ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΚΡΗΤΗΣ

Σχολή Τεχνολογικών Εφαρμογών
Τμήμα Εφαρμοσμένης Πληροφορικής & Πολυμέσων

Πτυχιακή Εργασία

"Δημιουργία βασικής υποδομής υποστήριξης παιχνιδιών εικονικής πραγματικότητας "

ΟΝΟΜΑΤΕΠΩΝΥΜΟ: **ΚΟΤΑΝΙΤΣΗ ΕΛΕΝΗ**
ΗΜΕΡΟΜΗΝΙΑ:

ΕΙΣΗΓΗΤΗΣ: **ΜΑΛΑΜΟΣ ΑΘΑΝΑΣΙΟΣ**

Ευχαριστίες

Με την ολοκλήρωση της πτυχιακής μου εργασίας, θα ήθελα να ευχαριστήσω τους ανθρώπους οι οποίοι βοήθησαν στην περάτωση αυτής της εργασίας. Θα ήταν παράλειψη να μην αναφερθώ σε όλους εκείνους που συμπαραστάθηκαν σε αυτήν την προσπάθεια.

Κατά κύριο λόγο, οφείλω να ευχαριστήσω τον επιβλέποντά μου από το Α.Τ.Ε.Ι Κρήτης και διδάκτορα, κ. Μαλάμο Αθανάσιο ο οποίος με υποστήριξε καθ'όλη τη διάρκεια της πτυχιακής εργασίας. Ευγνωμοσύνη οφείλω και στον εργαστηριακό συνεργάτη Γεώργιο Μαμάκη ο οποίος σε συνεργασία με τον κ. Μαλάμο Αθανάσιο εξασφάλισε την παροχή πλούσιας υλικοτεχνικής υποδομής, πολύτιμης για μια εργασία όπως αυτή. Επίσης θα ήθελα να ευχαριστήσω την Σύμπα Παρασκευή για τη βοήθεια της στη δημιουργία του γραφικού περιβάλλοντος. Αισθάνομαι όσο λίγοι, ευνοημένη που στο διάστημα αυτό ένιωθα πάντα τη σιγουριά ότι σε κάθε βήμα είχα την υλική αλλά και ηθική βοήθεια που χρειαζόμουν για να προχωρήσω στην ολοκλήρωση αυτής της εργασίας.

Τέλος, ευχαριστώ όλους εκείνους που ήταν δίπλα μου σε όλη αυτή την προσπάθεια παρέχοντας απεριόριστη ψυχολογική υποστήριξη και κατανόηση.

Ηράκλειο, Σεπτέμβριος 2007

Περιεχόμενα

Περιεχόμενα Σχημάτων

Περιεχόμενα Πινάκων

1. Εισαγωγή

1.1 Γενική περιγραφή

Το θέμα με το οποίο ασχολείται η συγκεκριμένη πτυχιακή εργασία είναι η δημιουργία μιας πλατφόρμας που στηρίζεται στη μηχανή εικονικής αναπαράστασης X3D και στη γλώσσα προγραμματισμού Java. Η πλατφόρμα που υλοποιήθηκε δημιουργεί την υποδομή για την ενσωμάτωση αντικειμένων στον εικονικό χώρο, τα οποία περιγράφουν σε μορφή ιδιοτήτων την εμφάνιση τους στο χώρο,την αλληλεπίδραση τους με το περιβάλλον και το ρόλο που κατέχουν στο σενάριο του παιχνιδιού. Σύμφωνα με το σενάριο η εισαγωγή αντικειμένων στο χώρο συνοδεύεται από έλεγχο που πραγματοποιείται με τη χρήση αρχείων XML που είναι αποθηκευμένα εξωτερικά υλοποιείται έτσι η ύπαρξη εξωτερικού σεναρίου που περιγράφει τη δράση του παιχνιδιού. Η πλατφόρμα υλοποιεί ένα προσομοιωτή εικονικής πραγματικότητας με σκοπό την εξοικείωση του χρήστη-μαθητή με κάποιες απλές, βασικές μαθηματικές αρχές. Υλοποιείται η ιδέα παίζω και μαθαίνω μίας και για να εισαχθούν τα αντικείμενα στο χώρο πρέπει ο χρήστης να εφαρμόσει τις μαθηματικές του γνώσεις. Η εφαρμογή δηλαδή προσπαθεί να εμφυσήσει στον μαθητή μέσα από τη χρήση ενός παιχνιδιού με οπτικό περιβάλλον τον μαθηματικό τρόπο σκέψης.

1.2 Στόχος

Ο στόχος της εργασίας είναι η υλοποίηση μιας εφαρμογής η οποία χρησιμοποιώντας το τρισδιάστατο γραφικό και διαδραστικό περιβάλλον θα παρέχει μια πιο ελκυστική και διασκεδαστική όψη των μαθηματικών στο χρήστη. Η επίτευξη αυτού του στόχου περιλαμβάνει την χρήση των τρισδιάστατων χαρακτηριστικών που αποτελεί πόλο έλξης στο χρήστη. Ο συνδυασμός του X3D και της Java για την υλοποίηση της εφαρμογής είναι ακόμη ένας στόχος που υλοποιήθηκε χρησιμοποιώντας την κατά μία έννοια ύπαρξη εξωτερικού σεναρίου δίνοντας έτσι τη δυνατότητα της αλλαγής του σεναρίου του παιχνιδιού που υλοποιήθηκε χωρίς να χρειαστεί να αλλάξει ολόκληρος ο κώδικας του προγράμματος.

1.3 Εργαλεία υλοποίησης

Για την υλοποίηση αυτής της εφαρμογής χρησιμοποιήθηκαν τρία διαφορετικά εργαλεία. Η βασική ιδέα για την υλοποίηση της πλατφόρμας ήταν η χρήση του X3D σε συνδυασμό με τη Java.Η XML θεωρήθηκε σωστό να χρησιμοποιηθεί ώστε να υλοποιηθεί η εισαγωγή εξωτερικού ελέγχου και κατά κάποιο τρόπο και σεναρίου . Για να υλοποιηθεί αυτή η εφαρμογή χρησιμοποιήθηκε κυρίως η γλώσσα Java, έτσι έπρεπε να εντοπιστεί ένα εργαλείο που θα μπορεί να υποστηρίξει την πλατφόρμα. Υπάρχουν διάφορα

εργαλεία που υποστηρίζουν X3D και Java μαζί . Κάποια από αυτά δεν υποστηρίζουν πλήρως τη Java και κάποια άλλα δεν είναι δωρεάν. Έτσι χρησιμοποιήθηκε το Xj3D που και είναι δωρεάν, και υποστηρίζει πλήρως τη γλώσσα.

1.3.1 Java

Η Java αποτελεί την κυρίαρχη και βασική ιδέα στην εφαρμογή. Όλα όσα διαδραματίζονται διαχειρίζονται από αυτή. Ο ρόλος της Java λοιπόν, ήταν να αποδώσει τη διάδραση ανάμεσα στον κόσμο του παιχνιδιού (X3D) τις δράσεις του χρήστη και την αυτοματοποίηση λειτουργιών που σχετίζονται με την κύρια ιδέα του παιχνιδιού (χρόνος ,γεγονότα,αλληλεπιδράσεις κλπ).Στην ουσία η Java αποδίδει στον 3D κόσμο τη λειτουργικότητα και τη διαδραστικότητα, έχοντας τον πλήρη έλεγχο όλων των γεγονότων που δημιουργούνται στον κόσμο του παιχνιδιού. Χρησιμοποιώντας τις κλάσεις του X3D αποκτάται επικοινωνία ανάμεσά στην Java και τον τρισδιάστατο κόσμο.

