEF node. Ωστόσο, αυτό δεν κατέστη δυνατό, καθώς παρότι η xj3d επιτρέπει να βρεις τους DEF κόμβους με τις μεθόδους getNamedNodes()/getNamedNode(), δεν προσφέρει κάποια μέθοδο για να ορίσεις έναν κόμβο ως DEF. Επίσης, οι κόμβοι λαμβάνονται από την μνήμη, όπου φορτώνονται, επομένως, ακόμα και αν βρίσκαμε έναν τρόπο να παρέμβουμε εκεί μετατρέποντας έναν Transform κόμβο σε DEF, αυτό θα είχε μόνο προσωρινή λειτουργικότητα, καθώς θα έπρεπε αυτός ο DEF κόμβος να ορίζεται και στο αρχείο, από όπου αντλεί τα item sources η verl.Αυτό τελικά συνεπάγεται πως ο χρήστης πρέπει χειροκίνητα να ανοίξει τον κώδικα ενός 3d αρχείου και να ονοματίσει ένα Transform του ως DEF αν θέλει να το χρησιμοποιήσει οπωσδήποτε στον κόσμο.

6.3 Ορισμός του Item

Το επόμενο ζητούμενο που έπρεπε να αντιμετωπίσουμε είναι να εισάγουμε τους DEF X3DNode κόμβους στον εικονικό κόσμο. Εδώ είναι που γίνεται και η σύνδεση μεταξύ των δυο μεταπτυχιακών διατριβών, μέσω της μετατροπής ενός κομβου σε item για να προχωρήσουμε στη σχεδίαση του εικονικού κόσμου. Για αυτό το σκοπό, δημιουργούμε μια ξεχωριστή κλάση (Item.java), όπου α) κρατάμε τις πληροφορίες από τον X3Dnod εκόμβο που χαρακτηρίζουν ένα item, όπως είναι το όνομα του DEF κόμβου (που μας χρειάζεται για το virtual source) και το όνομα και η τοποθεσία του αρχείου από όπου παίρνουμε το αντικείμενο (που μας χρειάζεται για το itemGroup source) και β) δημιουργούμε μεθόδους για να ορίσουμε το όνομα ενός item καθώς και τα βασικά του συστατικά (virtualmodel, accessmodel, semanticmodel).

Πέρα από τον κόμβο για να δημιουργήσουμε ένα item, χρειαζόμαστε και μια σκηνή, που θα είναι και η σκηνή του κόσμου μας. Αυτό, γιατί το item προϋποθέτει έναν εξωτερικό Transform κόμβο, που θα είναι ο κόμβος-ρίζα του γραφήματος σκηνής που αποτελεί το αντικείμενο, πάνω στον οποίον θα γίνονται όλα τα transformations. Χρειαζόμαστε, δηλαδή, την X3DScene σκηνή, πέρα από την απεικόνιση του κόσμου, για να χρησιμοποιήσουμε την μέθοδο createNode("Transform") του xj3d που φτιάχνει έναν Transform κόμβο, ώστε ως παιδί του να ορίσουμε τον X3Dκόμβο που επιλέγεται από το γράφημα σκηνής. Έπρεπε, λοιπόν, να αρχικοποιήσουμε τη σκηνή και για αυτό τον σκοπό χρησιμοποιούμε ένα dummy vrml αρχείο, το test.wrl, φορτώνοντάς το για να δημιουργήσουμε τη σκηνή και αφαιρώντας αμέσως μετά το περιεχόμενό του, ώστε η σκηνή να μείνει κενή για χρήση. Παρακάτω, ο κώδικας για το βασικό constructor του item:

```
//Item constructor with def name
    public Item(X3DScene mainScene, X3DNode node, String name, String path){
        this.mainScene = mainScene;
        this.copy = node;
        this.location = path;
        ////////////////////////////////////////////////////
        SFVec3f translation = (SFVec3f)node.getField("translation");
System.out.println("Original translation "+translation.toString());
        ////////////////////////////////////////////////////
        //Create a new root Transform node to add the object under it
        //in order to be able to handle its transformations separately
        transform = mainScene.createNode("Transform");
        children = (MFNode)(transform.getField("children"));
        children.append(node);
        //Create the item and save its DEF name
```

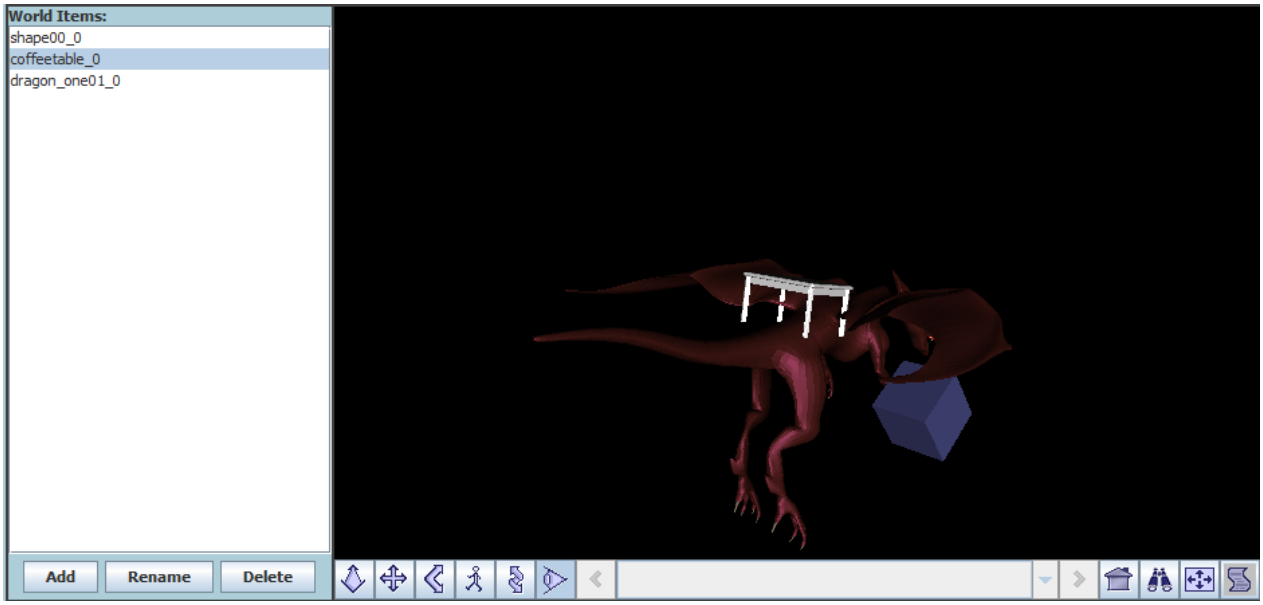
```
        createItem(transform,name);
        //Give a default name to the item added
        setDefaultName();
        //Set a default semantic and access model with null values
        setDefaultSemanticModel();
        setDefaultAccessModel();
   }//end constructor
```

Έχοντας, λοιπόν, 1)τον επιλεγμένο DEF κόμβο, 2)τη σκηνή του κόσμου και 3)τον ορισμό του item, μπορέσαμε να προσθέσουμε ένα αντικείμενο ως item στην σκηνή, εμπλουτίζονταςτον κώδικα των σχετικών κουμπιών (Add Node/DEF to world).

Σε αυτό το σημείο, οφείλουμε να πούμε πως υπάρχει ένα ουσιαστικό πρόβλημα που δεν λύθηκε στα πλαίσια αυτής της μεταπτυχιακής διατριβής. Η εισαγωγή ενός item στην σκηνή δεν το ευθυγραμμίζει, όπως η Reve, στην αρχή των αξόνων του κόσμου, δηλαδή δεν τοποθετείται στο σημείο (0,0,0) στην οθόνη του κόσμου αλλά το αντικείμενο διατηρεί το transformationπου έχει από τον VRML κώδικα, με αποτέλεσμα να εμφανίζεται σε εκείνες τις συντεταγμένες στον κόσμο. Αυτό σημαίνει πως το εξωτερικό Transform που προστίθεται για την λειτουργία του virtual model, λαμβάνει ως σημείο αναφοράς το σημείο που βρίσκεται κάθε φορά το εισαγόμενο αντικείμενο και όχι την αρχή των αξόνων. Επομένως, το virtual model είναι σχετικό με την θέση και το συνολικό transformation κάθε αντικειμένου και όχι με την αρχή των αξόνων.

Παρότι έγινε προσπάθεια το αντικείμενο να ευθυγραμμιστεί στην αρχή των αξόνων, μέσω προσθαφαίρεσης των translations και των rotation του, αυτό δεν κατέστη εφικτό καθώς δημιουργούσε προβλήματα, με το αντικείμενο να έχει τυχαία συμπεριφορά ως προς την θέση του στον κόσμο. Πέραν τούτου, η συγκεκριμένη προσθαφαίρεση θα έπρεπε να γίνει σε όλα τα εσωτερικά transformations από τα οποία αποτελείται ένα αντικείμενο, ενώ αν υπήρχε και scale στο αντικείμενο δημιουργούσε περαιτέρω προβλήματα. Δεδομένου ότι ο κ. Αναστασάκης έχει υλοποιημένο κώδικα για να επιτευχθεί η ευθυγράμμιση στην αρχή των αξόνων, αποφασίστηκε, σε συνεννόηση μαζί του, αυτή η διόρθωση να γίνει σε δεύτερο χρόνο είτε από τον ίδιο είτε από άλλον συνάδελφο, εφόσον απαιτούσε αρκετό χρόνο και η μεταπτυχιακή διατριβή προσφέρεται για περαιτέρω συνέχεια. Ωστόσο, υπάρχουν σχόλια στον κώδικα όπου θεωρούμε πως πρέπει να τοποθετηθεί το συγκεκριμένο κομμάτι κώδικα, συγκεκριμένα στον constructor του Item class.

Επιστρέφοντας στο προηγούμενο θέμα, κατά την εισαγωγή του αντικειμένου στην σκηνή του κόσμου ενημερώνουμε δυο αλληλένδετες δομές δεδομένων όπου πρέπει να αποθηκεύουμε την πληροφορία του κόσμου, που είναι δυο λίστες, μια με τα ονόματα των αντικειμένων και μια με τα items που υπάρχουν στον κόσμο. Αυτές οι δομές δεδομένων μοναδικοποιούνται μέσω του ονόματος του αντικειμένου που προσδιορίζει την ταυτότητά του. Η διάκριση μεταξύ των δυο δομών έγινε καθαρά για να μπορεί ο χρήστης να επιλέξει ένα αντικείμενο βάσει του ονόματός του και να το χειριστεί από τη σχετική Jlist που εμφανίζεται στην εφαρμογή υπό το JLabel "World Items" (Εικόνα 6.11).

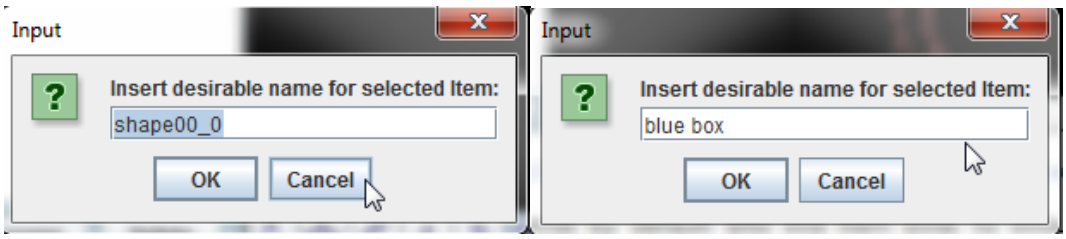


Εικόνα 6.11: Τα ονόματα των items (αριστερά) και τα itemsστην σκηνή (δεξιά)

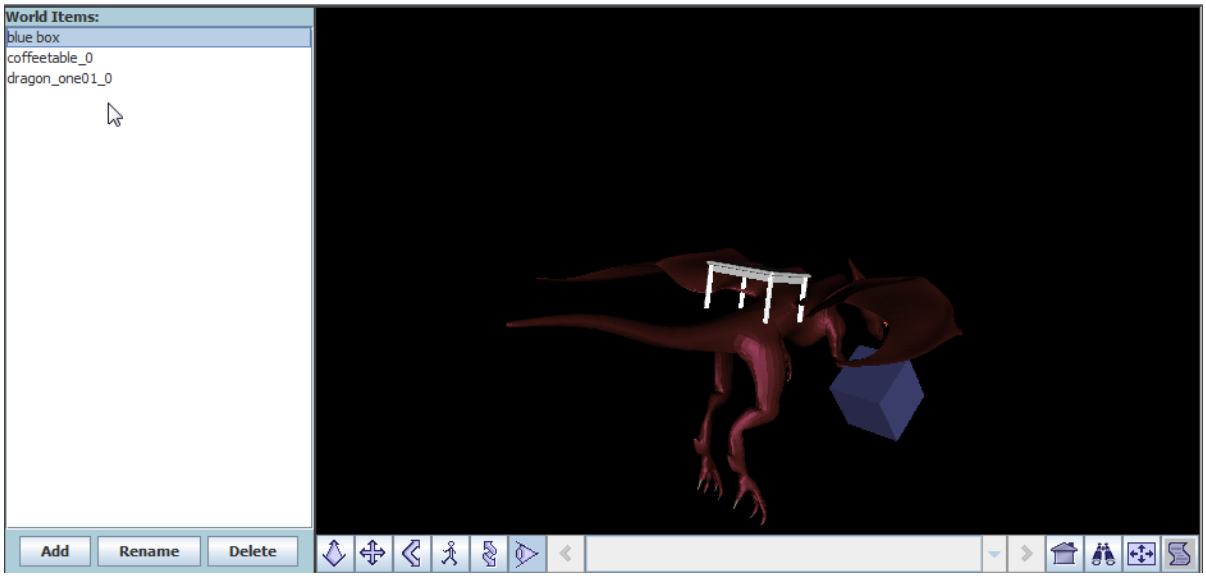
6.4 Μενού Επιλογών για Item

6.4.1. Μετονομασία ενός Item

Το όνομα που λαμβάνεται by default από ένα item είναι το όνομα του DEF κόμβου του συνοδευμένο από κάτω παύλα και έναν αριθμό που αυξάνει κάθε φορά που εισάγουμε τον ίδιο κόμβο στη σκηνή. Αυτό, γιατί έπρεπε να δώσουμε τη δυνατότητα στον χρήστη να εισάγει πολλές φορές το ίδιο αντικείμενο, καθιστώντας το κάθε φορά μοναδικό βάσει του ονόματός του. Ωστόσο, το όνομα δεν είναι δεσμευτικό και ο χρήστης μπορεί να το αλλάξει για να είναι πιο εύκολα κατανοητό στον ίδιο, μέσω της επιλογής Rename, όπως φαίνεται και στην Εικόνα 6.12, που υπάρχει σε ένα μενού κάτω από τη λίστα των αντικειμένων.



(α) (β)



(γ)

Εικόνα 6.12: Μετονομασία <item>

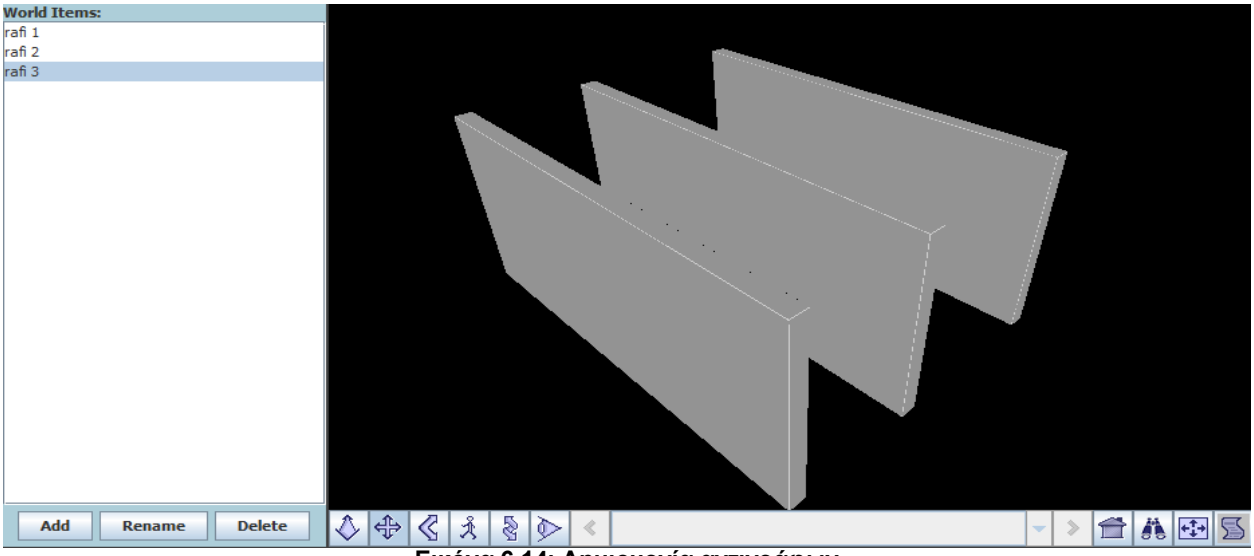
Για το rename έπρεπε να αντιμετωπίσουμε την περίπτωση που ο χρήστης δεν δίνει όνομα και την περίπτωση που ο χρήστης δίνει ένα όνομα που ήδη χρησιμοποιείται. Δεδομένου ότι το όνομα του item το μοναδικοποιεί στην λίστα, έπρεπε να προβλέψουμε αυτές τις περιπτώσεις, ώστε να αποφεύγονται λάθη. Για παράδειγμα, αν αφήναμε δυο items να έχουν το ίδιο όνομα και αλλάζαμε το virtual model του ενός, αυτό επηρέαζε και το άλλο, εφόσον το όνομα του item αποτελεί για εμάς κλειδί σε αυτή τη δομή δεδομένων. Για αυτές τις περιπτώσεις, λοιπόν, ενημερώνουμε τον χρήστη με σχετικό warning (Εικόνα 6.13).

Εικόνα 6.13: Περίπτωση κενού ονόματος (αριστερά) και υπάρχοντος ονόματος (δεξιά)

Πέρα από το Rename προσφέρουμε άλλες δυο βασικές επιλογές στο χρήστη μέσα από τη λίστα ονομάτων, τις οποίες τις ενσωματώσαμε και χωρικά κάτω από τα items για εύκολη πρόσβαση, διάκριση από το κεντρικό μενού και ακριβώς επειδή θεωρούμε ότι είναι οι πιο συχνές που μπορεί να εκτελέσει ο χρήστης αναφορικά με ένα item. Αυτές είναι το κουμπί Add και Delete.

6.4.2. Προσθήκη Αντιγράφου ενός Item

Η επιλογή Add προσθέτει ένα αντίγραφο του αντικειμένου στον κόσμο. Αυτό σημαίνει πως κατά την προσθήκη ενός item,αυτό συμπίπτει πάνω στο άλλο item καθώς κληρονομεί όλες τις ιδιότητές του, δηλαδή, με εξαίρεση το όνομα, έχει το ίδιο virtual model, access modelκαι semantic model. Αυτό είναι χρήσιμο σε περίπτωση που ο χρήστης θέλει να προσθέσει πανομοιότυπα αντικείμενα στον κόσμο που θα βρίσκονται απλά σε άλλη θέση, πχ αν έχουμε ένα ράφι μπορούμε να φτιάξουμε ένα αντίγραφό του και μετατοπίζοντάς το μονάχα ως προς έναν άξονα, πχ τον y, να φτιάξουμε μια σειρά από ράφια, χωρίς να χρειάζεται να φορτώνουμε το ίδιο αρχείο και να εκτελούμε όλα τα απαραίτητα transformations στο virtual model κάθε φορά. Αυτή η λειτουργικότητα δεν θα αλλάξει με την διόρθωση για τοποθέτηση των αντικειμένων στην αρχή των αξόνων, εφόσον τα virtual models απλά θα έχουν άλλο σημείο αναφοράς (την αρχή των αξόνων αντί για τον εαυτό τους), αλλά οι τιμές μεταξύ τους θα είναι ίδιες.



Εικόνα 6.14: Δημιουργία αντιγράφων

Για να επιτύχουμε την δημιουργία αντιγράφου μέσω του Add button, δεν μπορούσαμε να χρησιμοποιήσουμε την μέθοδο addItem() που δημιουργήσαμε για την προσθήκη αντικειμένου στον κόσμο, ούτε απλά να δημιουργήσουμε ένα αντίγραφό του σε μια άλλη Item μεταβλητή, καθώς τα παραπάνω μας δημιουργούσαν προβλήματα.

Στην πρώτη περίπτωση, το αντικείμενο φτιαχνόταν ως αντίγραφο του πρώτου αλλά υπήρχε σχέση γονέα-παιδιού μεταξύ τους και δεν ήταν ξεχωριστά items, με αποτέλεσμα το παιδί να έχει λανθασμένο virtual model καθώς το σημείο αναφοράς (0,0,0) αντιστοιχούσε στο κέντρο του γονέα του. Συνεπακόλουθα, ο γονέας και το αντίγραφο λειτουργούσαν ως γράφημα σκηνής που επηρέαζε το ένα το άλλο. Δηλαδή, η μεταβολή του virtual modelτου γονέα οδηγούσε και σε μεταβολή της χωρικής προβολής του αντιγράφου-παιδιού, χωρίς, όμως, να ενημερώνεται κατάλληλα το virtual modelτου. Πέρα από αυτό το πρόβλημα, είχαμε θέμα να κρατήσουμε αυτή την λειτουγικότητα για άλλο σκοπό,καθώς η σχέση των αλληλεξαρτώμενων κόμβων δημιουργούσε προβλήματα αν σβήναμε τον γονέα, όπου το παιδί παρέμενε στη σκηνή αλλά είχε διαφορετικό σημείο αναφοράς στο χώρο, αυτό του γονέα που δεν υπήρχε πια.

Στην δεύτερη περίπτωση, το αντίγραφο έβλεπε την ίδια θέση μνήμης με το αντικείμενο στη σκηνή, οπότε αλλάζοντας το virtual model του ενός άλλαζε και του άλλου. Αυτό θα μπορούσε να χρησιμοποιηθεί για διαφορετικό σκοπό, όμως, στην περίπτωση που διαγράφαμε το ένα αντικείμενο διαγραφόταν και το άλλο λόγω αυτής της σχέσης εξάρτησης, καθώς είχαν αναφορά στον ίδιο κόμβο. Για να επιλύσουμε τα παραπάνω, έπρεπε να διατηρήσουμε την πληροφορία για το X3Dnode κάθε αντικειμένου και να δημιουργήσουμε ένα νέο item από αυτόν τον κόμβο, όπου του θέταμε το virtual, semantic και access model του αρχικού. Για αυτό δημιουργήσαμε μια ξεχωριστή μέθοδο, την copyItem(Item item).

6.4.3. Διαγραφή ενός Item

Όσον αφορά το Delete button, όπως μαρτυρά το όνομά του, διαγράφει ένα επιλεγμένο αντικείμενο από τον κόσμο. Αυτό συνεπάγεται ότι αφαιρείται το όνομά του από την λίστα και η απεικόνισή του από την σκηνή. Παρότι θεωρούσαμε ότι με την μέθοδο removeRootNode() της xj3d, θα επρόκειτο για μια απλή υπόθεση βάζοντας σαν όρισμα της μεθόδου τον σχετικό κόμβο που θέλαμε να διαγράψουμε, απεδείχθη πως δεν ήταν τόσο εύκολο.

Με απλή χρήση της μεθόδου removeRootNode(), το αντικείμενο δεν αφαιρούνταν από τη σκηνή, παρά μόνο αν ήταν το τελευταίο αντικείμενο που είχε εισαχθεί στον κόσμο. Καθώς κάτι τέτοιο δεν αναφερόταν στο σχετικό tutorial, συνέβαινε γιατί, με την εισαγωγή μέσω της addRootNode() ένός κόμβου στη σκηνή (δεδομένου ότι δεν δινόταν άλλος τρόπος από την xj3d), μόνο ο τελευταίος θεωρούνταν πραγματικά κόμβος-

ρίζα (rootNode). Επομένως, είχαμε δυο επιλογές. Η μία ήταν να αναδιατάσσουμε κάθε φορά τη σκηνή, διαγράφοντας όλα τα αντικείμενα και εισάγοντάς τα ώστε το επιλεγμένο να εισάγεται τελευταίο για να μπορεί να διαγραφεί και η δεύτερη ήταν να αφαιρέσουμε διαδοχικά όλα τα αντικείμενα στη σκηνή και να τα προσθέσουμε ξανά πλην του επιλεγμένου προς διαγραφή κόμβου.

Η πρώτη επιλογή αύξανε την πολυπλοκότητα και απαιτούσε περισσότερη μνήμη και υπολογιστική ισχύ από την δεύτερη, που τελικά προτιμήθηκε. Εν τέλει, η διαγραφή ενός αντικειμένου επιτυγχάνεται με την διαδοχική αφαίρεση (μέσω ενός βρόγχου), των αντικειμένων στη σκηνή, με την αντίστροφη σειρά από την οποία εισήχθησαν (Last In First Out) και στη συνέχεια την επαναπροσθήκη των αντικειμένων στη σκηνή, εκτός εκείνου που είχε επιλεχθεί να διαγραφεί. Είναι κατανοητό πως η διαγραφή, λοιπόν, πρακτικά δεν είναι μια απλή αφαίρεση ενός αντικειμένου από τον κόσμο αλλά η επανασύνθεση του κόσμου χωρίς αυτό το αντικείμενο.

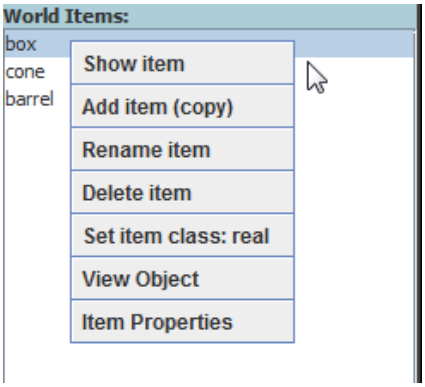
Σχετικά με το Add και το Delete button, έπρεπε και εδώ να διασφαλίσουμε ότι τα κουμπιά ενεργοποιούνται μονάχα όταν είναι επιλεγμένο ένα αντικείμενο από την λίστα και η λειτουργικότητά τους αφορά αυτό το συγκεκριμένο αντικείμενο. Σε διαφορετική περίπτωση, ο χρήστης μπορεί να πατούσε Add και να προστίθετο το αντικείμενο που είχε διατηρηθεί τελευταίο στην μνήμη χωρίς να είναι αυτό που ήθελε ή ακόμα χειρότερα, να πατάει Add και το τελευταίο αντικείμενο που υπήρχε στην μνήμη να είχε διαγραφεί, οπότε προέκυπταν σχετικά σφάλματα.

Αντίστοιχα, με το Delete μπορεί ο χρήστης να πατούσε διαδοχικά το κουμπί και να πήγαινε να αφαιρέσει ένα αντικείμενο που δεν υπήρχε πλέον ή να πατήσει Delete ενώ η σκηνή είναι ήδη άδεια και ομοίως να προκύπτουν σφάλματα στην εφαρμογή. Παρόμοια προβλήματα υπήρχαν και με το Rename, αν ο χρήστης το πάταγε χωρίς να έχει επιλέξει όνομα από την λίστα.

Για να ικανοποιήσουμε την προϋπόθεση τα παραπάνω κουμπιά να σχετίζονται με ένα επιλεγμένο αντικείμενο, αποφεύγοντας τα παραπάνω σφάλματα, ενεργοποιούμε τα 'Add', 'Rename', 'Delete', μονάχα όταν ένα αντικείμενο από την λίστα είναι επιλεγμένο, ενώ τα απενεργοποιούμε κατά την εκκίνηση της εφαρμογής, όταν η σκηνή είναι άδεια και όταν τα έχουμε μόλις πατήσει (οπότε απεπιλέγεται και το αντικείμενο από την λίστα).

6.4.4. Επιπλέον δυνατότητες

Προς διευκόλυνση του χρήστη, οι παραπάνω επιλογές εμφανίζονται κι όταν ο χρήστης επιλέγει με δεξί κλικ ένα αντικείμενο, οπότε του εμφανίζεται ένα μενού επιλογών (JPopupMenu της swing) και με αυτές και κάποιες επιπλέον δυνατότητες. Αντίστοιχα με πάνω, το μενού ενεργοποιείται όταν διασφαλίζεται πως ένα αντικείμενο είναι επιλεγμένο. Από τις δοθείσες επιλογές, το 'Add item (copy)' αντιστοιχεί στο κουμπί 'Add', το 'Rename item' στο κουμί 'Rename' και το 'Delete item' στο 'Delete' και έχουν ακριβώς την ίδια λειτουργικότητα. Οι υπόλοιπες επιλογές περιγράφονται παρακάτω.



Εικόνα 6.15: Μενού επιλογών για item (με δεξί κλικ)

Το 'Show Item' κάνει διακριτό στον χρήστη το επιλεγμένο αντικείμενο στον κόσμο. Η λειτουργικότητα αυτή επιτυγχάνει ο χρήστης να αντιλαμβάνεται σε ποιο ακριβώς αντικείμενο αντιστοιχεί το όνομα που επιλέγει από την λίστα, ώστε να μπορεί να μεταβάλλει το σωστό αντικείμενο στον κόσμο. Μπορούμε να φανταστούμε πόσο σημαντική είναι αυτή η δυνατότητα, ιδίως για εικονικούς κόσμους που αποτελούνται από δεκάδες αντικείμενα. Ωστόσο, η εντολή αυτή μας προβλημάτισε αρκετά ως προς την υλοποίησή της, καθώς δυσκολευτήκαμε να βρούμε ένα χαρακτηριστικό που θα κάνει το αντικείμενο διακριτά ορατό στον χρήστη και κάμποσες προσπάθειες απέβησαν άκαρπες.

Μια πρώτη σκέψη ήταν το αντικείμενο να αλλάζει χρώμα και να γίνεται πχ. λευκό, ωστόσο, δεν μπορούσαμε να αποκλείσουμε την πιθανότητα ένα άλλο αντικείμενο στον κόσμο να έχει το ίδιο χρώμα, γεγονός που κατέρριπτε αυτόματα την εν λόγω λειτουργικότητα. Μια δεύτερη σκέψη ήταν να εμφανίζεται το αντικείμενο σε ένα ξεχωριστό παράθυρο αλλά και πάλι αυτό δεν απέκλειε την πιθανότητα να υπάρχουν δυο όμοια αντικείμενα στον κόσμο, οπότε και πάλι να καταρρίπτεται η λειτουργικότητα. Μια τρίτη σκέψη ήταν να δημιουργείται ένα αντικείμενο που αντιστοιχεί σε Textnode της VRML που να τοποθετείται πάνω από το αντικείμενο στον κόσμο, υποδεικνύοντας το όνομά του. Ωστόσο και αυτή η προσπάθεια είχε αρκετά προβλήματα. Πρώτον, γιατί η τοποθέτηση του ονόματος είναι σχετική με τη θέση και το μέγεθος του αντικειμένου και δεν μπορούσαμε να ορίσουμε μια default θέση για το όνομα, εφόσον κάθε αντικείμενο έχει άλλα χαρακτηριστικά. Έτσι, θα έπρεπε να επέμβουμε στον κώδικα του αντικειμένου για να τοποθετήσουμε εκεί ένα Textfield μέσα στο DEF Transform του. Ωστόσο, αυτό σήμαινε ότι θα έπρεπε να τοποθετηθεί το ίδιο Textfield κάτω από κάθε DEF Transform, για να μπορεί ο χρήστης να επιλέγει οποιονδήποτε κόμβο, γεγονός που πέραν του ότι δεν κατέστη δυνατό, αύξανε την πολυπλοκότητα της εφαρμογής χωρίς ουσιαστικό λόγο. Πέραν τούτου, ακόμα και αν καταφέρναμε να εισάγουμε το όνομα έτσι, θα είχαμε προβλήματα κατά το rename, όπου εκ νέου θα έπρεπε να διαγράψουμε και να ξαναφορτώσουμε τη σκηνή για να τοποθετήσουμε το αντικείμενο με το νέο όνομα, καθυστερώντας τον χρόνο απόκρισης της εφαρμογής για μεγάλους κόσμους.

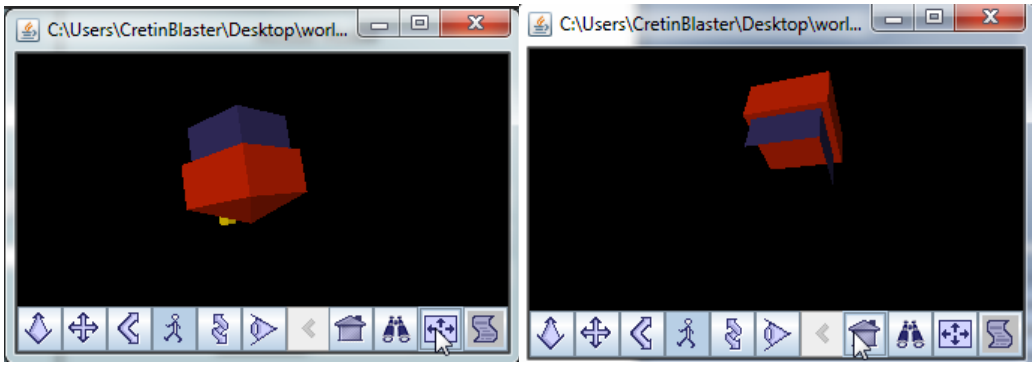
Εν τέλει, σκεφτήκαμε να αξιοποιήσουμε τις υπάρχουσες δυνατότητες που δίνονται μέσα από το virtual model. Ως εκ τούτου το 'Show item' κάνει διακριτό ένα αντικείμενο στην σκηνή, κάνοντάς το scale προς όλους τους άξονες κατά 30% επί του αρχικού μεγέθους, ώστε η εντολή να είναι ανεξάρτητη από το μέγεθος του αντικειμένου και ταυτόχρονα το αντικείμενο να μεγενθύνεται αρκετά ώστε να γίνεται ορατό στον χρήστη. Αυτό επιτυγχάνεται ουσιαστικά αλλάζοντας το scale στο virtual model του επιλεγμένου αντικειμένου για 1", «παγώνοντας» την εφαρμογή μέσω της μεθόδου Thread.sleep() της java. Μετά από αυτό το δευτερόλεπτο, το scale του αντικειμένου επιστρέφει στις τιμές που είχε προηγουμένως. Το ίδιο πράγμα συμβαίνει όταν ο χρήστης επιλέγει με αριστερό κλικ ένα αντικείμενο από τη λίστα, απλά η λειτουργία κρατάει τον μισό χρόνο, 0.5", ώστε να μην καθυστερεί σημαντικά την περιήγηση του χρήστη και τις όποιες εντολές θέλει να εκτελέσει. Παρακάτω ο σχετικός κώδικας, όπου το «it» αναφέρεται στο επιλεγμένο Item.

```
//save item scale values
                  float tempScale[] = new float[3];
                  tempScale[0] = it.getScale()[0];
                  tempScale[1] = it.getScale()[1];
                  tempScale[2] = it.getScale()[2];
//when selected apply a +0.5 scale for 0.5" to show selected item
try {
                    // do something
                    // pause to avoid churning
                    it.setScale(it.getScale()[0]+(it.getScale()[0]*0.3),
                                it.getScale()[1]+(it.getScale()[1]*0.3),
                                it.getScale()[2]+(it.getScale()[2]*0.3));
                     Thread.sleep(500);
                   } catch (InterruptedException ex) {

Logger.getLogger(WorldScene.class.getName()).log(Level.SEVERE, null, ex);
                    }
                  //restore scale before selection
                  it.setScale(tempScale[0], tempScale[1], tempScale[2]);
```

Η επιλογή 'Set item class' δίνει στον χρήστη τη δυνατότητα να αλλάξει την παράμετρο itemClass του item. Να υπενθυμίσουμε πως η συγκεκριμένη παράμετρος λαμβάνει δυο διακριτές τιμές, είτε την real (real.general) είτε την unreal (unreal.general) που χαρακτηρίζουν τον τύπο του αντικειμένου ως προς την αντίληψη του πράκτορα. Ένα αντικείμενο που ορίζεται ως real, είτε είναι υλικό είτε άυλο, είναι αντιληπτό από τον πράκτορα και ορίζεται συγκεκριμένη σημασιολογία για αυτό στα πλαίσια του εικονικού κόσμου και ταυτόχρονα, διατίθεται πληροφορία για τη σημασιολογία αυτή μέσω του semantic aspect του. Ένα αντικείμενο που χαρακτηρίζεται ως unreal δεν είναι ορατό από τον πράκτορα. Χαρακτηρίζει αντικείμενα τα οποία δεν μπορούν να γίνουν αντιληπτά παρά μόνο μέσω των αποτελεσμάτων τους και της επιρροής τους στον εικονικό κόσμο, πχ το φως ή ο χρόνος είναι τέτοια αντικείμενα.

Κατά την δημιουργία του αντικειμένου πήραμε την παραδοχή το item class να ορίζεται ως real. Στην επιλογή του 'Set item class' εμφανίζεται ο αντίθετος τύπος από αυτόν που έχει το αντικείμενο, ώστε πατώντας αυτή την επιλογή ο τύπος να πάρει την αντίθετη από την τρέχουσα τιμή του αντικειμένου. Επομένως, αν ένα item έχει τύπο real τότε στην επιλογή εμφανίζεται 'Set item class: unreal', ενώ αν το item έχει τύπο unreal τότε στην επιλογή εμφανίζεται 'Set item class: real'.

Η επιλογή 'View Object' δείχνει το αντικείμενο που αποτελεί την πηγή του item, δηλαδή το virtual sourcetou, σε ένα ξεχωριστό παράθυρο, να περιστρέφεται γύρω από τον εαυτό του με γωνία προς όλες τις κατευθύνσεις (x,y,z) ώστε ο χρήστης να αποκτά μια περίοπτη και σφαιρική εικόνα του αντικειμένου που έχει προσθέσει στον κόσμο, χωρίς να χρειάζεται να ανατρέξει στο συγκεκριμένο αρχείο από όπου το φόρτωσε. Προϋπόθεση ήταν η δημιουργία μιας νέας σκηνής από τον κόμβο του επιλεγμένου item, όπου θα τοποθετείται το περιστρεφόμενο αντικείμενο στο νέο παράθυρο, άρα πρώτα έπρεπε να αρχικοποιήσουμε την σκηνή με το επιλεγμένο αντικείμενο. Επίσης, για να επιτύχουμε την περιστροφή έπρεπε να δημιουργήσουμε έναν OrientationInterpolator κόμβο, για να ορίσουμε τη γωνία και τους άξονες περιστροφής, όπως και έναν TimeSensor κόμβο, για να ορίσουμε τη χρονική διάρκεια της περιστροφής (σε αυτή την περίπτωση μέχρι ο χρήστης να κλείσει το παράθυρο) και να τους συνδέσουμε με το αντικείμενο που θέλουμε να περιστρέψουμε.

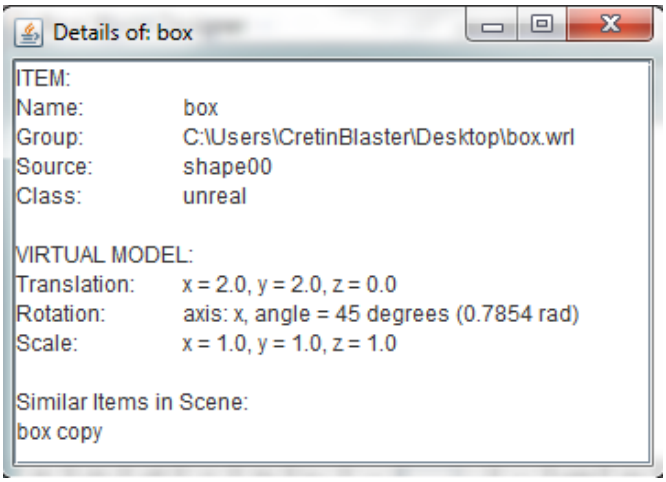


Εικόνα 6.16: Παράθυρο εμφάνισης περιστρεφόμενου αντικειμένου

Εδώ αντιμετωπίσαμε δυο βασικά προβλήματα. Το πρώτο είναι πως δεν αρκούσε, όπως στη δημιουργία του item, μια απλή προσθήκη Transform κόμβου πάνω από τον DEF κόμβο του αντικειμένου για να περιστρέψουμε το νέο εξωτερικό Transform που θα περιλάμβανε το αντικείμενο. Μια τέτοια προσέγγιση οδηγούσε το αντικείμενο να περιστρέφεται με σημείο αναφοράς το transformation που οριζόταν στον DEF κόμβο και όχι γύρω από τον εαυτό του, δηλαδή να κάνει έναν κύκλο στο χώρο και όχι μια περιστροφή. Η προσέγγιση που ακολουθήσαμε για να διορθώσουμε αυτό το πρόβλημα ήταν να πάρουμε ένα προς ένα τα παιδιά (children) του DEF κόμβου και να τοποθετήσουμε αυτά κάτω από τον νέο Transform κόμβο, οπότε να εξαλείψουμε τον ίδιο τον DEF κόμβο με το transformation του, μεταφέροντας την υπόλοιπη δομή κάτω από τον νέο Transform κόμβο, ώστε η περιστροφή να εφαρμοστεί στο σύνολο του αντικειμένου. Επρόκειτο ουσιαστικά για μια επανασύνθεση του DEF κόμβου υπό την νέα Transform ρίζα, χωρίς όμως το transformation του DEF κόμβου. Επίσης, έπρεπε και τον νέο Transform κόμβο να τον εισάγουμε κάτω από έναν Group κόμβο, όπως ορίζει η VRML, διαφορετικά η περιστροφή δεν ήταν δυνατή.

Το δεύτερο πρόβλημα ήταν πως θέλαμε το αντικείμενο να καταλαμβάνει όλο το χώρο του νέου παραθύρου, στην ουσία να ενεργοποιήσουμε αυτόματα την λειτουργικότητα του 'Fit to World' του xj3d browser. Ωστόσο, δεν βρήκαμε σχετική εντολή από το tutorial που να μας δίνει αυτή την δυνατότητα, ενώ δεν μπορούσαμε να εφαρμόσουμε ένα default transformation για να προσομοιώσουμε αυτή την λειτουργία, εφόσον όπως και στην περίπτωση προσθήκης του item, το αντικείμενο δεν εμφανίζεται στην αρχή των αξόνων αλλα η θέση του είναι σχετική με το εσωτερικό translation του. Αυτό εχει ως αποτελέσμα αντικείμενα με μεγάλα translations να είναι εκτός της οπτικής γωνίας του κόσμου όταν ανοίγει το παράθυρο και ο χρήστης να πρέπει να πατήσει το κουμπί 'Fit to World' για να δει ένα αντικείμενο να περιστρέφεται. Θεωρούμε, όμως, πως εφαρμόζοντας και εδώ τον κώδικα του κ. Αναστασάκη για την ευθυγράμμιση του αντικειμένου στην αρχή των αξόνων, το πρόβλημα θα λυθεί.