1.3.2 X3D

Ο κόσμος του παιχνιδιού δημιουργήθηκε με τη χρήση του X3D.Το X3D είναι μια XML γλώσσα και αποτελεί ένα ISO πρότυπο για την αναπαράσταση σε πραγματικό χρόνο 3D γραφικών ,είναι ο διάδοχος της Virtual Reality Modeling Language (VRML) και έχει τη δυνατότητα να αναπαριστά και να αποδίδει τρισδιάστατες σκηνές και αντικείμενα χρησιμοποιώντας τον τρόπο λειτουργίας και την δομή της XML. Στην συγκεκριμένη πλατφόρμα το X3D αναπαριστά τον κόσμο στον οποίο διαδραματίζονται τα γεγονότα(events που διαχειρίζεται η java) και στον οποίο υλοποιείται το σενάριο του παιχνιδιού. Τα αντικείμενα που εισάγονται στον εικονικής πραγματικότητας κόσμο είναι και αυτά X3D. Σε αυτό το σημείο πρέπει να επισημανθεί ότι τα αντικείμενα (κτίρια, δρόμοι, πάρκα) που έχει τη δυνατότητα να τοποθετήσει ο χρήστης στο γραφικό περιβάλλον καθώς και το ίδιο το γραφικό περιβάλλον της εφαρμογής είναι κατασκευασμένα με το 3D Studio Max και στη συνέχεια μέσω του exporter ο οποίος τα μετατρέπει σε X3D, δόθηκε η δυνατότητα να χρησιμοποιηθούν στη συγκεκριμένη εργασία.

1.3.3 XML

Η περιγραφή του σεναρίου και συγκεκριμένα των ιδιοτήτων των αντικειμένων, της κατάστασης του κόσμου “status” ,των ερωτήσεων που εμφανίζονται στην εφαρμογή και των καταστροφών που λαμβάνουν χώρα, γίνεται με την χρήση XML αρχείων. Η XML είναι ένα απλό και πολύ ευέλικτο είδος κειμένου, μια γενικού σκοπού markup language. Είναι μια γλώσσα που συνδυάζει κείμενο μαζί με περισσότερες πληροφορίες σχετικά με το κείμενο, οι οποίες εκτός των άλλων περιλαμβάνουν τη δομή και την εμφάνιση του. Ο κύριος σκοπός της είναι να διευκολύνει τη μεταφορά δομημένης πληροφορίας

ανάμεσα σε διαφορετικά συστήματα. Η γλώσσα XML κατατάσσεται στις γλώσσες υψηλού επιπέδου μιας και επιτρέπει στο χρήστη να τροποποιήσει ή να εμπλουτίσει τη σύνταξη της και να δημιουργήσει τα δικά του tags.

1.3.4 Xj3D

Τέλος για την παρουσίαση της εφαρμογής χρησιμοποιήθηκε το Xj3D. Το Xj3D είναι έργο της Web3D Consortium ,είναι ένα εργαλείο γραμμένο αποκλειστικά σε java και λειτουργεί ως browser της εφαρμογής. Το Xj3D υποστηρίζει έναν stand alone viewer δηλαδή ένα browser που λειτουργεί μόνος του χωρίς να χρειάζεται κάποια άλλη εφαρμογή. Το συγκεκριμένο εργαλείο επιλέχθηκε μεταξύ άλλων, λόγω της δυνατότητας υποστήριξης, του Java κώδικα.

2. Η εφαρμογή

2.1 Γενική παρουσίαση της εφαρμογής

Η γενική ιδέα της εφαρμογής βασίζεται στην ύπαρξη ενός τρισδιάστατου γραφικού διαδραστικού περιβάλλοντος το οποίο υποστηρίζει την εισαγωγή τρισδιάστατων αντικειμένων. Για να μπορέσει να συγκεκριμενοποιηθεί το θέμα, επιλέχθηκε η υλοποίηση ενός σεναρίου κατά το οποίο στο περιβάλλον εισάγονται τρισδιάστατα κτίρια,σε μορφή X3D μετά από εντολή του χρήστη. Στην εφαρμογή ο χρήστης εκτός των άλλων έχει τη δυνατότητα να επιλέξει από το μενού το κτίριο που θέλει, να χτίσει στον κόσμο του. Για να το πετύχει αυτό θα πρέπει να απαντήσει σωστά σε απλές μαθηματικές ερωτήσεις που του ζητούνται από την εφαρμογή. Σε αυτό το σημείο εισάγεται στο περιβάλλον η μαθηματική σκέψη,δημιουργείται έτσι μια εφαρμογή που συνδυάζει το παιχνίδι με την γνώση. Με αυτόν τον τρόπο δίνεται η δυνατότητα στο χρήστη να εφαρμόσει τη μαθηματική του σκέψη σε έναν κόσμο που δημιουργεί ο ίδιος,και ο οποίος διέπεται από επίπεδα και παραμέτρους. Καθώς τα επίπεδα αυξάνονται η τρέχουσα κατάσταση του κόσμου αλλάζει. Επιπλέον στα πλαίσια της υλοποίησης της εφαρμογής δίνεται η δυνατότητα στο χρήστη να αποθηκεύσει την τρέχουσα κατάσταση το παιχνιδιού, να επαναφέρει την τελευταία αποθηκευμένη κατάσταση και να ελέγξει την τρέχουσα κατάσταση.

2.2 Σενάριο

Το σενάριο αφορά στην δημιουργία μιας πόλης, ενός κόσμου, από τον χρήστη. Η δημιουργία γίνεται καθώς ο χρήστης επιλέγει να αναγείρει οικοδομήματα στον εικονικής πραγματικότητας κόσμο. Στην αρχή του παιχνιδιού δίνεται ένα αρχικό ποσό στον χρήστη ώστε να μπορέσει να κάνει τις πρώτες του κινήσεις. Στόχος του χρήστη είναι η δημιουργία μίας υγιούς υποδομής,που θα περιέχει τον κόσμο παρά τις δυσκολίες που εμφανίζονται

κατά τη διάρκεια του παιχνιδιού. Για να επιτευχθεί η ανέγερση ενός κτιρίου πρέπει να ικανοποιούνται κάποιες συνθήκες και παράμετροι. Καθώς ο κόσμος επεκτείνεται αυξάνονται οι αναπτυσσόμενοι δείκτες και η οικονομία του παίχτη. Εφ' όσων ικανοποιούνται οι συνθήκες για την ανέγερση του εκάστοτε κτιρίου, η δυνατότητα ανέγερσης κτιρίου αποκτάται μόνο όταν απαντηθεί σωστά το μαθηματικό ζήτημα που τίθεται κάθε φορά από το πρόγραμμα. Στόχος του χρήστη είναι να καταφέρει να φτάσει στο τρίτο επίπεδο ανάπτυξης χτίζοντας όσο περισσότερα κτίρια και απαντώντας σε όσο το δυνατόν περισσότερες ερωτήσεις. Στον τρισδιάστατο κόσμο έχουν υλοποιηθεί καταστροφές οι οποίες εμφανίζονται σε τυχαίες χρονικές στιγμές και είναι ικανές να καταστρέψουν οικοδομήματα και να μειώσουν την οικονομία του χρήστη ανάλογα με το μέγεθος τους το οποίο κάθε φορά είναι διαφορετικό.

2.2.1 Παράμετροι του κόσμου

Μία από τις βασικές παραμέτρους τού σεναρίου είναι οι πόντοι. Οι πόντοι **points** είναι τα χρήματα που κερδίζει ο χρήστης καθώς εξελίσσεται το παιχνίδι και αναπτύσσεται η πόλη. Άλλοι παράμετροι του παιχνιδιού είναι οι ακόλουθοι:**Economy**,με την έναρξη του παιχνιδιού η οικονομία είναι μηδενική εκτός από το αρχικό ποσό που δίνεται στο χρήστη. Όσο περισσότερο επεκτείνεται ο κόσμος και ανάλογα με τα κτίρια που επιλέγει ο χρήστης να αναγείρει αυξάνεται και η οικονομία του παιχνιδιού. **Environment**, στόχος του χρήστη που θέλει να αυξήσει τη συγκεκριμένη μεταβλητή, είναι η τοποθέτηση στον κόσμο πάρκων και κατασκευών που είναι φιλικά προς το περιβάλλον. **Health**, καλό είναι ο κόσμος να έχεί μεγάλο δείκτή υγείας. Έτσι ο χρήστης α πρέπει να τοποθετήσει νοσοκομείο στον κόσμο του. **Culture**, για να αυξηθεί η παράμετρός και να μπορέσει να εισαχθεί η έννοια της κουλτούρας σε έναν εικονικής πραγματικότητας κόσμο θα πρέπει ο χρήστης να εγκαταστήσει ένα εκπαιδευτικό σύστημα. Εισάγοντας σχολεία και πανεπιστήμια στον κόσμο,εκτός από την κουλτούρα ο χρήστης κερδίζει και πόντους. **Civil protection**, Η αύξηση της παραμέτρου γίνεται με τη δημιουργία τμημάτων που προστατεύουν τον πολίτη και βρίσκονται συνεχώς στις υπηρεσίες του όπως το αστυνομικό τμήμα και η πυροσβεστική. **Transportation,** ένας ολοκληρωμένος κόσμος θα πρέπει να περιέχει και δρόμους έτσι επιλέγοντας να δημιουργηθούν δρόμοι, ο χρήστης κερδίζει εκτός από την αύξηση της παραμέτρου transportation και την αύξηση της οικονομίας του. **Quality of life**, η παράμετρός αυτή δείχνεί την εξέλιξη του κόσμου και καλή κατάσταση του εικονικού κόσμου .Καθώς εξελίσσεται ο κόσμος και δημιουργούνται κτίρια, η ποιότητα της ζωής αυξάνεται. **Education**, είναι προφανές ότι για την αύξηση αυτής της παραμέτρου ευθύνεται η δημιουργία εκπαιδευτικών ιδρυμάτων. **Technology,** η τεχνολογία αυξάνεται με τη δημιουργία τεχνολογικών πάρκων και ιδρυμάτων ερευνάς.**Constructions** τέλος για τη συγκεκριμένη παράμετρο απαιτείται η δημιουργία όλων των ειδών κτιρίων τα οποία συνεισφέρουν το καθένα με διαφορετικό τρόπο στην ανάπτυξη του κόσμου.

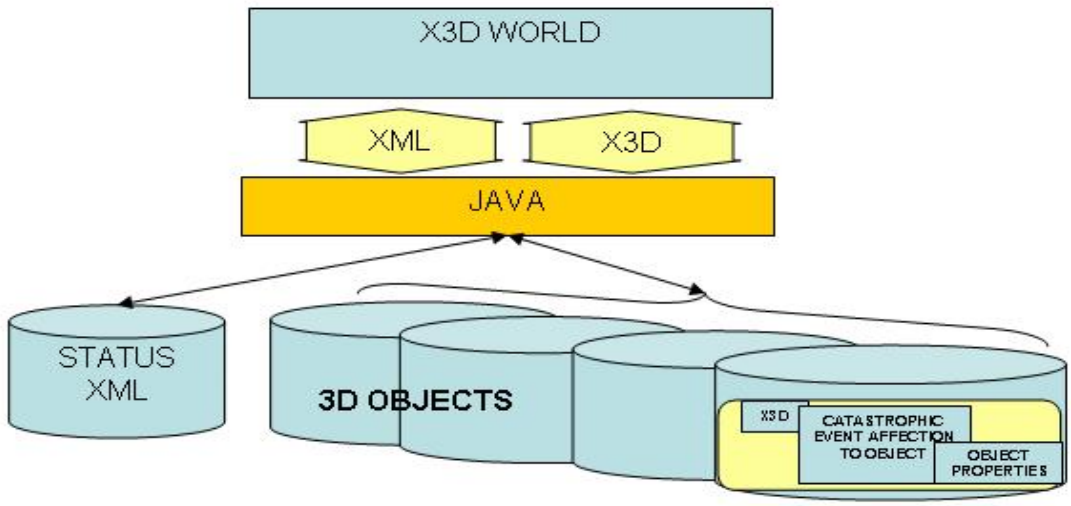
Με λίγα λόγια ανάλογα με το αντικείμενο που εισάγεται αυξάνονται οι συγκεκριμένοι παράμετροι που επηρεάζονται από αυτό. Και τελικά η εξέλιξη του παιχνιδιού έχει σαν αποτέλεσμα την αύξησή όλων των παραμέτρων-μεταβλητών.

Τέλος υπάρχει μια ακόμη παράμετρος η οποία δημιουργεί απρόσμενες καταστάσεις στο παιχνίδι. Η παράμετρος αυτή είναι τα **Catastrophic events** τα οποία προσομοιώνουν καταστροφικά γεγονότα που υπάρχουν και στην πραγματικότητα όπως σεισμούς, πλημμύρες και πυρκαγιές,και τα οποία όπως αναφέρθηκε παραπάνω έχουν σαν αποτέλεσμα είτε την καταστροφή μέρους του κόσμου είτε την αφαίρεση ποσού από αυτό που διαθέτει ο χρήστης. Οι καταστροφές αυτές εμφανίζονται σε τυχαίες χρονικές στιγμές και ο χρήστης δε μπορεί να κάνει τίποτα για να τις αποφύγει. Ο χρόνος εμφάνισής τους καθώς και το μέγεθος τους είναι τυχαίος. Έτσι οι ζημιές είναι ανάλογες με το βαθμό που επηρεάζεται το κάθε κτίριο από την εκάστοτε καταστροφή.

2.3 Αρχιτεκτονικός σχεδιασμός εφαρμογής

2.3.1 Δομή εφαρμογής

Η πλατφόρμα που υλοποιήθηκε αποτελείται από τρία βασικά τμήματα.1.) Το X3D που ουσιαστικά είναι ένα XML κείμενο που έχει τη δυνατότητα να δημιουργεί και να απεικονίζει 3D αντικείμενα και περιβάλλοντα. 2.)Τα XML αρχεία που χρησιμοποιήθηκαν για την απόδωσε του σεναρίου του παιχνιδιού και τέλος 3.) τη Java που αποτελεί το συνδετικό κρίκο ανάμεσα στο X3D και τα XML αρχεία, και διαχειρίζεται όλα τα events του παιχνιδιού. Το σχήμα που ακολουθεί αποδίδει τη δομή της πλατφόρμας:



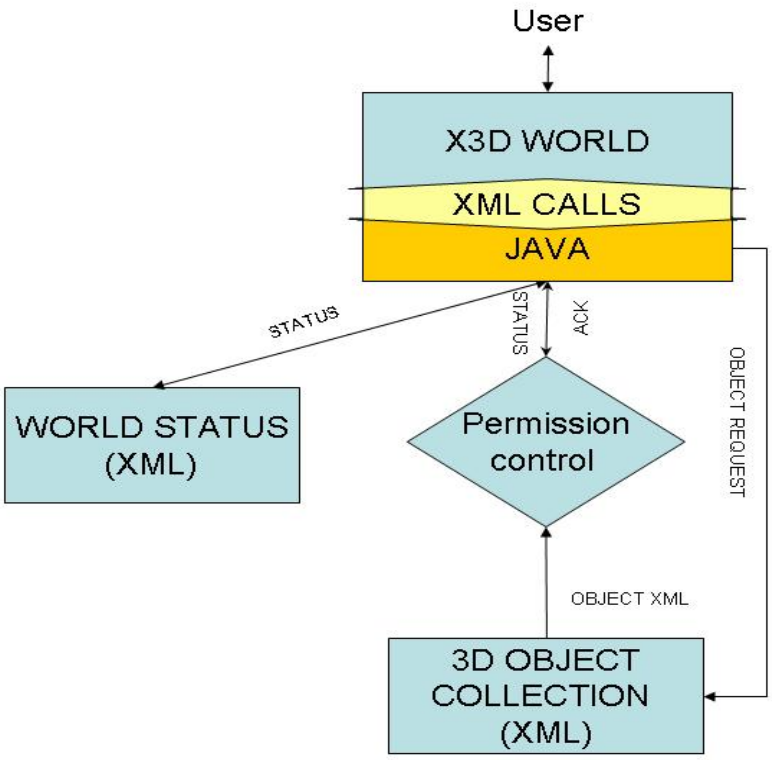
Σχήμα 1

Όπως φαίνεται στο σχήμα η πλατφόρμα περιλαμβάνει το X3D το XML και τη Java.Ο κόσμός αποτελείται από xml και X3D αρχεία. Το αρχείο Status είναι ένα XML αρχείο στο οποίο αποθηκεύεται οι παράμετροι του κόσμου και εμπεριέχει τον καθορισμό των επιπέδων (levels) του παιχνιδιού. Έτσι όποτε υπάρχει κάποια αλλαγή στις παραμέτρους και άρα στον ίδιο τον κόσμο το αρχείο αυτό καλείται από τη Java για να διαχειριστεί. Στη συνέχεια και όπως φαίνεται στο σχήμα 1 υπάρχουν τα αντικείμενα που εισάγονται στον 3D κόσμο. Τα αντικείμενα αυτά αποτελούνται από το αρχείο X3D που αποτελεί την εικονική αναπαράσταση τους,το XML αρχείο με τις επιδράσεις που ασκούνται πάνω στο αντικείμενο από τα καταστροφικά γεγονότα., το XML αρχείο με τις ιδιότητες του αντικειμένου καθώς επίσης και την φωτογραφία του αντικειμένου. Όλα αυτά τα αρχεία αποτελούνε τα αντικείμενα που χρησιμοποιούνται στο παιχνίδι και χρησιμοποιούνται από τη Java κάθε φορά που εισάγεται ένα νέο αντικείμενο στο χώρο.

2.3.2 Σχεδιασμός του αλγορίθμου εισαγωγής αντικειμένου (3D object)

Η πλατφόρμα υποστηρίζει την εισαγωγή αντικείμενων X3D.Για να επιτευχθεί αυτό σχεδιάστηκε ένας αλγόριθμος για την υλοποίηση της συγκεκριμένης λειτουργίας. Ο αλγόριθμος αυτός φαίνεται στο σχήμα 2 που ακολουθεί. Συγκεκριμένα όταν ο χρήστης δώσει την εντολή για την εισαγωγή ενός νέου κτιρίου στην εφαρμογή κάνοντας μια διάδραση με το περιβάλλον, η πλατφόρμα καλεί τη Java η οποία διαβάζεί τα XML αρχεία (status) και (properties).Έτσι καλείται η κλάση Housemanager (σχήμα 4)

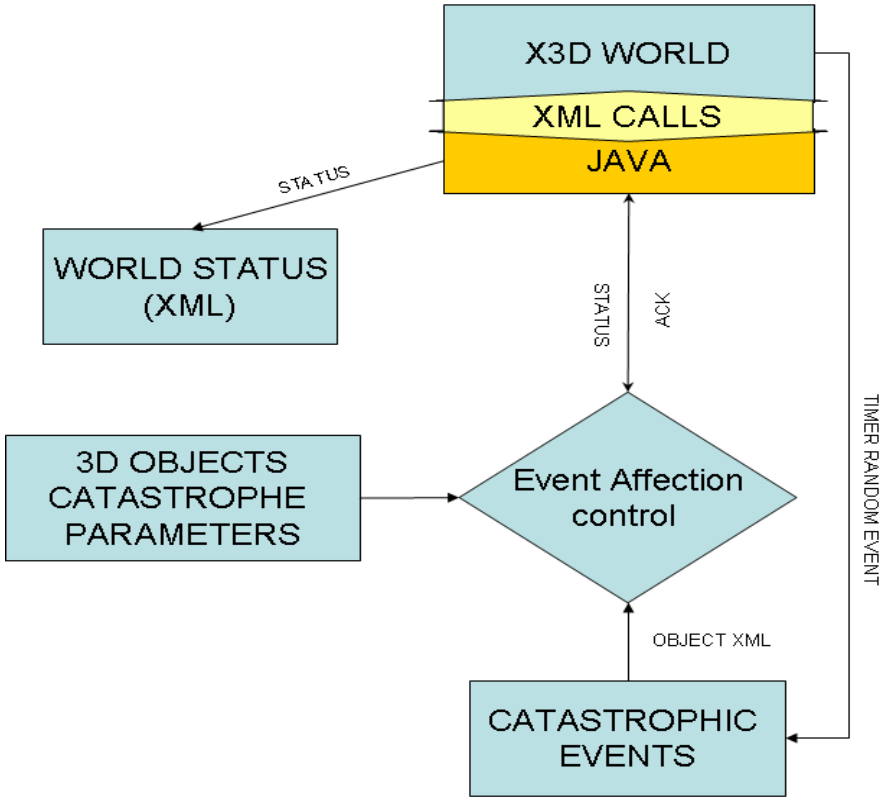
Το αρχείο Status περιέχει τις τιμές των παραμέτρων που χρησιμοποιεί η εφαρμογή και τον καθορισμό των levels.Το αρχείο properties για κάθε αντικείμενο είναι διαφορετικό,και περιέχει τις τιμές των παραμέτρων που αποδίδει στον κόσμο το κάθε κτίριο,καθώς και τα προαπαιτούμενα κτίρια που χρειάζονται για να μπορέσει να εισαχθεί το εκάστοτε αντικείμενο στο χώρο. Έτσι μόλις η Java αποκτήσει πρόσβαση σε αυτά τα δύο αρχεία γίνεται ο έλεγχος για την ικανοποίηση των προαπαιτούμενων όπως φαίνεται στο σχήμα 2 . Εάν το αποτέλεσμα του ελέγχου είναι θετικό και άρα ικανοποιούνται οι συνθήκες για την εισαγωγή του νέου αντικειμένου στο χώρο τότε το αντικείμενο αυτό είναι έτοιμο για να εισαχθεί στον κόσμο. Σε αυτό το σημείο καλείται από τη Java η κλάση WorldManager η οποία με τη σειρά της χρησιμοποιεί άλλες κλάσεις για να εισαχθεί το αντικείμενο στο κόσμο όπως φαίνεται στο σχήμα 4.Πριν όμως από την κλάση WorldManager καλούνται οι κλάσεις JQuestion και QuestionManager οι οποίες είναι υπεύθυνες για τη δημιουργία και τον έλεγχο των ερωτήσεων και απαντήσεων που δίνει ο χρήστης στην πλατφόρμα. Στη συνέχεια μέσω της Java και της κλάσης WorldManager γίνετε ανανέωση του αρχείου status με την πρόσθεση των παραμέτρων του νέου κτιρίου που εισήχθη στο τρισδιάστατο περιβάλλον,έτσι ώστε η τρέχουσα κατάσταση του κόσμου να είναι ενήμερη για τις αλλαγές.



Σχήμα 2

2.3.3 Σχεδιασμός του αλγορίθμου καταστροφικών γεγονότων (Catastrophic Events)

Στα πλαίσια της υλοποίησης του σεναρίου της πλατφόρμας που υλοποιήθηκε, σχεδιάστηκε και αναπτύχθηκέ ένας αλγόριθμος για την προσομοίωση των καταστροφικών γεγονότων που εμφανίζονται στο παιχνίδι κατά τη διάρκεια της εκτέλεσής του. Στο σχήμα που ακολουθεί φαίνεται ο τρόπος λειτουργίας του συγκεκριμένου αλγορίθμου.