Τέλος, η επιλογή 'Item Properties' από το μενού, εμφανίζει ένα καινούριο μικρό παράθυρο με βασικές πληροφορίες για το συγκεκριμένο item, ώστε ο χρήστης να έχει μια πλήρη γνώση για τα στοιχεία του item.



Εικόνα 6.17: ItemPropertiesWindow

Οι πληροφορίες κατηγοριοποιούνται σε τρία διακριτά σύνολα. Το πρώτο αναγράφει:

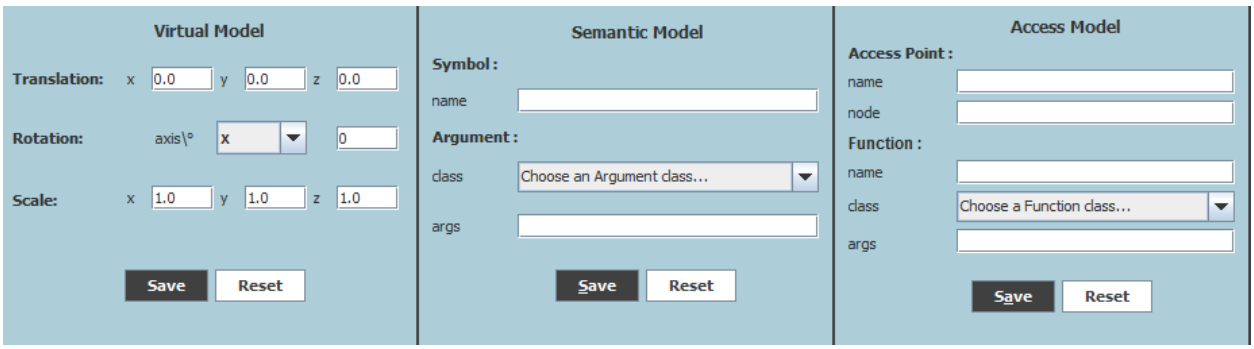
- το όνομα του item, το itemGroup του, δηλαδή την πλήρη τοποθεσία του αρχείου από όπου φορτώθηκε το item
- το όνομα του virtual source του, δηλαδή του DEF κόμβου από τον οποίο δημιουργήθηκε και
- τον τύπο του, δηλαδή real ή unreal.

Το δεύτερο σύνολο περιλαμβάνει τις τιμές του virtual model (translation, rotation, scale) με τη γωνία του rotation να εμφανίζεται τόσο σε μοίρες όσο και σε rad, καθώς και τις σχετικές τιμές του semantic model και access model, αν ορίζονται.

Το τρίτο σύνολο εμφανίζει μια λίστα ονομάτων των items που έχουν τον ίδιο DEFκόμβο ως πηγή, δηλαδή είτε είναι αντίγραφα αυτού του item είτε έχουν προέλθει γενικά από το ίδιο αντικείμενο. Θεωρούμε ότι αυτή η πληροφορία είναι χρήσιμη κυρίως για το αν θέλει ο χρήστης να χειριστεί παρόμοια αντικείμενα με τον ίδιο τρόπο, πχ να σβήσει όλα τα αντικείμενα που έχουν το ίδιο σχήμα σε μια σκηνή. Επίσης, δίνει την δυνατότητα στον χρήστη να βλέπει παράλληλα τα στοιχεία ενός item ενώ επεξεργάζεται ένα άλλο.

6.5 Τα models του Item

Συνεχίζοντας με τις υπόλοιπες δυνατότητες της εφαρμογής θα εστιάσουμε στην δημιουργία των virtual, access και semantic model που ορίζει η verl, όπως παρουσιάζονται στην παρακάτω εικόνα.



Εικόνα 6.18: UI των virtual, semantic και access model

Το κάθε μοντέλο κινείται σε 2 επίπεδα στην εφαρμογή. Το ένα είναι η εμφάνιση των τιμών που έχει ένα item στα αντίστοιχα πεδία των models του στο UI, όπου τραβάμε τις τιμές με get μεθόδους. Το δεύτερο είναι η εισαγωγή των τιμών που ορίζει χρήστη από το UI στα σχετικά πεδία των models του item, που επιτυγχάνεται με set μεθόδους. Επομένως, όταν επιλέγουμε ένα αντικείμενο από τη λίστα ονομάτων, αυτόματα γεμίζουν τα πεδία των models του με τις αντίστοιχες τιμές. Και όταν αλλάζουμε τις τιμές των πεδίων ενός μοντέλου από το UI και επιλέγουμε Save, αυτές οι τιμές ενημερώνουν τα σχετικά πεδία του item. Από την άλλη, το Reset επαναφέρει το item στις default τιμές για το συγκεκριμένο μοντέλο. Για το semantic και το accessmodel, οι default τιμές αντιστοιχούν σε κενά πεδία, που συνεπάγετα ότι το item δεν έχει semantickαι acces model. Στο virtual model οι default τιμές είναι αυτές που φαίνονται στον παρακάτω πίνακα:

MODEL	ACTION	AXIS			ANGLE
		x	y	z	rad
Virtual Model	Translation	0	0	0	N/A
	Rotation	1	0	0	0
	Scale	1	1	1	N/A

Εικόνα 6.19: UI των virtual, semantic και access model

Το γεγονός ότι έχουμε ένα Savekαι Reset κουμπί για κάθε μοντέλο, οφείλεται τόσο στο ότι αναπαριστούν διαφορετική λειουργικότητα για κάθε μοντέλο, όσο στο ότι θέλαμε να διατηρήσουμε την αυτονομία κάθε μοντέλου ως ξεχωριστή διασταση του item, όπως ορίζει και η αναπαράσταση REVE. Επομένως, για να αλλάξουμε τις τιμές ενός μοντέλου επιβάλλεται να πατήσουμε το Save του συγκεκριμένου μοντέλου.Επίσης, τα κουμπιά Save και Reset επιδρούν μονάχα στο επιλεγμένο item από τη λίστα, ενώ σε περίπτωση που δεν είναι επιλεγμένο κάποιο item, τα κουμπιά είναι απενεργοποιημένα, αποφεύγοντας σκασίματα λόγω null pointer exceptions της εφαρμογής. Παρακάτω εξετάζουμε το κάθε μοντέλο του item ξεχωριστά.

6.5.1. Το Virtual Model

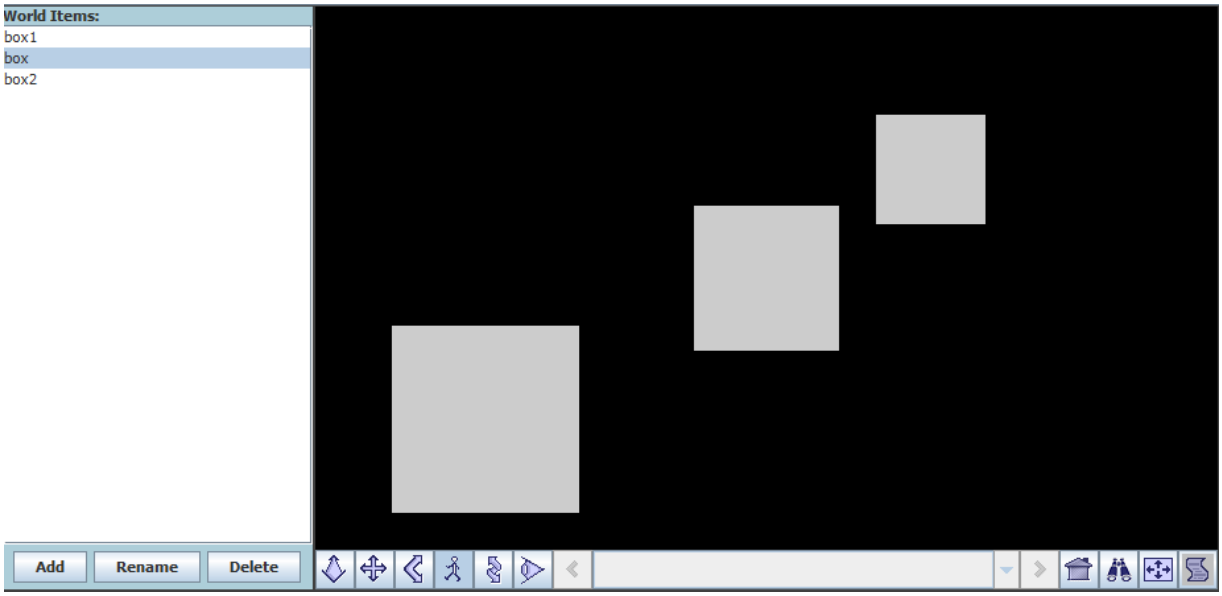
Το virtual model επιδρά στο physical aspect του αντικειμένου στον κόσμο. Το physical aspect αφορά στοιχεία όπως η γεωμετρία, η δομή, οι διαστάσεις και το υλικό του αντικειμένου, δηλαδή στοιχεία αναπαράστασης του αντικειμένου, καθώς και τη θέση του στο χώρο, το μέγεθός του και τον προσανατολισμό του. Εκμεταλλευόμενοι το γράφημα σκηνής που αναπαριστά ένα αντικείμενο, ορίσαμε έναν εξωτερικό Transform κόμβο-ρίζα κατά τη δημιουργία ενός item για να μπορέσουμε να παρέμβουμε στην μετατόπιση, την περιστροφή και την αλλαγή κλιμακας ενός αντικειμένου, στοιχεία που αφορούν το Virtual Model. Ως εκ τούτου, μέσα από το UI, στο Jpanel του Virtual Model εμφανίζονται οι τιμές του translation, rotation και scale ενός αντικειμένου και δίνεται η δυνατότητα στον χρήστη να αλλάζει τις τιμές αυτές βάσει κάποιου άξονα.

Για το translation, ο χρήστης μπορεί να ορίσει μετατόπιση, σε μέτρα, κατά μήκος ενός άξονα, εισάγοντας έναν ακέραιο ή δεκαδικό αριθμό. Αυτό γίνεται αλλάζοντας τις τιμές του SFVec3f πεδίουτου εξωτερικού

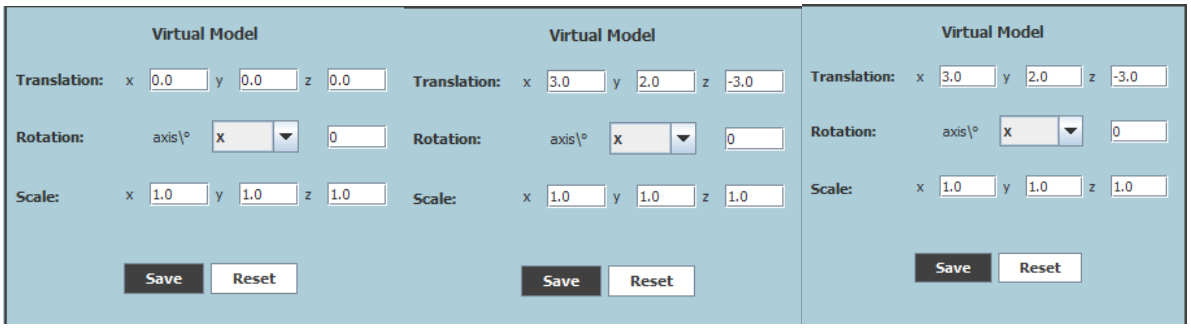
Transform που δημιουργήσαμε κατά τον ορισμό του item, όπως φαίνεται στον παρακάτω κώδικα της κλάσης Item.java:

```
//Given a number per axis, set the new translation of the item
    public void setTranslation(double x, double y, double z){
        translation = (SFVec3f)transform.getField("translation");
        translate[0]=(float) x;
        translate[1]=(float) y;
        translate[2]=(float) z;
        translation.setValue(translate);
}
```

Βλέποντας το current Viewpoint του κόσμου, αν εισάγουμε έναν θετικό αριθμό στον άξονα x, το αντικείμενο μετακινείται προς τα δεξιά, ενώ αν εισάγουμε έναν αρνητικό αριθμό το αντικείμενο μετακινείται προς τα αριστερά. Αν εισάγουμε έναν θετικό αριθμός στον άξονα y το αντικείμενο κινείται προς τα πάνω, ενώ αν εισάγουμε έναν αρνητικό αριθμό το αντικείμενο μετακινείται προς τα κάτω. Τέλος, αν εισάγουμε έναν θετικό αριθμό στον άξονα z, που αφορά την προοπτική, το αντικείμενο έρχεται προς τα εμπρός, σαν να «πλησιάζει» τον χρήστη ενώ αν εισάγουμε μια αρνητική τιμή το αντικείμενο πηγαίνει προς τα πίσω, σαν να απομακρύνεται από τον χρήστη. Εισάγοντας τιμές σε όλους τους άξονες, η κίνηση του αντικειμένου είναι συνδυαστική. Στην παρακάτω εικόνα φαίνεται ένα τέτοιο παράδειγμα, όπου το boxείναι το αρχικό αντικείμενο με μηδενικό translation και τα box1 και box2 είναι αντίγραφά του με διαφορετικά translations.

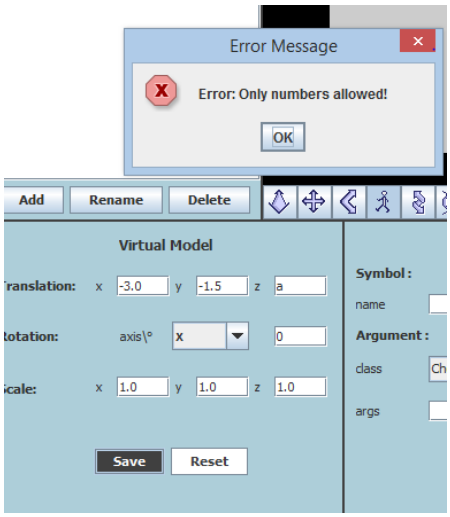


Εικόνα 6.20: box1 (πάνω δεξια), box (κέντρο), box2 (κάτω αριστερά)



Εικόνα 6.21: VM box (αριστερά), VM box1 (κέντρο), VM box2 (δεξιά)

Σε περίπτωση εισαγωγής μη έγκυρων τιμών, πχ αλφαριθμητικού αντί για αριθμό, ο χρήστης προειδοποιείται ανάλογα κατά το Save και η τιμή του translation παραμένει ίδια με αυτήν που είχε το item προηγουμένως.



Εικόνα 6.22: Προειδοποίηση για μη έγκυρες τιμές

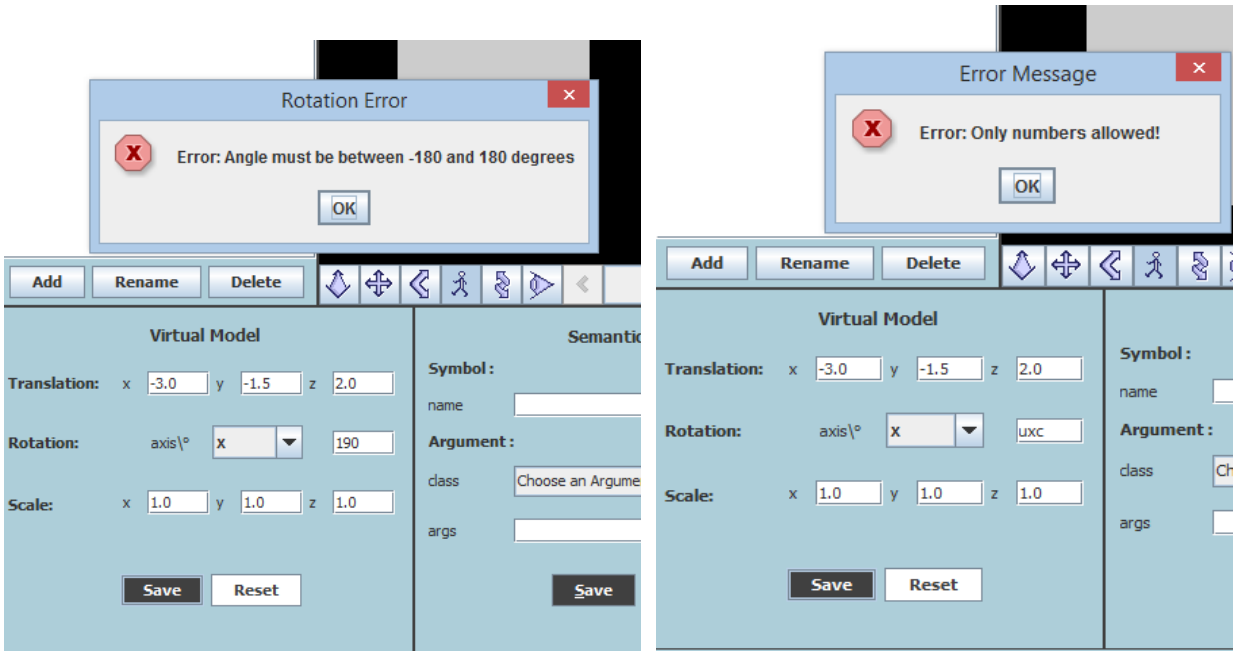
Τα βασικά θέματα του translation είναι δυο. Το πρώτο είναι πως η μετατόπιση του αντικειμένου είναι σχετική με την θέση του αντικειμένου και όχι την αρχή των αξόνων, εφόσον το αντικείμενο δεν ευθυγραμμίζεται με την αρχή των αξόνων όπως δηλώθηκε προηγουμένως. Επομένως, η μετατόπιση του αντικειμένου γίνεται με βάση το τοπικό του σύστημα συντενταγμένων και όχι το global σύστημα συντεταγμένων του κόσμου. Αυτό έχει ως αποτέλεσμα δυο αντικείμενα που μπορεί να έχουν ίδιες τιμές στο translation, να μην συμπίπτουν στο χώρο γιατί έχουν διαφορετικό σημείο αναφοράς λόγω του τοπικού συστήματος συντεταγμένων τους, εφόσον στον Def κόμβο τους ορίζεται μη-μηδενικό translation. Το δεύτερο θέμα είναι πως η μετατόπιση του αντικειμένου, λόγω του μαύρου φόντου της σκηνής, δεν κάνει διακριτή την κίνηση στους άξονες, δηλαδή προτιμούσαμε να ορίσουμε ένα πλέγμα κατά μήκος των αξόνων (x,yz,) όπως συμβαίνει στην Reve, αλλά αυτό δεν κατέστη δυνατό με την τρέχουσα υλοποίηση, καθώς δεν βρήκαμε τρόπο να αλλάξουμε την υπάρχουσα εμφάνιση της browser μέσω του xj3d tutorial.

Η δεύτερη επιλογή, το rotation, επιτρέπει στον χρήστη να επιλέξει έναν άξονα περιστροφής και να ορίσει την γωνία περιστροφής σε μοίρες. Προτιμήσαμε τις μοίρες καθώς θεωρούμε πως με αυτό το σύστημα μέτρησης ο χρήστης είναι πιο εξοικειωμένος. Αυτό βέβαια συνεπάγεται πως οι μοίρες μετατρέπονται σε rad πριν ενημερώσουν τα πεδία του rotation, ώστε να γίνει σωστά η περιστροφή, εφόσον σε rad υπολογίζεται. Αντίστροφα, κατά την επιλογή ενός item τα rad μετατρέπονται σε μοίρες για να εμφανιστούν σωστά στο UI. Επίσης, δεδομένου πως αλλιώς ενημερώνεται το rotation μέσω του UI κι αλλιώς μέσω άλλου item (πχ στην

περίπτωση δημιουργίας αντιγράφου), έπρεπε να φτιάξουμε δυο μεθόδους που ενημερώνουν το πεδίο SFRotation του εξωτερικού Transform κόμβου του item, όπως φαίνεται στον παρακάτω κώδικα:

```
//Given an axis and an angle in rad, set the item's rotation
     public void setRotation(String axis, double angle){
        this.axis = axis;
        this.angle = angle;
        rotation = (SFRotation)transform.getField("rotation");
        switch(axis){
            case "x" :
                rotate[0]= 1;
                rotate[1]= 0;
                rotate[2]= 0;
                break;
            case "y" :
                rotate[0]= 0;
                rotate[1]= 1;
                rotate[2]= 0;
                break;
            case "z":
                rotate[0]= 0;
                rotate[1]= 0;
                rotate[2]= 1;
                break;
        }
        rotate[3]=(float) angle;
        rotation.setValue(rotate);
    }

     //Used to set rotation by getting the values from another item or
loading world
     public void setRotation(float r1, float r2, float r3, double angle){
         rotate[0] = r1;
         rotate[1] = r2;
         rotate[2] = r3;
rotate[3] = (float) angle;

         if(rotate[0] == 1)
             this.axis = "x";
         if(rotate[1] == 1)
             this.axis = "y";
         if(rotate[2] == 1)
             this.axis = "z";

         rotation = (SFRotation)transform.getField("rotation");
         rotation.setValue(rotate);

         setRotation(this.axis, rotate[3]);
}
```

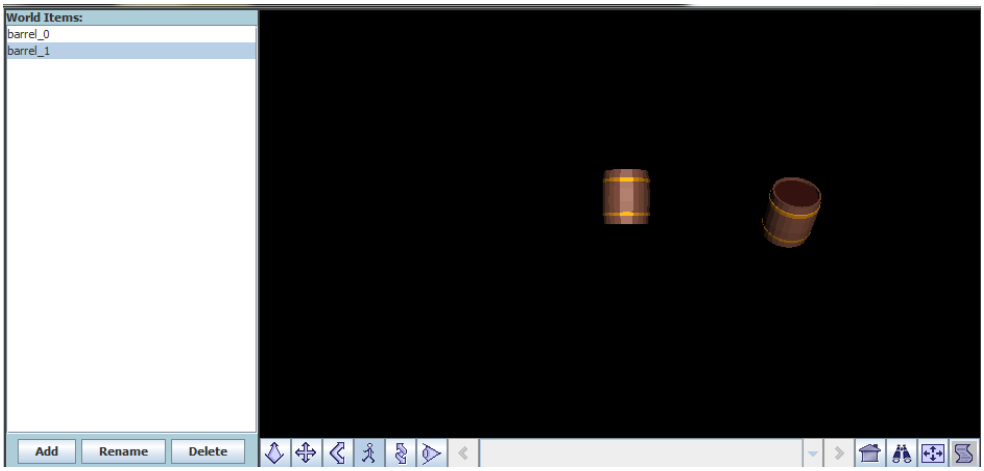
Οι επιτρεπτές τιμές της γωνίας περιστροφής είναι ακέραιοι και δεκαδικοί για το εύρος τιμών [-180,180] μοίρες οπότε θεωρείται πως καλύπτεται μια πλήρης περιστροφή. Σε διαφορετική περίπτωση ο χρήστης προειδοποιείται με σχετικό μήνυμα.

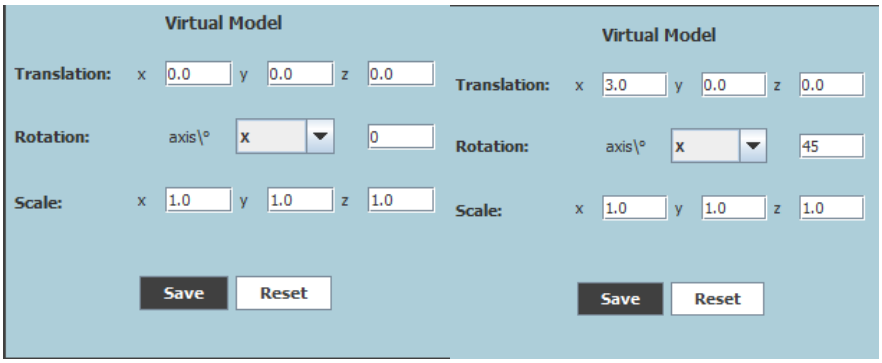


Εικόνα 6.23: Προειδοποιήσεις για μη έγκυρες τιμές στο rotation

Αντίστοιχα με το translation, η περιστροφή είναι σχετική με την περιστροφή που έχει ήδη ένα αντικείμενο στον DEF κόμβο του, λόγω μη ευθυγράμμισης στην αρχή των αξόνων, όπως συμβαίνει στην Reve. Σε περίπτωση που ένα αντικείμενο δεν έχει εσωτερική περιστροφή, τότε η περιστροφή μέσω του UI απεικονίζεται σωστά, δηλαδή οι μοίρες που εισάγει ο χρήστης είναι πράγματι η γωνία κατά την οποία έχει περιστραφεί το αντικείμενο. Αν, όμως, ο DEF κόμβος του αντικειμένου έχει ήδη περιστροφή, τότε η περιστροφή μέσω του UI έχει ως σύστημα αναφοράς την εσωτερική περιστροφή του, οπότε για παράδειγμα, μια περιστροφή κατά 45 μοίρες μπορεί στην πραγματικότητα να είναι μια περιστροφή 50 μοιρών, αν το αντικείμενο έχει ήδη εσωτερική περιστροφή 5 μοιρών στον ίδιο άξονα στον DEF κόμβο.

Στην παρακάτω εικόνα παρουσιάζουμε δυο βαρέλια, όπου το ένα είναι αντίγραφο του άλλου, περιστρεφόμενο κατα 45 μοίρες στον άξονα x.





Εικόνα 6.24: Περιστρεφόμενο βαρέλι κατά 45 μοίρες (δεξιά)

Η τελευταία επιλογή, το scale, επιτρέπει στον χρήστη να αλλάξει το μέγεθος ενός αντικειμένου κατά κάποιον άξονα. Όπως προηγούμενως, από τον εξωτερικό Transform κόμβο του item, παίρνουμε το πεδίο SFVec3F για να του ορίσουμε τις τιμές του scale που εισάγει ο χρήστης, βάσει του παρακάτω κώδικα:

```
//Given a number per axis set scale for item
     public void setScale(double x, double y, double z){
        scale = (SFVec3f)transform.getField("scale");
        scaling[0]=(float) x;
        scaling[1]=(float) y;
        scaling[2]=(float) z;
        scale.setValue(scaling);
}
```

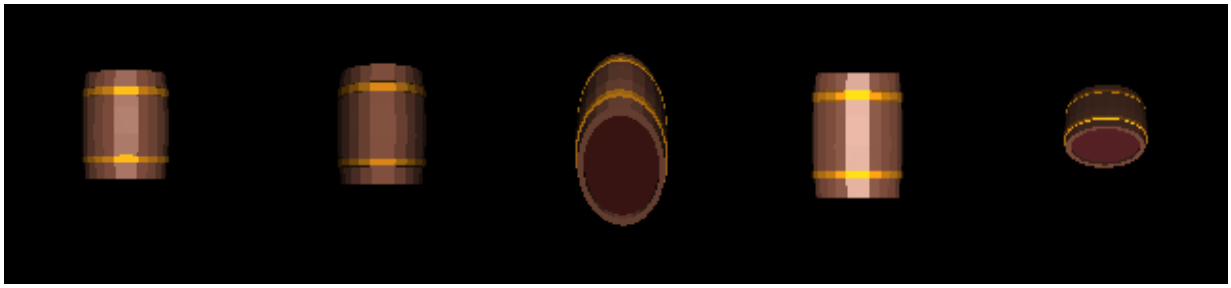
Βάζοντας τιμή μεγαλύτερη του 1 στον άξονα x, το πλάτος του αντικειμένου αυξάνει, ενώ βάζοντας τιμή μικρότερη του 1 το πλάτος του αντικειμένου μειώνεται (Εικόν 6.25).

Εικόνα 6.25: Αρχικό αντικείμενο (αριστερά), scale κατά 2 στον x (κέντρο), scale 0.5 στον άξονα x (δεξιά)

Βάζοντας τιμή μεγαλύτερη του 1 στον άξονα y, το ύψος του αντικειμένου αυξάνει, ενώ βάζοντας τιμή μικρότερη του 1 το ύψος του αντικειμένου μειώνεται (Εικόνα 6.26).

Εικόνα 6.26: Αρχικό αντικείμενο (αριστερά), scale 2 στον άξονα y (κέντρο), scale 0.5 στον y (δεξιά)

Βάζοντας τιμή μεγαλύτερη του 1 στον άξονα z, το μήκος του αντικειμένου αυξάνει, ενώ βάζοντας τιμή μικρότερη του z, το μήκος του αντικειμένου μειώνεται (Εικόνα 6.27).

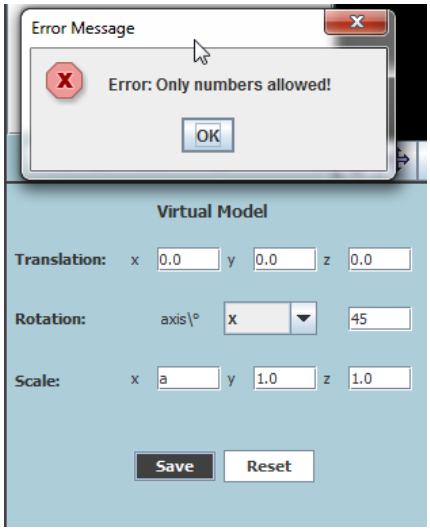


Εικόνα 6.27: Αρχικό αντικείμενο (αριστερά), scale κατά 2 στον άξονα z (2η & 3η), scale κατά 0.5 στον άξονα z (4η & 5η)

Αν εισάγουμε τις ίδιες τιμές σε όλους τους άξονες του scale τότε για τιμές μεγαλύτερες του 1 μπορούμε να μεγενθύνουμε το αντικείμενο ομοιόμορφα, ενώ για τιμές μικρότερες του 1 μπορούμε να μικρύνουμε ομοιόμορφα το αντικείμενο (Εικόνα 6.28).

Εικόνα 6.28: Αρχικό αντικείμενο (αριστερά), μεγέθυνση κατά 2.0 (κέντρο), σμίκρυνση κατά 2.0 (δεξιά)

Αν ο χρήστης δεν εισάγει αριθμό στις τιμές του scale,προειδοποιείται με σχετικό μήνυμα κατά το πάτημα του Save.



Εικόνα 6.29: Μήνυμα για μη έγκυρες τιμές

Σε αντίθεση με το translation και το rotation, αν το αντικείμενο έχει ήδη scale στον DEF κόμβο του, η αλλαγή του scale στο virtual model αναπαριστά την πραγματική αλλαγή κλίμακας, εφόσον το πραγματικό μέγεθος του αντικειμένου είναι αυτό που βλέπουμε.

Τέλος, για διευκόλυνση του χρήστη ώστε να αποφεύγει το συνεχές πήγαινε-έλα με το ποντίκι, η ενεργοποίηση του Save κουμπιού του Virtual Model μπορεί να γίνει και με το πάτημα συνδυαστικά του Alt+Enter από το πληκτρολόγιο. Από την άλλη, το Reset όπως αναφέρθηκε επαναφέρει τις default τιμές του Virtual Model στο item, όπως φαίνεται στον παρακάτω κώδικα:

```
//Set default values of an item's virtual model
public void setDefaultVirtualModel(){
        setTranslation(0, 0, 0);
        setRotation("x", 0);
        setScale(1, 1, 1);
    }
```

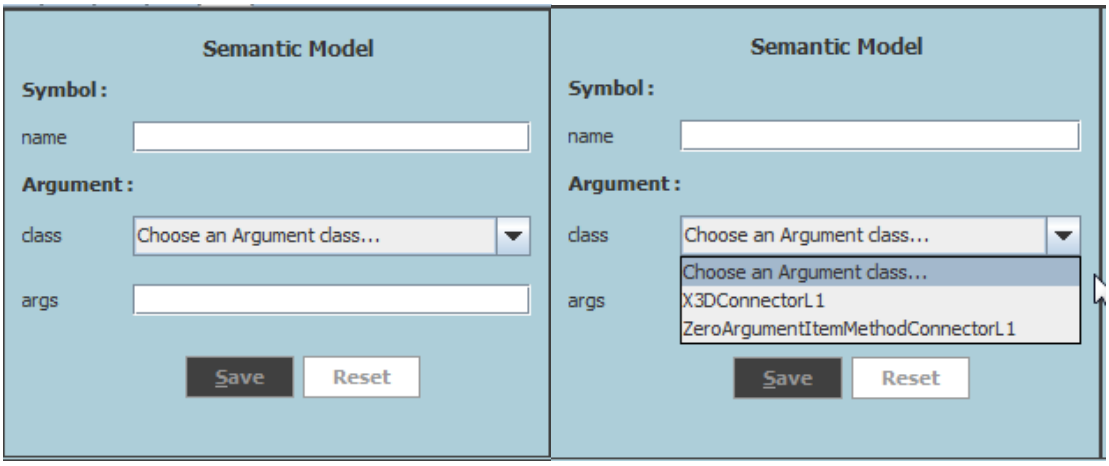
6.5.2. Το Semantic Model

Το semantic model αφορά στο semantic aspect ενός αντικειμένου, δηλαδή την σημασιολογική του αναπαράσταση στον κόσμο. Παρέχει τόσο γενική πληροφορία για την υπόσταση του αντικειμένου στον κόσμο όσο και μια εννοιολογική γέφυρα αλληλεπίδρασης με ένα item, ορίζοντας έννοιες που περιγράφουν το αντικείμενο σε ένα αφαιρετικό επίπεδο.

Όπως χαρακτηριστικά περιγράφεται στην διδακτορική διατριβή του κ. Αναστασάκη «*το semantic aspect παρέχει πληροφορία και για έναν αριθμό χαρακτηριστικών του item που αφορούν μία προσεγγιστική, αφαιρετική άποψη της ενσώματης υπόστασής του στον εικονικό κόσμο. Τα χαρακτηριστικά αυτά είναι η θέση, ο προσανατολισμόςκαι οι διαστάσεις για δεδομένης μορφής σχήμα που περικλείει το αντικείμενο. Η χρήση των τριών αυτών χαρακτηριστικών είναι ιδιαίτερα σημαντική για μεγάλο εύρος εφαρμογών, αν όχι στο σύνολό τους. Αρκεί να αναφερθούν τομείς όπως η ανίχνευση συγκρούσεων (collision detection), ή εύρεση διαδρομών (path finding), η αυτόματη πλοήγηση, κ.α., όπου η προσεγγιστική αναπαράσταση του χώρου που καταλαμβάνει, κατά κύριο λόγο, ένα συγκεκριμένο αντικείμενο, μέσω της έννοιας του περικλείοντος σχήματος, είναι ιδιαίτερης σημασίας, τόσο σε επίπεδο σημασιολογίας όσο και απόδοσης σε επίπεδο λογισμικού*».

Πέραν, όμως, της παραπάνω πληροφορίας για το bounding box ενός αντικειμένου, το semantic aspect παρέχει πληροφορία σχετικά με τη διαθέσιμη λειτουργικότητα ενός item, αναπαριστώντας σε αφηρημένο επίπεδο, μέσω συμβόλων, δυνατότητες αλληλεπίδρασης με εικονικά αντικείμενα στα πλαίσια εικονικών κόσμων.Το semantic model της εφαρμογής σχεδιάστηκε για να καταφέρει να εξυπηρετήσει αυτόν τον σκοπό.

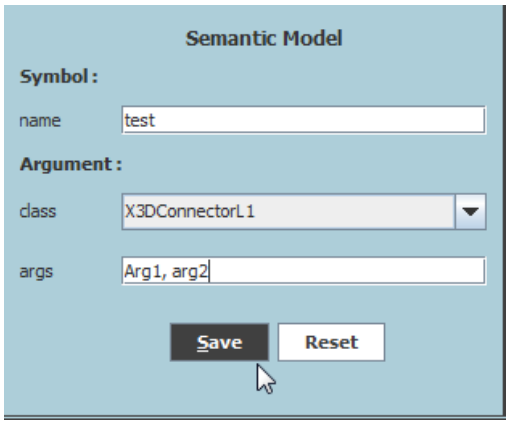
Όπως είχαμε περιγράψει και στο Κεφάλαιο 3, το semantic model ορίζεται από symbols και arguments που έχουν ως ορίσματα μια class και κάποια args.Η βασική δομή, λοιπόν, σχηματοποιηθήκε στο UI για ένα symbol και ένα argument πάνω στο item,όπως φαίνεται στην παρακάτω εικόνα και στον παρακάτω κώδικα:



Εικόνα 6.30: UI αναπαράσταση με τα βασικά στοιχεία του SemanticModel

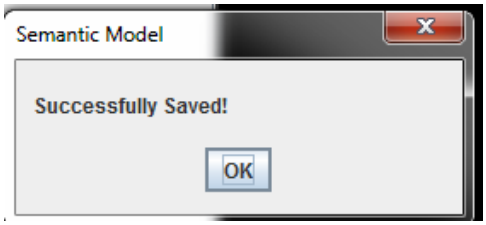
```
//set value for symbol's attribute name
     public void setSymbolName(String symbolName){
         this.symbolName = symbolName;
     }

     //set value for argument's attribute class
     public void setArgumentClass(String argumentClass){
this.argumentClass = argumentClass;
     }

     //set value for argument's attribute args
     public void setArgumentArgs(String args){
this.argumentArgs = args;
```

Ο χρήστης, αφού έχει επιλέξει ένα item, μπορεί να εισάγει το όνομα του symbol του καθώς και ένα ή παραπάνω args (χωρισμένα με κόμματα) και να τα αποθηκεύσει στο item πατώντας Save.



Εικόνα 6.31: Εισαγωγή τιμών στο Semantic Model

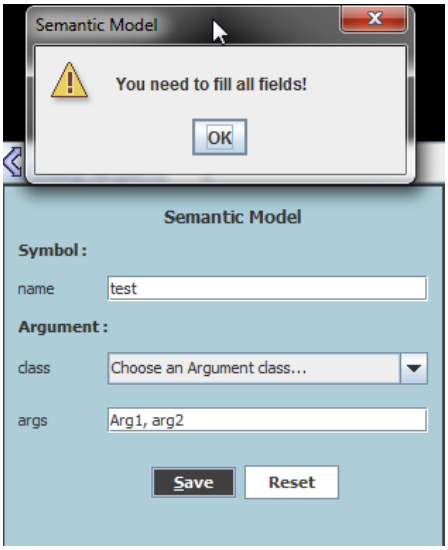
Στην πραγματικότητα, αν και το όνομα του symbol είναι στην ευχέρεια του χρήστη, τα ορίσματα του args πρέπει να σχετίζονται με την επιλεγμένη class, αλλιώς η σημασιολογία χάνει το νόημά της και η πλατφόρμα Reve πετάει erro rπως δεν μπορεί να συνδέσει την σημασιολογία με το αντικείμενο. Για παράδειγμα, για την X3DConnectorL1 class τα args ορίσματα πρέπει να αναφέρονται σε έναν DEFκόμβο ή πεδίο που σχετίζεται με κάποιον InterPolator κόμβο και ένα πεδίο του, καθώς και στη δυνατότητα μεταβολής τους. Δεδομένου ότι δεν υπάρχει τρόπος να επαληθευτεί μέσα από την τρέχουσα εφαρμογή η λειτουργικότητα του semantic modelκαι ούτε αποτελεί σκοπό αυτής της μεταπτυχιακής διατριβής, ο χρήστης αντιλαμβάνεται πως το semantic modelπου έχει ορίσει έχει αποθηκευτεί στο item με σχετικό μήνυμα.



Εικόνα 6.32: Ειδοποίηση χρήστη για επιτυχής αλλαγή του Semantic Model του item

Από την άλλη, αν ο χρήστης πάει να αποθηκεύσει ένα Semantic Model στο οποίο δεν έχει ορίσει τιμές σε όλα τα πεδία, του εμφανίζεται σχετικό μήνυμα λάθους ώστε να δώσει τιμές σε όλα τα πεδία, όπως επιβάλλει η σύνταξη της verl. Τέλος, όσον αφορά το Reset button, η ενεργοποίησή του αφαιρεί το semantic model από το επιλεγμένο item.

```
//Set default values for item's semantic model
   public void setDefaultSemanticModel(){
        setSymbolName("");
        setArgumentClass("");
setArgumentArgs("");
    }
```



Εικόνα 6.33: Προειδοποίηση για κενά πεδία

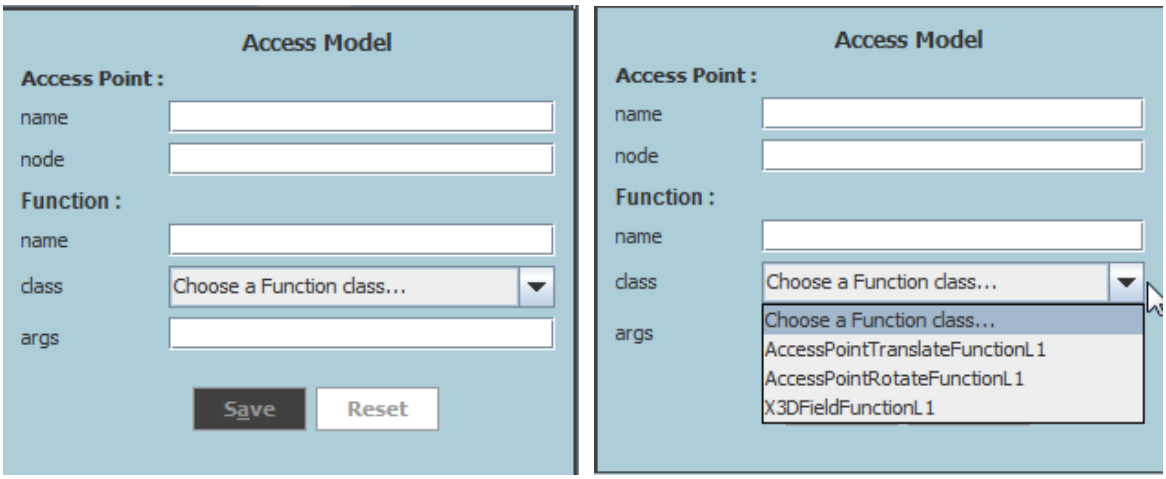
Όπως γίνεται αντιληπτό, το Semantic Model στα πλαίσια αυτής της μεταπτυχιακής διατριβής δίνει στον χρήστη τη δυνατότητα να μπορέσει να ορίσει ένα semantic model για ένα item, παρουσιάζοντάς του μια χρήσιμη δομή χωρίς να εντρυφεί στη ορθότητα του ορισμού του. Επομένως, εδώ δεν μας ενδιαφέρει να αν ο χρήστης βάζει τιμές που έχουν νόημα στο Semantic Model, καθώς αυτό ελέγχεται στην πλατφόρμα Reve, αλλά μας ενδιαφέρει μόνο η ορθότητα της σύνταξης, πχ εξάλειψη κενών τιμών. Επίσης, η μορφή του ορισμού προσφέρει μονάχα ένα πρώτο στάδιο, καθώς ένα item μπορεί να έχει ένα ή παραπάνω symbols και ένα symbol να έχει ένα ή παραπάνω arguments, που δεν καλύπτεται στα πλαίσια αυτής της μεταπτυχιακής διατριβής αλλά μπορεί να επεκταθεί, δίνοντας την δυνατότητα εμπλουτισμού της εφαρμογής.

6.5.3. Το Access Model

Το access model αφορά στο access aspect του αντικειμένου, δηλαδή στην αναπαράσταση ενδεχόμενης αλληλεπίδρασης με το item, ορίζοντας έναν ή παραπάνω κόμβους αλληλεπίδρασής και έναν ή παραπάνω τρόπους αλληλεπίδρασης με αυτούς. Το access model,λοιπόν, αποτελείται από access points και functions.

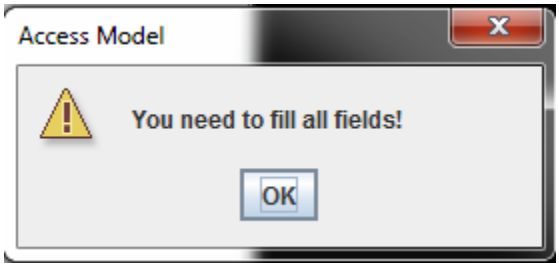
Το access point name λαμβάνει ένα όνομα που ορίζει ο χρήστης και το node ορίζει τον κόμβο του item που θα υποστεί κάποια μεταβολή, καθιστώντας τον εντοπίσιμο στην δομή του αντικειμένου, δηλαδή πρέπει να παίρνει το DEF όνομα αυτού του κόμβου. Ο τρόπος αλληλεπίδρασης με αυτό το accessp oint ορίζεται μέσω μιας function. Το όνομα της είναι στην ευχέρεια του χρήστη, όμως η class αφορά τον τύπο μεταβολής που θα λάβει χώρα και μπορεί να είναι είτε translation (AccessPointTranslateFunctionL1), είτε rotation (AccessPointRotateFunctionL1) ή κάποια άλλη ενέργεια που ορίζεται στη δομή του αντικειμένου (X3DFieldFunctionL1), ενώ τα args είναι πιθανά ορίσματα της συνάρτησης που μπορεί να παραμένουν και κενά.

Η λογική της σχεδίασης και υλοποίησης του access model είναι αντίστοιχη με αυτή του semantic model. Μέσω του UI, δηλαδή, δίνεται η δυνατότητα στον χρήστη να ορίσει ένα (μόνο) access point και function επί αυτού, διατηρώντας την ορθότητα της σύνταξης της verl χωρίς, όμως, να ελέγχονται τυχόν λογικά λάθη στα ορίσματα, παρά μόνο κενές τιμές που οδηγούν σε λάθος σύνταξη του κόσμου.

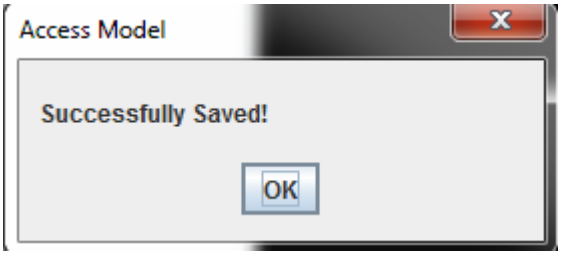


Εικόνα 6.34: UI αναπαράσταση με τα βασικά στοιχεία του Access Model

Λογικά λάθη προκύπτουν και εδώ αν ο χρήστης ορίσει ένα accessPoint για κόμβο που δεν υπάρχει στην δομή του αντικειμένου ή αν ορίσει accessPoint που ναι μεν υπάρχει αλλά ο τρόπος αλληλεπίδρασης είναι λανθασμένος. Πιο συγκεκριμένα, για να μην προκύψουν λάθη, ο χρήστης μπορεί να εισάγει στο access model, στην τιμή node του access point, το όνομα ενός υπαρκτού DEF Transform κόμβου του επιλεγμένου αντικειμένου. Στη συνέχεια, μπορεί να χρησιμοποιήσει είτε την AccessPointTranslateFunctionL1 είτε την AccessPointRotateFunctionL1 με κενά ορίσματα (args), εφόσον κάθε transform κόμβος μπορεί να υποστεί μετατόπιση ή περιστροφή. Αν, όμως, πάει να θέσει την X3DFieldFunctionL1, θα πρέπει να έχει βεβαιωθεί ότι αυτός ο κόμβος επιδρά με κάποιον Interpolator κόμβο και στα ορίσματα να θέσει τυχόν πεδία που είναι απαραίτητα, αντίστοιχα με το semantic model, αλλιώς θα προκύψει λογικό σφάλμα στην Reve. Όσον αφορά τα κενά ορίσματα ο χρήστης προειδοποιείται σχετικά, όπως φαίνεται στην Εικόνα 6.32, ενώ η επιτυχής αποθήκευση του μοντέλου στο επιλεγμένο item εμφανίζει σχετικό μήνυμα στον χρήστη κατά το πάτημα του Save (Εικόνα 6.33).



Εικόνα 6.35: Προειδοποίηση για κενά ορίσματα



Εικόνα 6.36: Μήνυμα επιτυχούς αποθήκευσης access model

Το Reset Button, αντίστοιχα με το Reset του semantic model αδειάζει το μοντέλο από τιμές, στην ουσία αφαιρώντας το από το item. Αυτό βέβαια ισχύει γιατί το access modelεπιτρέπει προς στιγμήν μόνο ένα

accesspoint. Στην περίπτωση που υπήρχαν παραπάνω θα έπρεπε ο χρήστης να ερωτάται αν θέλει να σετάρει εκ νέου μόνο το συγκεκριμένο access point ή όλο το accessmodel.

```
//Set default values for the access model of an item
    public void setDefaultAccessModel(){
        setAMName("");
        setAMNode("");
        setAMFunctionName("");
setAMFunctionClass("");
setAMArgument("");
    }
```

Δίνεται, συνεπώς, όπως και με το semantic model,η δυνατότητα επέκτασης της εφαρμογής ώστε ο χρήστης να μπορεί να ορίσει παραπάνω από ένα access point και functions επί αυτών σε ένα item. Επίσης, παρότι δεν απασχολεί την τρέχουσα μεταπτυχιακή διατριβή, θα μπορούσε να ελέγχεται η ορθότητα των εισαγόμενων ορισμάτων μέσα από την δομή του αντικειμένου, πχ να προειδοποιείται ο χρήστης όταν εισάγει ονόματα κόμβων που δεν υπάρχουν στην δομή του αντικειμένου ή υπάρχουν αλλά δεν συνδέονται με κάποιον Interpolator κ.ά.

6.6 Αναίρεση και Επανεκτέλεση Εντολών (Undo/Redo)

Βασικό στοιχείο κάθε εφαρμογής είναι να δίνει την ευκαιρία στον χρήστη να αναιρέσει μια λανθασμένη ενέργεια, εφόσον ο ανθρώπινος παράγοντας είναι απροσδιόριστος και μπορεί να κάνει λάθη στην εφαρμογή ή να λαμβάνει αποφάσεις που στη συνέχεια θέλει να ακυρώσει. Ως εκ τούτου, θεωρήθηκε πως επιβάλλεται να υπάρχει ένας undo-redo μηχανισμός, τουλάχιστον για τις βασικές ενέργειες που εκτελεί ο χρήστης, ώστε η εφαρμογή να είναι όσο το δυνατόν πιο user-friendly.

Η δυνατότητα αυτή επιτεύχθηκε αξιοποιώντας τα σχετικό πακέτα που προσφέρει η swing και φαίνονται στο παρακάτω import, προσαρμόζοντάς τη λειτουργικότητα στις δικές μας ανάγκες.

```
import javax.swing.undo.AbstractUndoableEdit;
import javax.swing.undo.UndoManager;
import javax.swing.undo.UndoableEdit;
```

Χρησιμοποιούμε έναν UndoManager, ο οποίος διαχειρίζεται μια λίστα από UndoableEdits. Το UndoableEdit είναι ένα interface που εκφράζει μια επεξεργασία (edit) και ως interface μας επιτρέπει να ορίσουμε διαφορετική λειτουργικότητά ανά εντολή. Η επεξεργασία μπορεί να αναιρεθεί(να γίνει undo) και αν έχει γίνει undo να επανεκτελεστεί (να γίνει redo). Στην ουσία, ο UndoManager διατηρεί μια διατεταγμένη λίστα από edits και γνωρίζει τον δείκτη της επόμενης επεξεργασίας στην λίστα. Αυτός αντιστοιχεί είτε στο μέγεθος της λίστας, είτε αν έχει γίνει κάποιο undo, αντιστοιχεί στο δείκτη της τελευταίας ενέργειας που αναιρέθηκε. Το undo λειτουργείμε αντίστροφη σειρά εισόδου των ενεργειών, επομένως όταν γίνεται ένα undo, ο δείκτης δείχνει στην πιο πρόσφατα εκτελεσμένη ενέργεια και όχι σε αυτήν που εισάχθηκε πρώτη.

Η υλοποίηση του undo/redo συγκεντρώνεται σε δυο κουμπιά στο toolbar της εφαρμογής, το 'Undo Last Action'  και το 'Redo Last Action '  , που όπως μαρτυρά το όνομά τους επιτρέπουν την αναίρεση της τελευταίας ενέργειας που εκτέλεσε ο χρήστης και την επαναφορά της τελευταίας ενέργειας που αναίρεσε ο χρήστης.

Εικόνα 6.37: Το undo-redo στο toolbar

Στα πλαίσια αυτής της διατριβής, τα undo/redo αφορούν την τελευταία – και μόνο - ενέργεια που εκτελείται από τον χρήστη και όχι παραπάνω. Σχετικά με το γιατί επιλέχθηκε το undo/redo να αναιρεί/επαναφέρει μόνο μια ενέργεια, η απάντηση είναι πως με τη χρήση πολλαπλών undo/redo δημιουργούνταν προβλήματα στην λειτουργία της εφαρμογής, καθώς μια ενέργεια μπορεί να είχε επίπτωση σε μια άλλη και χρειαζόταν προσοχή

μεταξύ διαδοχικών undo-redo. Προτιμήσαμε, σε πρώτο στάδιο, να κρατήσουμε απλή την εφαρμογή, δημιουργώντας μια σωστή βάση για την δυνατότητα επέκτασης αυτών των εντολών, παρά να αυξήσουμε την πολυπλοκότητα, καταναλώνοντας παραπάνω χρόνο στην υλοποίηση για λόγους εντυπωσιασμού, δεδομένου ότι όπως έχουμε ήδη αναφέρει η συγκεκριμένη διατριβή στοχεύει να είναι η βάση για να εμπλουτιστεί με επιπλέον πράγματα στο μέλλον.

Λόγω της συσχέτισης που υπάρχει μεταξύ των δυο κουμπιών, αποφασίσαμε όταν πατάμε το undo αυτό να απενεργοποιείται, εφόσον δεν επιτρέπεται άλλη αναίρεση και να ενεργοποιείται το redo, ώστε να μπορεί να γίνει επαναφορά της τελευταίας ακυρωμένης ενέργειας. Αυτό γιατί κατά τη διάρκεια των δοκιμών, παρατηρήσαμε πως αν ο χρήστης πατούσε undo και εκτελούσε άλλη ενέργεια μετά το undo, ενεργοποιώντας εκ νέου το undo ενώ και το redo παρέμενε ενεργοποιημένο, και στη συνέχεια πατούσε το redo χωρίς να έχει πατήσει πρώτα το undo, αυτό επανέφερε μια ενέργεια που είχε γίνει πολύ νωρίτερα και αποφασίστηκε με την εκτέλεση της επόμενης αναιρέσιμης ενέργειας, το redo να απενεργοποιείται ώστε να σχετίζεται πάντα με το τελευταίο undo, διότι διαφορετικά δημιουργούσε σύγχυση και μπέρδεμα στον χρήστη.

Το πρώτο βήμα για να επιτύχουμε την λειτουργικότητα ήταν, όπως μπορεί κάποιος να φανταστεί, σε κάθε ενέργεια που θέλουμε να την ορίσουμε ως αναιρέσιμη και επανεκτελέσιμη, να αποθηκεύουμε το item που επηρεάστηκε ώστε να μπορούμε να το μεταβάλλουμε ή να το επαναφέρουμε, όπως φαίνεται από τις παρακάτω μεθόδους της κλάσης WorldScene.java:

```
//Set item that was most recently used in an action for undo/redo purposes
    public void setActionItem(Item m){
this.backupItem = m;
    }

//Return item most recently used in an action
    public Item getActionItem(){
        return this.backupItem;
}
```

Κατά το άνοιγμα της εφαρμογής, τα δυο κουμπιά είναι απενεργοποιημένα και το undo ενεργοποιείται μόνο όταν εκτελείται μια ενέργεια που κληρονομεί το UndoableEdit interface, δηλαδή έχει χαρακτηριστεί ως αναιρέσιμη. Με την σειρά του το redo ενεργοποιείται άμα πατηθεί το undo, ενώ αντίστροφα το undo επανενεργοποιείται όταν πατηθεί το redo και αυτό συνεχίζεται μέχρι να εκτελεστεί μια άλλη undoable εντολή από τον χρήστη. Επίσης, τα δυο κουμπιά απενεργοποιούνται όταν καθαρίζουμε την σκηνή, εφόσον θεωρούμε πως θέλουμε να φτιάξουμε εξαρχής έναν κόσμο, όπως κι όταν φορτώνουμε εκ νέου έναν κόσμο, που πάλι θεωρείται πως η επεξεργασία της σκηνής αρχίζει εκ του μηδενός.

Όσον αφορά τις ενέργειες που επιτρέπονται μέσα από το undo-redo, αυτές είναι που αναλύσαμε και παραπάνω σε αυτό το κεφάλαιο και συνοψίζονται στις παρακάτω:

- Add Node to World (εντολή προσθήκης αντικειμένου στον κόσμο): Όταν πατάμε το undo έπειτα από αυτή την εντολή του toolbar,το τελευταίο item που εισήχθη στην σκηνή αφαιρείται, δηλαδή γίνεται delete. Όταν πατάμε το redo ο κόμβος που μόλις αφαιρέθηκε επαναπροστίθεται. Σε αυτή την επαναπροσθήκη δημιουργείται νέο item από το παλιό, που κληρονομεί όλα του τα χαρακτηριστικά (όνομα, properties και models). Για να διαχωρίσουμε το τελευταίο αντικείμενο που εισήχθη στην σκηνή από ένα αντικείμενο που αποθηκεύεται λόγω του undo/redo, φτιάξαμε άλλη μέθοδο, όπως φαίνεται στον παρακάτω κώδικα:

```
//Return item most recently added in scenegraph
  public Item getLastItem(){
    return this.item;
}
```

- Add (εντολή δημιουργίας αντιγράφου ενός item): Όταν πατάμε undo έπειτα από αυτή την εντολή, διαγράφεται το αντικείμενο που προστέθηκε ως αντίγραφο ενός άλλου αντικειμένου. Όταν πατάμε

redo, το αντίγραφο αυτό επαναπροστίθεται στην σκηνή, ως αντίγραφο του εαυτού του.Παρακάτω παραθέτουμε τον κώδικα του add button listener, κυρίως για να παρουσιάσουμε ένα παράδειγμα εφαρμογής της λειτουργικότητας του undo/redo:

```
//Listener for Add Button
    class AddListener implements ActionListener{
        //Activate undo redo possibility of functionality
         public boolean canUndo() { return true; }

         public boolean canRedo() { return true; }

        @Override
        public void actionPerformed(ActionEvent e){
            try{
               //if the selected item is not null
                if(it!=null){
                    //save the selected item for redo
                    final Item addedItem = it;
                    //Copy original item in world
                    copyItem(it);

                    UndoableEdit         undoableEdit         =         new
AbstractUndoableEdit(){
                     // Method that is called when we must redo the undone
action
                        @Override
                        public         void         redo()         throws
javax.swing.undo.CannotRedoException{
                            super.redo();
                            //get the deleted item
                            final Item redoItem = getActionItem();
//add the deleted item to scene and rename it with the name it had
                            copyItem(addedItem);
                            rename(getLastItem(), redoItem.getItemName());

                        }//end redo

                        @Override
                        public         void         undo()         throws
javax.swing.undo.CannotUndoException
                        {
                           super.undo();
                           //Delete last added item
                           delete(getActionItem());
                         }
                     };

                  // Add the undo action to manager
                  undoManager.addEdit(undoableEdit);

                }else{
                    JPanel panel = new JPanel();
                    JOptionPane.showMessageDialog(panel,      "No      item
selected!", "Warning",
```

```
JOptionPane.WARNING_MESSAGE);
                }

            }catch(NullPointerException n){
                repaint();
            }
        }
    }//end AddListener
```

- Rename (εντολή μετονομασίας item): Όταν πατάμε undo το item παίρνει το προηγούμενο όνομα που είχε. Όταν πατάμε redo το item παίρνει το πιο πρόσφατο όνομα που του δόθηκε. Για να το καταφέρουμε αυτό, έπρεπε πέρα από το item να κρατήσουμε πληροφορία και για το όνομα του item πριν την αλλαγή, καθώς δεν είχαμε διαφορετικά πρόσβαση στην ιστορικότητα του ονόματος από την στιγμή που το item παραμένει το ίδιο, δηλαδή δεν είναι όπως η διαγραφή, που στο undo δημιουργείται νέο item όπου την πληροφορία την αντλούμε από το παλιό item που έχουμε αποθηκεύσει. Εδώ επηρεάζεται το ίδιο το item, επομένως έπρεπε να κρατήσουμε πληροφορία για το πεδίο-όνομα. Οι σχετικές μέθοδοι φαίνεται στον παρακάτω κώδικα:

```
//Save item name of a selected item before it is renamed
  public void setPrevItemName(String name){
    this.prevItemName = name;
  }

  //Get previous item name of last item renamed
  public String getPrevItemName(){
    return this.prevItemName;
  }
```

- Delete (εντολή διαγραφής item): Όταν πατάμε undo έπειτα από αυτή την εντολή, το διεγραμμένο item επαναπροστίθεται στον κόσμο. Σε αυτή την επαναπροσθήκη δημιουργείται νέο item από το παλιό, που κληρονομεί όλα του τα χαρακτηριστικά (όνομα, properties και models). Με το redo το item αυτό επαναδιαγράφεται.
- Set item class: real/unreal: Όταν πατάμε undo έπειτα από αυτή την εντολή, επιστρέφει η προηγούμενη τιμή που είχε η παράμετρος item class, δηλαδή αν έγινε unreal επιστρέφει σε real, ένώ αν έγινε real επιστρέφει σε unreal. Το redo επιστρέφει την τιμή που είχε πριν την αναίρεση.
- Save Virtual Model:Αφού αποθηκεύσουμε το virtual model ενός item, αν πατήσουμε undo τότε αυτό επανέρχεται στις τιμές που είχε προηγουμένως, ενώ με το redo επιστρέφουν οι τιμές που καταχωρήθηκαν τελευταίες. Και εδώ συνεπάγεται πως έπρεπε να κρατήσουμε σε τοπικές μεταβλητές τις τιμές που είχε προηγουμένως το virtual model του item ώστε να μπορέσουμε να τις ανακτήσουμε.
- Reset Virtual Model:Το undo, ομοίως με το undo του Save, επαναφέρει τις προηγούμενες τιμές του Virtual model. Το redo βάζει τις default τιμές στο virtualmodel.
- Save Semantic Model: Αφού αποθηκεύσουμε το semantic model ενός item, αν πατήσουμε undo τότε αυτό επανέρχεται στις τιμές που είχε προηγουμένως, ενώ με το redo επιστρέφουν οι τιμές που καταχωρήθηκαν τελευταίες. Και εδώ συνεπάγεται πως έπρεπε να κρατήσουμε σε τοπικές μεταβλητές τις τιμές που είχε προηγουμένως το semantic model, ώστε να μπορέσουμε να τις ανακτήσουμε.
- Reset Semantic Model:Το undo, ομοίως με το undo του Save, επαναφέρει τις προηγούμενες τιμές του semantic model. Το redo βάζει τις default τιμές στο semantic model.
- Save Access Model: Αφού αποθηκεύσουμε το access model ενός item, αν πατήσουμε undo τότε αυτό επανέρχεται στις τιμές που είχε προηγουμένως ενώ με το redo επιστρέφουν οι τιμές που καταχωρήθηκαν τελευταίες. Και εδώ συνεπάγεται πως έπρεπε να κρατήσουμε σε τοπικές μεταβλητές τις τιμές που είχε προηγουμένως το access model, ώστε να μπορέσουμε να τις ανακτήσουμε.
- Reset Access Model: Το undo, ομοίως με το undo του Save, επαναφέρει τις προηγούμενες τιμές του access model. Το redo βάζει τις default τιμές στο access model.

Οι υπόλοιπες εντολές του δεξιού μενού (Show Item, View Object, Item Properties) δεν είναι undoable καθώς δεν μεταβάλλουν κάποιο χαρακτηριστικό του item, γιατί έχουν μόνο πληροφοριακό χαρακτήρα. Ως εκ τούτου το undo/redo παραμένει ενεργοποιημένο από προηγούμενη εντολή και δεν επηρεάζεται από την εκτέλεση αυτών των επιλογών. Οι υπόλοιπες εντολές που θα δούμε παρακάτω επίσης δεν είναι αναιρέσιμες, ενώ κάποιες μάλιστα οδηγούν σε απενεργοποίηση του undo/redo, όπως αναφέρθηκε και προηγούμενως.

6.7 Αποθήκευση Κόσμου

Η επόμενη βασική επιλογή για να ολοκληρωθεί η σχεδίαση ενός εικονικού κόσμου είναι η αποθήκευση του κόσμου σε κατανοητή μορφή από την πλατφόρμα Reve, δηλαδή η εξαγωγή του σε ένα αρχείο verl. Η λειτουργικότητα αυτή έχει ενσωματωθεί σε δυο κουμπιά του toolbar, το Save  και Save As  που η λειτουργικότητά τους διαφέρει σε κάποια ελάχιστα σημεία.

Εικόνα 6.38: Τα Save/SaveAs στη γραμμή εργαλειών

Αρχίζοντας από τα κοινά τους χαρακτηριστικά, τα δυο κουμπιά επιτυγχάνουν να δημιουργήσουν ένα verl αρχείο που αναπαριστά τον εικονικό κόσμο που έχει δημιουργήσει ο χρήστης. Για την εξαγωγή του κόσμου σε κώδικα verl δημιουργείται ένα στιγμιότυπο της κλάσης ExportScene.java που παράγει τον κόσμο, η οποία λαμβάνει ως παράμετρο την σκηνή του κόσμου και το όνομά του, στην ουσία το όνομα του αρχείου. Έπειτα, μελετώντας ένα-ένα τα αντικείμενα, χτίζει τον xml κώδικα αξιοποιώντας την JAXP βιβλιοθήκη της java, όπως αυτή περιγράφηκε στο Κεφάλαιο 5.

Αρχικά, γίνεται μια ταξινόμησή των items στη σκηνή βάσει της προέλευσής τους, δηλαδή του itemGroup τους, καθώς έτσι επιτάσσει η σωστή σύνταξη της verl. Σε διαφορετική περίπτωση, η Reve δεν φορτώνει τα items αν υπάγονται στο ίδιο itemGroup αλλά εμφανίζονται σε ξεχωριστά σύνολα στην σύνταξη της verl. Στη συνέχεια, ανατρέχοντας αναδρομικά τα items, δημιουργούμε τα διαφορετικά κύρια elements της verl, <world>, <itemGroups> και <items>, και τα επιμέρους elements, πχ <virtualModel>, <accessModel>, <semanticModel> και τα attributes τους, βάσει των χαρακτηριστικών που έχει αποθηκευμένο ένα item στον κόσμο. Οι διαστάσεις του κόσμου ορίζονται σε ένα default μέγεθος σε μέτρα, ωστόσο θα μπορούσε να προστεθεί σχετική επιλογή στο μέλλον, ώστε να λαμβάνεται ως παράμετρος εισόδου από τον χρήστη κατά την αποθήκευση του κόσμου.

Κατά το export του semantic model, γίνεται αντιστοίχιση των τιμών του symbol argument class στο πλήρες τους όνομα, πχ το X3DConnectorL1 θα μεταφραστεί σε reve.scene.item.semantic.argument.X3DConnectorL1. Το ίδιο ισχύει και για το accessPoint function class του access model, πχ το AccessPointTranslateFunctionL1 θα μεταφραστεί σε reve.scene.item.access.function.AccessPointTranslateFunctionL1, εφόσον αυτές είναι οι τιμές που αναγνωρίζει η Reve. Στην περίπτωση που το semanticmodelείναι κενό, δεν συμπεριλαμβάνεται ως μοντέλο του item στο αρχείο του κόσμου, καθώς αυτό παράγει errors στην Reve κατά την φόρτωση του κόσμου. Η ίδια λογική εφαρμόζεται και για το access model.

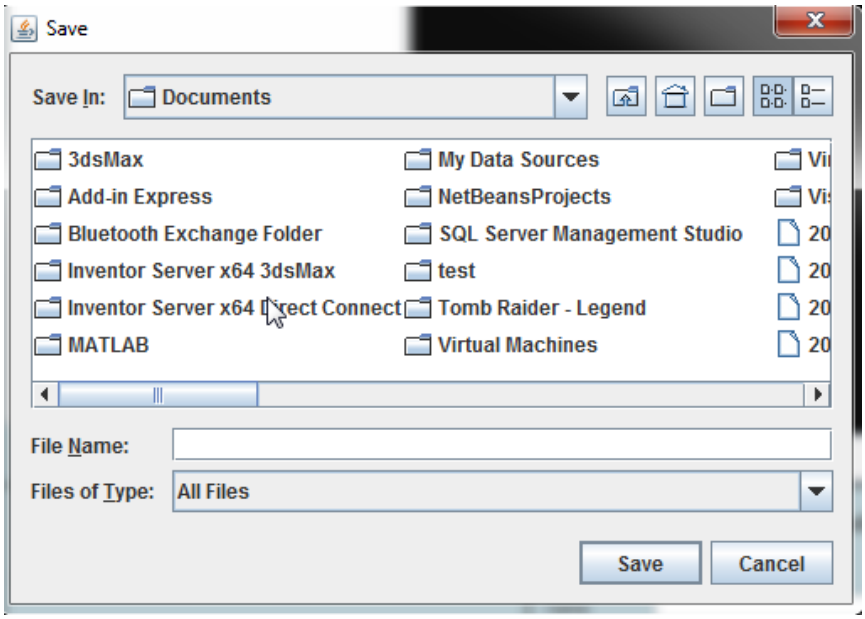
```
   Attr aclass = doc.createAttribute("class");
   String argClass = i.getArgumentClass();
    if (argClass.equals("X3DConnectorL1"))
                        argClass =
"reve.scene.item.semantic.argument.X3DConnectorL1";
                    else if
(argClass.equals("ZeroArgumentItemMethodConnectorL1"))
                        argClass =
"reve.scene.item.semantic.argument.ZeroArgumentItemMethodConnectorL1";
aclass.setValue(argClass);
```

```
argument.setAttributeNode(aclass);
```

Τέλος, πρέπει να πούμε σχετικά με το export, πως το attribute«fit» κάθε item το έχουμε ορίσει ως false, ενώ πρέπει να ορίζεται ως true, ώστε το τοπικό σύστημα συντεταγμένων ενός αντικειμένου να συμφωνεί με το σύστημα συντεταγμένων του κόσμου. Αυτό έγινε καθαρά μέχρι να επιτευχθεί η σχετική διόρθωση που ευθυγραμμίζει τα αντικείμενα στην αρχή των αξόνων, ώστε ο χρήστης να μπορεί να δει στην Reve την ίδια εικόνα του κόσμου που έχει υλοποιήσει από την εφαρμογή. Αυτό, όμως, θα πρέπει να αλλάξει σε μελλοντική βελτίωση της εφαρμογής. Επίσης, δεδομένου ότι θέλουμε η παράμετρος fit να παίρνει μόνο την τιμή true, παραλήφθηκε ως δυνατότητα επιλογής του χρήστη από το right-click μενού.

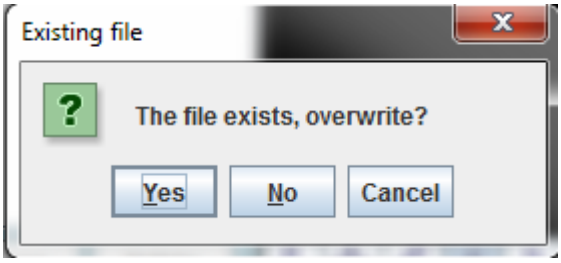
Σχετικά με τις ιδιαιτερότητες κάθε κουμπιού, το Save As button είναι πάντα ενεργοποιημένο, δίνοντας την δυνατότητα στον χρήστη να σώσει ακόμη και έναν κενό κόσμο. Σε γενικές γραμμές, το Save Asχρησιμοποιείται όταν σώζουμε για πρώτη φορά έναν κόσμο, οπότε δεν υπάρχει προηγούμενη πληροφορία για αυτόν, ή όταν θέλουμε να σώσουμε έναν κόσμο με διαφορετικό όνομα από το τρέχον ή σε διαφορετική τοποθεσία από την τρέχουσα, όπως για παράδειγμα συμβαίνει με ένα απλό MS Word αρχείο.

Όταν ο χρήστης πατάει το Save As, του ανοίγει ένα παράθυρο όπου πρέπει να εισάγει το όνομα του verl αρχείου (που αντιστοιχεί και στο όνομα του κόσμου) και να επιλέξει την τοποθεσία του υπολογιστή του όπου θέλει να το αποθηκεύσει. Ο φάκελος προεπιλογής είναι ο 'My Documents', αλλά ο χρήστης μπορεί να περιηγηθει και σε άλλους φακέλους. Το παράθυρο που ανοίγει απεικονίζεται στην παρακάτω εικόνα.



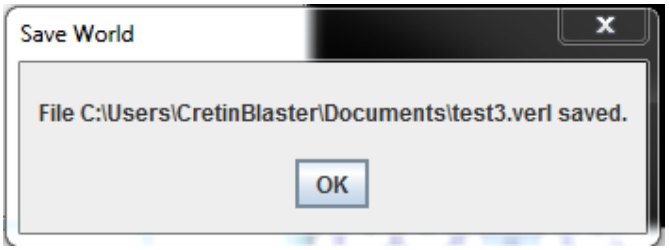
Εικόνα 6.39: Τα παράθυρο αποθήκευσης κόσμου

Σε περίπτωση που το όνομα του αρχείου χρησιμοποιείται ήδη εκεί που πάει να το αποθηκεύσει ο χρήστης, του εμφανίζεται προειδοποιητικό μήνυμα, ώστε να επιλέξει αν θέλει να ακυρώσει την αποθήκευση αν έκανε λάθος, για να μη χάσει το προηγούμενο αρχείο, ή να επιλέξει αν θέλει να το αποθηκεύσει ούτως ή άλλως, γράφοντας πάνω από το υπάρχον αρχείο. Στην περίπτωση της ακύρωσης, ο χρήστης θα πρέπει να ξαναπατήσει το Save As για να εισάγει άλλο όνομα αρχείου.



Εικόνα 6.40: Έλεγχος επιβεβαίωσης για υπάρχον αρχείο

Τέλος, ο χρήστης ενημερώνεται για την επιτυχή αποθήκευση του αρχείου με σχετικό μήνυμα, όπου εμφανίζεται το πλήρες όνομα του αρχείου. Εδώ έπρεπε να προσέξουμε αν ο χρήστης έβαζε την κατάληξη verl στο όνομα, αυτή να παραλείπεται κατά την αποθήκευση για να μην εισάγεται εις διπλούν, ενώ στην αντίθετη περίπτωση που ο χρήστης την παραλείπει, να εισάγεται. Χωρίς αυτόν τον έλεγχο ένα αρχείο μπορεί να σωζόταν ως test.verl.verl καθιστωντας το μη αναγνώσιμο από την Reve Worlds.



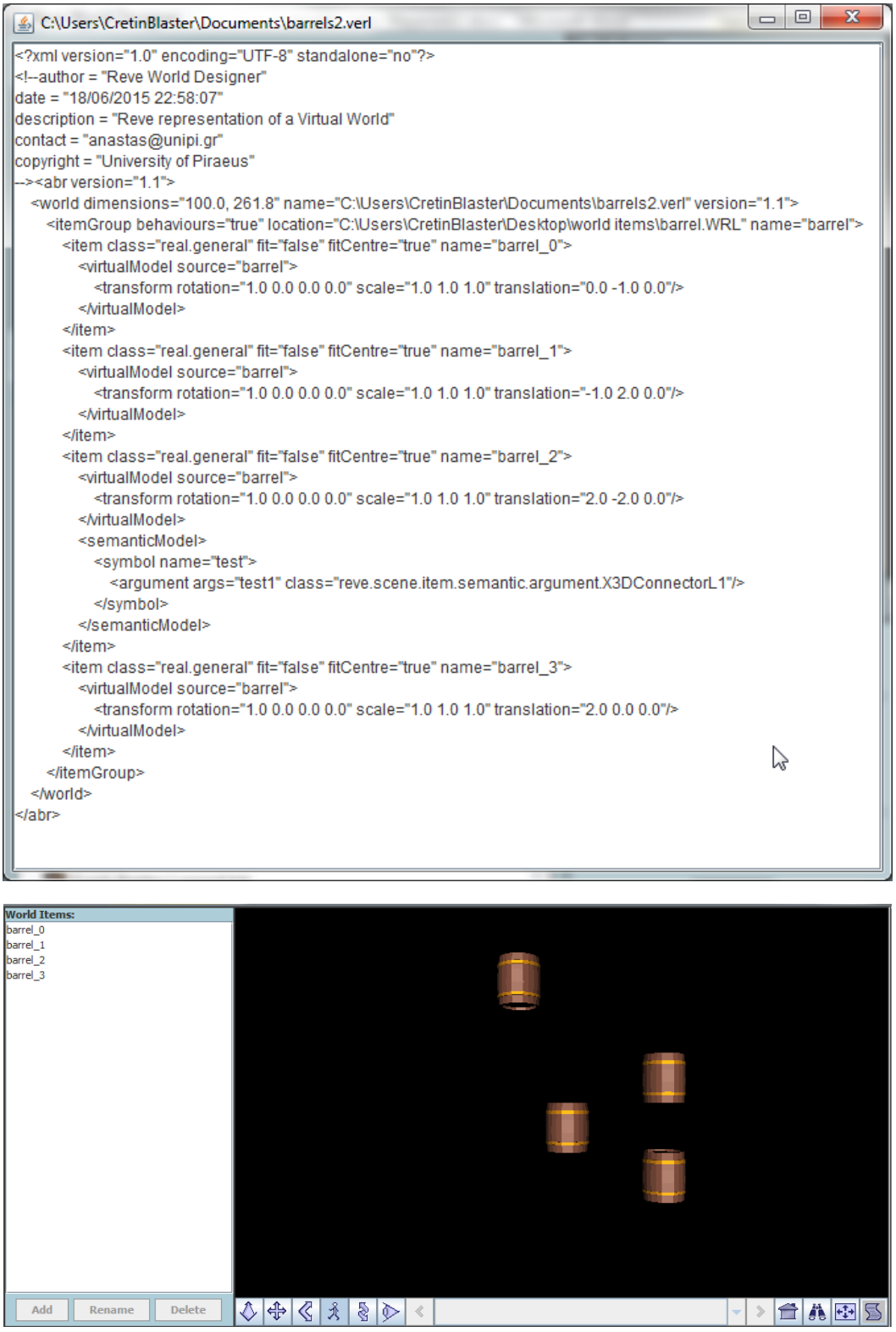
Εικόνα 6.41: Μήνυμα επιτυχούς αποθήκευσης κόσμου

Σε σχέση με το Save As, το Save είναι αρχικά απενεργοποιημένο μέχρι να εισαχθεί ένα αντικείμενο στον κόσμο και η σκηνή να μην είναι κενή. Αν είναι η πρωτη φορά που αποθηκεύουμε τον κόσμο, τότε το Save λειτουργεί ακριβώς με τον ίδιο τρόπο όπως το Save As. Η διαφορά έγκειται στο αν έχουμε ήδη αποθηκεύσει τον κόσμο που εμφανίζεται στην σκηνή, τότε το Save ενημερώνει το υπάρχον αρχείο με τις αλλαγές και εμφανίζει σχετικό μήνυμα ενημέρωσης, όπως το Save as.

Αφού αποθηκεύσουμε τον κόσμο (ή φορτώσουμε έναν υπάρχον κόσμο), ενεργοποιείται το κουμπί 'Show Code '  του toolbar. Πατώντας το, ο χρήστης μπορεί να δει σε ένα JPopUpπαράθυρο τον κώδικα του κόσμου που έχει δημιουργήσει. Βασική προϋπόθεση για να εμφανιστούν οι όποιες αλλαγές στον κώδικα είναι να έχει πατήσει Save για τον κόσμο, αφού μόνο έτσι θα περαστούν οι όποιες αλλαγές των items και στο αρχείο.

Σκεφτήκαμε να κάνουμε τον κώδικα αυτόν επεξεργάσιμο από τον χρήστη, προσθέτωντας κάποιο κουμπί στο παράθυρο για να αποθηκευτούν οι αλλαγές στο αρχείο,αλλά αυτό άνοιγε μια σωρεία προβλημάτων. Το κύριο πρόβλημα είναι πως μπλέκαμε με ένα σωρό ελέγχους που χρειάζονταν για αυτή την περίπτωση, καθώς ο χρήστης θα μπορούσε να αλλάξει οτιδήποτε σε ένα αρχείο και να κάνει άθελά του λάθη, από το να χαλάσει την σύνταξη της verl, μέχρι να προσθέσει αντικείμενα που δεν υπάρχουν στον υπολογιστή του κτλ, δημιουργώντας συντακτικά και λογικά σφάλματα στην εφαρμογή. Δεδομένου ότι η επεξεργασία του κώδικα θα έπρεπε να γίνεται με ασφάλεια και για περιορισμένα πράγματα, ελέγχοντας επιτρεπτές τιμές – που μέσα από την ανάγνωση αρχείου η διαδικασία είναι δύκολη και χρονοβόρα – προτιμήσαμε να μην αφήσουμε αυτή την επιλογή στον χρήστη, παρότι θα μπορούσε να αλλάξει στο μέλλον, αν η εφαρμογή υποστεί βελτιώσεις.

Στην παρακάτω εικόνα, όπου φαίνεται ο κώδικας ενός αποθηκευμένου κόσμου, μπορούμε να παρατηρήσουμε πως κατά την αποθήκευση του κόσμου από την εφαρμογή, δημιουργούνται σχετικά σχόλια στο αρχείο που υποδεικνύουν στον χρήστη την ακριβή ημερομηνία δημιουργίας του, καθώς και πληροφορίες για την εφαρμογή και στοιχεία επικοινωνίας.



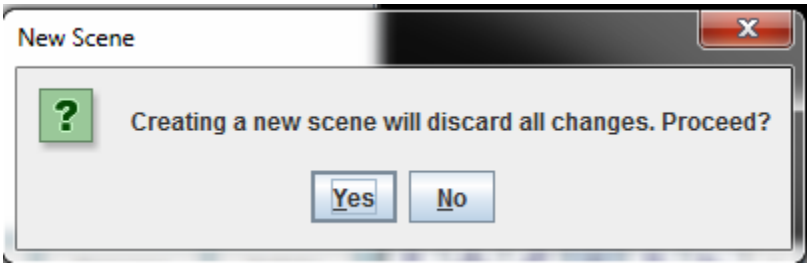
Εικόνα 6.42: Παράθυρο εμφάνισης κώδικα αποθηκευμένου κόσμου (επάνω) και ο κόσμος (κάτω)

Τέλος, να αναφέρουμε πως επιλέγοντας αποθήκευση κόσμου, οι αλλαγές στα models που μπορεί να έχουμε κάνει χωρίς να πατήσουμε Save στο αντίστοιχο μοντέλο, δεν θα συμπεριληφθούν στην αποθήκευση του κόσμου, καθώς αυτή απλά τραβάει τα στοιχεία που έχει κάθε item ήδη, και το item ενημερώνεται για αλλαγές στα models του μέσα από το πάτημα του Save, εφόσον το τελευταίο καταχωρεί τις εισαγόμενες τιμές του χρήστη για το item που είναι κάθε φορά επιλεγμένο και μόνο. Δεν μπορεί, λοιπόν, η αποθήκευση να λειτουργήσει για το σύνολο των items, καθώς δεν μπορεί να γνωρίζει ποιες τιμές αντιστοιχούν σε ποιο item διαφορετικά.

6.8 Δημιουργία Νέας Σκηνής

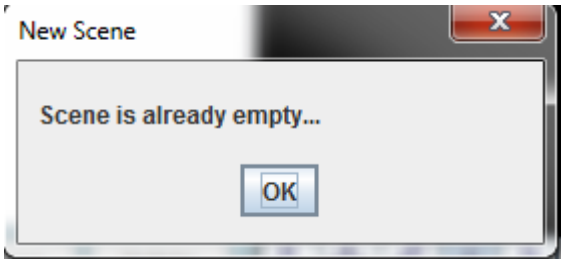
Μια σύνηθης ανάγκη που θέλαμε να καλύψουμε ήταν ο χρήστης να μπορεί να καθαρίσει την σκηνή και να αρχίσει εκ νέου τη σχεδίαση ενός κόσμου. Για αυτό τον λόγο, τοποθετήσαμε στο toolbar το κουμπί 'New Scene'  .

Πατώντας το, η σκηνή καθαρίζει από τον υπάρχον κόσμο, όλα τα models εμφανίζουν τις default τους τιμές, το undo και redo απενεργοποιούνται και καθαρίζεται το ιστορικό των ενεργειών, ενώ απενεργοποιείται και το Save, υποδεικνύοντας στον χρήστη ότι αρχίζει καινούρια σχεδίαση και δεν υπάρχει συσχέτιση με τον προηγούμενο κόσμο. Για να αποφευχθούν, βέβαια, λάθη ο χρήστης διερωτάται για επιβεβαίωση της απόφασης, όπως φαίνεται στην παρακάτω εικόνα, καθώς διαφορετικά μπορεί να έχανε ότι είχε κάνει ως εκείνη την στιγμή.



Εικόνα 6.43: Παράθυρο επιβεβαίωσης δημιουργίας νέας σκηνής

Επιλέγοντας 'No', διασφαλίζεται η διατήρηση του προηγούμενου κόσμου, ενώ επιλέγοντας 'Yes', εφαρμόζεται η δημιουργία της νέας σκηνής, όπως περιγράφηκε. Σε περίπτωση που ο χρήστης πατήσει το 'New Scene', ενώ η σκηνή είναι ήδη άδεια, παρότι τα προαναφερθέντα κουμπιά απενεργοποιούνται, τού εμφανίζεται σχετικό μήνυμα για να ξέρει να μην περιμένει κάτι πέρα από αυτό.