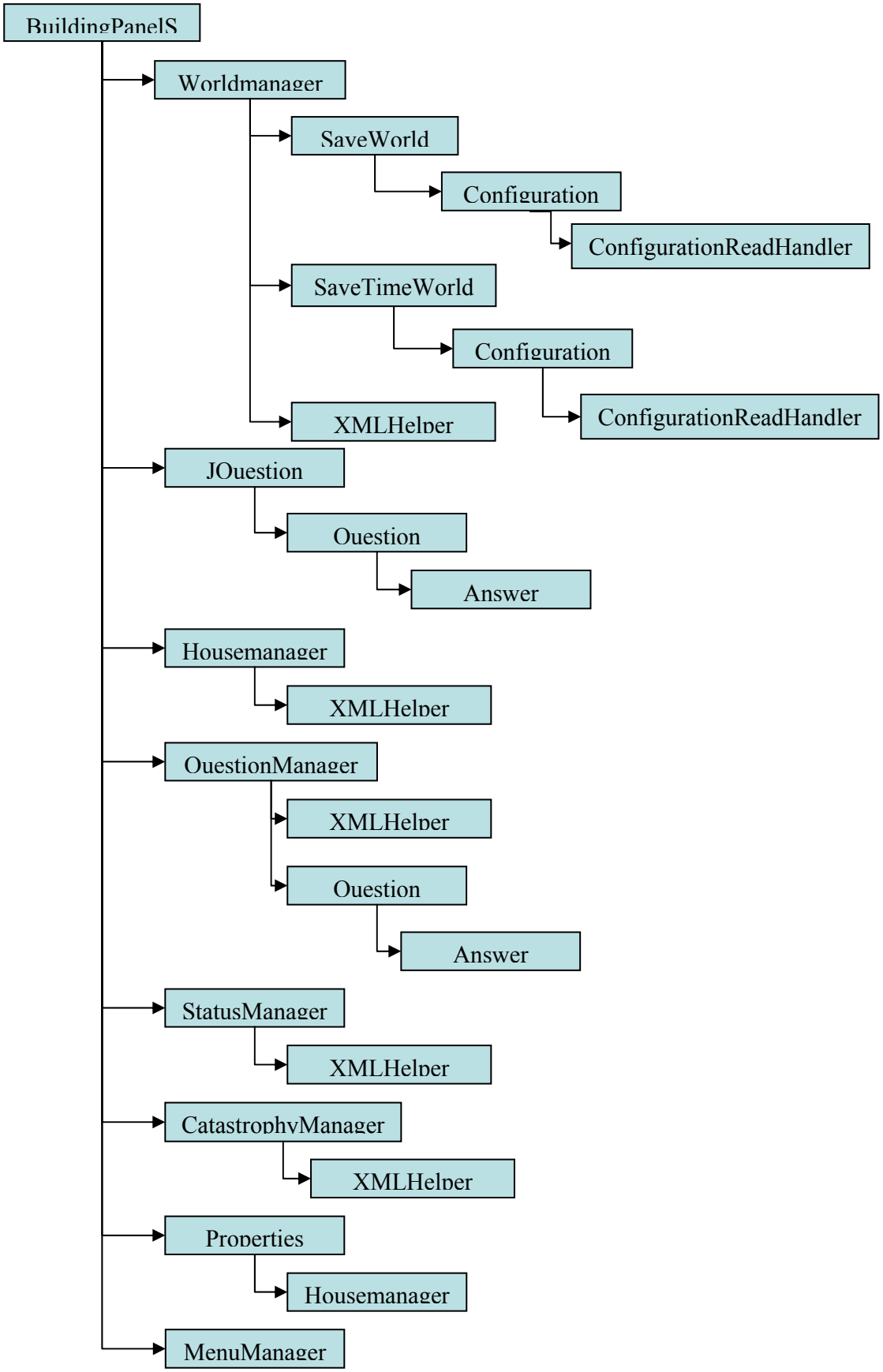
Σχήμα 3

Κατά τη διάρκεια εκτέλεσης της εφαρμογής μπορούν να συμβούνε τριών ειδών καταστροφές (σεισμός, πλημμύρα, φωτιά).Στον κόσμο X3D έχει ενσωματωθεί ένας timer ο οποίος δημιουργεί τέτοιου είδους events όταν φτάσει μια χρονική στιγμή,τυχαία κάθε φορά. Μόλις λοιπόν δημιουργηθεί ένα event τέτοιου τύπου,καλείται από τη Java η κλάση που είναι υπεύθυνη για τη διαχείριση των καταστροφικών γεγονότων και είναι η catastrophymanager. Αρχικά φορτώνεται το αρχείο της συγκεκριμένης καταστροφής το οποίο είναι XML αρχείο και υπολογίζεται το μέγεθος και το ποσοστό καταστροφής επί των συνολικών αντικειμένων του κόσμου. Στη συνέχεια εντοπίζονται τα αντικείμενα που επηρεάζονται από τη συγκεκριμένη καταστροφή χρησιμοποιώντας το αρχείο catastrophy του κάθε αντικειμένου το οποίο και καλείται από τη Java.Το αρχείο αυτό είναι διαφορετικό για κάθε αντικείμενο και περιέχει πληροφορίες σχετικά με το είδος των καταστροφών που επηρεάζουν το κάθε κτίριο , το μέγεθος της καταστροφής που μπορεί να αντέξει ένα αντικείμενο και πάνω από το οποίο καταστρέφεται και τέλος το ποσό που κοστίζει η διόρθωσή της φθοράς που προκλήθηκε στο κάθε κτίριο. Μετά υπολογίζονται οι συνολικές οικονομικές και οικοδομικές καταστροφές και αφού ενημερωθεί ο χρήστης για το μέγεθος της καταστροφής που συνέβη,καλείται από τη Java το αρχείο status και ενημερώνεται για τις νέες συνθήκες. Τέλος ενημερώνεται ο κόσμος για τις αλλαγές που προκλήθηκάν από την καταστροφή,αλλάζοντας την γραφική αναπαράσταση του κόσμου στην περίπτωση που κάποια κτίρια έχουν κατεδαφιστεί έτσι καλείται η κλάση

SaveTimeWorld η οποία είναι υπεύθυνη για την αποθήκευση της τρέχουσας κατάστασης του κόσμου και την ενημέρωση του timer που δημιουργεί τα events που είναι υπεύθυνα για τις καταστροφές .

3 Υλοποίηση

Για να υλοποιηθεί η εφαρμογή που παρουσιάζεται ήταν απαραίτητη η δημιουργία πολλών κλάσεων, έτσι θεωρήθηκε σωστό να γίνει μια παρουσίαση της δομής των κλάσεων με τέτοιο τρόπο ώστε να γίνει κατανοητή η μεταξύ τους επικοινωνία,σε μια προσπάθεια να αποδοθεί ο τρόπος κλήσης και η μεταξύ τους εξάρτηση. Στο γράφημά που ακολουθεί επιχειρείται η προσέγγιση της δομής του προγράμματος με δενδροειδή μορφή, ξεκινώντας από τη βασική κλάση της εφαρμογής την BuildingPanelSC. Στη συνέχεια παρουσιάζεται αναλυτική αναφορά για το ρόλο της κάθε κλάσης μέσα στον εφαρμογή. Αναλύοντας και παρουσιάζοντας ξεχωριστά την κάθε λειτουργία της πλατφόρμας εξηγείται ο τρόπος αλληλεπίδρασης μεταξύ των κλάσεων και ο λόγος της ύπαρξής τους.


BuildingPanelS
Worldmanager
SaveWorld
Configuration
ConfigurationReadHandler
SaveTimeWorld
Configuration
ConfigurationReadHandler
XMLHelper
JQuestion
Question
Answer
Housemanager
XMLHelper
QuestionManager
XMLHelper
Question
Answer
StatusManager
XMLHelper
CatastrophyManager
XMLHelper
Properties
Housemanager
MenuManager

Σχήμα 4

3.1 Εκτέλεση εφαρμογής

Για να εκτελεστεί η εφαρμογή, ο χρήστης πρέπει να φορτώσει στον Xj3D browser το αρχικό περιβάλλον. Το αρχικό περιβάλλον είναι ο τρισδιάστατος κόσμος με τον οποίο ο χρήστης δρα και εξελίσσει τον κόσμο του. Το αρχείο αυτό είναι το background.x3d. Σε αυτό το σημείο θα πρέπει να επισημανθεί ότι για να εκτελεστεί σωστά η πλατφόρμα θα πρέπει ολόκληρη η εφαρμογή να εγκατασταθεί μέσα στο φάκελο του Xj3D. Σε αυτό το σημείο κάνει την εμφάνιση της η Java και ο ρόλος της. Μόλις ο Xj3D browser ανοίξει το background αυτομάτως καλείται από τη java η κλάση BuildingPanelSC. Η κλάση αυτή είναι η κύρια κλάση που διαχειρίζεται όλα τα γεγονότα που διαδραματίζονται μέσα στο παιχνίδι και η οποία καλεί τις υπόλοιπες βασικές κλάσεις όποτε χρειάζεται.