Εικόνα 6.44: Μήνυμα ενημέρωσης για άδεια σκηνή

6.9 Φορτωση Υπάρχοντος Εικονικού Κόσμου

Όπως είχαμε προδιαγράψει και στην ανάλυση απαιτήσεων, θέλαμε ο χρήστης να μπορεί να φορτώσει έναν υπάρχον εικονικό κόσμο. Η επιλογή αυτή είναι εξαιρετικής σημασίας και θα μπορούσαμε να πούμε πως είναι

το ήμισυ κάθε εφαρμογής να επεξεργάζεται αρχεία τα οποία παράγει η ίδια. Σε αντίθετη περίπτωση, ο κόπος του χρήστη θα πήγαινε εν μέρει χαμένος, καθώς για να επεξεργαστεί έναν κόσμο σε δεύτερο χρόνο θα έπρεπε να τον ξαναφτιάξει από την αρχή αλλάζοντας στην νέα σκηνή τις παραμέτρους που τον ενδιαφέρουν. Με απλά λόγια, η εφαρμογή θα είχε χρησιμότητα μόνο για μια χρήση ανά κόσμο.

Για να επιτύχουμε αυτή την λειτουργικότητα, ακολουθήσαμε την αντίστροφη πορεία από αυτήν της αποθήκευσης του κόσμου. Ως εκ τούτου, βασιστήκαμε και πάλι στην JAXP τεχνολογία και την ικανότητα της java να διαβάζει αρχεία, για να διαβάσουμε ένα verl αρχείο, ακολουθώντας την σύνταξή της xml για τη φόρτωση του κόσμου. Έτσι δημιουργήσαμε την κλάση LoadVerl.javaπου λαμβάνει ως παραμέτρους το πλήρες όνομα του αρχείου που αναπαριστά τον κόσμο και την κεντρική σκηνή του κόσμου στην οποία πρέπει αυτός να φορτωθεί.

Διαβάζοντας ένα-ένα τα elementstου αρχείου, δημιουργούμε items στην σκηνή μέσα από το <item> της verl και ορίζουμε στις παραμέτρους του τις πληροφορίες που έχουν τα attributes και τα παιδιά του στο αρχείο. Για παράδειγμα, στον παρακάτω κώδικα, διαβάζουμε το transform του virtualmodel, για να πάρουμε στην συνέχεια τις τιμές του.

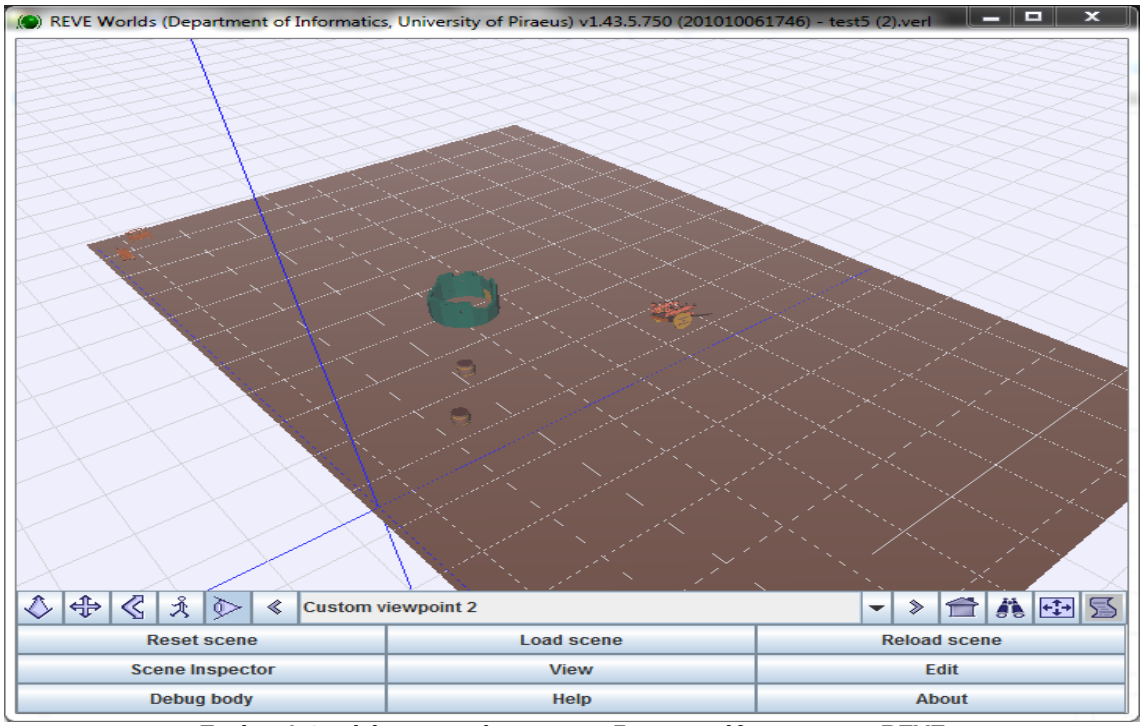
```
//Virtual Model
                try{
                   NodeList                        vmList                        =
itemElement.getElementsByTagName("virtualModel");
                   //System.out.println("-----------------------------");

                     Node vmNode = vmList.item(0);

                     //System.out.println("\nCurrent     Element     :"     +
vmNode.getNodeName());

                     if (vmNode.getNodeType() == Node.ELEMENT_NODE) {

                       Element vmElement = (Element) vmNode;

                       //System.out.println("Source        :        "        +
vmElement.getAttribute("source"));

//Get item source in order to search this DEF name in itemGroup file
                       itemSource = vmElement.getAttribute("source");
                       try{
                       //Add DEF node to the temporary main scene
                       X3DNode                        item                        =
tempWindow.getMainScene().getNamedNode(itemSource);
                       //Create  item  from  given  DEF  and  add  it  to  the
WorldScene
                       scene.addItem(item, itemSource, itemGroupLocation);
                       scene.rename(scene.getLastItem(),
resetName(scene.getLastItem(), itemName));
                       scene.getLastItem().setItemClass(itemClass);

//Get transform values of virtual model to apply them to Scene Graph
                       NodeList                transformList                =
vmElement.getElementsByTagName("transform");
NodetransformNode = transformList.item(0);
```

Για να φορτώσουμε τον κόσμο, αντιμετωπίσαμε δυο ζητήματα. Το πρώτο ήταν πως έπρεπε να μετατρέψουμε τις τιμές του αρχείου σε σωστούς τύπους για τις μεταβλητές του item, πχ για το translation έπρεπε να περαστούν οι τιμές του ως float τιμές στο setTranslation, τα rad να εμφανιστούν σε γωνίες στοUI, τα

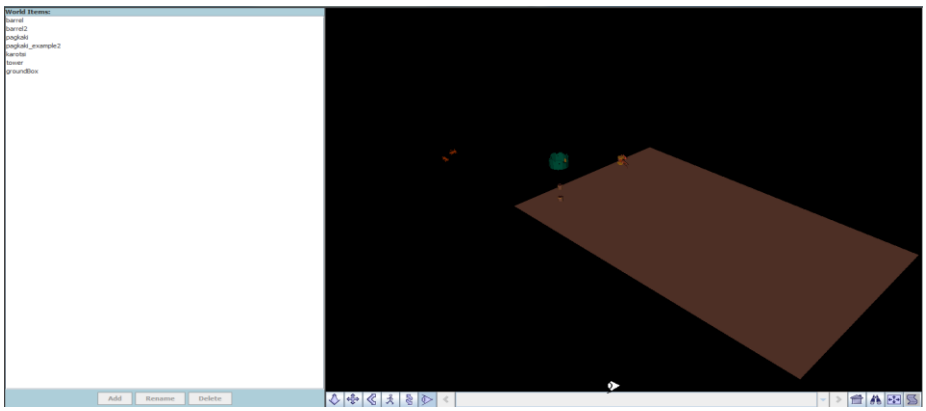
classes των semanticκαι access model να μεταφραστούν στις τιμές που μπορεί να επιλέξει ο χρήστης από το UI κοκ. Το δεύτερο και πιο βασικό ήταν πως έπρεπε να πιάσουμε τυχόν exceptions αν τα items δεν μπορούν να φορτωθούν στην εφαρμογή για κάποιο λόγο, όπως αν ο χρήστης έχει διαγράψει ένα αρχείο από όπου έπαιρνε ένα αντικείμενο ή αν του έχει αλλάξει τοποθεσία στον υπολογιστή. Σε αυτήν την περίπτωση, αποφασίσαμε να βγάζουμε προειδοποιητικό μήνυμα στον χρήστη για τα items που δεν κατάφεραν να φορτωθούν.

Εδώ οφείλουμε να πούμε πως λαμβάνουμε ως προϋπόθεση τα αρχεία που φορτώνονται να έχουν δημιουργηθεί από την εφαρμογή μας. Πρώτον, γιατί έτσι διασφαλίζουμε πως η τοποθεσία των <itemGroup>είναι πλήρης, καθιστώντας εντοπίσιμα τα αρχεία από όπου λαμβάνονται τα items του κόσμου. Δεύτερον, δεδομένου ότι στην εφαρμογή τα αντικείμενα δεν έχουν ακόμα ως σύστημα αναφοράς το σύστημα συντεταγμένων του κόσμου αλλά το δικό τους τοπικό σύστημα συντεταγμένων, βάσει του οποίου υπολογίζεται και το virtual model τους, η φόρτωση κόσμων που έχουν δημιουργηθεί βάσει του συστήματος αξόνων της Reve θα φορτώνονταν διαστρεβλωμένοι, ως προς τις τιμές τους, στην δική μας εφαρμογή.

Ένα τέτοιο παράδειγμα φαίνεται στην παρακάτω εικόνα, όπου φαίνεται η διαφορά ανάμεσα στον κόσμο που φτιάξαμε χειροκίνητα για την πλατφόρμα Reve και η αναπαράσταση αυτού του κόσμου στην εφαρμογή. Όπως βλέπουμε στις παρακάτω εικόνες, στην Reve τα αντικείμενα είναι σωστά τοποθετημένα πάνω στο έδαφος, ενώ στην εφαρμογή η τοποθέτησή τους αποκλίνει από αυτήν που θα θέλαμε, λόγω άλλου συστήματος συντεταγμένων.



Εικόνα 6.45: Φόρτωση κόσμου που δημιουργήθηκε για την REVE

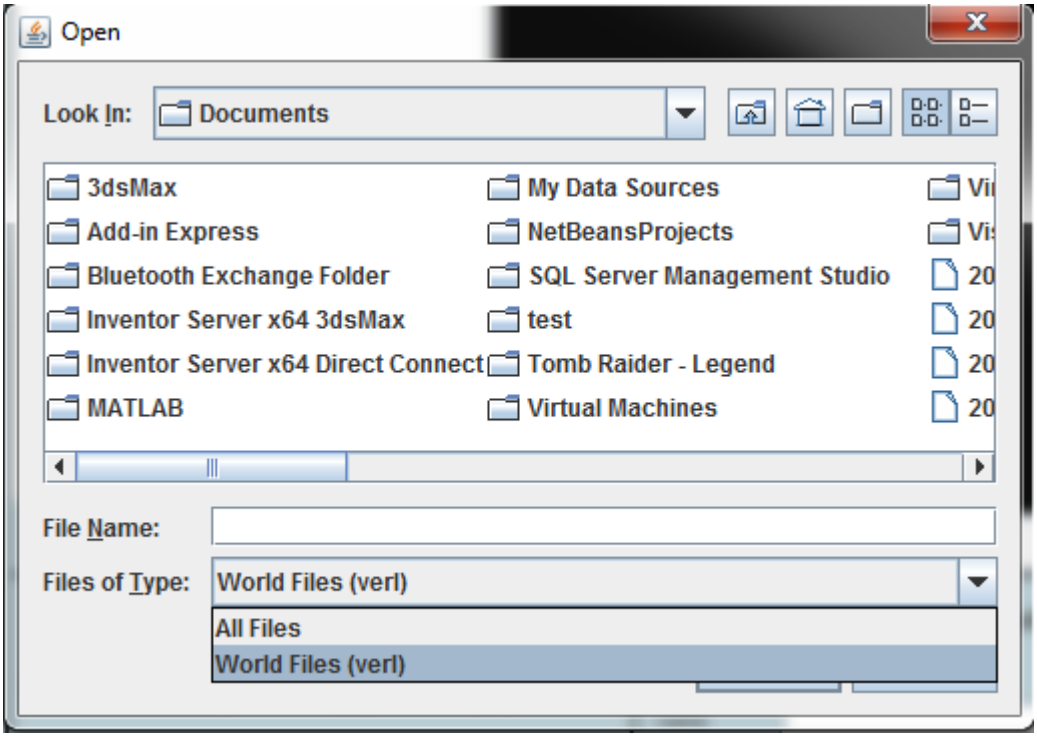


Εικόνα 6.46: Φόρτωση ίδιου κόσμου στην εφαρμογή

Τρίτον, η αποδοχή φορτώματος οποιουδήποτε verlαρχείου προκαλεί τα ίδια προβλήματα με την δυνατότητα χειροκίνητης επεξεργασίας του κώδικα ενός κόσμου από τον χρήστη, καθώς εκεί θα πρέπει να προβλεφθούν και να αντιμετωπιστούν όλα τα συντακτικά και λογικά σφάλματα που ενδέχεται να προκύψουν. Για παράδειγμα, στην προσπάθεια να φορτώσουμε έναν κόσμο που έχουμε φτιάξει χειροκίνητα προκύπτουν τα παρακάτω σφάλματα, γιατί σε κάποια virtual models ορίζεται μόνο το translation και όχι το rotation, ενώ η εφαρμογή αναμένει και τα δυο:

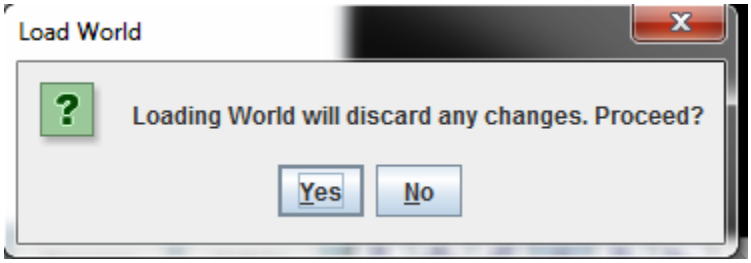
Εικόνα 6.47: Σφάλματα κατά την φόρτωση verl αρχείου που έχει φτιαχτεί από χρήστη

Η υλοποίηση για το φόρτωμα ενός εικονικού κόσμου στην εφαρμογή, ενσωματώθηκε στο κουμπί 'Load World'  του toolbar. Πατώντας το, ανοίγει στον χρήστη το παρακάτω παράθυρο, με φάκελο προεπιλογής το 'My Documents' και τύπο αρχείου το verl (World Files), ώστε να διευκολύνουν τον χρήστη στην αναζήτηση.



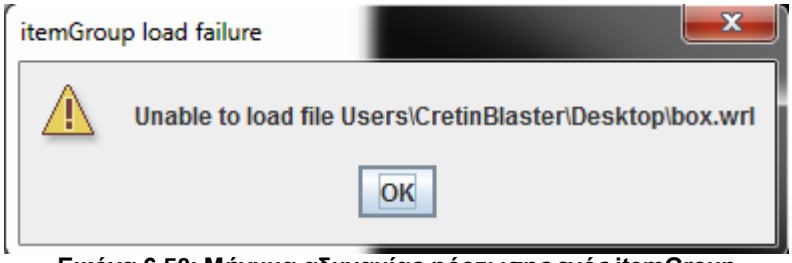
Εικόνα 6.48: Παράθυρο επιλογής αρχείου για φόρτωση

Αν η σκηνή είναι άδεια, το παραπάνω παράθυρο ανοίγει απευθείας. Διαφορετικά, ο χρήστης πρώτα διερωτάται για επιβεβαίωση φορτώματος του κόσμου, εφόσον, αντίστοιχα με την δημιουργία νέας σκηνής, ο κόσμος που έχει σχεδιαστεί μέχρι στιγμής θα χαθεί, ώστε να φορτωθεί ο επόμενος κόσμος.



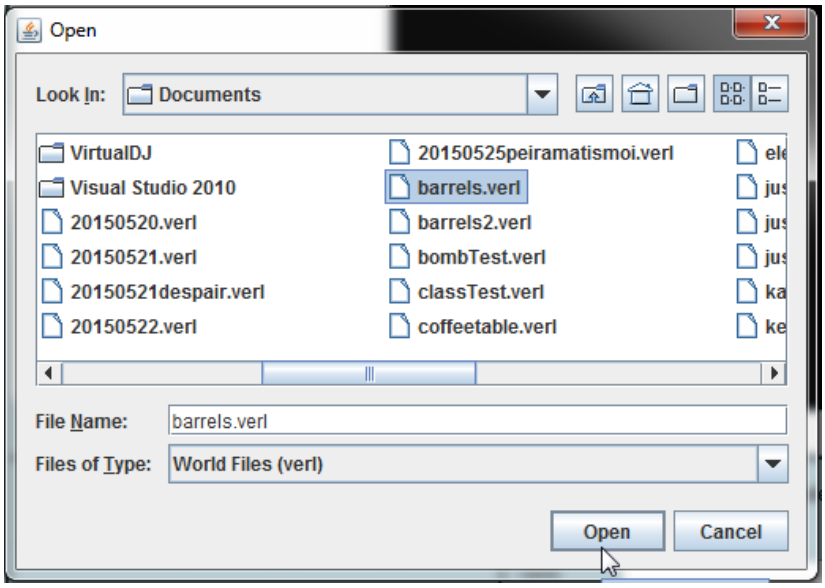
Εικόνα 6.49: Μήνυμα επιβεβαίωσης φόρτωσης κόσμου

Τέλος, όπως ειπώθηκε προηγουμένως, αν ένα ή παραπάνω items δεν μπορούν να φορτωθούν για κάποιο λόγο, ο χρήστης ενημερώνεται κατά την φόρτωση του κόσμου με σχετικό μήνυμα, όπως φαίνεται παρακάτω. Θεωρήσαμε πως δεν άξιζε να μην φορτώσουμε καθόλου τον κόσμο αν απλά λείπει ένα item, καθώς ο χρήστης θα μπορούσε αυτό να το διορθώσει χειροκίνητα στη συνέχεια. Από την άλλη, αν δεν μπορεί να φορτωθεί κανένα item, δηλαδή συνολικά ο κόσμος, τότε η προηγούμενη εικόνα που είχε η εφαρμογή παραμένει, ώστε να μην χαθεί η πληροφορία της προηγούμενης σχεδίασης. Προτιμήσαμε, δηλαδή, ο χρήστης να πατήσει ένα New Scene μετά από αποτυχημένη φόρτωση κόσμου αν θέλει να καθαρίσει οπωσδήποτε την σκηνή, παρά να χάσει και την υπάρχουσα σκηνή και να μην έχει φορτωθεί ο κόσμος που ήθελε.

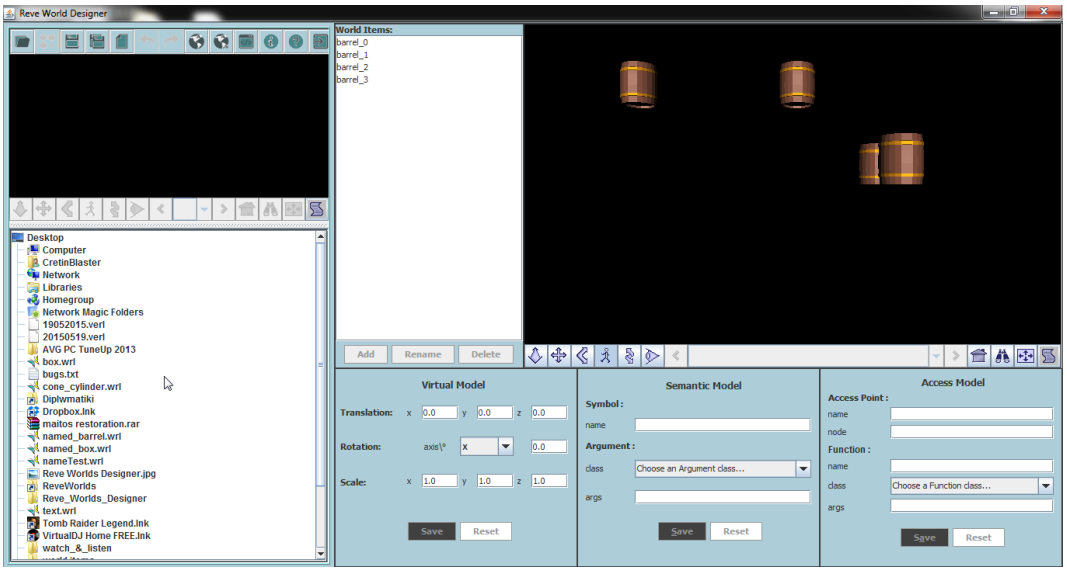


Εικόνα 6.50: Μήνυμα αδυναμίας φόρτωσης ενός itemGroup

Στα πλαίσια της φόρτωσης εικονικού κόσμου, δημιουργήσαμε και το κουμπί 'Add World', που στην ουσία δεν καθαρίζει την σκηνή, αλλά επιτρέπει στον χρήστη να φορτώσει έναν εικονικό κόσμο στην υπάρχουσα σκηνή, συγχωνεύοντας έτσι έναν ή παραπάνω κόσμους μαζί. Παρακάτω παρουσιάζουμε ένα παράδειγμα όπου φορτώνουμε έναν κόσμο με βαρέλια και στην συνέχεια προσθέτουμε στον υπάρχον κόσμο τον εαυτό του.

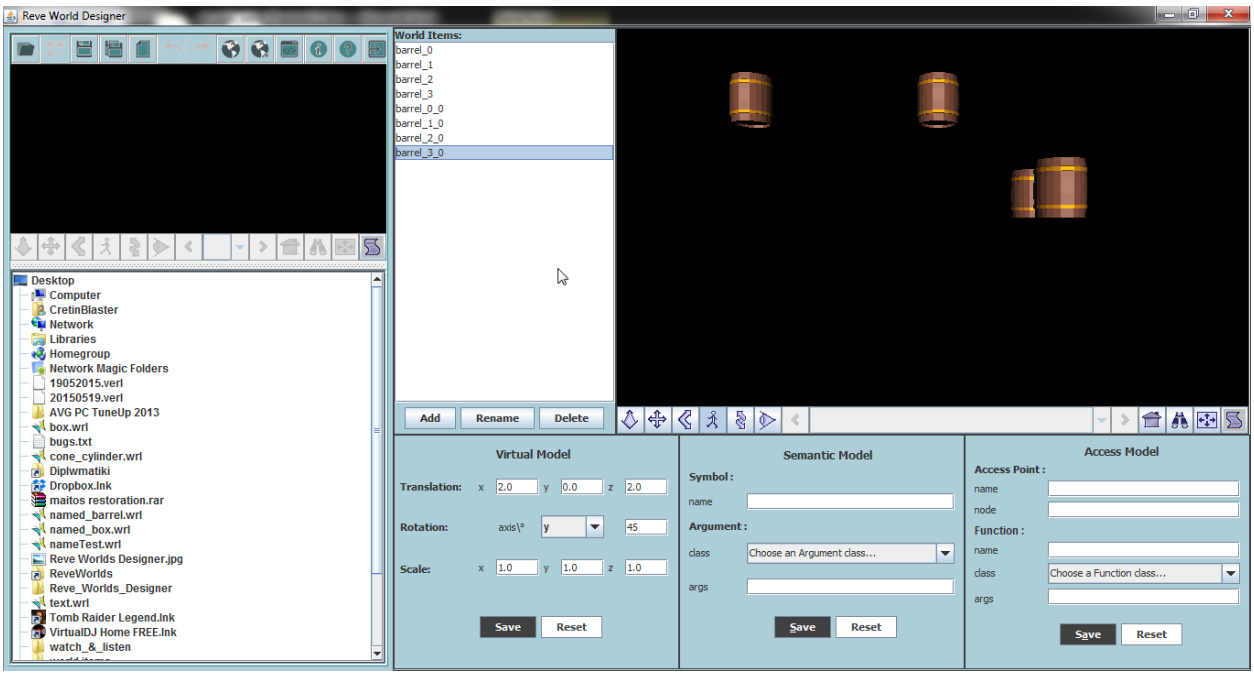


Εικόνα 6.51: Επιλογή του κόσμου barrels.verl για φόρτωση



Εικόνα 6.52: Φόρτωση κόσμου barrels.verl

Αυτό έχει ως αποτέλεσμα τα αντικείμενα να συμπίπτουν το ένα με το άλλο, ενώ είναι τα διπλά. Η χρησιμότητα σε αυτό το παράδειγμα είναι αν θέλουμε να πολλαπλασιάσουμε τα βαρέλια του υπάρχοντος κόσμου, επομενως αντί να δημιουργούμε διαρκώς αντίγραφα από ένα αντικείμενο, να ξαναφορτώσουμε τον ίδιο κόσμο και να αλλάξουμε απλά το virtual model των αντικειμένων.

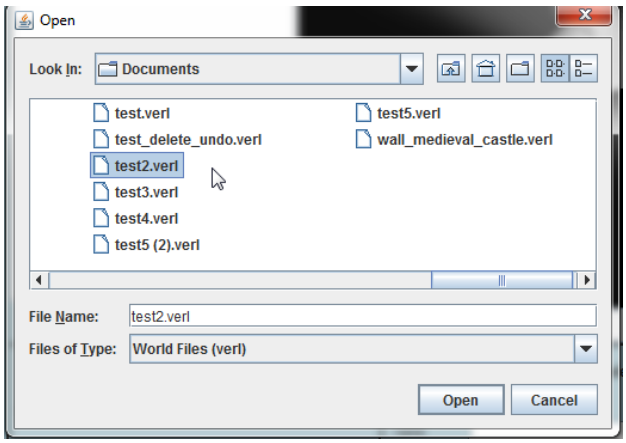


Εικόνα 6.53: Προσθήκη κόσμου barrels.verl

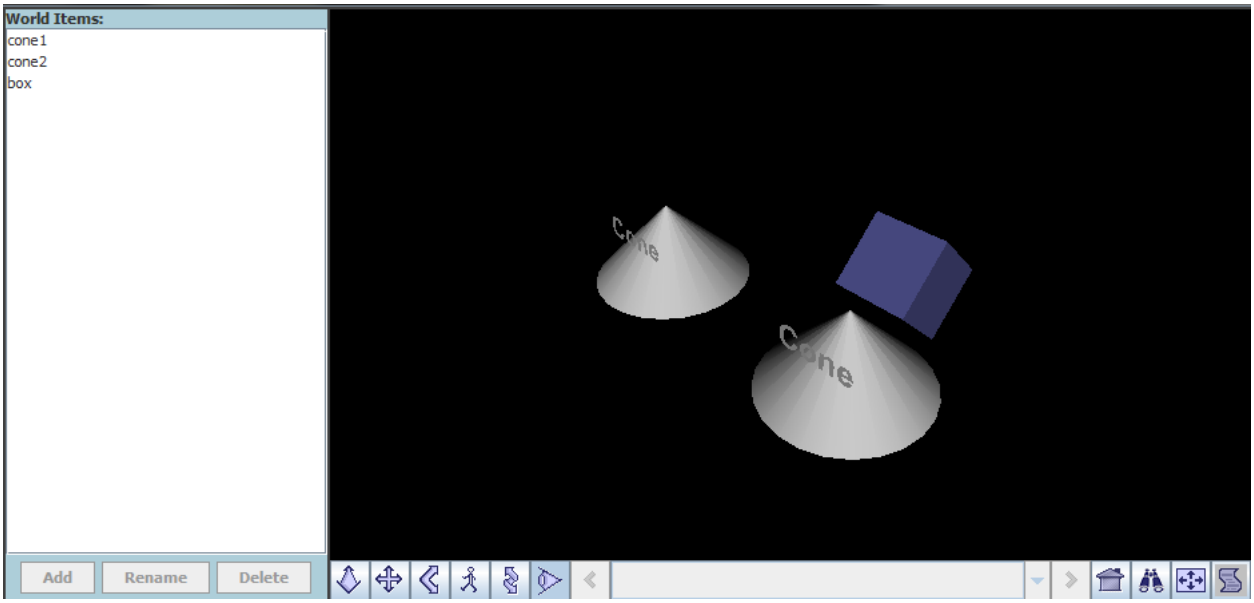
Ένα πρόβλημα που είχαμε στην προσθήκη κόσμου ήταν η ονομασία των items, καθώς σε μια τέτοια περίπτωση τα items θα είχαν το ίδιο όνομα, δημιουργώντας πρόβλημα στην ταυτοποίηση, όπως αναλύσαμε στην αρχή αυτού του κεφαλαίου. Επομένως, ελέγχουμε τα ονόματα των εισαγόμενων αντικειμένων, ώστε να τα μετονομάσουμε για να είναι όλα μοναδικά στην σκηνή. Αυτό δεν επηρεάζει τον κόσμο που φορτώνεται,

εφόσον επιλέγοντας Save, αν η σκηνή ήταν εξαρχής άδεια τότε δημιουργείται απλά καινούριος κόσμος, ενώ αν προσθέσαμε κόσμο σε υπάρχον αρχείο, οι όποιες αλλαγές θα επηρεάσουν το πρώτο αρχείο και σε ουδεμία περίπτωση αυτό που προστίθεται.

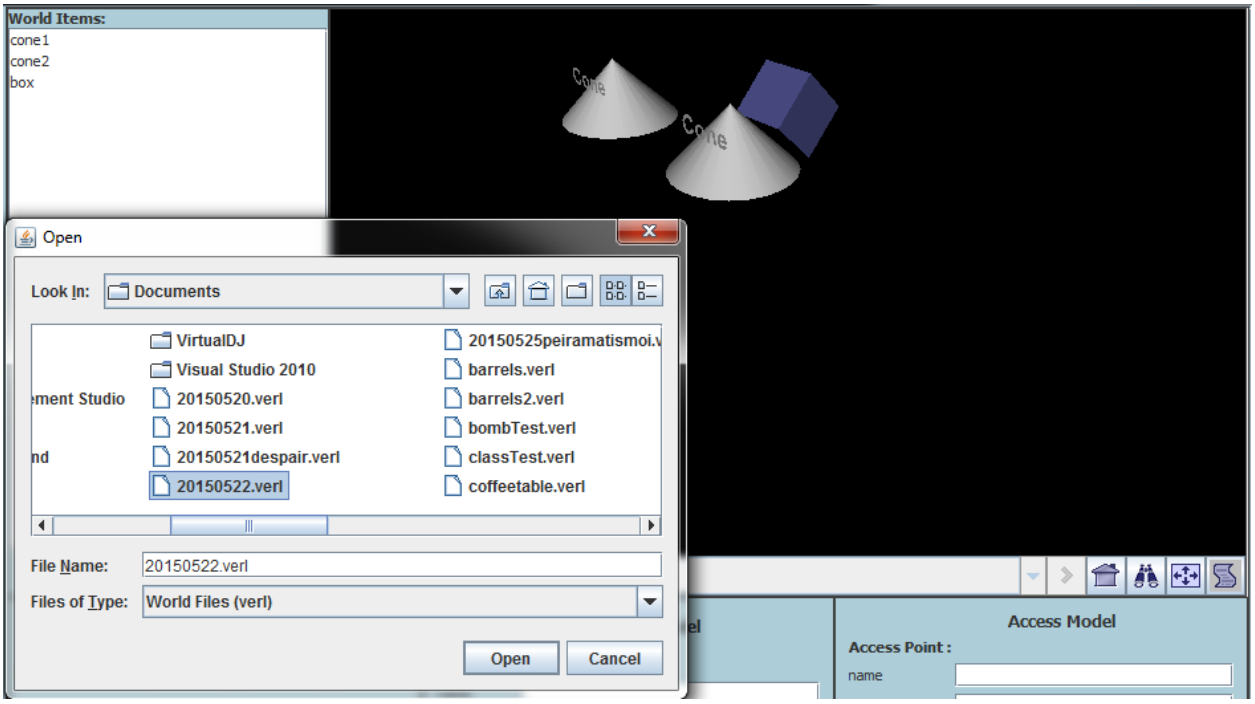
Παρακάτω παρουσιάζουμε ένα παράδειγμα, όπου φορτώνουμε αρχικά έναν κόσμο (test2.verl) και στην συνέχεια προσθέτουμε άλλον κόσμο (20150522.verl). Πατώντας Save, όπως φαίνεται και στο σχετικό μήνυμα, οι αλλαγές επηρεάζουν μόνο τον πρώτο κόσμο που φορτώθηκε.



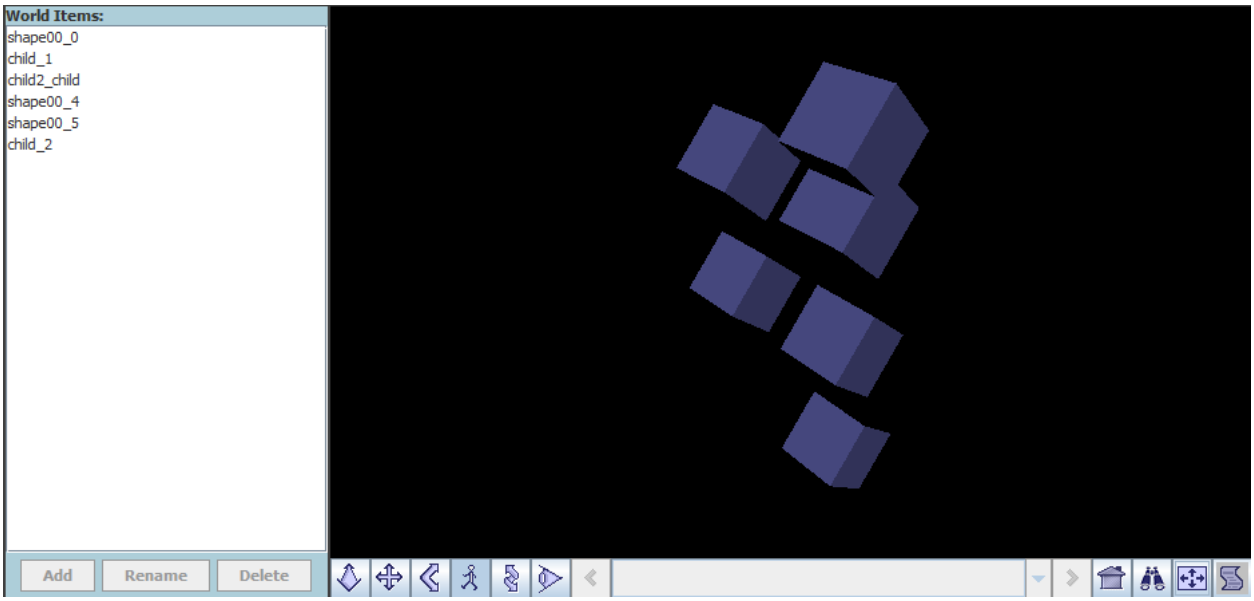
Εικόνα 6.54: Επιλογή φόρτωσης κόσμου test2.verl



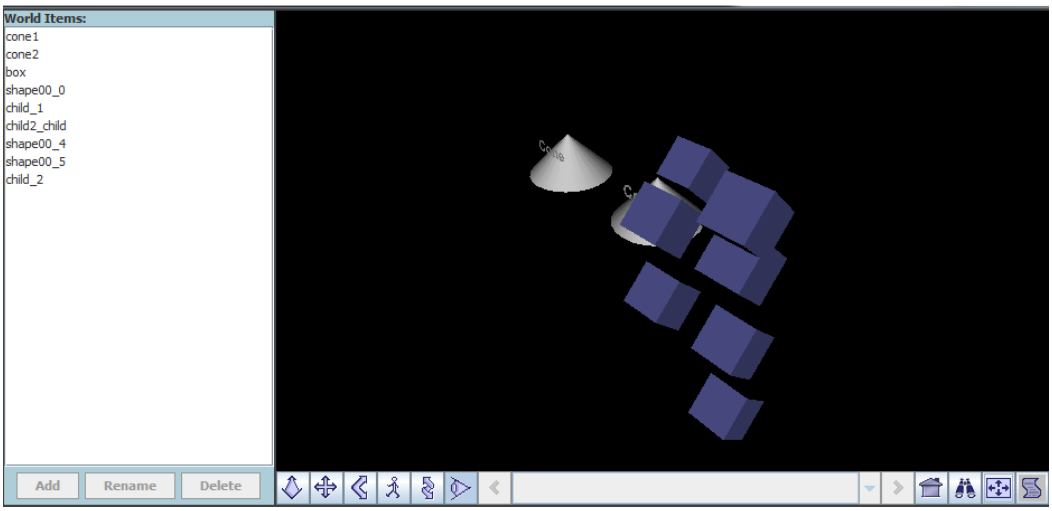
Εικόνα 6.55: Φόρτωση κόσμου test2.verl



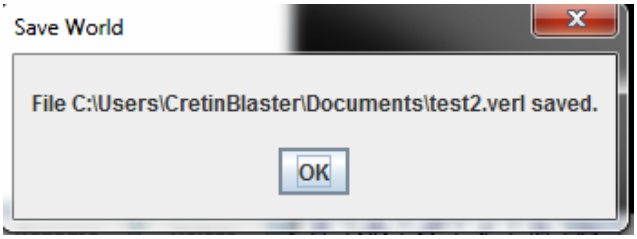
Εικόνα 6.56: Επιλογή προσθήκης κόσμου 20150522.verl



Εικόνα 6.57: Ο κόσμος 20150522.verl



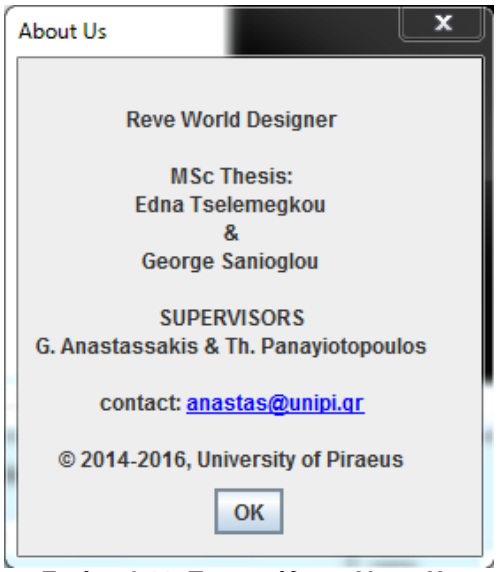
Εικόνα 6.58: Η σκηνή με τους 2 κόσμους μαζί



Εικόνα 6.59: Η αποθήκευση επηρεάζει μόνο τον πρώτο κόσμο

6.10 Πληροφορίες Εφαρμογής

Στο toolbar διαθέτουμε άλλα δυο κουμπιά. Το ένα, που λέγεται 'About Us'  απλά εμφανίζει στοιχεία για την δημιουργία της εφαρμογής και τα πνευματικά δικαιώματα, όπως φαίνεται παρακάτω.

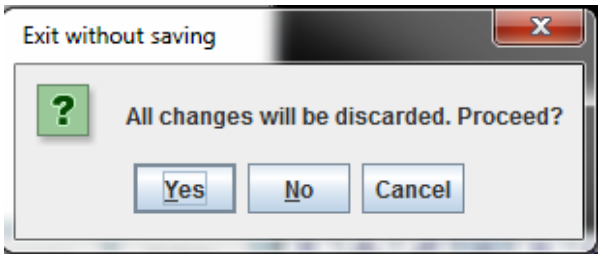


Εικόνα 6.60: Το παράθυρο About Us

Το δεύτερο, 'Help' ,  στοχεύει να κατατοπίσει τον χρήστη με πληροφορίες και οδηγίες χρήσης της εφαρμογής. Για αυτό το σκοπό έχουμε ενσωματώσει ένα pdf στο πακέτο της εφαρμογής το οποίο ανοίγει κατά το πάτημα του Help.

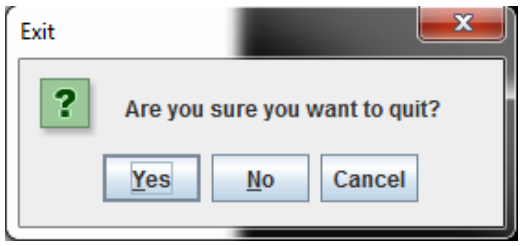
6.11 Έξοδος Από Την Εφαρμογή

Η τελευταία επιλογή που διατίθεται στον χρήστη, είναι βέβαια η έξοδος από την εφαρμογή. Για αυτό υπάρχει το κουμπί 'Exit without saving'  στο toolbar που ρωτάει τον χρήστη αν είναι βέβαιος για την απόφαση, γιατί οι όποιες μη σωσμένες αλλαγές θα χαθούν. Συνεπώς, για να κρατήσει ό,τι έχει κάνει, ο χρήστης θα πρέπει πρώτα να έχει πατήσει αποθήκευση και στην συνέχεια να πατήσει έξοδο.



Εικόνα 6.61: Επιβεβαίωση εξόδου από το 'Exit without Saving'

Πέρα από το συγκεκριμένο κουμπί στο toolbar, βέβαια, ο χρήστης μπορεί να κλείσει την εφαρμογή πατώντας απλά το 'X' του παραθύρου, όπου και εκεί διερωτάται για επιβεβαίωση της απόφασης ώστε να αποφευχούν λάθη.



Εικόνα 6.62: Επιβεβαίωση εξόδου από το 'X'

ΚΕΦΑΛΑΙΟ 7 – ΠΑΡΑΔΕΙΓΜΑ ΧΡΗΣΗΣ ΕΦΑΡΜΟΓΗΣ

7.1 Προδιαγραφές Λειτουργίας

Η εφαρμογή υποστηρίζεται από όλα τα λειτουργικά συστήματα, χάρη στην ανεξαρτησία που προσφέρει η Java από την πλατφόρμα εγκατάστασης. Έχει δοκιμαστεί επιτυχώς σε Windows Vista, Windows 7 και Windows 8, τόσο σε 32-bit όσο και σε 64-bit υπολογιστές.

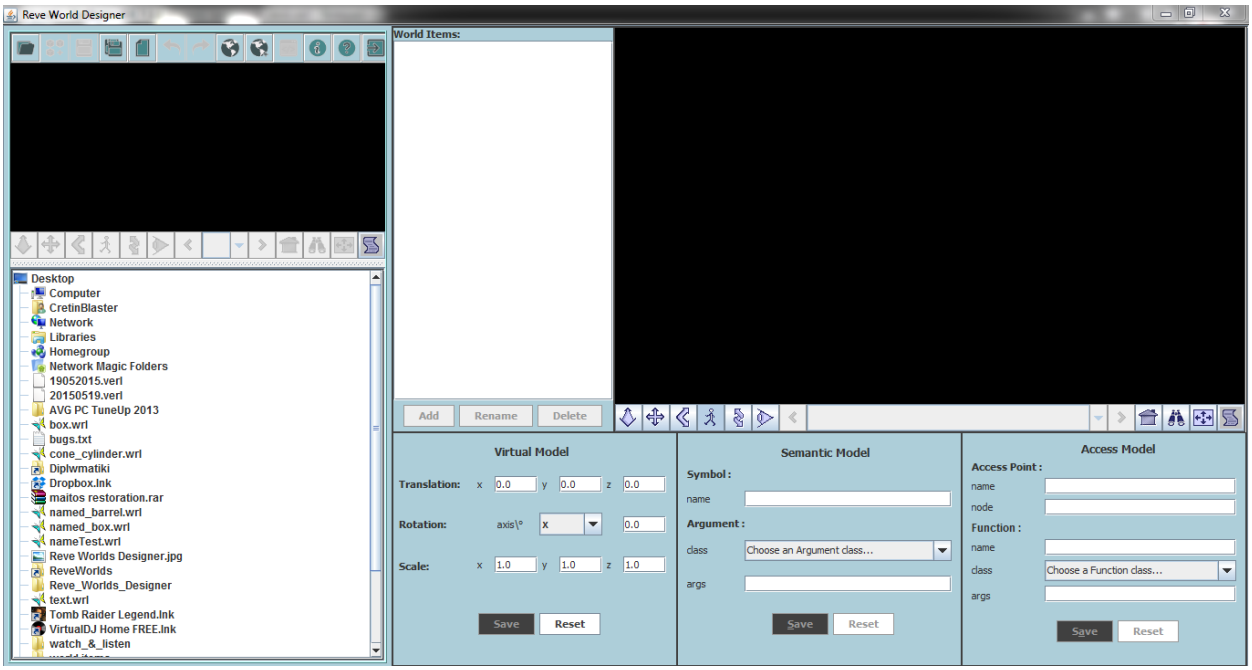
Βασική προϋπόθεση για να τρέξει η εφαρμογή είναι ο χρήστης να έχει στον υπολογιστή το Java Runtime Environment (JRE) έκδοση 7 και άνω για 32-bit πλατφόρμες. Στην περίπτωση που ο χρήστης έχει 64-bit υπολογιστή, θα πρέπει να φροντίσει να έχει 32-bit java, που θα την εντοπίσει στον φάκελο C:\Program Files (x86)\Java.

Για να εκτελέσει την εφαρμογή, ο χρήστης αντιγράφει τον φάκελο εγκατάστασης Reve World Designer στον υπολογιστή του, κατά προτίμηση στον φάκελο C:\ . Στην συνέχεια, αν έχει 32-bit υπολογιστή, εκτελεί το run.cmd αρχείο από τον φάκελο C:\Reve World Designer\x86, ενώ αν έχει 64-bit υπολογιστή εκτελεί το run.cmd αρχείο από τον φάκελο C:\Reve World Designer\x64. Σε περίπτωση που του προκύπτει σφάλμα θα πρέπει να σιγουρευτεί πως έχει το περιβάλλον της java σωστά εγκατεστημένο στον υπολογιστή του. Για τους 64-bit υπολογιστές, θα πρέπει να βεβαιωθεί πως στο αρχείο run.cmd το path της java είναι αυτό που έχει στον υπολογιστή του, διαφορετικά θα πρέπει να το αλλάξει με το σωστό.

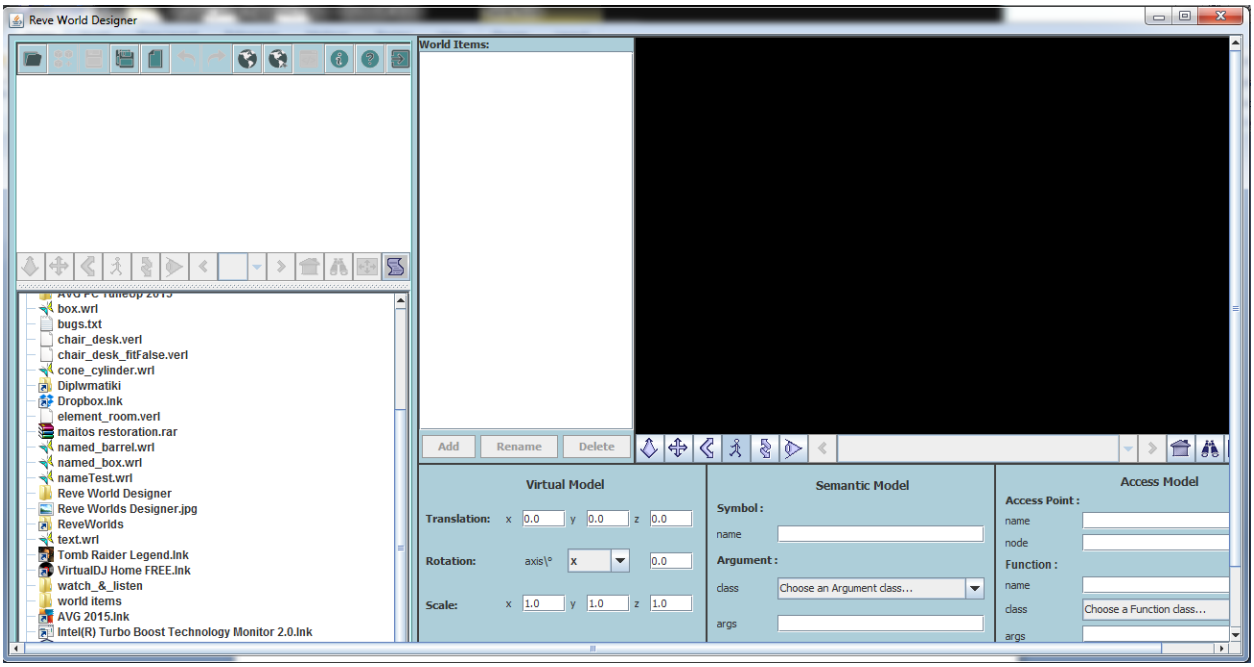
run.cmd

```
@echo off

set APP_CLASSPATH=ptyxiaki.jar
set MAIN_CLASS=ptyxiaki.ReveWorldsDesigner
set EXT_CLASSPATH=./ext
set JDK_MEM_OPTS=-Xmx512m

rem -----------------------------------------------------------------------------
rem Set application classpath...
rem -----------------------------------------------------------------------------

set CLASSPATH=%APP_CLASSPATH%;%EXT_CLASSPATH%


rem -----------------------------------------------------------------------------
rem Execute using console-based version of the Java VM...
rem -----------------------------------------------------------------------------

"C:\Program Files (x86)\Java\jre7\bin\java" %JDK_MEM_OPTS% -cp "%CLASSPATH%" %MAIN_CLASS% %1


rem -----------------------------------------------------------------------------
rem Pause on errors...
rem -----------------------------------------------------------------------------

if errorlevel 1 pause
```

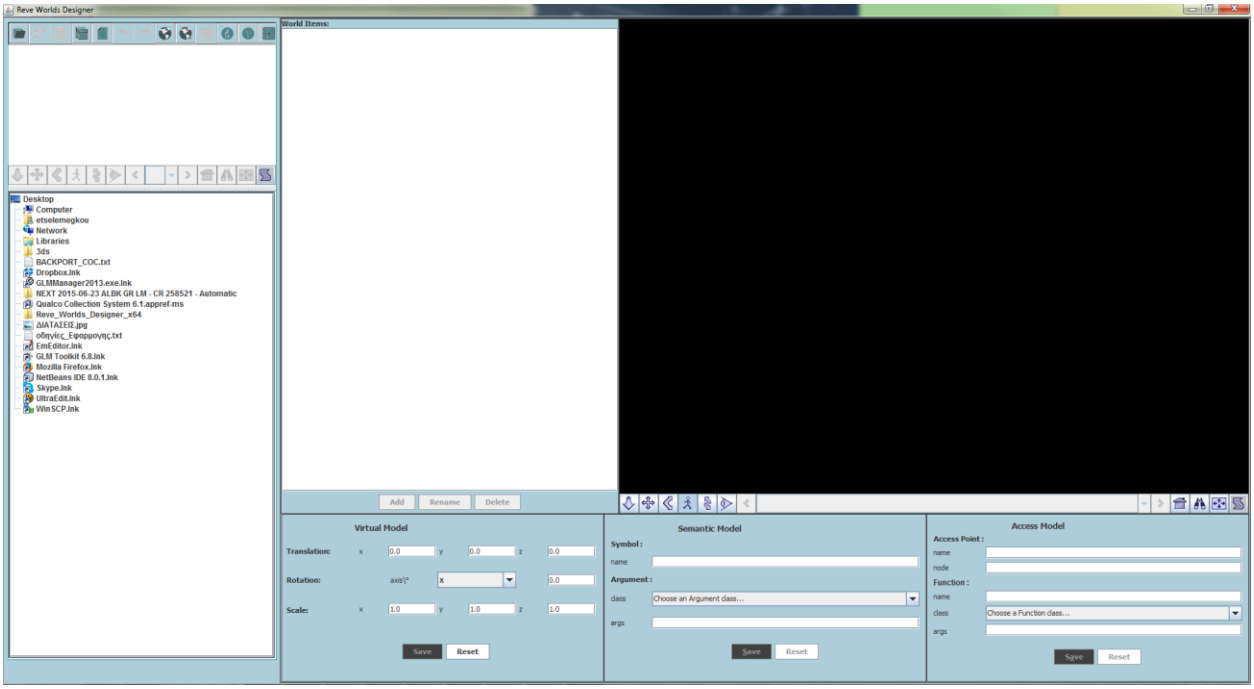
Κάνοντας τα παραπάνω βήματα σωστά, με την εκτέλεση του run.cmd ανοίγει η εφαρμογή όπως φαίνεται παρακάτω.



Εδώ να αναφέρουμε πως η εφαρμογή σχεδιάστηκε για οθόνη 15.6''. Για χρήση σε μικρότερη οθόνη η εφαρμογή «συρρικνώνεται» και ο χρήστης μπορεί να περιηγηθεί με τη χρήση των scrollbars που ανοίγουν δεξιά και κάτω από τα παράθυρα, όπως φαίνεται παρακάτω:



Τέλος, για χρήση σε μεγαλύτερη οθόνη η εφαρμογή «εκτείνεται» ανάλογα, όπως φαίνεται παρακάτω:

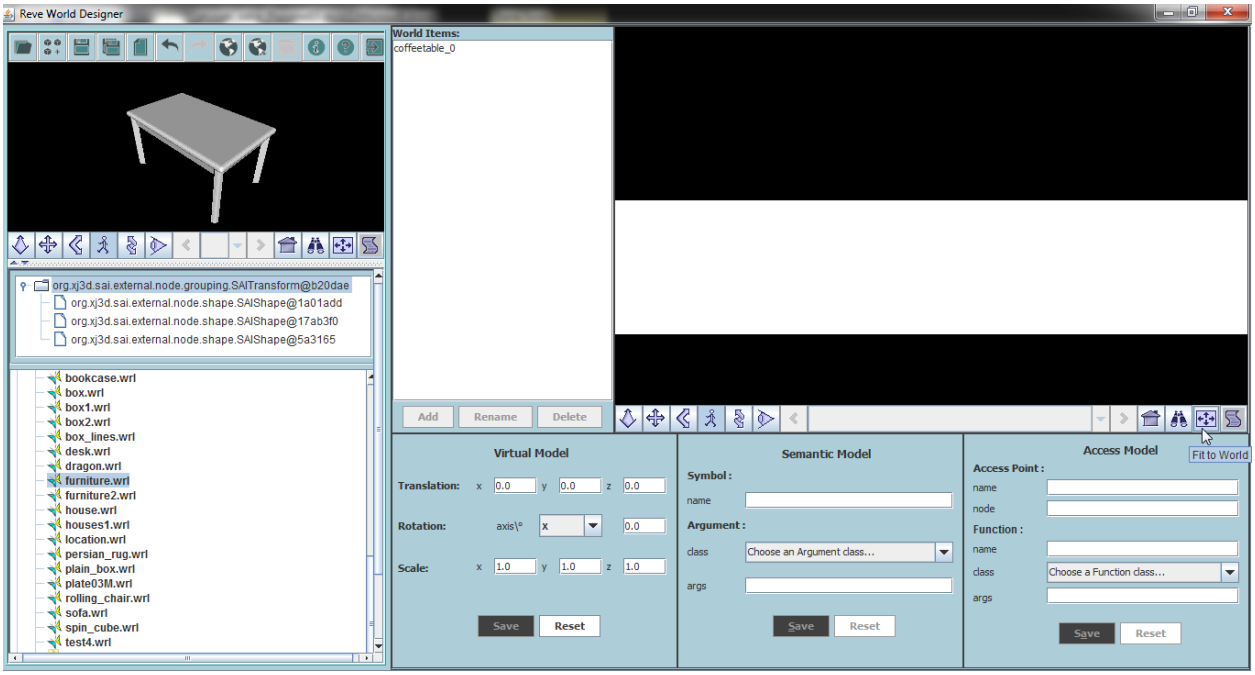


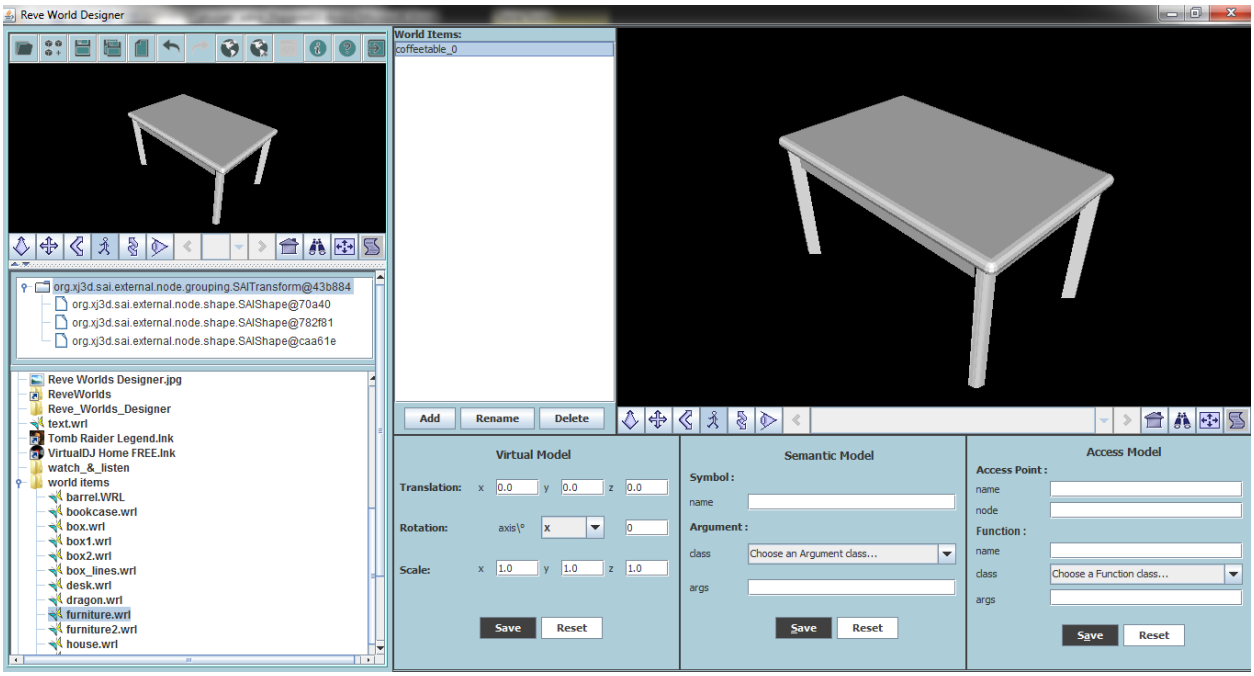
Σχετικές λειτουργίες εγκατάστασης υπάρχουν και στο αρχείο README.TXT. Με το άνοιγμα της εφαρμογής, ο χρήστης μπορεί πλέον να αρχίσει τη σχεδίαση ενός εικονικού κόσμου. Αν τέλος, ο χρήστης θέλει να απεγκαταστήσει την εφαρμογή αρκεί να διαγράψει τον φάκελο Reve World Designer από τον υπολογιστή του.

7.2 Παράδειγμα Σχεδίασης Ενός Εικονικού Κόσμου

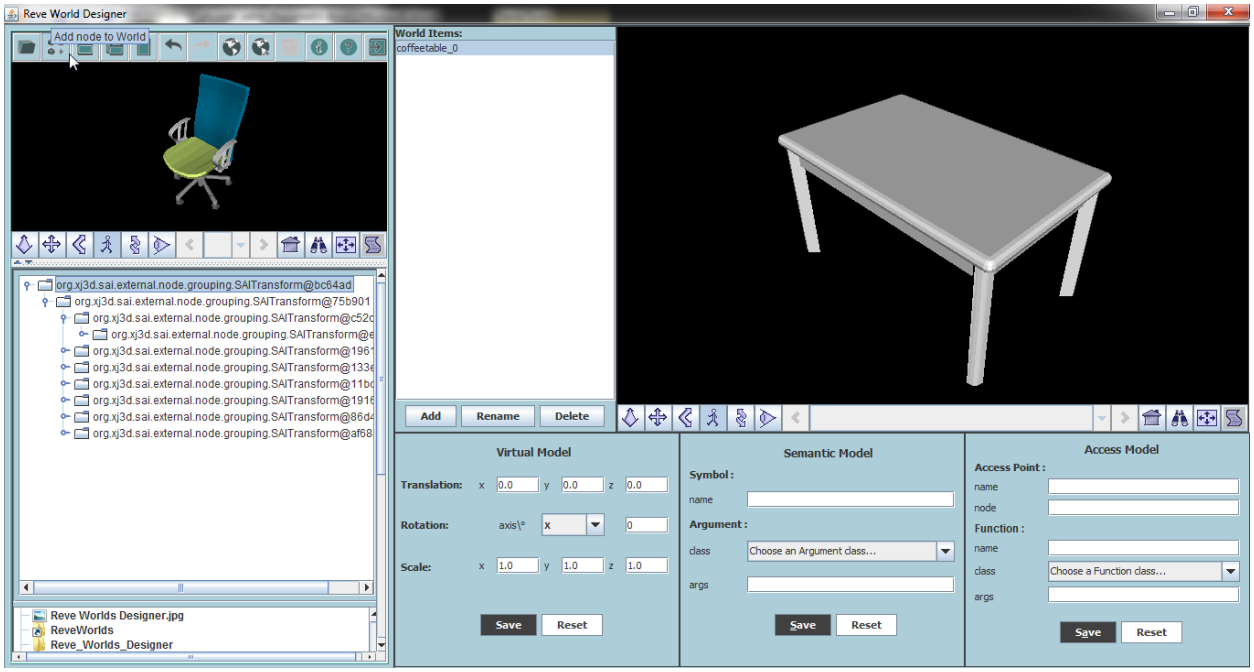
Σε αυτό το κεφάλαιο θα παρουσιάσουμε ένα πλήρες παράδειγμα σχεδίασης ενός εικονικού κόσμου μέσω της εφαρμογής Reve World Designer, παρουσιάζοντας βήμα βήμα τις επιλογές που κάνουμε μέσω εικόνων. Τέλος, θα δούμε πώς αυτός ο κόσμος εμφανίζεται στην πλατφόρμα Reve Worlds, σε συνάρτηση με τα ζητήματα που αναπτύξαμε στο Κεφάλαιο 6.

Χρησιμοποιούμε το αριστερό παράθυρο για να επιλέξουμε τα αντικείμενα που θέλουμε για τον κόσμο, εφόσον τα περισσότερα τα έχουμε σε φάκελο στην επιφάνεια εργασίας και μας βολεύει η αναζήτηση από εκεί. Αρχικά, επιλέγουμε το κύριο αντικείμενο που θέλουμε να αποτελέσει το κέντρο του κόσμου μας, σε αυτό το παράδειγμα το παρακάτω τραπέζι, το οποίο είναι αρκετά μεγάλο για την σκηνή μας, οπότε για να το δούμε πλήρως πρέπει να πατήσουμε την επιλογή 'Fit to World'.

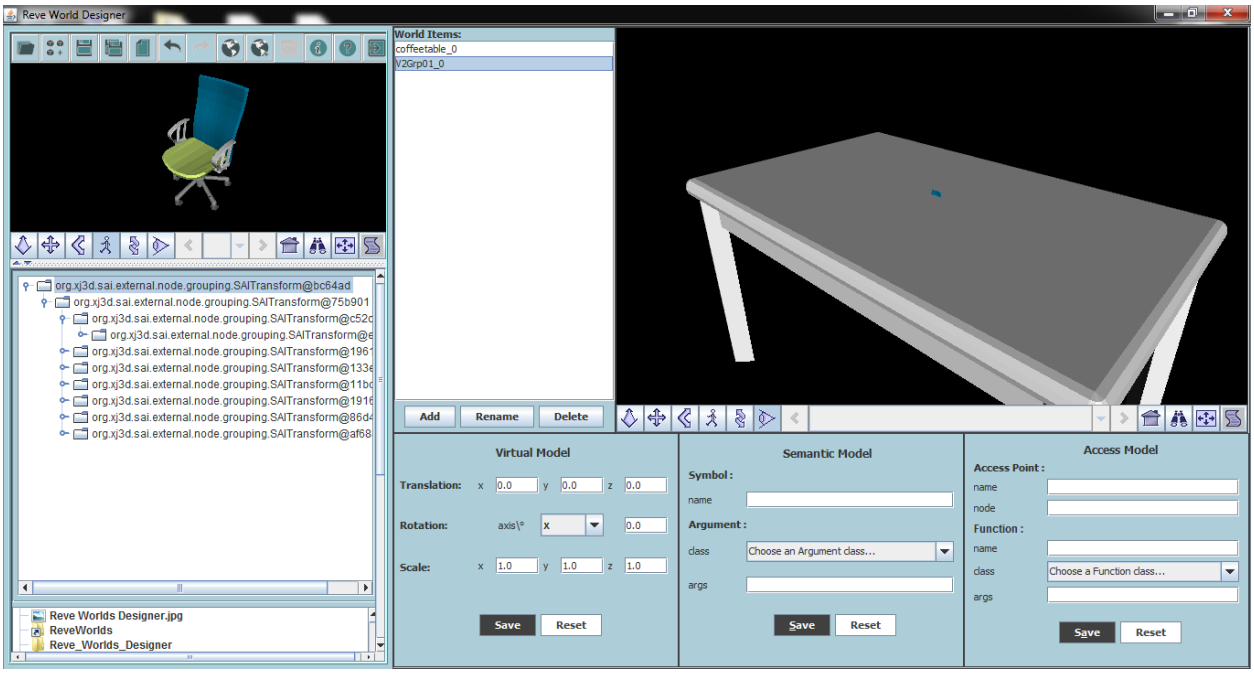




Έπειτα, επιλέγουμε το επόμενο αντικείμενο για να το προσθέσουμε στον κόσμο και είναι μια καρέκλα.

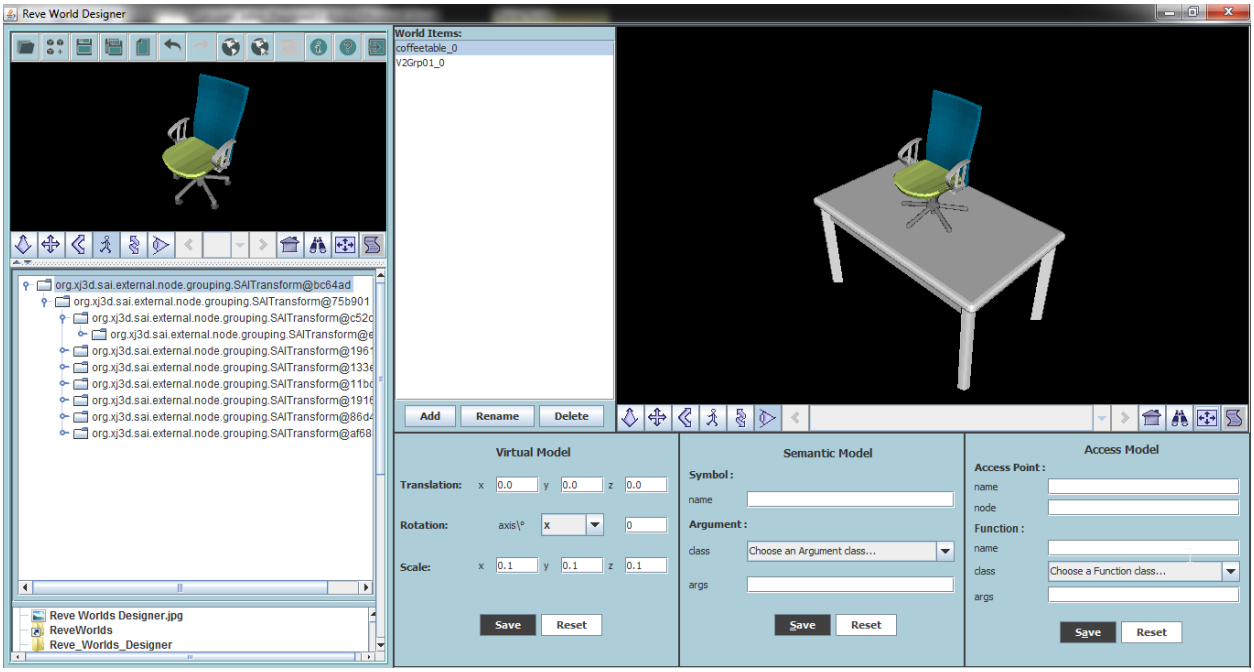


Παρατηρούμε πως προσθέτοντας την καρέκλα, αυτή εμφανίζεται χωρικά στη μέση του τραπεζιού και κρύβεται από αυτό, γιατί το τραπέζι έχει πολύ μεγάλο μέγεθος σε σχέση με την καρέκλα επικαλύπτοντάς την, όπως φαίνεται στην παρακάτω εικόνα, όπου ίσα που ξεχωρίζει μια λεπτή μπλε γραμμή από την καρέκλα πάνω από το τραπέζι.

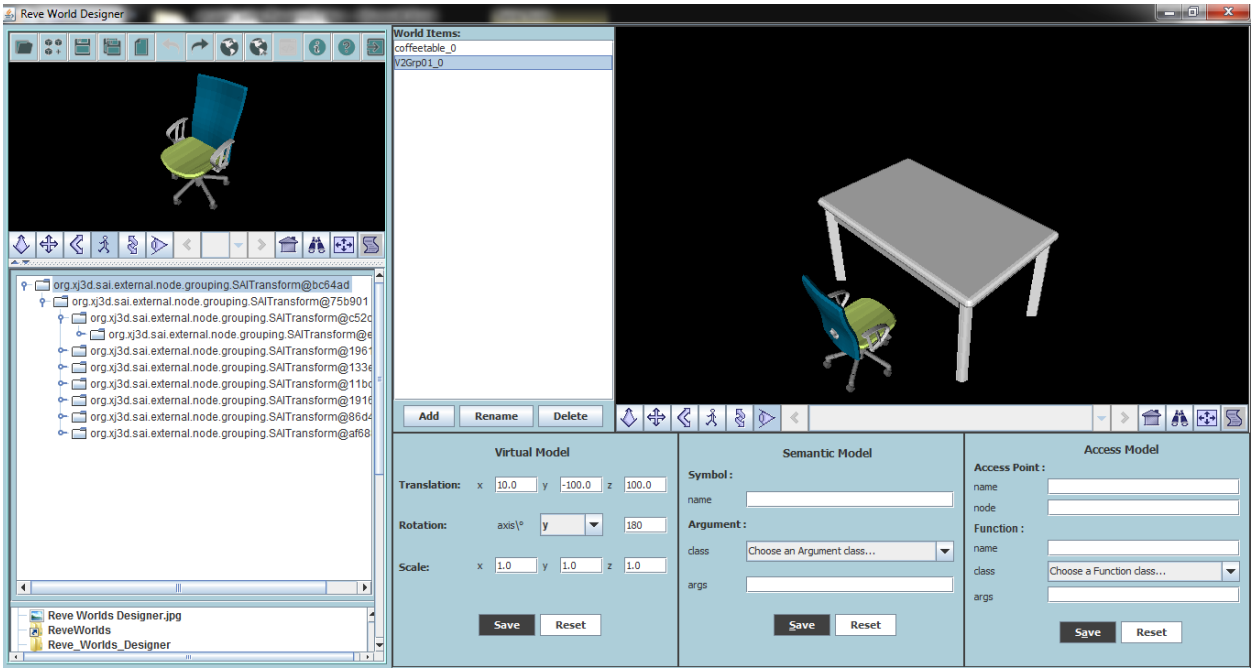


Αντιλαμβανόμαστε πως είτε πρέπει να μεγενθύνουμε την καρέκλα, είτε πρέπει να μικρύνουμε το τραπέζι και βέβαια, να μετακινήσουμε το ένα από τα δυο ώστε να μη συμπίπτει το ένα πάνω στο άλλο, αλλά η καρέκλα να βρίσκεται μπροστά από το τραπέζι. Όπως είδαμε κατά την προσθήκη του τραπεζιού στον κόσμο, είναι πολύ

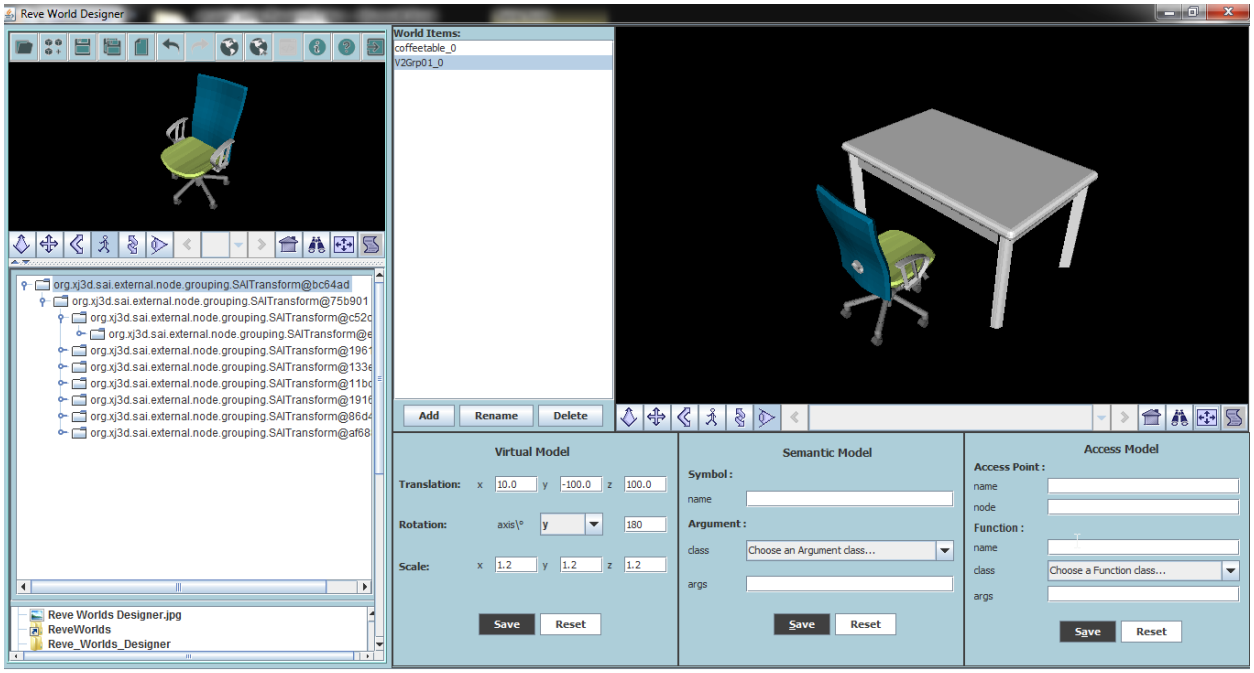
μεγάλο και αποφασίζουμε να πειράξουμε αυτό. Αρχικά, λοιπόν, πειραματιζόμενοι με την κλίμακα του τραπεζιού, το μικραίνουμε ομοιόμορφα κατά 0.1 φορά, βελτιώνοντας το μέγεθος του σε σχέση με την καρέκλα, όπως φαίνεται στην παρακάτω εικόνα.



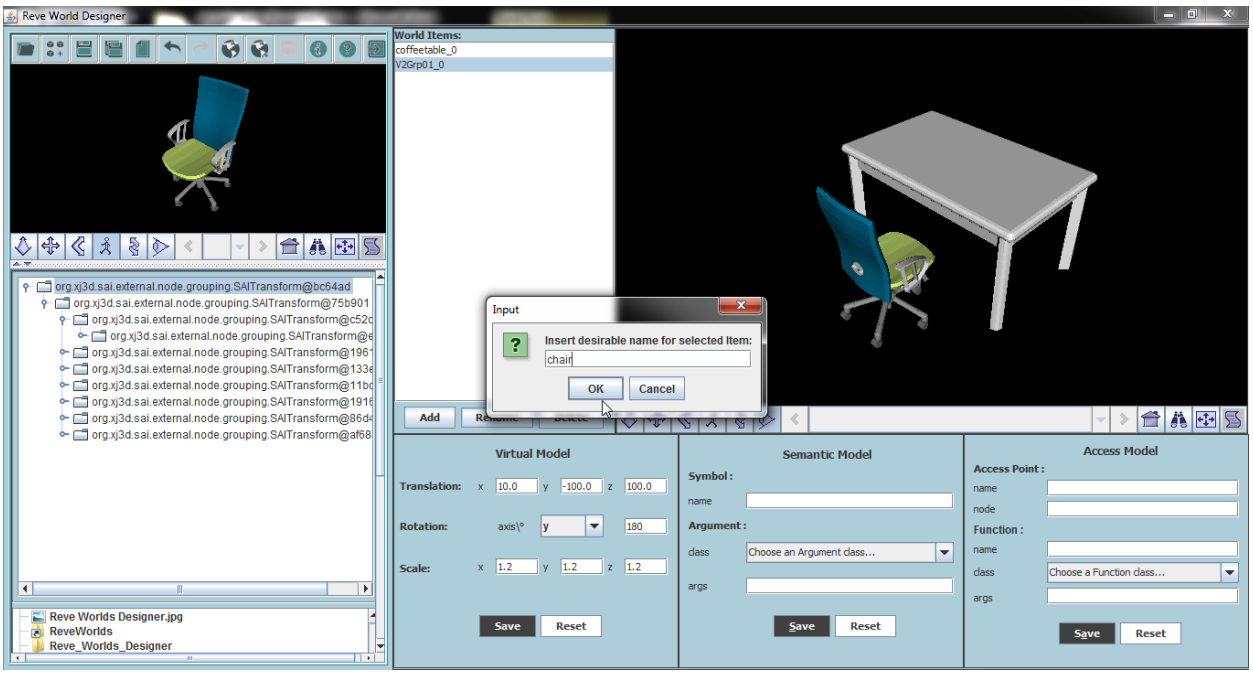
Το επόμενο βήμα είναι να φέρουμε την καρέκλα μπροστά από το τραπέζι. Αυτό συνεπάγεται η καρέκλα να μετακινηθεί α) προς τα εμπρός, δηλαδή θετικά ως προς τον άξονα z (100 μέτρα), β) προς τα δεξιά, δηλαδή θετικά ως προς τον άξονα x (10 μέτρα), κα ιγ) να χαμηλώσει ύψος, δηλαδή αρνητική μετακίνηση ως προς τον άξονα y (-100 μέτρα). Επίσης, πρέπει να περιστραφεί κατά 180 μοίρες ως προς τον άξονα y ώστε γυρίσει η πλάτη της από την άλλη πλευρά.



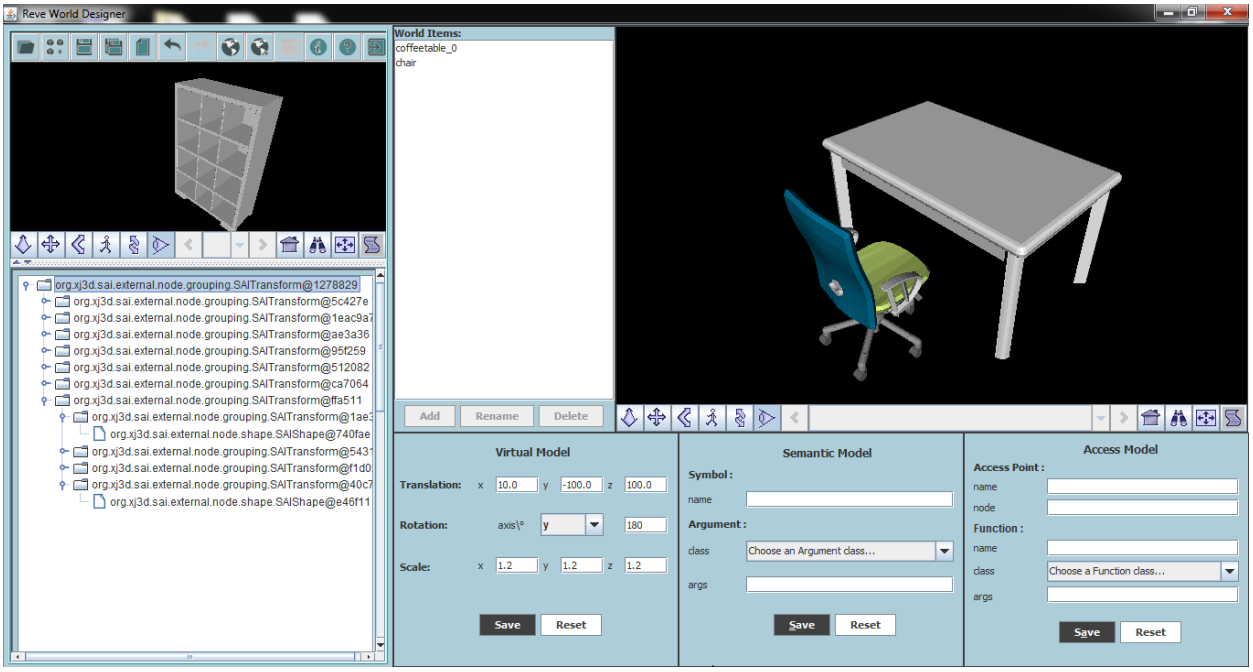
Το γεγονός ότι η μετακίνηση στους άξονες x και y είναι τόσο μεγάλη οφείλεται στο μέγεθος του κόσμου, που καθιστούν μια μετακίνηση ανά ένα μέτρο σχεδόν ανεπαίσθητη. Παρατηρούμε ακόμα πως αλλάζοντας την θέση της καρέκλας φαίνεται τελικά να είναι λίγο μικρή για το τραπέζι, οπότε την μεγενθύνουμε ομοιόμορφα κατά 1.2 φορές για να αντιστοιχεί στην ίδια αναλογία με το τραπέζι, όπως φαίνεται παρακάτω.



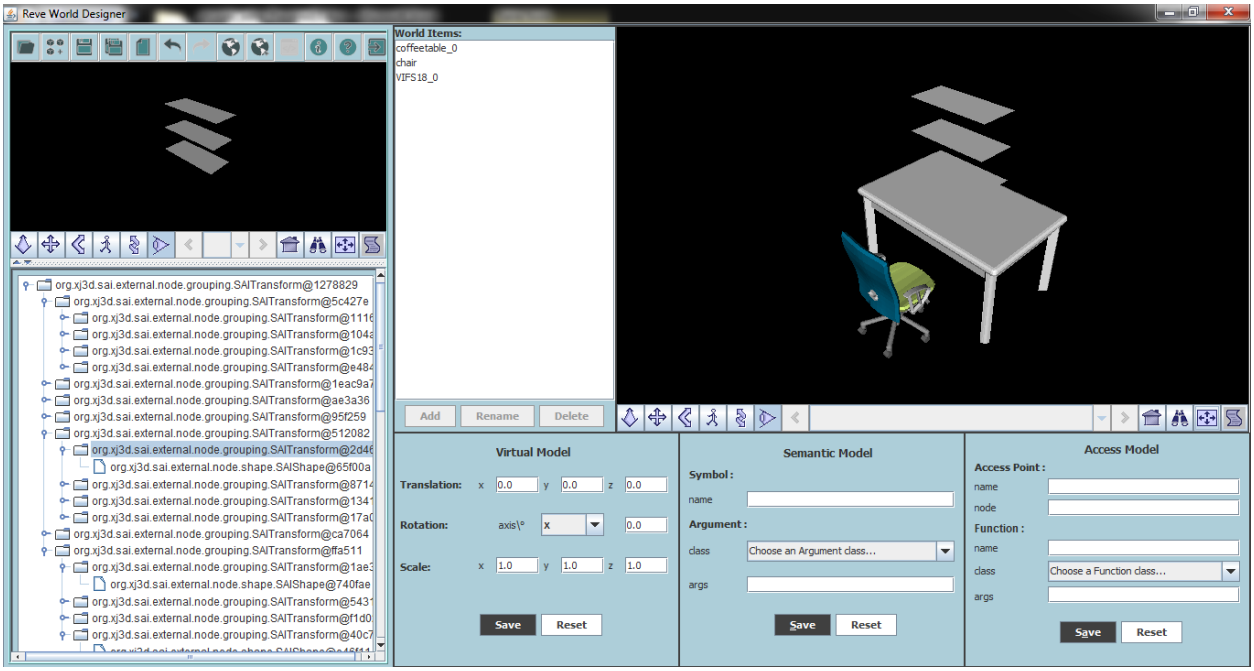
Τέλος, αλλάζουμε το όνομα της καρέκλας, γιατί είναι δυσνόητο και θέλουμε να προσθέσουμε και άλλα αντικείμενα, οπότε θέλουμε να έχουμε μια καλύτερη εικόνα των αντικειμένων που βρίσκονται στον κόσμο μας.