Με την εκτέλεση της κλάσης BuildingPanelSC, αρχικοποιούνται οι κλάσεις WorldManager, MenuManager, QuestionManager, housemanager καθώς και όλοι οι sensors που υπάρχουν στον κόσμο του X3D και είναι υπεύθυνοι για τη δημιουργία των γεγονότων .

Αρχικά στον κόσμο απεικονίζεται το background, δεξιά υπάρχει το μενού της εφαρμογής μέσω του οποίου ο χρήστης μπορεί να δημιουργήσει γεγονότα, δηλαδή μπορεί να προσθέσει ένα κτίριο στον εικονικής πραγματικότητας κόσμο,να το "κτίσει" απενεργοποιώντας ουσιαστικά τη δυνατότητα της μετακίνησης του στο χώρο, να ελέγξει την τρέχουσα κατάσταση του κόσμου δηλαδή τις παραμέτρους του κόσμου, να ελέγξει τις ιδιότητες των αντικειμένων-κτιρίων, να αποκρύψει και να εμφανίσει το μενού καθώς και να περιηγηθεί στον κόσμο είτε από το μενού που παρέχει ο Xj3D browser είτε αλλάζοντας viewpoint. Στην εφαρμογή που αναπτύχθηκε στα πλαίσια της πτυχιακής εργασίας, κρίθηκε σκόπιμο να ενσωματωθεί η δυνατότητα δυναμικής απεικόνισης των αντικειμένων από συγκεκριμένες οπτικές γωνίες,δίνοντας έτσι στον χρήστη την δυνατότητα σαφέστερης αντίληψης της δημιουργίας του και μία πιο ολοκληρωμένη άποψη της δυναμικής του χώρου. Έτσι δημιουργήθηκαν πέντε διαφορετικά viewpoints τα οποία δείχνουν τον εικονικό κόσμο από διαφορετικές οπτικές γωνίες(Front View, Rear View, Left View, Right View, Top View). Επίσης ο χρήστης μπορεί να αποκτήσει πρόσβασή σε περισσότερα από αυτά που αρχικά φαίνονται αντικείμενα, χρησιμοποιώντας τα βέλη που βρίσκονται στο μενού και τα οποία προσομοιώνουν την μπάρα κύλισης η αλλιώς scroll bar.

Στον εικονικής πραγματικότητας κόσμο ενεργοποιούνται οι sensors που είναι υπεύθυνοι για τη δημιουργία των γεγονότων, ένας TimeSensor που είναι υπεύθυνος για τη δημιουργία των καταστροφών (catastrophic events), και ένας TimeSensor που είναι υπεύθυνος για την περιστροφή της γης δηλαδή το κουμπί απόκρυψης και εμφάνισης του μενού που βρίσκεται κάτω δεξιά.

Σχετικά με την εφαρμογή και εφόσον δημιουργήθηκε η οπτική απεικόνιση του μενού η οποία αποδίδεται από το X3D αρχείο, έπρεπε να αποδοθεί η λειτουργιά των buttons στα boxes του X3D η αλλιώς στα εικονίδια που δημιουργήθηκαν και αποτελούν το μενού της εφαρμογής. Αυτό έγινε

δυνατό με την δημιουργία των fields μέσα στο X3D αρχείο τα οποία συνδέονται με τα αντικείμενα-εικόνες με routes.Με αυτόν τον τρόπο αποδίδεται ο ρόλος των buttons στο X3D. Η ενεργοποίηση της λειτουργίας των Buttons γίνεται από την κλάση BuildingPanelSC η οποία καλώντας το X3D αρχείο εντοπίζει τα fields που μας ενδιαφέρουν και στα οποία έχουν προστεθεί routes και τους προσθέτει event listeners.

3.2 Υλοποίηση: Scroll Bar

Η λειτουργία της μπάρας κύλισης ή scrollbar κρίθηκέ απαραίτητη για να μπορέσουν να ενσωματωθούν πολλά αντικείμενα στο μενού της εφαρμογής χωρίς αυτό να καλύπτει μεγάλο μέρος του παραθύρου με το οποίο ο χρήστης έρχεται σε επαφή και δρα. Έτσι υλοποιήθηκε αλγόριθμος που προσομοιώνει τη μπάρα κύλισης. Η λειτουργία της γίνεται εφικτή με την κλάση MenuManager η οποία καλείται από την κλάση BuildingPanelSC, κάθε φορά που ενεργοποιείται ο event listener από κάποιο από τα δυο buttons με το σχήμα του βέλους. Η κλάση Menumanager μετακινεί τα αντικείμενα-buttons, αποκρύπτοντας τα και αλλάζοντας τη θέση τους πάνω στη μπάρα του μενού, δηλαδή αλλάζοντας τη θέση τους στο χώρο. Για να επιτευχθεί αυτό χρησιμοποιούνται οι εκάστοτε τιμές των μεταβλητών translation και visibility του κάθε X3D αντικειμένου. Η μεταβλητή translation είναι υπεύθυνη την θέση του αντικειμένου στο χώρο και με τη κλάση MenuManager αλλάζουν οι τιμές της μετακινώντας έτσι τα αντικείμενα πάνω στο μενού. Η μεταβλητή visibility είναι υπεύθυνη για την εμφάνιση και την απόκρυψη του αντικειμένου. Κάθε φορά που επιλέγεται ένα από τα δύο βέλη καλούνται οι μέθοδοι goUp και goDown ανάλογα με την επιλογή.

Αρχικά είναι διαθέσιμή μονό η προς τα κάτω μετακίνηση έτσι θα παρακάτω θα εξηγηθεί αναλυτικά ο τρόπος που δουλεύει η συγκεκριμένη μέθοδος μιας και με παρόμοιο τρόπο δουλεύουν και οι άλλες. Επιλέγοντας λοιπόν το προς τα κάτω βέλος καλείται η μέθοδος goDown. Αρχικά γίνεται έλεγχος αν βρισκόμαστε στο τέλος της λίστα με τα αντικείμενα. Εφ' όσων υπάρχουν ακόμη αντικείμενα για να εμφανιστούν καλείται η μέθοδος hide. Η μέθοδος αυτή αποκρύπτει το πρώτο αντικείμενο-κουμπί που υπάρχεί στο μενού χρησιμοποιώντας τη μεταβλητή Visibility και δίνοντας της τιμή NotVisible. Στη συνέχεια καλείται η μέθοδος show η οποία αναλαμβάνει να κάνει ορατό το αντικείμενο που πρόκειται να εισαχθεί στη μπάρα του μενού και θα πάρει την τελευταία θέση. Αυτό γίνεται εφικτό δίνοντας στη μεταβλητή Visibility την τιμή Visible. Στη συνέχεια και μέχρι να σαρωθούν όλα τα αντικείμενα –κουμπιά εκτελείται η μέθοδος moveUp. Η μέθοδος αυτή εντοπίζει το κάθε αντικείμενο και παίρνει τη μεταβλητή translation που του αντιστοιχεί. Η μεταβλητή αυτή έχει αποθηκευμένη την θέση του αντικειμένου στο χώρο και αποτελείται από τρεις παραμέτρους που αντιστοιχούν στους τρεις άξονες του τρισδιάστατου συστήματος αναπαράστασης. Στη συνέχεια η μέθοδος αυτή αυξάνει την τιμή της παραμέτρου Y της μεταβλητής translation.