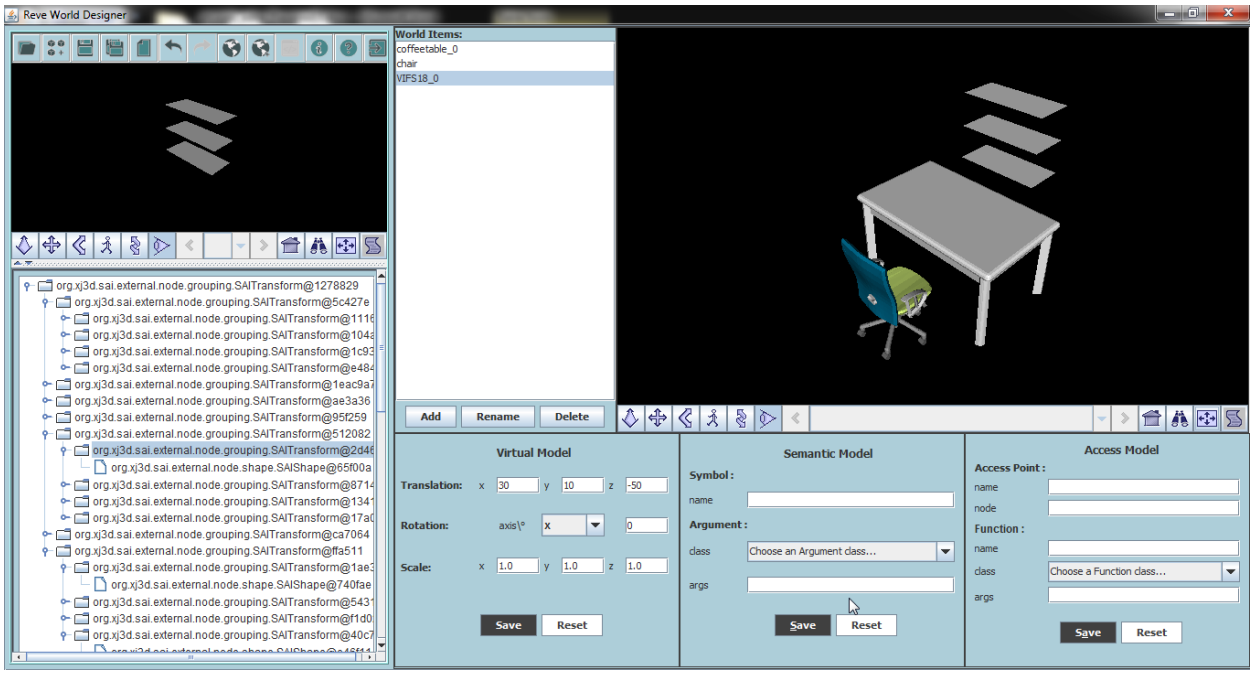
Το επόμενο αντικείμενο που θέλουμε να προσθέσουμε είναι μια βιβλιοθήκη.



Δεν μας αρέσει, όμως, το συγκεκριμένο έπιπλο, γιατί το βρίσκουμε πολύ μεγάλο και προτιμάμε να προσθέσουμε μόνο κάποια ράφια στον κόσμο μας, οπότε διαλέγουμε ένα υπο-γράφημα από το γράφημα σκηνής της βιβλιοθήκης και το προσθέτουμε στον κόσμο μας, όπως φαίνεται παρακάτω.

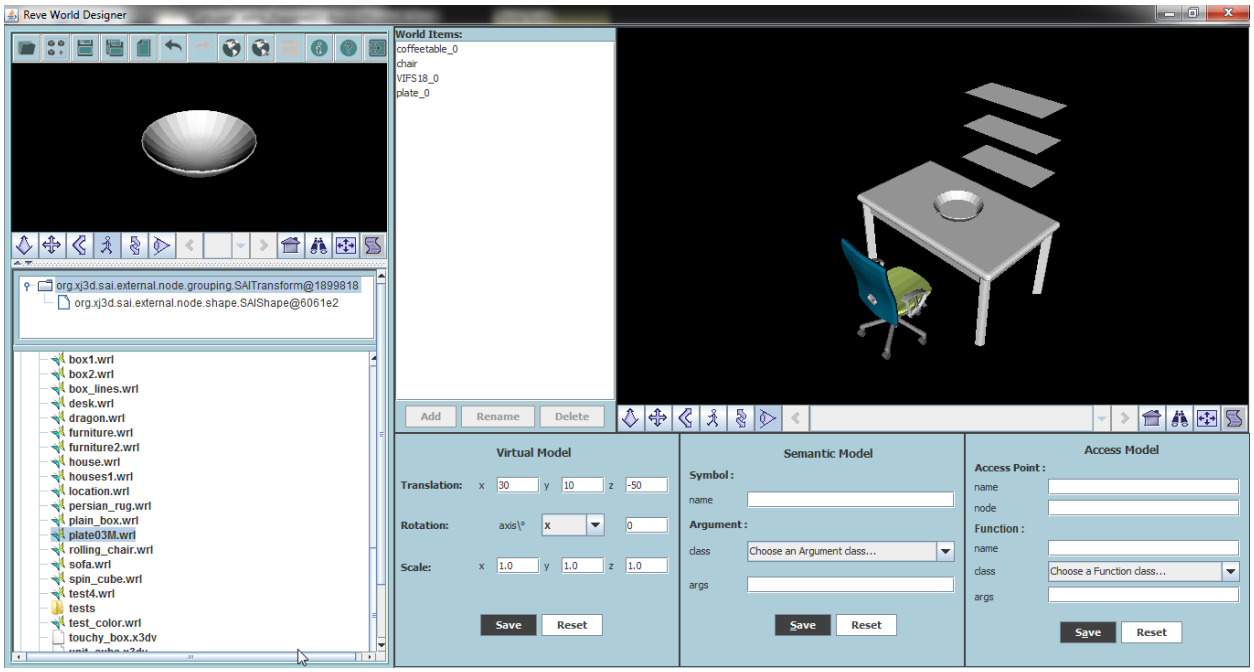


Τα ράφια συμπίπτουν με την θέση του τραπεζιού και δεν φαίνονται καλά, επομένως αυτό που κάνουμε είναι να αλλάξουμε το virtual model τους για να τα τοποθετήσουμε ψηλά και πίσω από το τραπέζι. Αλλάζουμε το translation ώστε τα ράφια να μετακινηθούν α) προς τα πίσω, δηλαδή αρνητική μετατόπιση στον άξονα z (-50 μέτρα), β) προς τα δεξιά, δηλαδή θετική μετατόπιση ως προς τον άξονα x (30 μέτρα) και γ) προς τα πάνω, δηλαδή θετική μετατόπιση ως προς τον άξονα y (10 μέτρα).

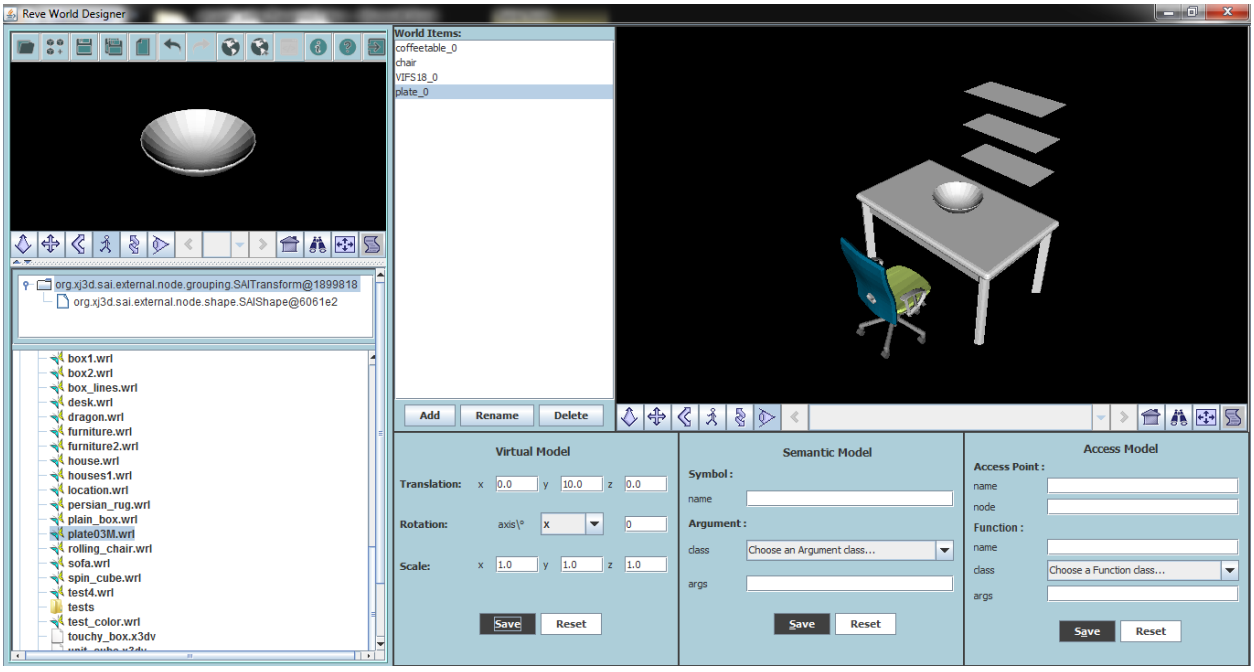


Στη συνέχεια, πάμε να τοποθετήσουμε κάποια αντικείμενα στα ράφια και στο τραπέζι για να μην είναι τόσο άδεια και ο κόσμος να θυμίζει περισσότερο έναν πραγματικό χώρο. Αρχικά, προσθέτουμε μια διακοσμητική

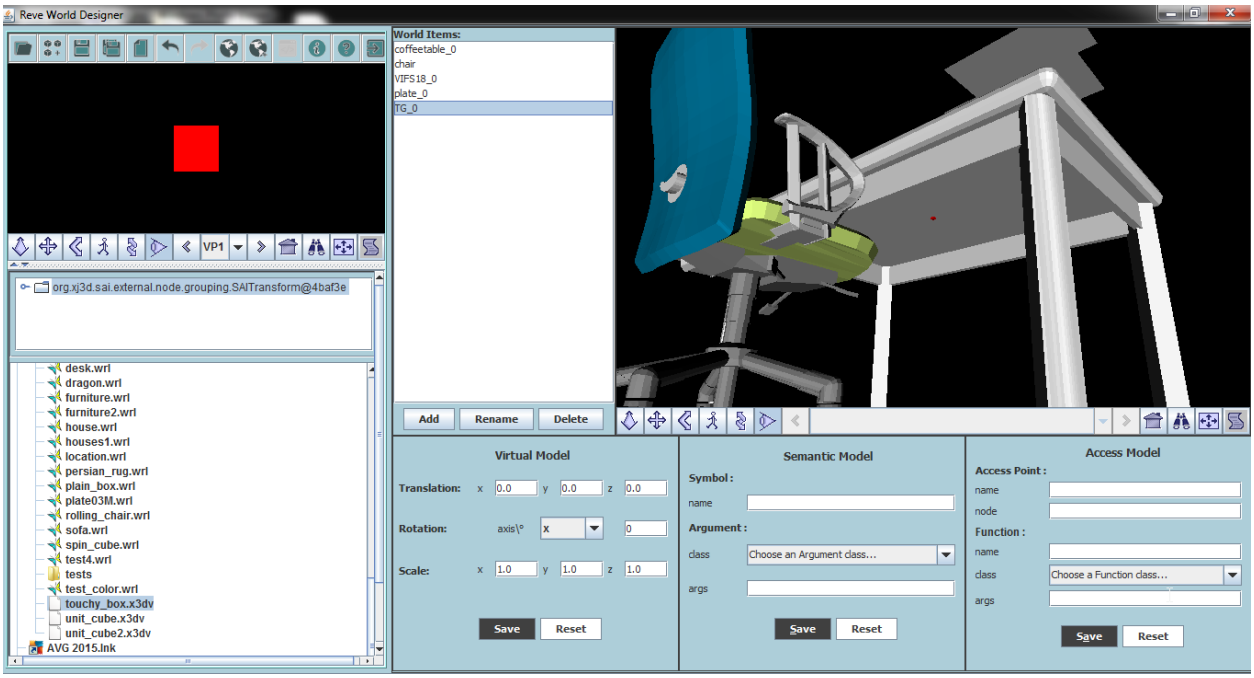
πιατέλα, η οποία τυχαίνει να συμπίπτει με το κέντρο του τραπεζιού, ωστόσο ο πάτος της χάνεται μέσα στο τραπέζι, σαν να βυθίζεται.



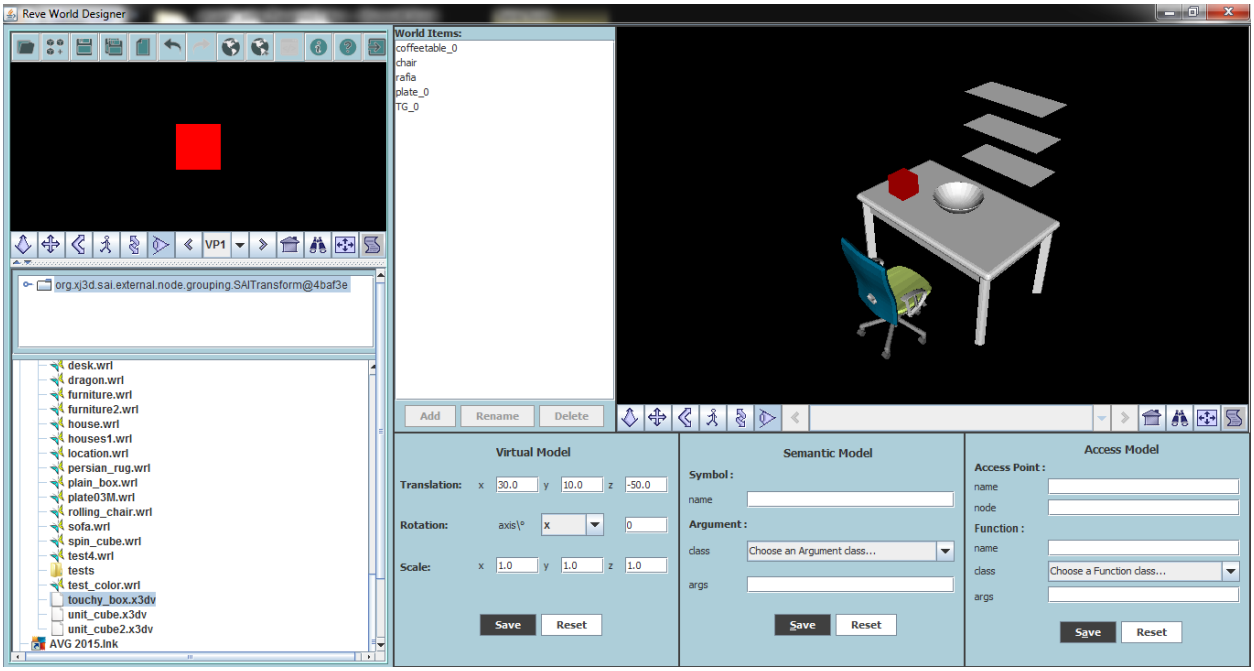
Για να φέρουμε την πιατέλα να πατάει πάνω στο τραπέζι, αλλάζουμε το translation της μετατοπίζοντάς την θετικά 10 μέτρα κατά τον άξονα y.



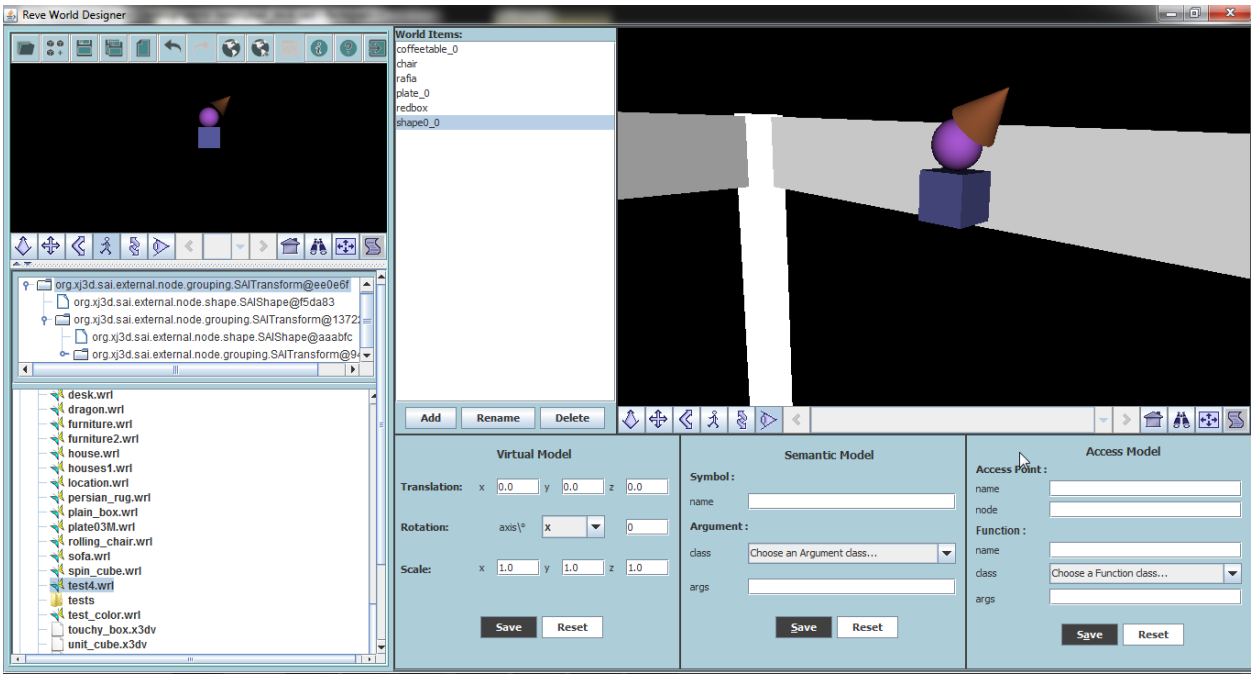
Αλλάζουμε και το όνομα των ραφιών, ώστε να είναι κατανοητό και προχωράμε στην επόμενη προσθήκη αντικειμένου, ενός μικρού κόκκινου κουτιού. Μάλιστα, το αντικείμενο είναι τόσο μικρό που δεν φαίνεται κατά την προσθήκη του στον κόσμο και μόνο με zoom-in στην σκηνή το βλέπουμε αμυδρά κάτω από το τραπέζι.



Επιβάλλεται, λοιπόν, να αλλάξουμε την κλίμακα του κουτιού, οπότε το μεγενθύνουμε ομοιόμορφα κατά 12.0 φορές για να έρθει στην ίδια αναλογία με τα υπόλοιπα. Επίσης, πειράζουμε και το translation του ώστε α) να ανέβει προς τα πάνω, δηλαδή θετική μετατόπιση στον άξονα y (10 μέτρα), β) να έρθει προς τα αριστερά, δηλαδή αρνητική μετατόπιση στον άξονα x (-60) και γ) προς τα μπροστά για να μην ακουμπάει την πιατέλα, με θετική μετατόπιση στον άξονα z (20). Το αποτέλεσμα είναι το παρακάτω.

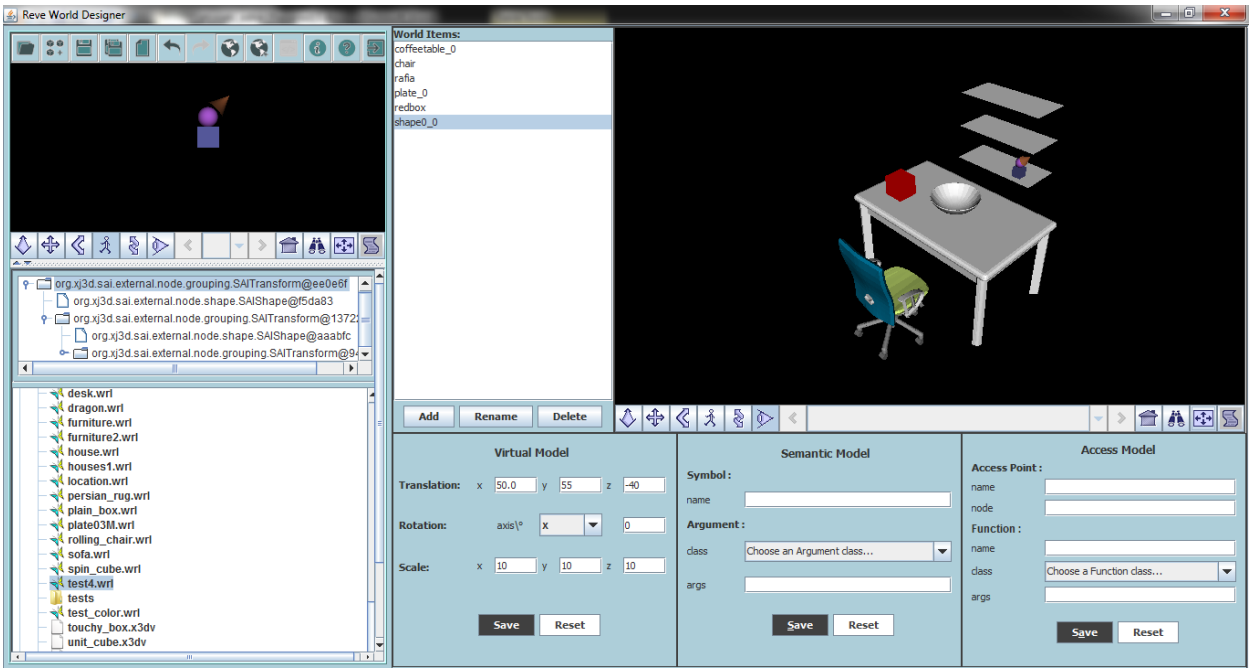


Μετονομάζουμε και αυτό το αντικείμενο σε 'red box' για να είναι διακριτό στην λίστα και προχωράμε στα επόμενα, με στόχο αυτά να τα προσθέσουμε στην ραφιέρα. Διαλέγουμε άλλο ένα διακοσμητικό αντικείμενο, το οποίο, όμως, είναι τόσο μικροσκοπικό σε σχέση με τα υπόλοιπα που κρύβεται και αυτό κάτω από το τραπέζι όπως το εντοπίσαμε με πολύ zoom-in.

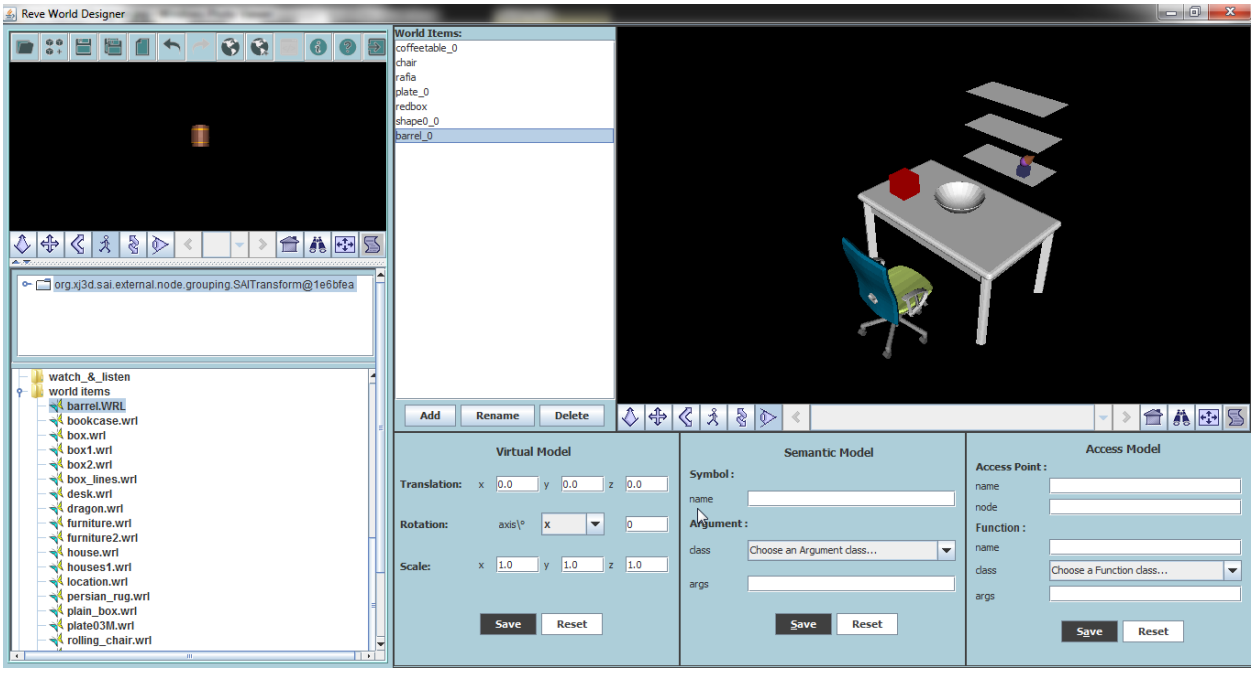


Αποφασίζουμε να το τοποθετήσουμε στη δεξιά γωνιά του πιο χαμηλού ραφιού. Αλλάζουμε πρώτα την κλιμακά του, μεγενθύνοντάς το ομοιόμορφα κατά 10 φορές. Έπειτα αλλάζουμε και το translation του, με θετική μετατόπιση στον άξονα y (55 μέτρα) για να ανέβει προς τα πάνω, θετική μετατόπιση στον άξονα x (50

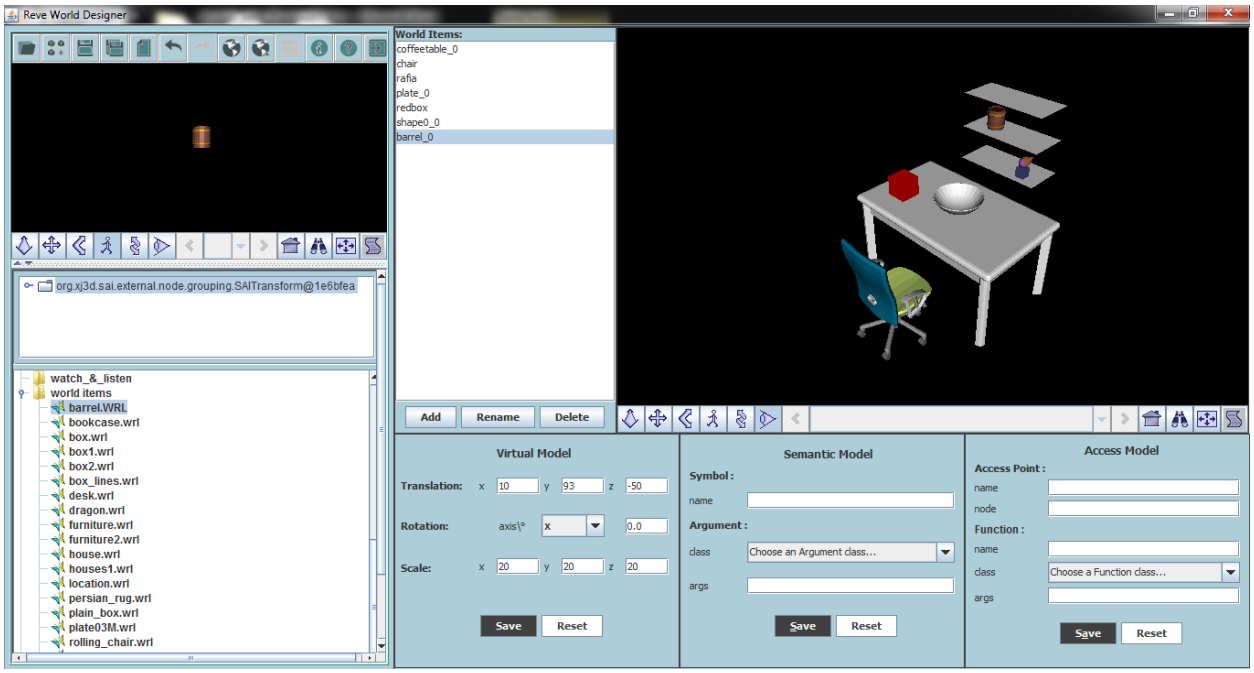
μέτρα) για να μετακινηθεί προς τα δεξιά και αρνητική μετατόπιση στον άξονα z (-40 μέτρα) για να μετακινηθεί προς τα πίσω.



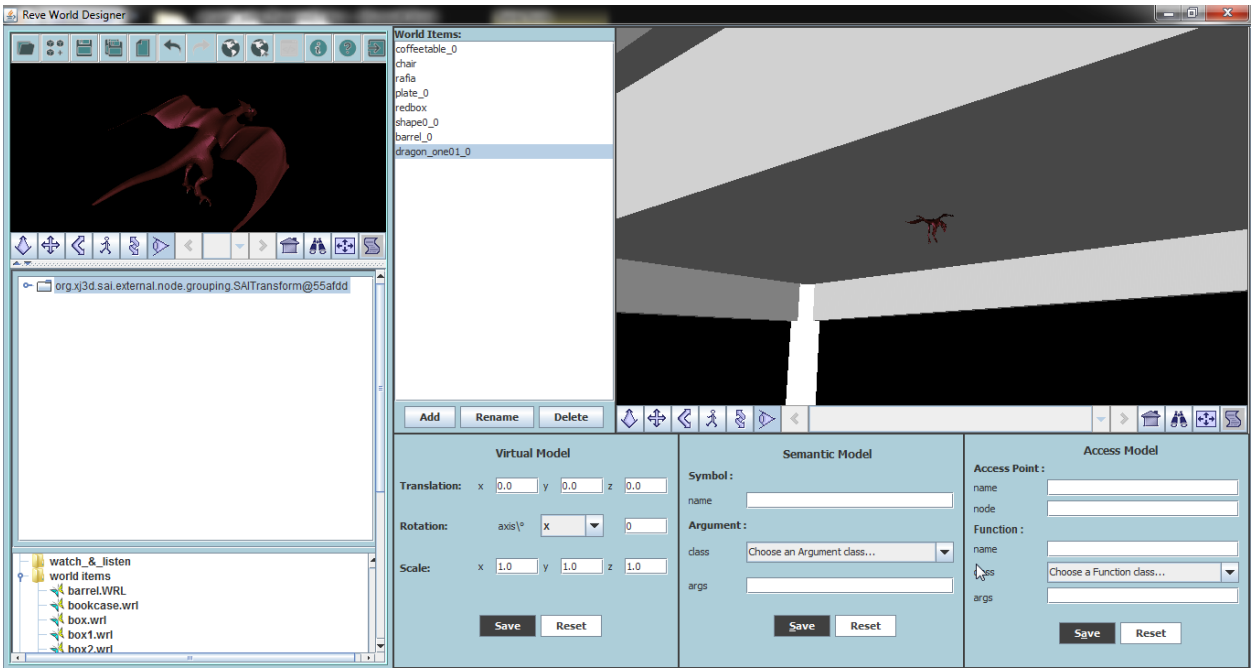
Στο δεύτερο ράφι αποφασίζουμε να τοποθετήσουμε ένα μικρό βαρέλι. Αρχικά είναι και αυτό πολύ μικρό για να φανεί κατά την προσθήκη στην σκηνή.



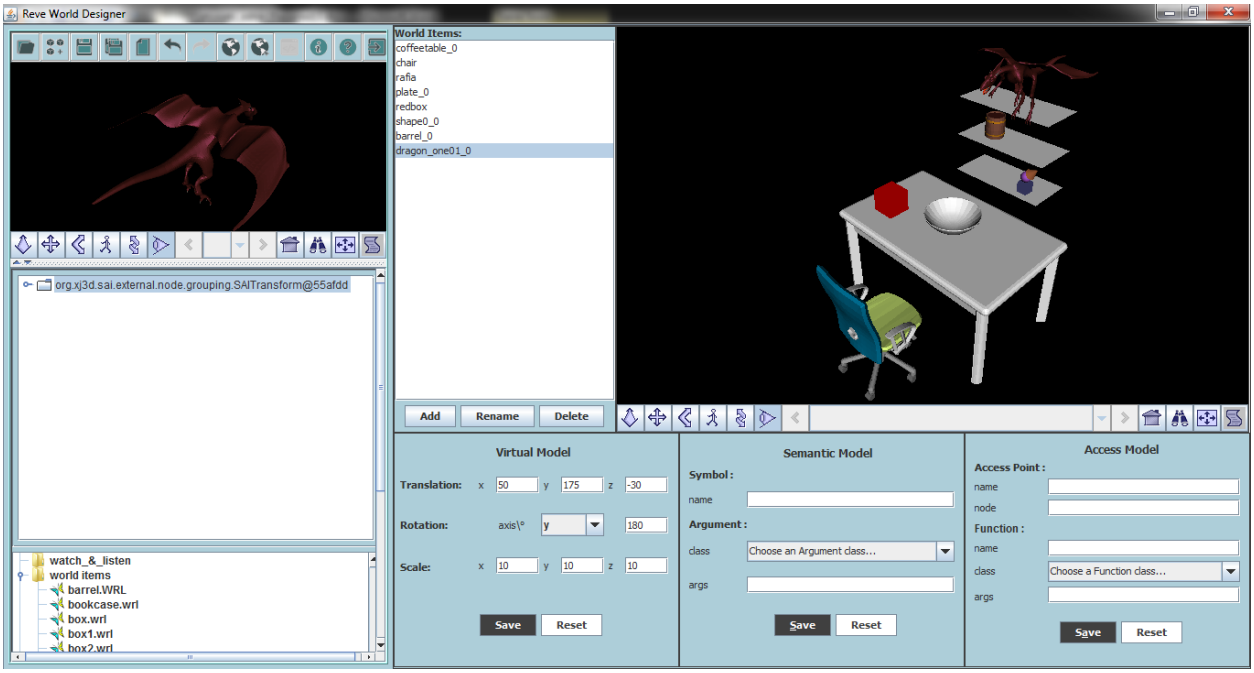
Αλλάζουμε το scale του, μεγενθύνοντάς το κατά 20 φορές και το translation του για να βρεθεί στην αριστερή γωνία του δεξιού ραφιού ως εξής: μετατόπιση κατά 10 μέτρα στον άξονα x, κατά 93 μέτρα στον άξονα y και κατά -50 στον άξονα z.



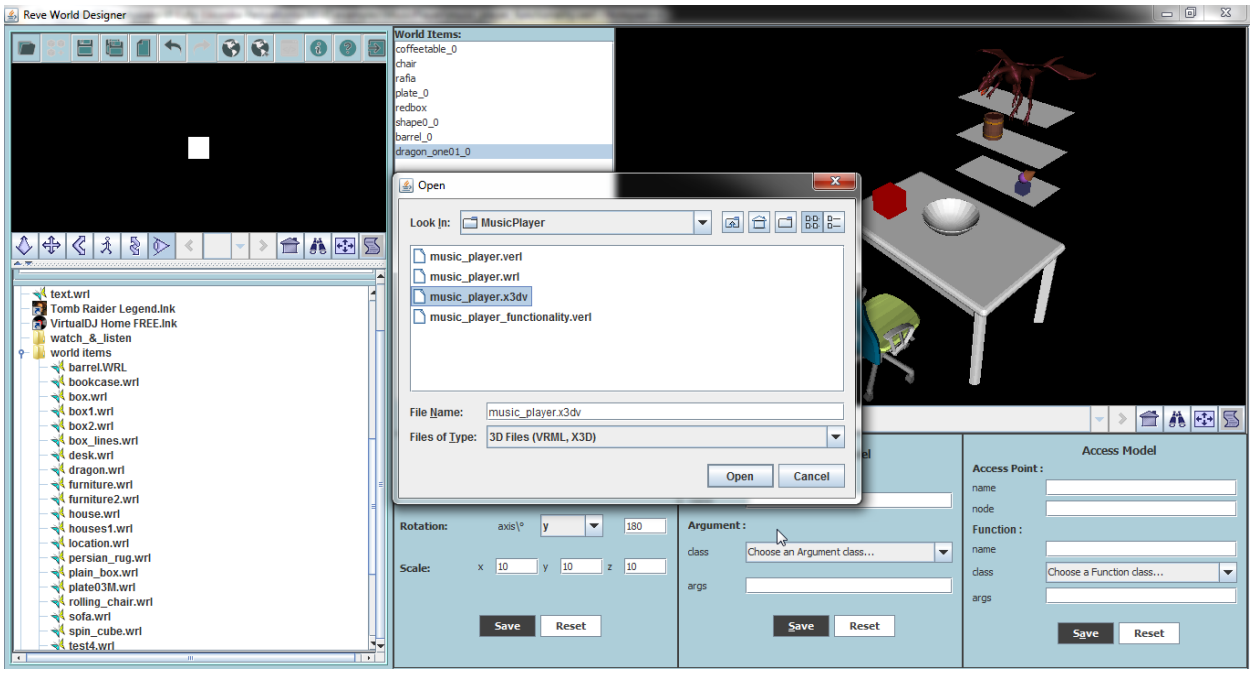
Στο τελευταίο ράφι αποφασίζουμε να προσθέσουμε έναν δράκο, ώστε να φαίνεται ουσιαστικά σαν παιχνίδι πάνω στο ράφι. Παρατηρούμε πως είναι πολύ μικρό το μέγεθός του κι επικαλύπτεται και αυτός από το τραπεζι κατά την προσθήκη του στον κόσμο, οπότε τον εντοπίζουμε με zoom-in στην σκηνή.

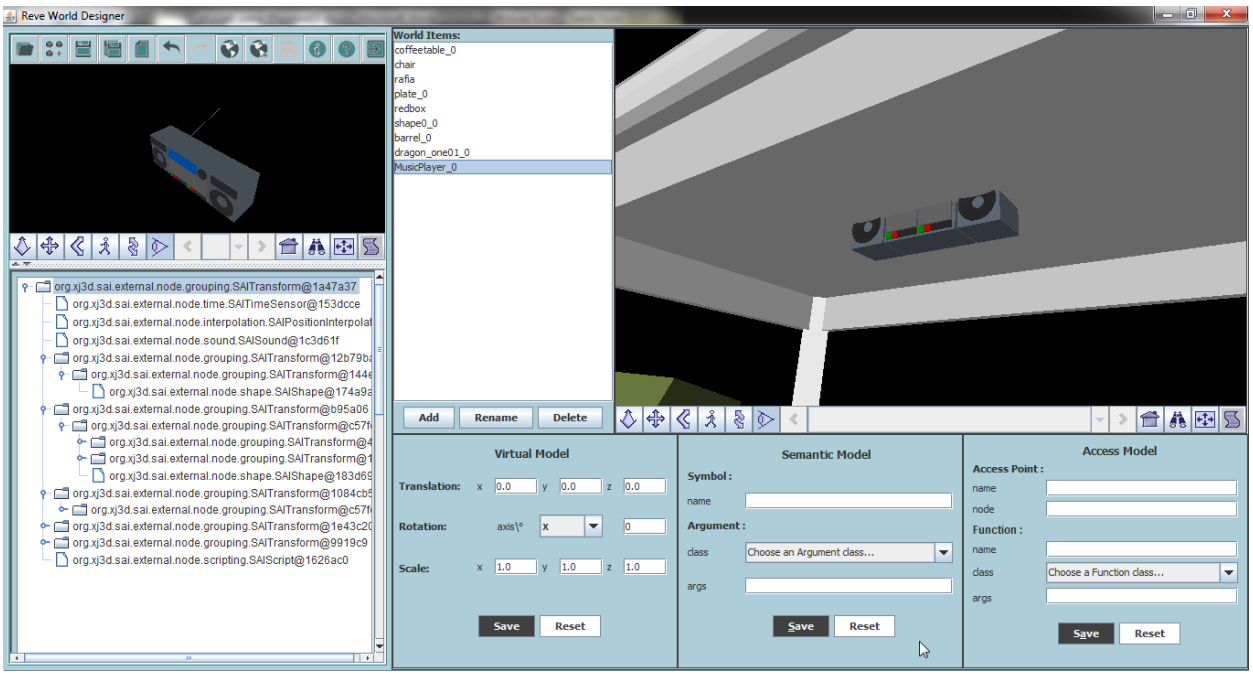


Αρχικά, αλλάζουμε το scale του μεγενθύνοντάς τον κατά 10 φορές. Στη συνέχεια τον περιστρέφουμε κατά 180 μοίρες στον άξονα y για να κοιτάει το πρόσωπό του το τραπέζι και τέλος,τον μετατοπίζουμε ως εξής: δεξιά 50 μέτρα στον άξονα x, πάνω 175 μέτρα στον άξονα y για να συμπίπτει με την επιφάνεια του τρίτου ραφιού και πίσω -30 μέτρα στον άξονα z.

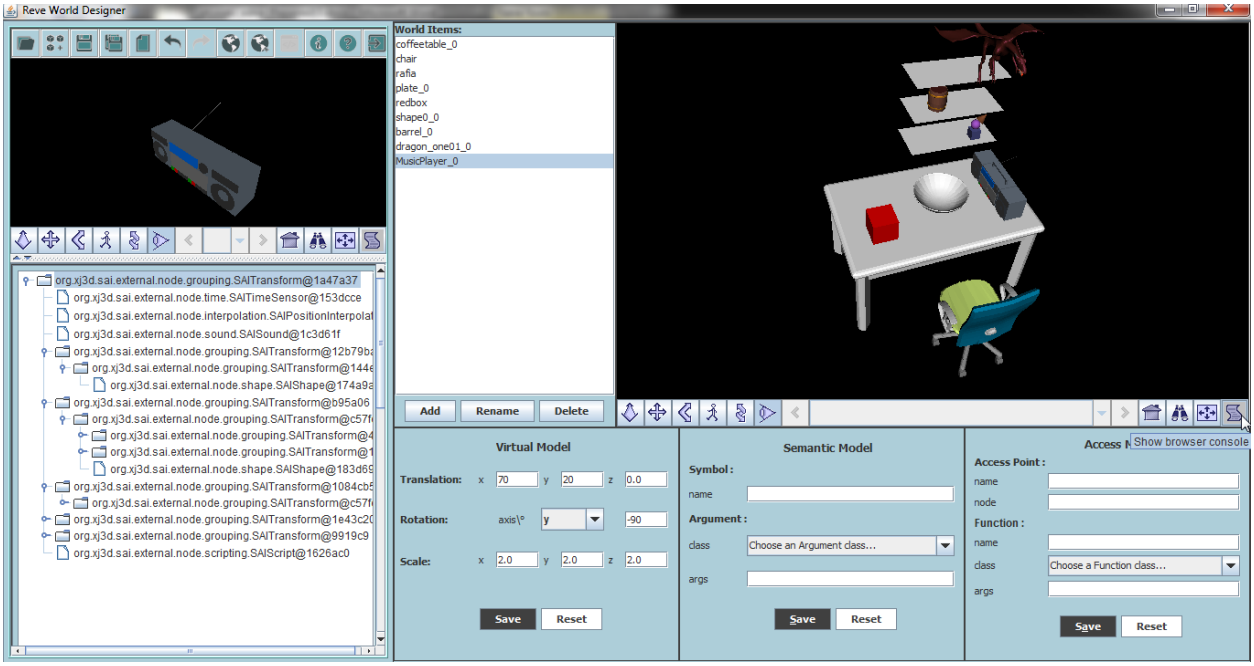


Τέλος, αποφασίζουμε να προσθέσουμε στο τραπέζι ένα κασετόφωνο, γιατί μας φαίνεται αρκετά άδειο. Δεδομένου ότι το αντικείμενο που ψάχνουμε δεν είναι πρόχειρο στην επιφάνεια εργασίας μας οπως τα υπόλοιπα, χρησιμοποιούμε το Open File για να το φορτώσουμε.

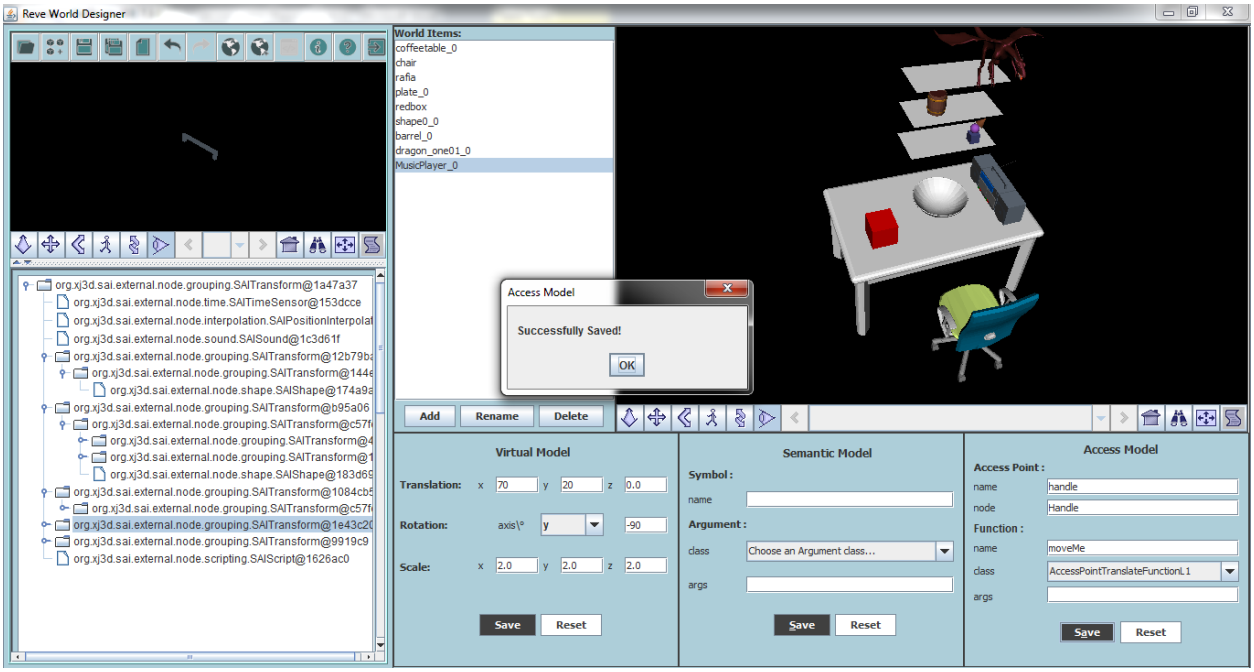




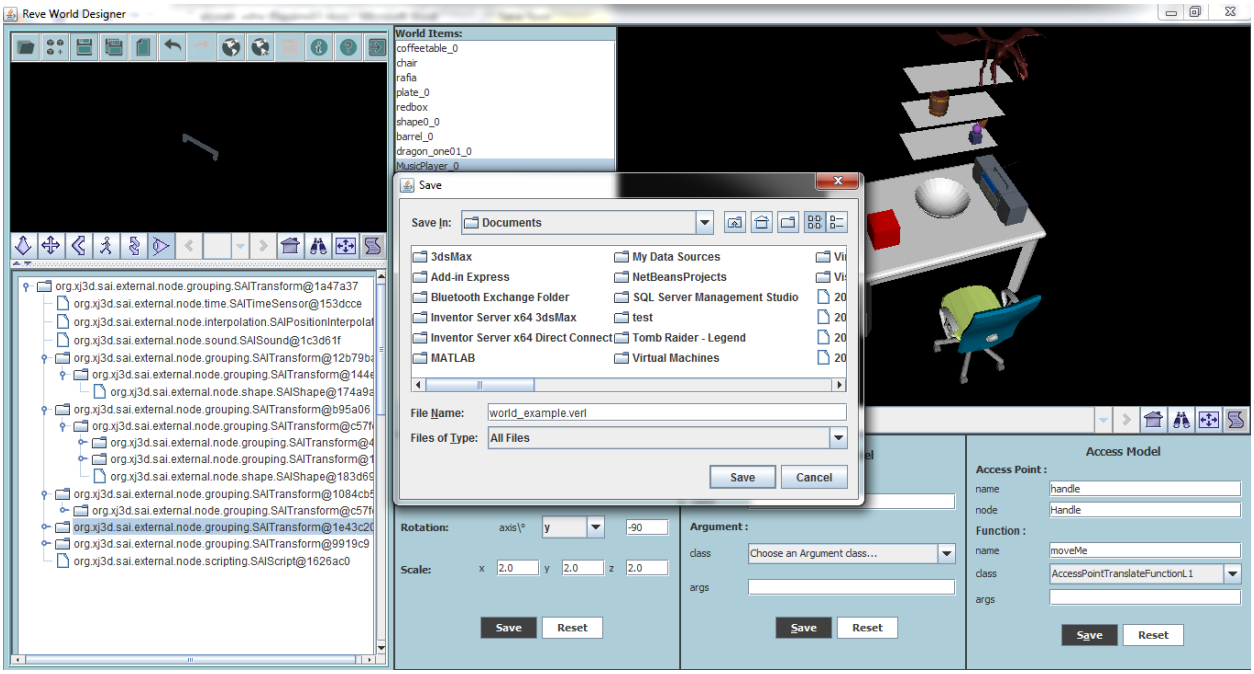
Αυτό, όπως τα υπόλοιπα αντικείμενα, αρχικά κρύβεται κάτω από το τραπέζι και πρέπει να υποστεί αλλαγή στο virtual model του για να εμφανιστεί πάνω στο τραπέζι. Επομένως, κάνουμε διπλάσιο το μέγεθος του αντικειμένου, το περιστρέφουμε -90 μοίρες κατά τον αξονα y ώστε τα ηχεία του να κοιτάνε την πιατέλα, το ανεβάζουμε στο τραπέζι με θετική μετακίνηση κατά 20 μέτρα στον άξονα y και το φέρνουμε την δεξιά άκρη του τραπεζιού με μετατόπιση κατά 70 μέτρα στον άξονα x.



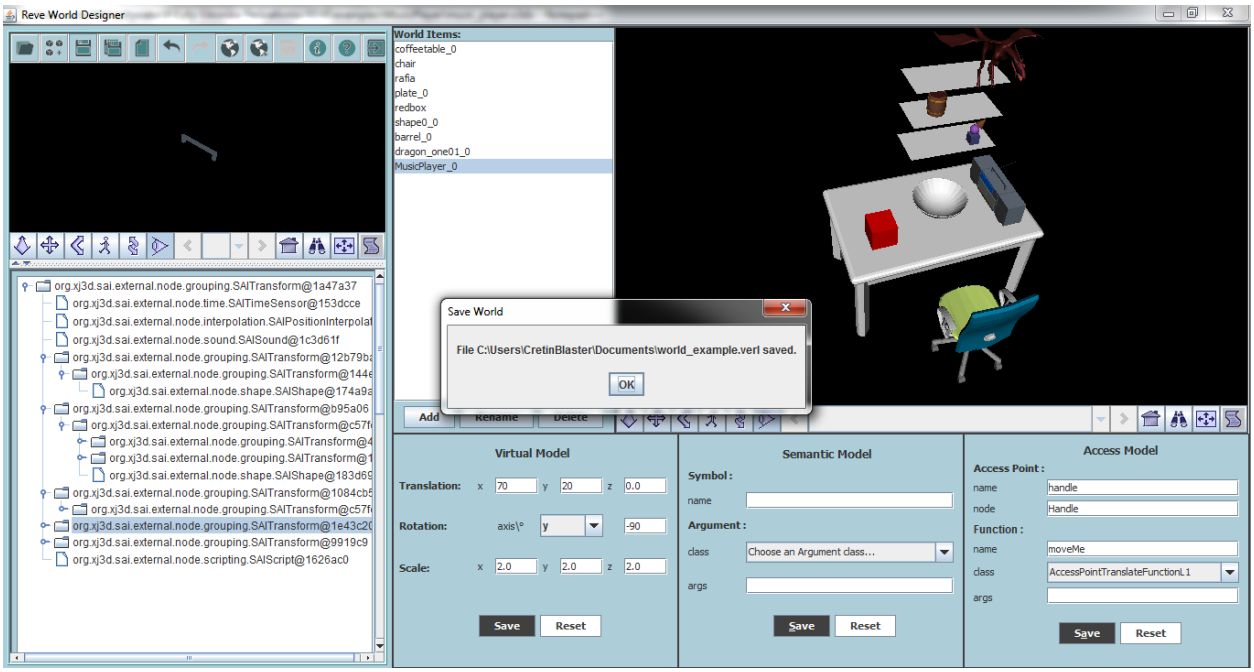
Στο γράφημα σκηνής του κασετοφώνου συναντάμε κόμβους που μπορούν να χρησιμοποιοηθούν για αλληλεπίδραση με έναν εικονικό πράκτορα, ορίζοντας και access model στο εν λόγω item. Ένας τέτοιος κόμβος είναι ο DEF Handle Transform που αναπαριστά την λαβή που βρίσκεται στο πάνω μέρος του κασετοφώνου και πάνω στον οποίο μπορούμε να ορίσουμε ένα access point με translate function, ώστε ο πράκτορας να μπορεί να το χρησιμοποιήσει για να μετακινήσει το κασετόφωνο. Ορίζουμε, λοιπόν, το παρακάτω access model στο κασετόφωνο.

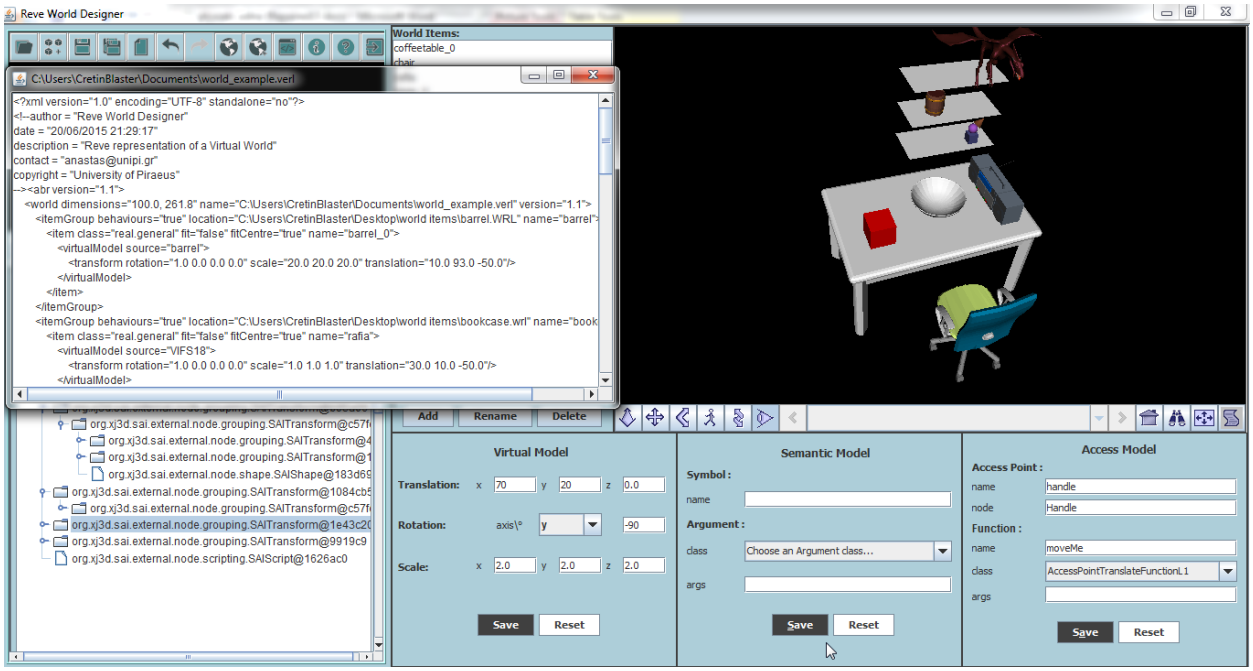


Θεωρώντας πως ο κόσμος μας επιδέχεται βελτίωσης, θα τον σώσουμε αρχικά σε αυτή την μορφή, για να επαληθεύσουμε πως μπορούμε να τον ανοίξουμε αργότερα και να τον επεξεργαστούμε. Μπορούμε, λοιπόν, να πατήσουμε Save ή Save As, που σε αυτή την περίπτωση έχουν πανομοιότυπη λειτουργικότητα εφόσον είναι η πρώτη φορά που σώζουμε τον κόσμο και να τον αποθηκεύσουμε το 'My Documents' με το όνομα world_example.verl.

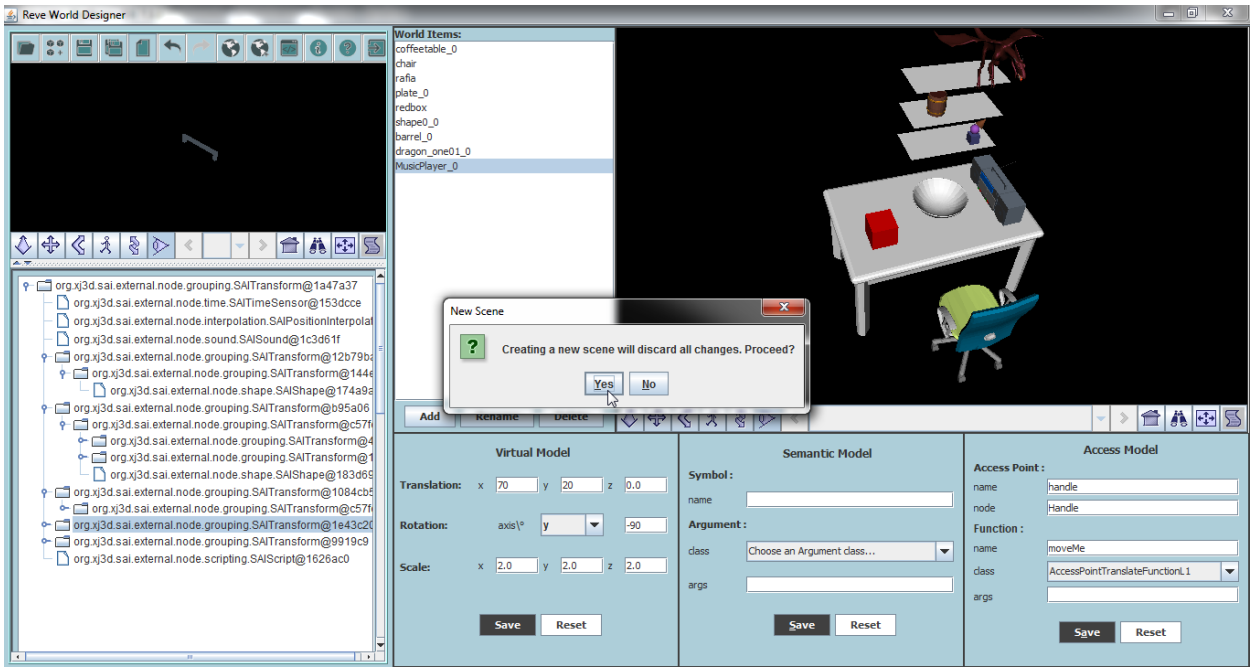


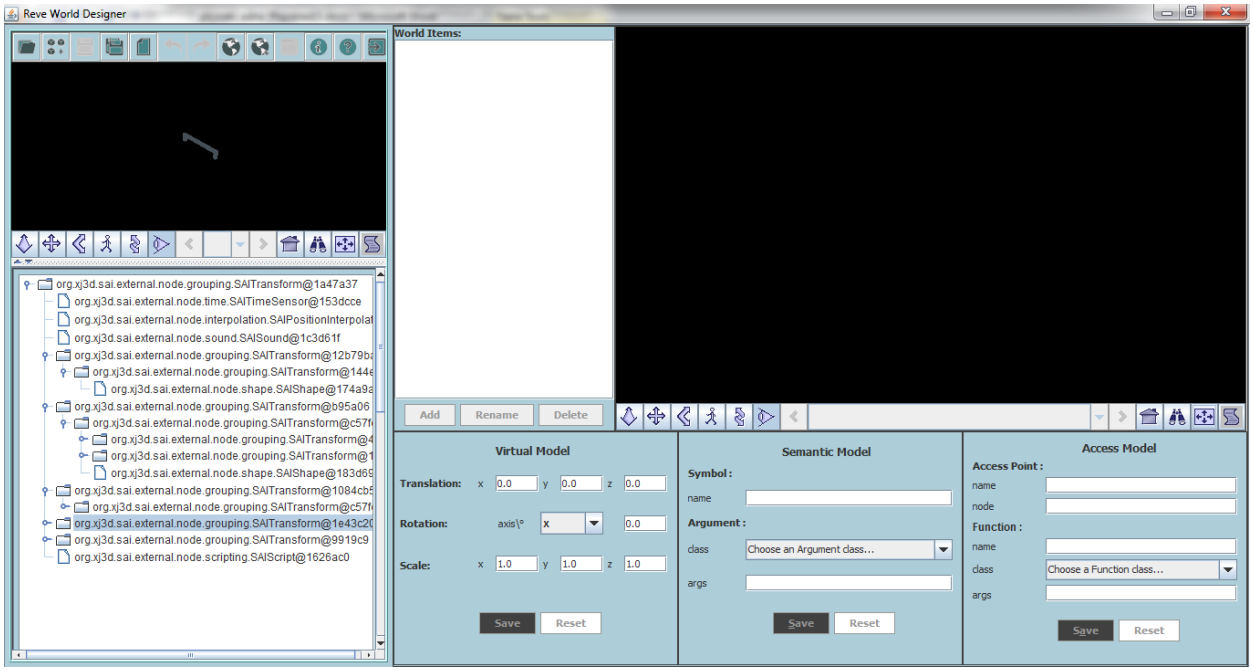
Η επιτυχής αποθήκευση του κόσμου επιβεβαιώνεται με σχετικό μήνυμα, ενώ ενεργοποιείται και το κουμπί 'Show Code', από όπου μπορούμε να δούμε την αναπαράσταση verl του κόσμου μας.



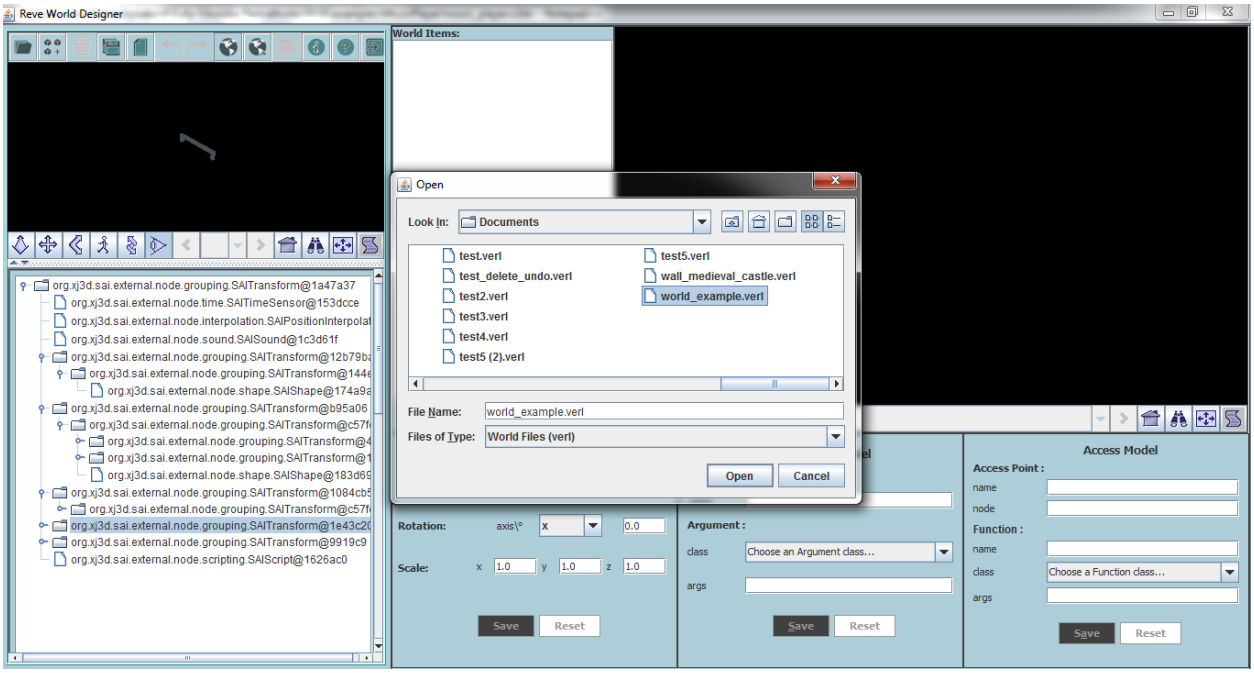


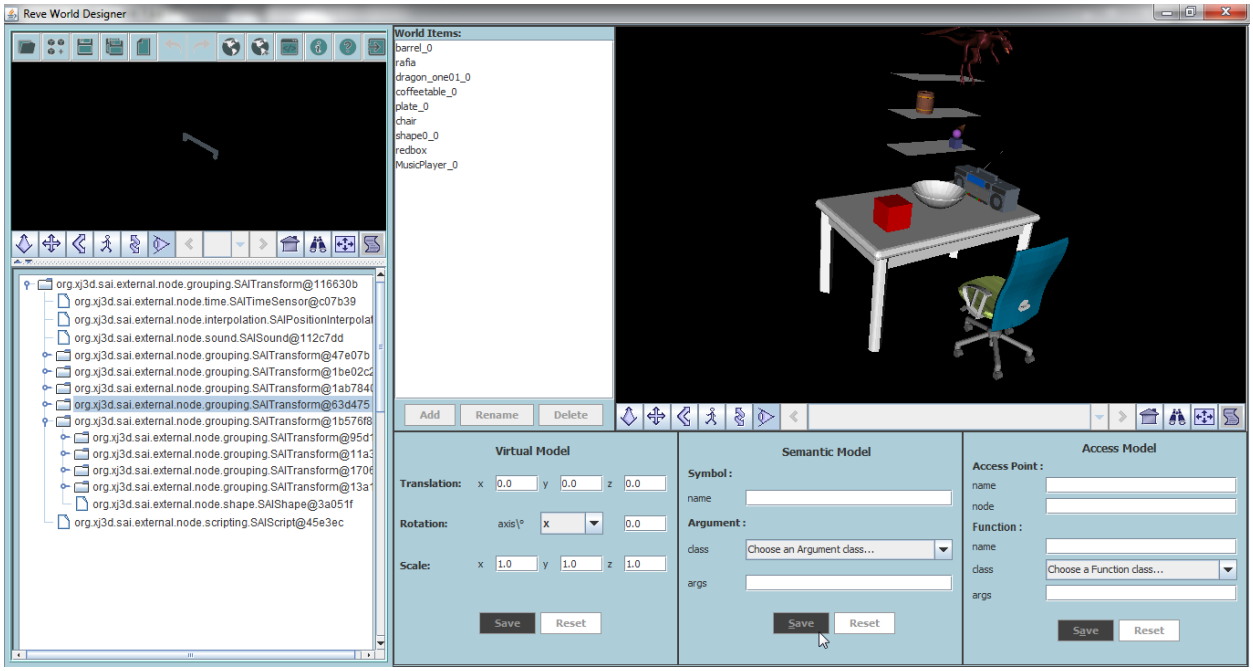
Στη συνέχεια, καθαρίζουμε την σκηνή μας, για να φορτώσουμε εκ νέου τον κόσμο που δημιουργήσαμε και να τον επεξεργαστούμε.



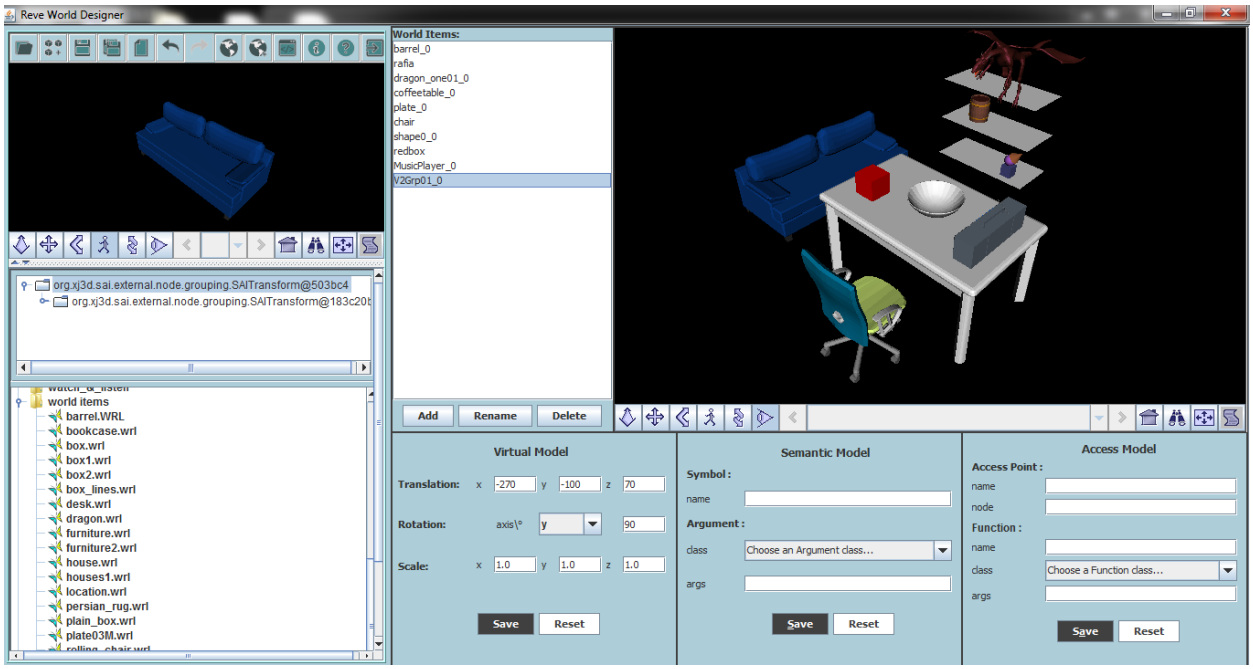


Φορτώνουμε τον κόσμο μας από το κουμπί Load World, όπου ορθά επανεμφανίζεται στην σκηνή μας.

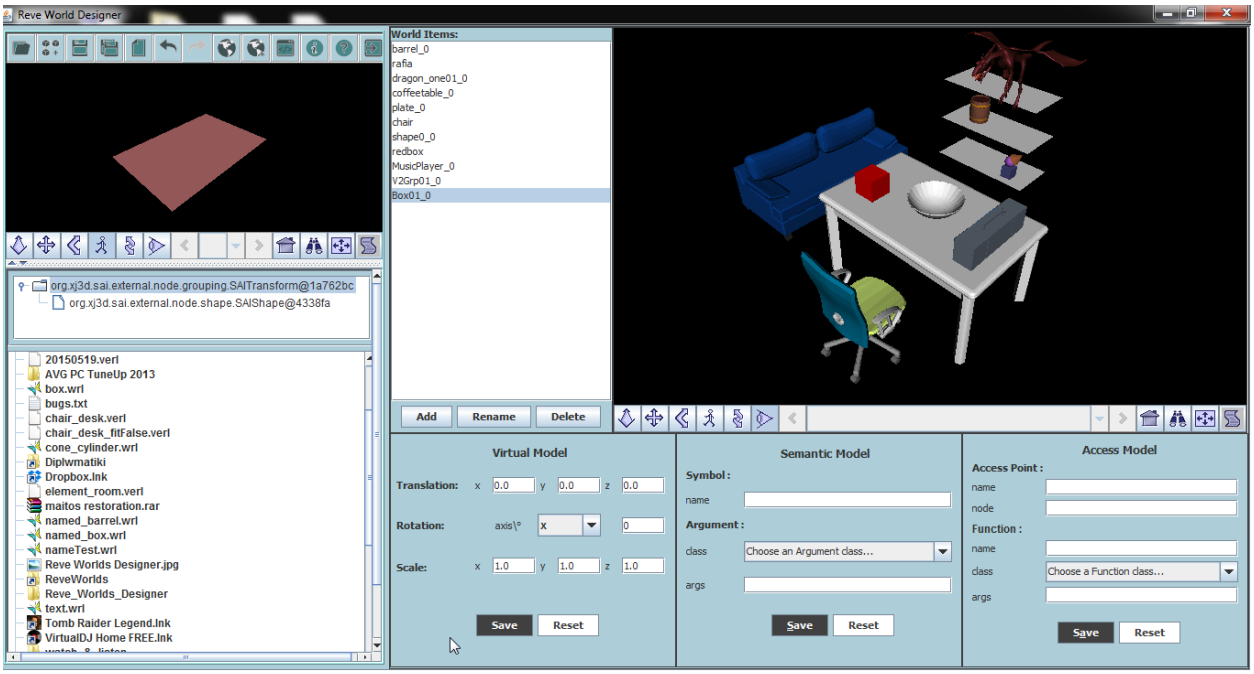




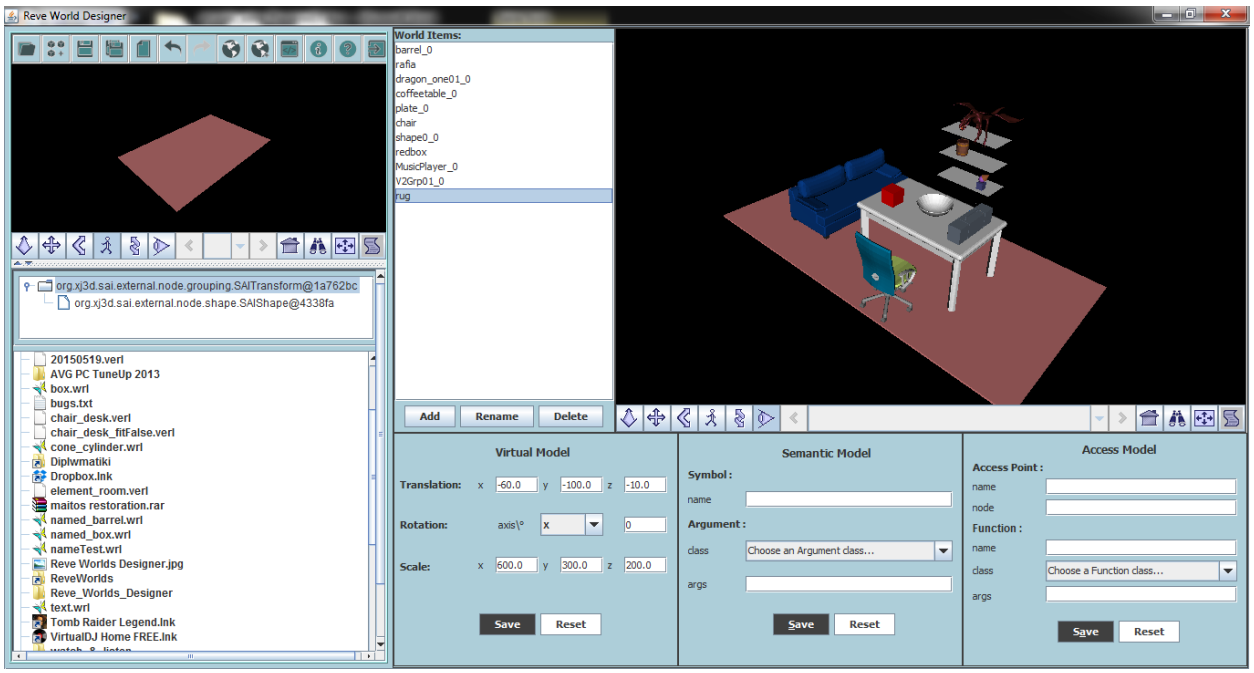
Αποφασίζουμε να επεξεργαστούμε τον κόσμο προσθέτοντας έναν καναπέ. Όπως γίνεται κατανοητό από την παρακάτω εικόνα, ο καναπές πρέπει να υποστεί μεταβολή στο virtual model του για να βρεθεί σε μια φυσιολογική θέστη στο χώρο, όπως είναι αριστερά από το τραπέζι. Για να γίνει αυτό πρέπει α) να χαμηλώσουμε τον καναπέ, δηλαδή να υποστεί αρνητική μετατόπιση στον άξονα y (-100), β) να τον μετακινήσουμε προς τα αριστερά, δηλαδή αρνητική μετατόπιση στον άξονα x (-270) και γ) να το περιστρέψουμε ως προς τον άξονα y κατά 90 μοίρες για να βλέπει το τραπέζι. Το αποτέλεσμα είναι το παρακάτω:



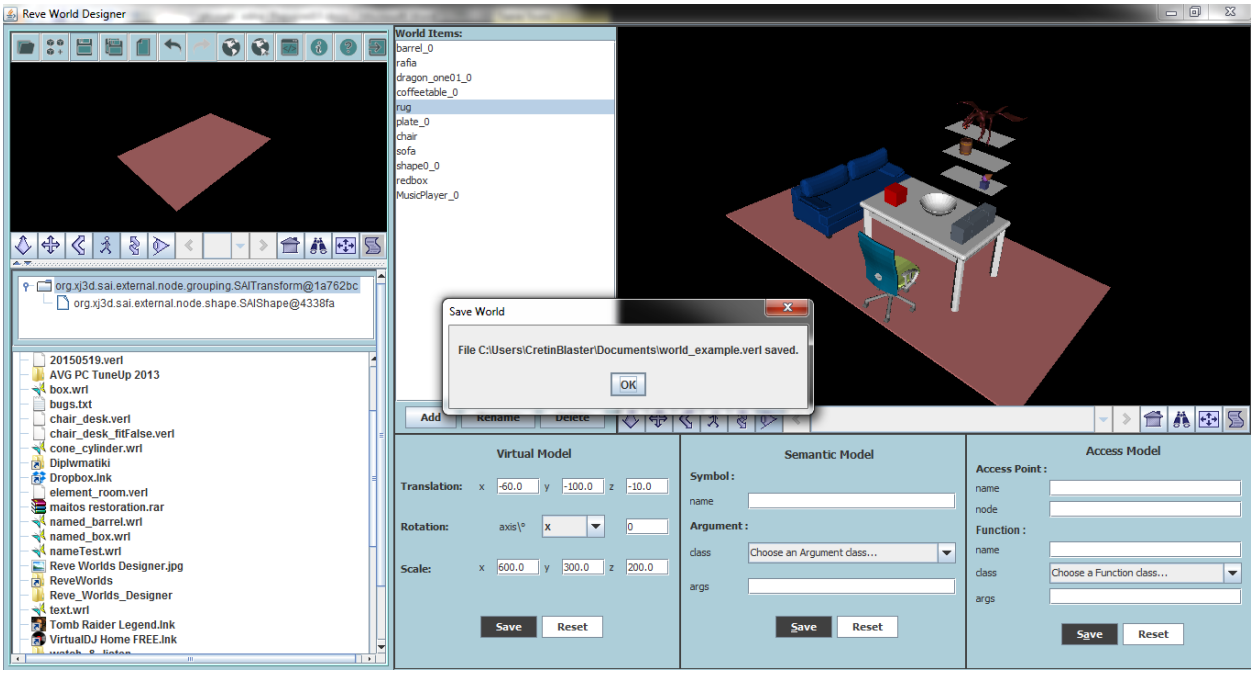
Τέλος, θα προσθέσουμε ένα χαλί στο χώρο, όπου θα πατάνε ο καναπές, το τραπέζι και η καρέκλα για να προσομοιάσουμε καλύτερα έναν φυσικό χώρο. Και σε αυτή την περίπτωση, το χαλί έχει μικρές διαστάσεις και κρύβεται κάτω από το τραπέζι, οπότε πρέπει να υποστεί transformations.



Μεγενθύνουμε το χαλί, αλλά όχι ομοιόμορφα αυτή την φορά, καθώς μας ενδιαφέρει το μήκος του να είναι πιο μακρύ, οπότε αυξάνουμε το scale ως εξής: x κατά 600, y κατά 300 και z κατά 200. Τέλος, αλλάζουμε και το translation του στις τιμές (-60, 100, -10) και το αποτέλεσμα είναι το παρακάτω.



Πατώντας Save αυτή τη φορά οι αλλαγές αποθηκεύονται στον αρχικό μας κόσμο και εδώ ολοκληρώνεται η σχεδίαση του εικονικού μας κόσμου.



Ο κώδικας που αναπαριστά τον κόσμο μας είναι ο παρακάτω:

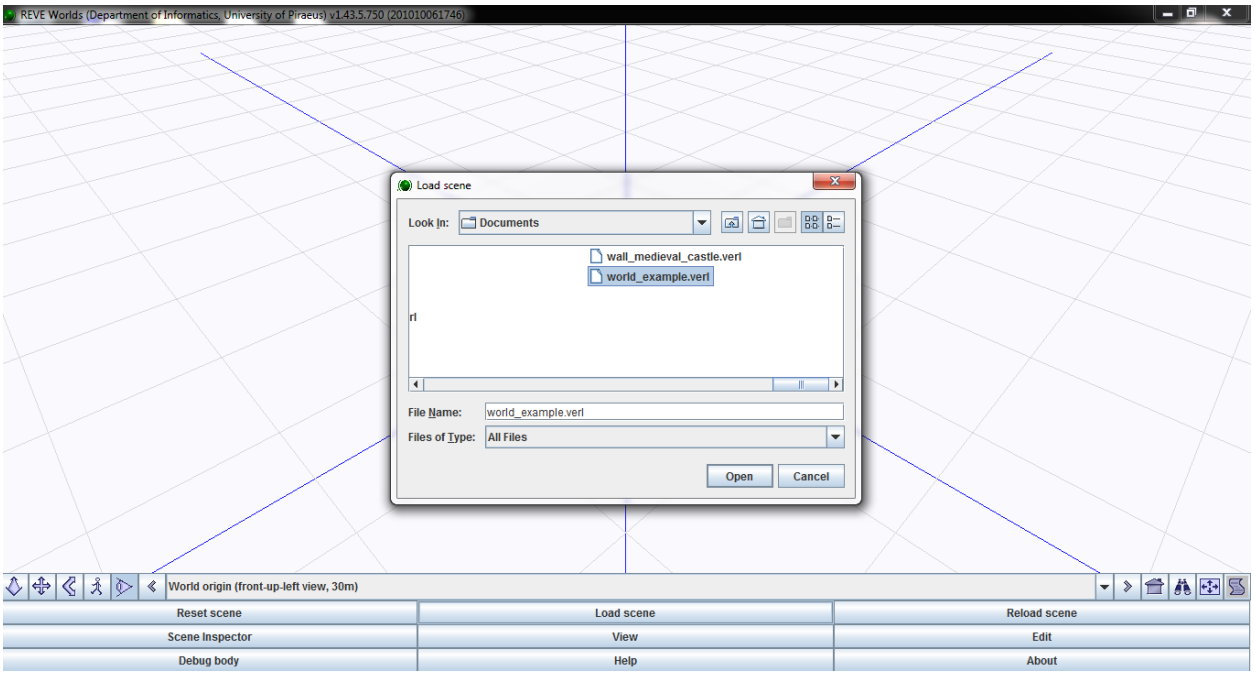
```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!--author = "Reve World Designer"
date = "20/06/2015 22:21:23"
description = "Reve representation of a Virtual World"
contact = "anastas@unipi.gr"
copyright = "University of Piraeus"
--><abr version="1.1">
    <world                          dimensions="100.0,                          261.8"
name="C:\Users\CretinBlaster\Documents\world_example.verl" version="1.1">
        <itemGroup                                              behaviours="true"
location="C:\Users\CretinBlaster\Desktop\world                  items\barrel.WRL"
name="barrel">
            <item     class="real.general"     fit="false"     fitCentre="true"
name="barrel_0">
                <virtualModel source="barrel">
                    <transform  rotation="1.0  0.0  0.0  0.0"  scale="20.0  20.0
20.0" translation="10.0 93.0 -50.0"/>
                </virtualModel>
            </item>
        </itemGroup>
        <itemGroup                                              behaviours="true"
location="C:\Users\CretinBlaster\Desktop\world                 items\bookcase.wrl"
name="bookcase">
            <item     class="real.general"     fit="false"     fitCentre="true"
name="rafia">
```