Η παράμετρός Y αντιστοιχεί τον κάθετο άξονα, με αυτόν τον τρόπο γίνεται εφικτή η προς τα πάνω κίνηση του αντικειμένου.

Παρακάτω παρατίθεται η κλάση Menumanager:

```
package xj3dp2p;
import org.web3d.x3d.sai.*;

public class MenuManager {

   private static final float   BUTTONS_HEIGHT = 0.25f;
   private static final int     BUTTONS_COUNT  = 13;
   private static final int     BUTTONS_COUNT1  = 13;
   private static final int     BUTTONS_DISPLAYED  = 5;
   private static final int     BUTTONS_DISPLAYED1  = 5;
   private static final float[] VISIBLE = new float[] {1, 1, 1};
   private static final float[] NOTVISIBLE = new float[] {0, 0, 0};
   private SFVec3f[] visibility, translation;
   private X3DScene scene;
   private X3DNode[] buttons = new X3DNode[BUTTONS_COUNT];
   private X3DNode[] buttons1 = new X3DNode[BUTTONS_COUNT1];
   private int topmost_button = 0;
   private int topmost_button1 = 0;

   public MenuManager(X3DScene s, SFVec3f[] v, SFVec3f[] t) {
      this.scene = s;
      this.visibility = v;
      this.translation = t;
   }
   public void goUp() {
      int i;
      System.out.println("[MM] Aye aye! Going Up!");
      if (topmost_button > 0) {
                  this.show(topmost_button-1);
                 System.out.println("topmost:                          "+topmost_button+"hiding
"+(topmost_button+BUTTONS_DISPLAYED-1));
         this.hide(topmost_button+BUTTONS_DISPLAYED-1);
         for (i=0; i<BUTTONS_COUNT; i++)
            this.moveDown(i);
         topmost_button--;
      }
   }
     public void goUp1() {
      int i;
      System.out.println("[MM] Aye aye! Going Up!");
      if (topmost_button1 > 0) {
                  this.show1(topmost_button1-1);
                 System.out.println("topmost:                          "+topmost_button1+"hiding
"+(topmost_button1+BUTTONS_DISPLAYED1-1));
         this.hide1(topmost_button1+BUTTONS_DISPLAYED1-1);
         for (i=0; i<BUTTONS_COUNT1; i++)
            this.moveDown1(i);
         topmost_button1--;
      }
   }
```

```
public void goDown() {
    int i;
    System.out.println("[MM] Aye aye! Going Down!");
    if (topmost_button+BUTTONS_DISPLAYED < BUTTONS_COUNT) {
        this.hide(topmost_button);
        this.show(topmost_button+BUTTONS_DISPLAYED);
        for (i=0; i<BUTTONS_COUNT; i++)
            this.moveUp(i);
        topmost_button++;
    }
}
 public void goDown1() {
    int i;
    System.out.println("[MM] Aye aye! Going Down!");
    if (topmost_button1+BUTTONS_DISPLAYED1 < BUTTONS_COUNT1) {
        this.hide1(topmost_button1);
        this.show1(topmost_button1+BUTTONS_DISPLAYED1);
        for (i=0; i<BUTTONS_COUNT1; i++)
            this.moveUp1(i);
        topmost_button1++;
    }
}
public void hide(int num) {
    System.out.println("[MM] Hiding button "+num);
    visibility[num].setValue(NOTVISIBLE);
}
 public void hide1(int num) {
   int num1=num+13;
    System.out.println("[MM] Hiding button "+num1);
    visibility[num1].setValue(NOTVISIBLE);
}
public void show(int num) {
    System.out.println("[MM] Showing button "+num);
    visibility[num].setValue(VISIBLE);
}
public void show1(int num) {
    int num1=num+13;
    System.out.println("[MM] Showing button "+num1);
    visibility[num1].setValue(VISIBLE);
}
public void moveUp(int num) {
    float[] vec = {0,0,0};
    System.out.println("[MM] Moving up "+num);
    X3DNode bp = scene.getNamedNode("Build"+(num+1));
    SFVec3f transform = (SFVec3f) bp.getField("translation");
    transform.getValue(vec);
    vec[1] = vec[1]+BUTTONS_HEIGHT;
    translation[num].setValue(vec);
}
  public void moveUp1(int num) {
    int num1=num+13;
    float[] vec = {0,0,0};
    System.out.println("[MM] Moving up "+num1);
    X3DNode bp = scene.getNamedNode("Build"+(num1+1)+"P");
    SFVec3f transform = (SFVec3f) bp.getField("translation");
```

```
        transform.getValue(vec);
        vec[1] = vec[1]+BUTTONS_HEIGHT;
        translation[num1].setValue(vec);
    }
    public void moveDown(int num) {
        float[] vec = {0,0,0};
        System.out.println("[MM] Moving down "+num);
        X3DNode bp = scene.getNamedNode("Build"+(num+1));
        SFVec3f transform = (SFVec3f) bp.getField("translation");
        transform.getValue(vec);
        vec[1] = vec[1]-BUTTONS_HEIGHT;
        translation[num].setValue(vec);
    }
     public void moveDown1(int num) {
        int num1=num+13;
        float[] vec = {0,0,0};
        System.out.println("[MM] Moving down "+num1);
        X3DNode bp = scene.getNamedNode("Build"+(num1+1)+"P");
        SFVec3f transform = (SFVec3f) bp.getField("translation");
        transform.getValue(vec);
        vec[1] = vec[1]-BUTTONS_HEIGHT;
        translation[num1].setValue(vec);
    }
}
```

Πίνακας 1

3.3 Έλεγχος Συνθηκών

Για να μπορέσει ο παίκτης να δημιουργήσει τον κόσμο θα πρέπει να αρχίσει να χτίζει κτίρια. Για να επιλέξει και να χτίσει κάποιο κτίριο θα πρέπει να ικανοποιούνται οι συνθήκες που το συγκεκριμένο αντικείμενο απαιτεί. Έτσι κάθε φορά που επιλέγεται κάποιο αντικείμενο καλείται η μέθοδος popUpQuestion η οποία καλεί την κλάση housemanager.Η κλάση αυτή διαχειρίζεται τα XML αρχεία των κτιρίων (properties), και της κατάστασης του κόσμου (w_status). Εφόσον αποκτηθεί η πρόσβαση στα αρχεία, ελέγχεται αν υπάρχουν τα προαπαιτούμενα κτίρια στο αρχείο w_status πράγμα που σημαίνει ότι έχουν προηγουμένως προσαρτηθεί στον κόσμο, και συγκρίνεται η τρέχουσα οικονομική κατάσταση του χρήστη με το κόστος ανέγερσης του εκάστοτε κτιρίου. Εάν όλες οι συνθήκες ικανοποιούνται τότε μπορεί να προχωρήσει η προσάρτηση του αντικείμένου στον τρισδιάστατο κόσμο. Διαφορετικά στην οθόνη εμφανίζεται ένα μήνυμα που ενημερώνει το χρήστη ότι δε μπορεί να προχωρήσει η ανέγερση του συγκεκριμένου κτιρίου. Εικόνα2.