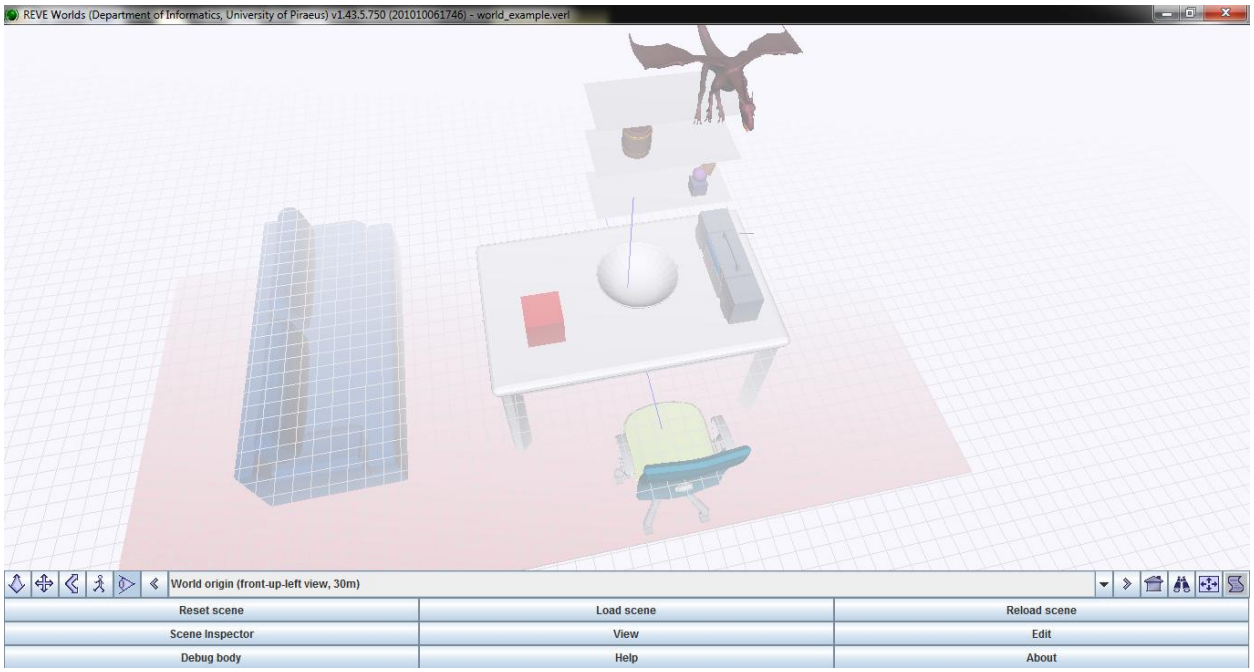
```
                <virtualModel source="VIFS18">
                    <transform rotation="1.0 0.0 0.0 0.0" scale="1.0 1.0 1.0"
translation="30.0 10.0 -50.0"/>
                </virtualModel>
            </item>
        </itemGroup>
        <itemGroup                                              behaviours="true"
location="C:\Users\CretinBlaster\Desktop\world                items\dragon.wrl"
name="dragon">
            <item     class="real.general"     fit="false"     fitCentre="true"
name="dragon_one01_0">
                <virtualModel source="dragon_one01">
                    <transform  rotation="0.0  1.0  0.0  3.1415927"  scale="10.0
10.0 10.0" translation="50.0 175.0 -30.0"/>
                </virtualModel>
            </item>
        </itemGroup>
        <itemGroup                                              behaviours="true"
location="C:\Users\CretinBlaster\Desktop\world            items\furniture.wrl"
name="furniture">
            <item     class="real.general"     fit="false"     fitCentre="true"
name="coffeetable_0">
                <virtualModel source="coffeetable">
                    <transform rotation="1.0 0.0 0.0 0.0" scale="0.1 0.1 0.1"
translation="0.0 0.0 0.0"/>
                </virtualModel>
            </item>
        </itemGroup>
        <itemGroup                                              behaviours="true"
location="C:\Users\CretinBlaster\Desktop\world          items\persian_rug.wrl"
name="persian_rug">
            <item     class="real.general"     fit="false"     fitCentre="true"
name="rug">
                <virtualModel source="Box01">
                    <transform  rotation="1.0  0.0  0.0  0.0"  scale="600.0  300.0
200.0" translation="-60.0 -100.0 -10.0"/>
                </virtualModel>
            </item>
        </itemGroup>
        <itemGroup                                              behaviours="true"
location="C:\Users\CretinBlaster\Desktop\world             items\plate03M.wrl"
name="plate03M">
            <item     class="real.general"     fit="false"     fitCentre="true"
name="plate_0">
                <virtualModel source="plate">
                    <transform rotation="1.0 0.0 0.0 0.0" scale="1.0 1.0 1.0"
translation="0.0 10.0 0.0"/>
                </virtualModel>
            </item>
        </itemGroup>
        <itemGroup                                              behaviours="true"
location="C:\Users\CretinBlaster\Desktop\world        items\rolling_chair.wrl"
name="rolling_chair">
            <item     class="real.general"     fit="false"     fitCentre="true"
name="chair">
```

```
                <virtualModel source="V2Grp01">
                    <transform  rotation="0.0  1.0  0.0  3.1415927"  scale="1.2
1.2 1.2" translation="10.0 -100.0 100.0"/>
                </virtualModel>
            </item>
        </itemGroup>
        <itemGroup                                              behaviours="true"
location="C:\Users\CretinBlaster\Desktop\world items\sofa.wrl" name="sofa">
            <item     class="real.general"     fit="false"     fitCentre="true"
name="sofa">
                <virtualModel source="V2Grp01">
                    <transform  rotation="0.0  1.0  0.0  1.5707964"  scale="1.0
1.0 1.0" translation="-270.0 -100.0 70.0"/>
                </virtualModel>
            </item>
        </itemGroup>                              <itemGroup    behaviours="true"
location="C:\Users\CretinBlaster\Desktop\world items\test4.wrl" name="test4">
            <item     class="real.general"     fit="false"     fitCentre="true"
name="shape0_0">
                <virtualModel source="shape0">
                    <transform  rotation="1.0  0.0  0.0  0.0"  scale="10.0  10.0
10.0" translation="50.0 55.0 -40.0"/>
                </virtualModel>
            </item>
        </itemGroup>
        <itemGroup                                              behaviours="true"
location="C:\Users\CretinBlaster\Desktop\world          items\touchy_box.x3dv"
name="touchy_box">
            <item     class="real.general"     fit="false"     fitCentre="true"
name="redbox">
                <virtualModel source="TG">
                    <transform  rotation="1.0  0.0  0.0  0.0"  scale="12.0  12.0
12.0" translation="-60.0 20.0 20.0"/>
                </virtualModel>
            </item>
        </itemGroup>
        <itemGroup                                              behaviours="true"
location="E:\Documents\Metaptyxiako\B\Eyfyi                             Eikonika
Perivallonta\REVE\examples\MusicPlayer\music_player.x3dv"
name="music_player">
            <item     class="real.general"     fit="false"     fitCentre="true"
name="MusicPlayer_0">
                <virtualModel source="MusicPlayer">
                    <transform  rotation="0.0  1.0  0.0  -1.5707964"  scale="2.0
2.0 2.0" translation="70.0 20.0 0.0"/>
                </virtualModel>
                <accessModel>
                    <accessPoint name="handle" node="Handle">
                        <function                   args="                    "
class="reve.scene.item.access.function.AccessPointTranslateFunctionL1"
name="moveMe"/>
                    </accessPoint>
                </accessModel>
            </item>
        </itemGroup>
```

```
    </world>
</abr>
```

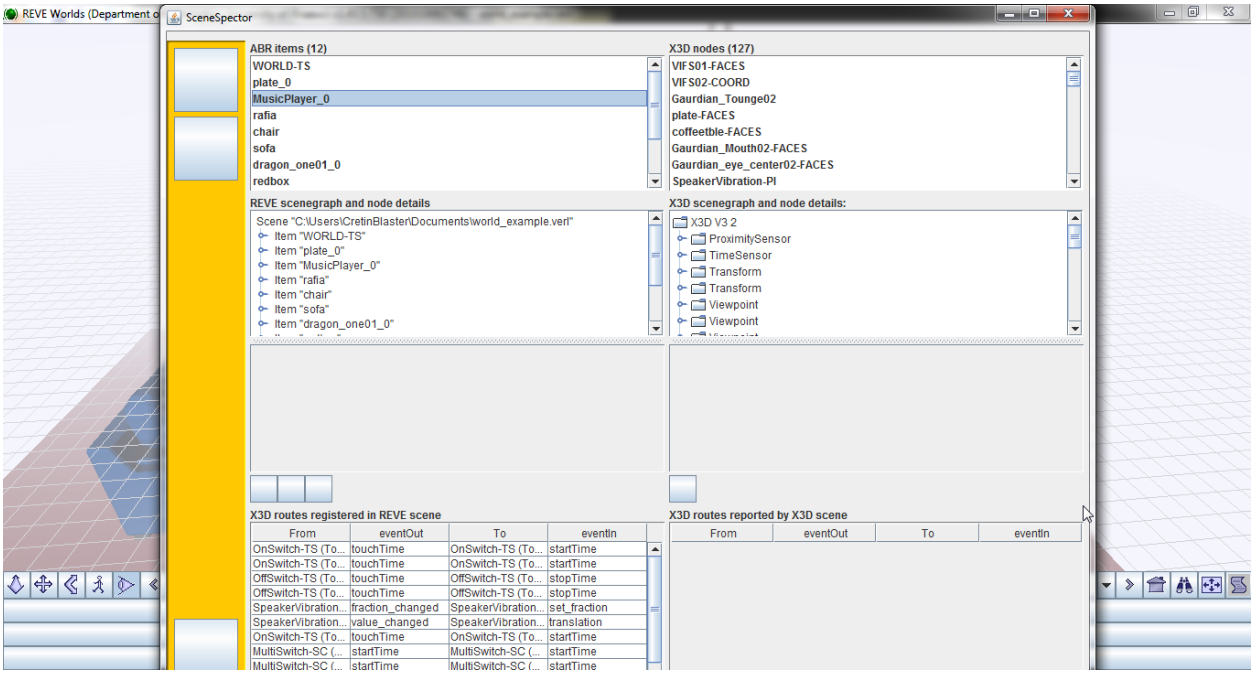
Παρακάτω δείχνουμε πως φαίνεται ο κόσμος που σχεδιάσαμε στην πλατφόρμα Reve Worlds.



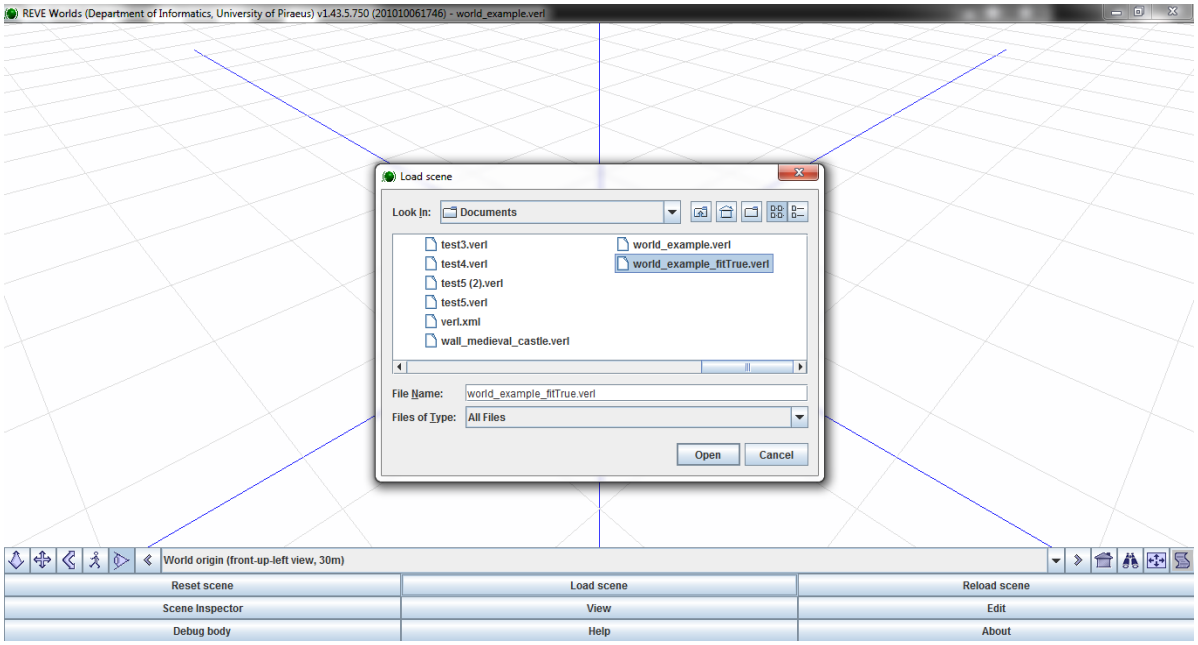


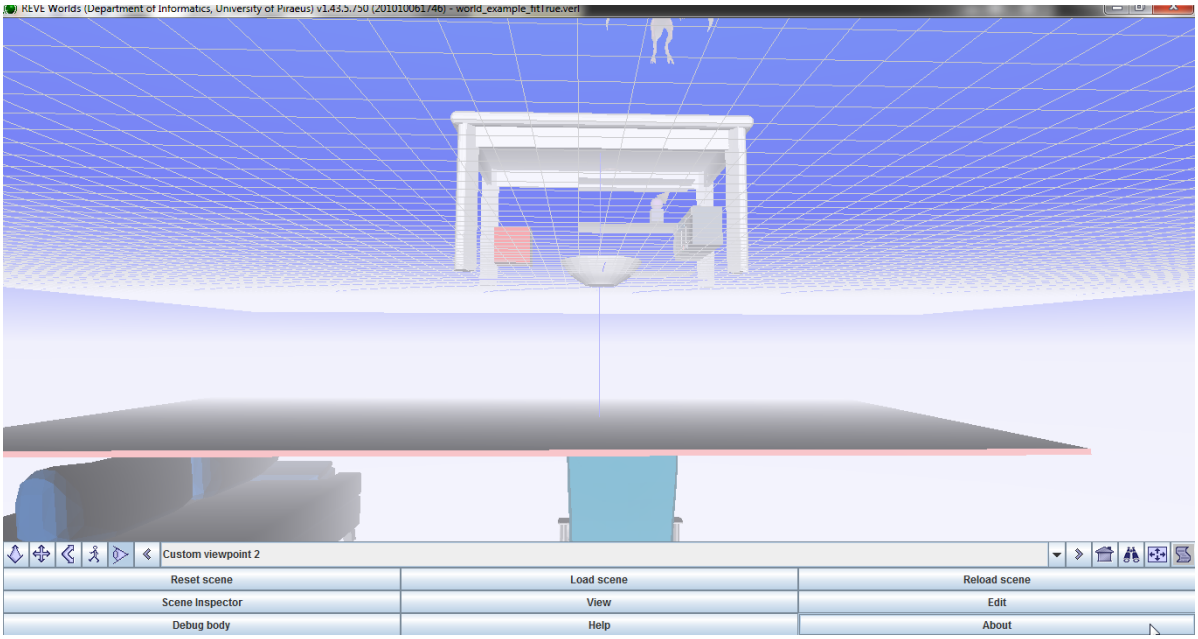
Παρατηρούμε πως ακριβώς επειδή τα αντικείμενα δεν ευθυγραμμίζονται στην αρχή των αξόνων κατά την εισαγωγή τους στον κόσμο, έχουμε το φαινόμενο όπου αυτά μπορεί να βρίσκονται σε μη λογικές θέσεις σε σχέση με τις συντεταγμένες του κόσμου,όπως συμβαίνει με τον καναπέ, όπου φαίνεται πως βρίσκεται σε

αρνητικές συντεταγμένες του άξονα y, δίνοντας την εντύπωση πως είναι κάτω από το έδαφος. Τέτοια φαινόμενα θα εξαλειφθούν με την διόρθωση που αναφέραμε πως απαιτείται. Ανοίγοντας τον Scene Inspector μπορούμε να δούμε τα αντικείμενα της σκηνής μας και πληροφορίες για αυτά.



Αν αλλάξουμε το verl αρχείο και ορίσουμε όλα τα fit αντί για false ως true κι έπειτα φορτώσουμε τον κόσμο στην Reve Worlds, θα διαπιστώσουμε αυτό που αναφέραμε και παραπάνω, όπου τώρα ζητάμε το τοπικό σύστημα συντεταγμενων κάθε αντικειμένου να ταυτίζεται με το σύστημα συντεταγμένων του κόσμου, με αποτελέσμα οι τιμές του virtual model να έχουν άλλη επίδραση, αφού πλέον έχουν ως σύστημα αναφορας το σύστημα συντεταγμένων του κόσμου και όχι το τοπικό σύστημα συντεταγμένων του κάθε αντικειμένου.





Προφανώς και αυτή η περίπτωση θα διορθωθεί με την ενσωμάτωση του κώδικα του κ. Αναστασάκη στην εφαρμογή μας, για ευθυγραμμιση των αντικειμένων στην αρχή των αξόνων κατά την εισαγωγή τους στον κόσμο, ώστε όλα να έχουν το ίδιο σύστημα αναφοράς, δηλαδή το σύστημα συντεταγμένων του κόσμου.

Το παραπάνω παράδειγμα διατίθεται μαζί με τον εκτελέσιμο κώδικα της εφαρμογής και είναι τα αρχεία world_example.verl και world_example_fitTrue.verl για τις περιπτώσεις όπου τα item έχουν attribute fit="false" και fit="true" αντίστοιχα.

ΣΥΜΠΕΡΑΣΜΑΤΑ

Στόχος της παρούσας μεταπτυχιακής διατριβής ήταν η ανάπτυξη μιας εφαρμογής σχεδίασης εικονικών κόσμων αξιοποιώντας την αναπαράσταση Reve. Σκοπός ήταν η εφαρμογή να αποτελέσει την βάση για μια επεκτάσιμη πλατφόρμα σύνθεσης εικονικών κόσμων με αντικείμενα vrml και x3d, χωρίς να είναι βέλτιστη ούτε από άποψη πολυπλοκότητας κώδικα ούτε από άποψη κάλυψης πλήρους λειτουργικότητας της αναπαράστασης Reve. Για αυτό τον λόγο, άλλωστε, η υλοποίηση εστίασε στην φυσική αναπαράσταση ενός κόσμου και όχι τόσο στην σημασιολογική αναπαράσταση του και στην δυνατότητα αλληλεπίδρασης με αυτόν, παρότι έγινε μια πρώτη προσέγγιση για την ανάπτυξη των εν λόγω μοντέλων.

Ανατρέχοντας στην ανάλυση απαιτήσεων, ως την πρωταρχική άποψη για το τι θέλαμε ιδανικά να καλύψουμε στα πλαίσια αυτής της εφαρμογής, μπορούμε να πούμε πως κατά την υλοποίηση ένα μεγάλο κομμάτι της ανάλυσης επιτεύχθηκε και αφορά στο κομμάτι που αποτελεί και την βασική λειτουργικότητα της τρέχουσας διατριβής, δηλαδή την σχεδίαση ενός εικονικού κόσμου.

Πιο συγκεκριμένα, η εφαρμογή πράγματι διαθέτει ένα φιλικό προς τον χρήστη περιβάλλον λειτουργίας, με την ανάπτυξης ενός user interface που περιλαμβάνει ένα σύνολο κουμπιών για τις εντολές που μπορεί να εκτελέσει. Τα κουμπιά αυτά διαθέτουν περιγραφή για το τι κάνουν ή οδηγίες για την ενεργοποίησή τους από το πληκτρολόγιο (πχ Save-> Alt+S) και τοποθετούνται σε θέση σχετική με την λειτουργικότητα που επιτελούν, ώστε να είναι εύχρηστα και κατανοητά από τον χρήστη. Έτσι, έχουμε κουμπιά στο toolbar για τις κύριες λειτουργίες της εφαρμογής, στην λίστα των αντικειμένων που υπάρχουν στην σκηνή του κόσμου για τις βασικές εντολές που μπορούν να εκτελεστούν πάνω σε ένα item (add, rename, delete), σε καθένα από τα μοντέλα αναπαράστασης ενός item, καθώς και σε μενού (που εμφανίζεται με δεξί κλικ πάνω στο item) για πιο γρήγορη εκτέλεση των εντολών και για να δώσουμε στον χρήστη μια πληθώρα επιλογών χωρίς να υπερφορτώσουμε το παράθυρο με πολλή πληροφορία.

Επιτύχαμε να διαγράφουμε ένα item από την σκηνή, να δημιουργούμε αντίγραφά του στην σκηνή και να το μετονομάζουμε, διατηρώντας την ταυτότητά του, μέσα από μοναδικοποίηση του ονόματός του. Ακόμα, επιτύχαμε να δείχνουμε ποιο item είναι αυτό που επιλέγεται από την λίστα με τα αντικείμενα της σκηνής, αξιοποιώντας την scale ιδιότητά του, καθώς και να δείχνουμε το επιλεγμένο αντικείμενο να περιστρέφεται γύρω από τον εαυτό του, ώστε ο χρήστης να έχει μια πλήρη εικόνα για τον κόμβο-ρίζα του item. Πέραν τούτου, προσθέσαμε την δυνατότητα ο χρήστης να ορίζει ως 'real' ή 'unreal' ένα item στην σκηνή, με τα σημασιολογικά αποτελέσματα που αυτό έχει στην Reve Worlds, ενώ εμφανίζουμε σε ξεχωριστό παράθυρο τα βασικά στοιχεία που αποτελούν ένα item, ώστε ο χρήστης να έχει μια καλύτερη εικόνα για την προέλευσή του και τις ιδιότητές του. Ακόμα, επιτύχαμε ο χρήστης να μπορεί να αναιρεί μια ενέργεια, σε περίπτωση λάθους, καθώς και να επανεκτελεί την ενέργεια που μόλις αναίρεσε.

Επιπροσθέτως, επιτύχαμε η δομή της εφαρμογής να διευκολύνει τον χρήστη στην επιλογή αντικειμένων που θα εισάγει στον κόσμο με το να αποτελείται από επιμέρους παράθυρα με ξεχωριστή λειτουργικότητα. Ως εκ τούτου, η εφαρμογή από την μια πλευρά εμφανίζει τα αντικείμενα που επιλέγει ο χρήστης και την εσωτερική τους δομή, επιτρέποντάς τον να τα περιεργαστεί συνολικά και ελέγχοντας ποιους κόμβους μπορεί να εισάγει στον κόσμο βάσει της προδιαγραφής verl κι από την άλλη πλευρά, διατηρείται μια αυτονομία, εφόσον ο χρήστης μπορεί να ασχοληθεί ανεξάρτητα με την επεξεργασία ενός εικονικού κόσμου.

Επίσης, καταφέραμε να διατηρήσουμε την αυτονομία των διαφορετικών διαστάσεων που χαρακτηρίζουν ένα item μέσα από την αναπαράσταση του virtual, semantic και access model του σε διακριτές θέσεις στην παραθυρική εφαρμογή και τη δυνατότητα παρέμβασης του χρήστη σε αυτά. Ως εκ τούτου, επιτύχαμε τον βασικό στόχο να σχεδιάσουμε έναν εικονικό κόσμο από ετερόκλητα αντικείμενα ή κομμάτια αντικειμένων από το γράφημα σκηνής τους, αλλάζοντας την θέση τους στο χώρο, το μέγεθός τους ή και την περιστροφή τους, δηλαδη μεταβάλλοντας το virtual model τους. Ακόμα, επιτρέψαμε στον χρήστη να ορίσει ένα υποτυπώδες semantic ή access model σε ένα item, για να βεβαιωθούμε ότι και αυτή η λειτουργικότητα είναι δυνατή μέσα από την τρέχουσα εφαρμογή, χωρίς να επεκταθούμε περισσότερο σε αυτή την έκφανση του item.

Τέλος, επιτύχαμε να εξάγουμε έναν εικονικό κόσμο με την αναπαράσταση Reve, δημιουργώντας ένα μοναδικό verl αρχείο που το αποθηκεύουμε στην τοποθεσία που ορίζει ο χρήστης, προσθέτωντας την δυνατότητα ο χρήστης να μπορεί να βλέπει απευθείας τον verl κώδικα που αναπαριστά την σκηνή του. Ακόμα, δώσαμε την δυνατότητα ο χρήστης να φορτώσει, αντιστρόφως, έναν εικονικό κόσμο που δημιουργείται από

την εφαρμογή, εισάγοντας ένα verl αρχείο σε αυτήν. Πέραν τούτου, ο χρήστης μπορεί να καθαρίζει την σκηνή ξεκινώντας μια νέα σχεδίαση από την αρχή ή να προσθέτει έναν εικονικό κόσμο στον υπάρχον που σχεδιάζει, για να τον εμπλουτίσει πιο γρήγορα, συγχωνεύοντας πλήθος αντικειμένων σε μια νέα σχεδίαση.

Ωστόσο, κάποια πράγματα που αναφέραμε στην ανάλυση απαιτήσεων δεν καταφέραμε να τα πραγματοποιήσουμε, είτε επειδή το μέγεθος της δυσκολίας τους δεν κατέστησε εφικτή την υλοποίησή τους στην παρούσα φάση, είτε επειδή θεωρήθηκαν δευτερεύοντα ως προς τον πρωτεύον μας στόχο και θα μας ενέτασσαν σε μια αέναη διαδικασία διαρκούς βελτίωσης της εφαρμογής, την στιγμή που έπρεπε, για αντικειμενικούς λόγους, να μπουν συγκεκριμένα χρονικά πλαίσια για την ολοκλήρωσή της.

Ως εκ τούτου, αναφέρουμε κάποια βασικά πράγματα που δεν καταφέραμε να πραγματοποιήσουμε βάσει της ανάλυσης απαιτήσεων ή που δεν αναφέραμε στην ανάλυση απαιτήσεων, αλλά στην πορεία θεωρούμε πως θα ήταν ενδιαφέρον και εφικτό να πραγματοποιηθούν για την βελτιστοποίηση, τον εμπλουτισμό και την επέκταση της εφαρμογής.

Το πρώτο και κύριο που επιβάλλεται να διορθωθεί για την σωστή αναπαράσταση του κόσμου στην πλατφόρμα Reve Worlds και αποτελεί έλλειψη της τρέχουσας υλοποίησης είναι να εισαχθεί ο κώδικας του κ. Αναστασάκη στην εφαρμογή, κατά την δημιουργία ενός item, ώστε το αντικείμενο που προστίθεται στον κόσμο να εμφανίζεται στην αρχή των αξόνων του κόσμου, έχοντας αφαιρέσει το εσωτερικό του translation και rotation. Εν ολίγοις, να ευθυγραμμιστεί το αντικείμενο με την αρχή των αξόνων του κόσμου και το virtual model του να έχει ως σύστημα αναφοράς το σύστημα συντενταγμένων του κόσμου και όχι το τοπικό σύστημα συντεταγμένων του αντικειμένου. Επίσης, συνδυαστικά με αυτό, κατά την αποθήκευση του κόσμου, η παράμετρος "fit" του item πρέπει να πάρει την τιμή "true" και όχι "false", για να εφαρμόζεται το παραπάνω και στο Reve Worlds κατά την φόρτωση του κόσμου.

Δεύτερον, μπορεί να προστεθεί δυνατότητα πέρα από DEF Transform κόμβους να εισάγονται και Shape κόμβοι στην σκηνή. Ακόμα μπορεί να δίνεται η δυνατότητα στον χρήστη να κάνει drag and drop έναν κόμβο στην σκηνή, αξιοποιώντας την σχετική δυνατότητα της swing και εμφανίζοντας σχετικό μήνυμα όταν αυτός ο κόμβος δεν πληρεί τις προϋποθέσεις προσθήκης στον εικονικό κόσμο.

Τρίτον, μπορεί να αυξηθεί το πλήθος των undo-redo ενεργειών που επιτρέπονται στον χρήστη, αυξάνοντας το limit του undoManager σε έναν λογικό αριθμό. Ο αριθμός αυτός θα πρέπει να μελετηθεί σε σχέση με άλλα εργαλεία σχεδίασης εικονικών κόσμων αλλά και τις απαιτήσεις σε μνήμη που μπορεί να υπάρχουν για αυτή την περίπτωση από την εφαρμογή.

Τέταρτον, μπορεί να δοθεί η δυνατότητα επεξεργασίας πολλαπλών αντικειμένων ταυτόχρονα, όπως είναι η περίπτωση επιλογής πολλών items από την λίστα της σκηνής και την δημιουργία αντιγράφων τους με το πάτημα του 'Add' μια φορά ή την ταυτόχρονη διαγραφή τους με το πάτημα του 'Delete' μια φορά. Σε αντίθεση με την τρέχουσα υλοποίηση, που κρατάει σε μια Item μεταβλητή το επιλεγμένο item, εδώ θα πρέπει να διατηρούνται τα επιλεγμένα αντικείμενα σε μια λίστα ή σε μια στοίβα από Items για να μπορούν να επεξεργαστούν ταυτόχρονα. Επίσης, θα πρέπει αυτή η δυνατότητα να μπορεί να αναιρεθεί ή να επανεκτελεστεί μέσω του undo-redo μηχανισμού, γιατί διαφορετικά μπορεί να έχει τραγικές συνέπειες, εξού και η σημασία της προσωρινής αποθήκευσής τους σε μια ευέλικτη δομή δεδομένων.

Πέμπτον, η επίτευξη της παραπάνω δυνατότητας θα ανοίξει τον δρόμο και για την αναίρεση της προσθήκης ενός κόσμου (Add World) στον υπάρχον, γλιτώνοντας τον χρήστη από την αγγαρεία να διαγράψει ένα-ένα τα αντικείμενα που προστέθηκαν με τον νέο κόσμο αν στην πορεία μετανιώσει αυτή την απόφαση ή αν κατα λάθος προσθέσει άλλον κόσμο από τον επιθυμητό. Σε σχέση με αυτό, θα μπορούσε να δίνεται η δυνατότητα να εμφανίζεται ο κόσμος που ο χρήστης θέλει να προσθέσει στον υπάρχον σε ένα νέο παράθυρο, όπως συμβαίνει με την εμφάνιση του περιστρεφόμενου αντικειμένου σε ένα νέο παράθυρο.

Έκτον, όσον αφορά το semantic και access model, μπορεί και θα εξυπηρετούσε να προστεθεί έλεγχος για την περίπτωση λογικών σφαλμάτων στον ορισμό παραμέτρων, ώστε να αποφεύγονται λάθη κατά την φόρτωση στην πλατφόρμα Reve Worlds. Τέτοια λογικά σφάλματα είναι η εισαγωγή κόμβων που δεν ορίζονται στην δομή του αντικειμένου ή η εισαγωγή κόμβων που ορίζονται αλλά δεν συνδυάζονται με την function που πάει να αποθηκεύσει ο χρήστης. Επίσης, θα πρέπει να προστεθεί κάποιο κουμπί στο semantic και access model, όπως ένα σύμβολο '+', που θα επιτρέπει στον χρήστη να προσθέσει παραπάνω από ένα symbol ή access point στο επιλεγμένο item.

Όγδοον, θα μπορούσε να αλλάξει ο φόντος της σκηνής από μαύρος σε λευκός και να προστεθούν grids κατά μήκος των (x,y,z) αξόνων ώστε ο χρήστης να αντιλαμβάνεται καλύτερα πώς πρέπει να μετατοπίσει ένα αντικείμενο για να επιτύχει το επιθυμητό αποτέλεσμα. Συνδυαστικά με αυτό, θα μπορούσαν να προστεθούν κάποια διαφορετικά default viewpoints στην σκηνή του κόσμου, ώστε ο χρήστης να μπορεί να αλλάξει εύκολα και γρήγορα οπτική γωνία στον κόσμο, όπως συμβαίνει στην πλατφόρμα Reve Worlds.

Ένατον, θα πρέπει να εμπλουτιστεί ο exception μηχανισμός της φόρτωσης ενός κόσμου, ώστε να μπορεί πιο εύκολα να φορτωθεί ένα verl αρχείο που έχει γραφτεί με το χέρι και δεν έχει παραχθεί από την εφαρμογή, για να μην υπάρχουν σκασίματα και να αυξηθεί το εύρος των εικονικών κόσμων που πράγματι μπορούν να φορτωθούν χωρίς να περιορίζεται σε αυτούς που έχει φτιάξει η εφαρμογή. Αυτό θα ανοίξει την δυνατότητα ο χρήστης να μπορεί να επεξεργάζεται και χειροκίνητα τον κώδικα που παράγεται κατά την αποθήκευση ενός κόσμου, παρότι αυτό με την σειρά του συνεπάγεται την εκ νέου φόρτωση του κόσμου με τις νέες αλλαγές.

Δέκατον, θα μπορούσε να βελτιωθεί το υπάρχον user interface, προσαρμόζοντας καλύτερα το παράθυρο σε οθόνες μικρότερης ή μεγαλύτερης ανάλυσης ή δίνοντας την δυνατότητα στον χρήστη να αυξομειώσει το μέγεθος των διακριτών παραθυρικών τμημάτων της εφαρμογής, όπως του κόσμου ή του Object Navigator.

Τέλος, θα μπορούσε να δίνεται η δυνατότητα ο κόσμος να γίνεται preview μετά την αποθήκευσή του απευθείας στην πλατφόρμα Reve Worlds, ενώ μια άλλη ιδέα που μας άρεσε, αν και αναγνωρίζουμε πως η υλοποίησή της είναι αρκετά δύσκολη, είναι η εφαρμογή να επιτρέπει την σύνδεση πολλαπλών χρηστών σε αυτήν που θα σχεδιάζουν παράλληλα έναν εικονικό κόσμο. Αυτό θα είχε ιδιαίτερο ενδιαφέρον για τη χρήση της σε ένα εργαστηριακό μάθημα όπου πολλοί φοιτητές καλούνται να συνεργαστούν για αυτόν τον σκοπό.

Όπως αναλύθηκε στα προηγούμενα κεφάλαια και γίνεται κατανοητό από τα παραπάνω, συναντήσαμε αρκετές δυσκολίες κατά την υλοποίηση της εφαρμογής, οι οποίες δεν ήταν αμελητέες στον χρόνο που χρειαστήκαμε για να ολοκληρώσουμε την παρούσα διατριβή και μάλιστα κάποιες από αυτές δεν καταφέραμε να τις επιλύσουμε. Άλλωστε, το γεγονός ότι η εφαρμογή «έσπασε» σε δυο κομμάτια, όπως και οι ελλείψεις που αναδείχθηκαν στα πλαίσια της υλοποίησης, μαρτυρούν το μέγεθος της δουλειάς που κρυβόταν από πίσω καθώς και το σύνολο των δυσκολιών με τις οποίες ήρθαμε αντιμέτωποι.

Παρότι, λοιπόν, θεωρητικά ο χρήστης μπορεί με μια πρώτη ματιά να μην εντυπωσιαστεί με το πλήθος των δυνατοτήτων που δίνονται από το Reve World Designer, οφείλει να έχει υπόψη του πως η υλοποίηση έπρεπε να λάβει και να προβλέψει ένα σύνολο παραγόντων και να συνδυάσει ένα σύνολο διαφορετικών εντολών που πολλές φορές οδηγούσαν την εφαρμογή σε απρόβλεπτα σκασίματα. Επίσης, η υλοποίηση βασίστηκε ελάχιστα στο σχετικό tutorial της βιβλιοθήκη xj3d, εφόσον τις περισσότερες φορές τα links δεν λειτουργούσαν, τα παραδείγματα ήταν ανεπαρκή ενώ δεν υπήρχε καν documentation για τις υπάρχουσες μεθόδους, γεγονός που μας ανάγκασε πολλές φορές να προχωράμε στα τυφλά και να ανακαλύπτουμε τις δυνατότητες μιας μεθόδου μέσα από πειραματισμούς. Ένα χαρακτηριστικό παράδειγμα που θα μπορούσαμε να είχαμε γλιτώσει χρόνο εαν υπήρχε σχετική ενημέρωση στο tutorial του xj3d ήταν η διαγραφή ενός αντικειμένου από τον κόσμο, που προϋποθέτει να είναι το τελευταίο που έχει εισαχθεί για να γίνει σωστά, εφόσον θεωρείται ο κόμβος ρίζα.

Παραταύτα, θεωρούμε πως ο βασικός στόχος της μεταπτυχιακής μας διατριβής, δηλαδή η δημιουργία μιας παραθυρικής εφαρμογής που θα σχεδιάζει έναν εικονικό κόσμο από μηδενική βάση, έχει επιτευχθεί. Η βελτιστοποίηση της εφαρμογής ήταν εξαρχής γνωστό πως θα χρειαζόταν, εφόσον οι δυνατότητες που ανοίγονται στην περίπτωση σχεδίασης ενός εικονικού κόσμου είναι πολλές και καμιά φορά απρόβλεπτες, ενώ κάθε φορά θα μπορούσαμε να επανέλθουμε με μια καινούρια ιδέα υλοποίησης χωρίς να θέτουμε όριο στην υλοποίηση. Για αυτόν τον λόγο, η εφαρμογή κάλυψε ένα βασικό, πρωταρχικό κομμάτι λειτουργικότητας και σχεδιάστηκε με τρόπο ώστε να είναι επεξεργάσιμη και επεκτάσιμη μελλοντικά.

Σε κάθε περίπτωση, η εφαρμογή Reve World Designer είναι ένα πρωτοποριακό σχεδιαστικό εργαλείο, εφόσον αξιοποιεί το γράφημα σκηνής για την σύνθεση εικονικών κόσμων μέσω της αναπαράστασης REVE, που καθιστά εύκολη και προσιτή στον απλό χρήστη την σχεδίαση εικονικών κόσμων.

ΒΙΒΛΙΟΓΡΑΦΙΑ

Anastassakis G., 2010, "***Αναπαράσταση και Λειτουργία Εικονικών Περιβάλλοντων***", PhD Thesis at University of Piraeus

Anastassakis G., Panayiotopoulos Th., "***A Unified Model for Representing Objects with Physical Properties, Semantics and Functionality in Virtual Environments***"

Anastassakis G., Panayiotopoulos Th., "***Intelligent Virtual Environment Development with the REVE Platform as an Educational Aid***"

Anastassakis G., Panayiotopoulos Th., 2012, "***A transparent and Decentralized Model of Action for Intelligent Virtual Agents***", IEEE 24th International Conference on Tools with Artificial Intelligence

Aylett R., Cavazza M., 2001, "***Intelligent Virtual Environments - A State-of-the-art Report***"

Aylett R., Luck M., 2000, "***Applying Artificial Intelligence to Virtual Reality: Intelligent Virtual Environments***", Applied Artificial Intelligence: An International Journal, 14:1, 3-32, Taylor & Francis X πηγή: http://dx.doi.org/10.1080/088395100117142

Bowman D., North Ch., Chen j., Polys N., Pyta P., Yilmaz U., 2008, "***Information-Rich Virtual Environments Theory, Tools and Research Agenta***", from the Department of Computer Science and Center of Human-Computer Interaction, Blacksburg Virginia, USA

Bricken M., "***Virtual Words: No Interface to Design***", technical report for the Human Interface Technology Laboratory of University of Washington, Seattle

Broll W., Meier E., Schardt T, 2000, "***The Virtual Round Table: a Collaborative Augmented Multi-User Environment***" ACM on Collaborative Virtual Environments http://orgwis.gmd.de/~broll/

Brooks Fr., 1999, "***What's Real About Virtual Reality***", special report in IEEE Computer Graphics and Applications, November/December 1999

Carazo-Chandler Ch., 2000, "***Conceptualising Geography in a Virtual World Environment***", eCOSM Project

Coninx K., De Troyer O., Raymaekers Ch., Kleinermann Fr., 2006, "***VR-DeMo: a Tool-supported Approach Facilitating Flexible Development of Virtual Environments using Conceptual Modelling***"., Proceedings of Virtual Concept, Nov. 27th - Dec. 1st, Cancun, Mexico

Deitel & Deitel, 2006, "***Java Προγραμματισμός, Έκτη Έκδοση***", Μ. Γκιούρδας, Αθήνα

Dix A., Finlay J. Abowd Gr., Beale R., 2004, "***Επικοινωνία Ανθρώπου-Υπολογιστή, Τρίτη Έκδοση***", Μ. Γκιούρδας, Αθήνα

Ellis St., "***What are Virtual Environments?***", NASA Ames Research Center

Falloon G., 2010, "***Using avatars and virtual environments in learning: What do they have to offer?***", British Journal of Educational Technology, Vol. 41 No. 1, pp. 108-122

Fitzmaurice W., Balakrishnan R., Kurtenbach Gordon, 1999, "***Sampling, Synthesis and Input Devices***" Communications of the ACM

http://portal.acm.org

Gobbetti E., Scateni R., 1998, "***Virtual Reality: Past, Present and Future***", essay from the Centre of Advanced Studies, Research and Development in Sardinia, Cagliari, Italy

Greenfield E., Wloka M., 1995, "***The Virtual Tricorder: A Uniform Interface for Virtual Reality***", in UIST 95,Pitsburg PA, USA

Macedonia M., Brutzman D., Zyda M., Pratt D., Barham P., 1995, *"**NPSNET: A Multi-Player 3D Virtual Environment Over the Internet**"*, ACM - Symposium on Interactive 3D Graphics
http://www.npsnet.org/~zyda/pubs/Symposium95.pdf

Mannecke B., McNeill D., Roche E., Bray D., Townsend A., Lester J., 2008, "**Second *Life and Other Virtual Worlds: A Roadmap Research***", in Communication of the Association for Information Systems (CAIS), Volume 22, Article 20, pp. 371-388, March 2008

Robertson G., Mackinlay J., Card St., 1991, "***Cone Trees: Animated 3D Visualizations of Hierarchical Information***", paper for the Xerox Palo Alto Research Center, Coyote Hill Road, Canada

Robertson G., Card S. K., Mackinlay J. D., 1993, "***Non-Immersive Virtual Reality***", IEEE Computer Magazine
http://www2.parc.com/istl/projects/uir/UIR-R-1993-07.pdf

Schroeder R., 2008, "***Defining Virtual Worlds and Virtual Environments***", Journal of Virtual World Research, Vol. 1 No. 1, πηγή: www.jvwresearch.org

Shenchang E., 1995, "***Quick Time VR- An Image Based Approach to Virtual Environment Navigation***", for Apple Computer Inc

Links

http://blogs.sch.gr/billbas/2009/03/19/%CF%84%CE%B9-%CE%B5%CE%AF%CE%BD%CE%B1%CE%B9-%CF%84%CE%BF-second-life/

http://www.madata.gr/33/35/6584.html

http://en.wikipedia.org/wiki/Second_Life

http://www.it.uom.gr/project/mycomputer/v_card/perigr1.html

http://www.it.uom.gr/project/mycomputer/v_card/info/3d_grap1.htm

http://en.wikipedia.org/wiki/Scene_graphhttp://www.techsoft3d.com/developers/getting-started/hoops-visualize/what-is-a-scenegraph http://www.drdobbs.com/jvm/understanding-scene-graphs/184405094

http://www.acm.org/tsc/vrml.html

http://www.web3d.org/realtime-3d/x3d/profiles

http://en.wikipedia.org/wiki/Java_3Dhttp://www.java3d.org/introduction.html

http://docs.unity3d.com/Manual/LearningtheInterface.html

http://dl.acm.org/citation.cfm?id=1185461

http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=00871557

http://en.wikipedia.org/wiki/3D_modeling

http://www.tziola.gr/gr/PreviewBook.asp?ID=15

http://reactiongrid.myshopify.com/collections/virtual-worlds

http://www.tutorialspoint.com/java/index.htm

http://docs.unity3d.com/Manual/UnityBasics.html

http://www.give-lab.cs.uu.nl/movie/moviemodels/specifications/ISO-IEC-19775-FDIS-X3dAbstractSpecification/Part02/servRef.html#replaceWorld

https://savage.nps.edu/Xj3D.nps/docs/javadoc/xj3d/browser/Xj3DBrowser.html

http://onaslant.ndsu.edu/x3d_content/layering/xj3d/exmaples.htm

http://www.web3d.org/example

http://www.bitmanagement.com/developer/spec/x3d/html/java/index-all.html

http://www.xj3d.org/

http://svn.xj3d.org/xj3d_code/branches/NPS/parsetest/sai/external/

http://www.cims.nyu.edu/~meyersr/links/xj3d/vrml/field/MFNode.html

http://circe.univ-fcomte.fr/Docs/Java/Xj3D.2.0-Doc/org/web3d/vrml/renderer/ogl/browser/OGLStandardBrowserCore.html

http://www.cims.nyu.edu/~meyersr/links/xj3d/org/web3d/vrml/lang/BasicScene.html

http://www.cims.nyu.edu/~meyersr/links/xj3d/org/web3d/vrml/lang/AbstractScene.html

http://www.codejava.net/java-se/swing/jtree-basic-tutorial-and-examples

https://github.com/search?utf8=%E2%9C%93&q=xj3d+ExternalBrowser&type=Code&ref=searchresults

http://tutorials.jenkov.com/java-collections/list.html#generics

http://javalessons.com/cgi-bin/ui/java-swing.cgi?1cd=pop&sid=a0789&j2ee=jsp

http://circe.univ-fcomte.fr/Docs/Java/Xj3D.1.0-Doc/org/web3d/x3d/sai/SFVec3f.html#getValue(float[])

http://www.web3d.org/documents/specifications/19776-1/V3.0/Part01/DTD.html

http://xmsf.sourceforge.net/xsbc.html

https://savage.nps.edu/Xj3D.nps/docs/javadoc/

http://www.tutorialspoint.com/java/util/observable_notifyobservers_object.htm

http://tecfa.unige.ch/guides/vrml/vrmlman/node22.html

http://www.c3.hu/cryptogram/vrmltut/part2.html

https://docs.oracle.com/javase/tutorial/jaxp/