Συγκεκριμένα, αρχικά δίνεται το όνομα του κτιρίου που θα προσαρτηθεί με τη μέθοδο popUpQuestion στη συνέχεια και αφού κληθεί η κλάση housemanager χρησιμοποιείται η κλάση XMLHelper για να μπορέσει η εφαρμογή να διαβάσει τα XML αρχεία με τις ιδιότητες του κτιρίου και την τρέχουσα κατάσταση του κτιρίου. Εφ' όσων αποκτηθεί η πρόσβαση στα αρχεία που χρειάζονται για την υλοποίηση της συγκεκριμένης διαδικασίας και τα οποία αναφέρθηκαν παραπάνω,επιλέγονται από το αρχείο properties το

πεδίο με τα προαπαιτούμενα αντικείμενα και το πεδίο με το κόστος ανέγερσης του κτιρίου. Στη συνέχεια επιλέγονται από το αρχείο w_status το πεδίο με τα συνολικά χρήματα του χρήστη και το πεδίο που αποθηκεύονται τα κτίρια που έχουν ήδη εισαχθεί στον κόσμο και που αναλόγως με το αντικείμενο που έχει επιλεχθεί για να προσαρτηθεί στον κόσμο είναι διαφορετικό. Στη συνέχεια γίνεται σύγκριση του συνολικού ποσού που διαθέτει ο χρήστης με το κόστος ανέγερσης του αντικειμένου. Εφ' όσων τα χρήματα που έχει συγκεντρώσει ο χρήστης καλύπτουν τα έξοδα για τη δημιουργία του νέου κτίριου γίνεται ο επόμενος έλεγχος. Σε αυτό το βήμα ελέγχεται αν το προαπαιτούμενο κτίριο υπάρχει στη λίστα με τα ήδη δημιουργημένα κτίρια. Εάν ικανοποιείται και αυτή η συνθήκη η εφαρμογή προχωρά περαιτέρω διαφορετικά εμφανίζεται στην οθόνη το μήνυμα που αναφέρθηκε και παραπάνω

Στο σχήμα 2 φαίνεται ο τρόπος δράσης της εφαρμογής κατά την εισαγωγή νέου αντικειμένου στο χώρο.

3.4 Διαχείριση Ερωτήσεων

Για να προστεθεί ένα αντικείμενο στο χώρο,ο χρήστης πρέπει να επιλέξει τη σωστή απάντηση στο μαθηματικό ζήτημα που του τίθεται κάθε φορά. Σε αυτό το σημείο, ένα παράθυρο διαλόγου εμφανίζεται στην οθόνη και η πλατφόρμα γίνεται ανενεργή μέχρι ο χρήστης να επιλέξει μια απάντηση. Στην Εικόνα 2 φαίνεται η συγκεκριμένη κατάσταση.

Για την υλοποίηση της συγκεκριμένης διαδικασίας η κλάση BuldingPanelSC καλεί τις κλάσεις QuestionManager και JQuestion οι οποίες με τη σειρά τους καλούν τις υποκλάσεις Question και Answer, όπως φαίνεται στο Σχήμα 1.Η κλάση QuestionManager είναι υπεύθυνη για την κλήση του αρχείου με τις ερωτήσεις. Για να γίνει η επικοινωνία ανάμεσα στο XML αρχείο και τη Java καλείται η κλάση XMLHelper. Οι ερωτήσεις που εμφανίζονται στο χρήστη και καλούνται από το πρόγραμμα είναι αποθηκευμένες εξωτερικά σε μορφή XML. Με αυτό τον τρόπο αλλαγή ή η τροποποίηση τους μπορεί να γινεί δυναμικά χωρίς να χρειάζεται επέμβαση σε κάποιο κομμάτι του κώδικα του προγράμματος, άλλα αλλάζοντας το αρχείο των ερωτήσεων. Η κλάση QuestionManager εντοπίζει το αρχείο με τις ερωτήσεις και τις προβάλει στην οθόνη με τη σειρά. Αυτό γίνεται με τη βοήθεια ενός counter που κάθε φορά δείχνει στην επόμενη ερώτηση. Επίσης μέσα από το αρχείο των ερωτήσεων παίρνει τις απαντήσεις και γίνεται ο εντοπισμός της σωστής απάντησης ελέγχοντας την παράμετρο correct που έχει το πεδίο με της απαντήσεις.

Η κλάση JQuestion χρησιμοποιώντας το αρχείο με τις ερωτήσεις που έχει ήδη καλέσει η κλάση QuestionManager και καλώντας τις κλάσείς Answer και Question ,όπως αναφέρθηκε και πιο πάνω, ασχολείται με τον τρόπο εμφάνισης των ερωτήσεων στη οθόνη και ελέγχει την απάντηση του χρήστη.

Συγκεκριμένα δημιουργεί ένα παράθυρο μέσα στο οποίο θα εμφανιστούν οι ερωτήσεις και οι απαντήσεις καθώς επίσης και η εικόνα του αντικειμένου. Στη συνέχεια κάθε απάντηση συνδέεται με ένα radio button στο οποίο εγκαθιστάται ένας listener. Όταν ο χρήστης επιλέξει μία απάντηση, με τη χρήση της κλάσης Question και Answer, γίνεταί έλεγχος της ορθότητας της.

Εάν η απάντηση που θα δοθεί είναι σωστή τότε η ροή της εφαρμογής για την εισαγωγή ενός νέου αντικειμένου συνεχίζεται διαφορετικά σταματά σε αυτό το σημείο.

3.5 Εισαγωγή αντικειμένου στο χώρο

Όταν ο χρήστης δώσει μια απάντηση η εφαρμογή ελέγχει αν η απάντηση είναι σωστή,στην περίπτωση αυτή το κτίριο που ζητήθηκε εισάγεται στον εικονικό κόσμο. Έτσι η βασική κλάση της εφαρμογής BuildingPanelSC καλεί την WorldManager. Η κλάση αυτή είναι υπεύθυνη εκτός των άλλων και για την εισαγωγή ενός νέου αντικειμένου στον κόσμο,έτσι καλείται η μέθοδος AddNewBuilding η οποία παρατίθεται στον πίνακα που ακολουθεί.

```
public synchronized void addNewBuilding(String name)  {
      /* Load the building file */
      X3DScene buildingScene = x3dBrowser.createX3DFromURL( new String[] { name + 
".x3d"} );

      /* Get the elements we need from the scene */
      /* Element to add to the scene */
      X3DNode buildingAndSensor = buildingScene.getNamedNode(name);
      /* The building structure */
      X3DNode building = buildingScene.getNamedNode("building");

      /* Update the maps  */
      this.nodeToName.put(building,name);
      VMID myID=new VMID();
      log.debug("New '"+name+"' with id '"+myID+"'");
      this.nodeToNodeID.put(building,myID.toString());
      this.nodeIDToNode.put(myID.toString(),building);

      mainScene.addRootNode(buildingAndSensor);

      mainScene.addRoute(sensor, "translation_changed",building, "set_translation" );

      /* get world  meta*/
      mainScene = (X3DScene) x3dBrowser.getExecutionContext();
      X3DNode meta_world = mainScene.getNamedNode("meta");
      MFString meta_w_value=(MFString)meta_world.getField("value");
      String xml_world =meta_w_value.get1Value(0);

      /* Update World */
      updatestatus(xml_world,name);
   }
```

Πίνακας 2

Αρχικά εντοπίζονται τα στοιχεία που χρειάζονται μέσα από τον κόσμο του X3D και τον browser τα στοιχεία αυτά είναι το όνομα του αντικειμένου και το πεδίο building που του αντιστοιχεί. Έπειτα κάθε αντικείμενο που δημιουργείται παίρνει μια μοναδική ταυτότητα, και μαζί με το όνομα του αποθηκεύεται σε ένα hashmap ένα είδος πίνακα (nodeToNodeID) που

ενημερώνεται με την εισαγωγή κάθε νέου αντικειμένου. Στη συνέχεια στην σκηνή προστίθεται το νέο αντικείμενο. Αμέσως μετά εντοπίζεται το αρχείο στο οποίο αποθηκεύεται η τρέχουσα κατάσταση του παιχνιδιού χρησιμοποιώντας το πεδίο metadata του X3D, έτσι σειρά έχει η κλήση της μεθόδου updatestatus η οποία ενημερώνει το αρχείο στο οποίο αποθηκεύεται η εκάστοτε κατάσταση του κόσμου. Η updatestatus χρησιμοποιεί τηη κλάση XMLHelper για να διαβάσει το αρχείο με τις ιδιότητες του αντικειμένου που προστέθηκε(properties) και το αρχείο της κατάστασης του κόσμου(w-status).Έτσι αφού εντοπίσουν και διαβάσουν τα δύο αρχεία,το αρχείο που αποθηκεύονται οι παράμετροι του παιχνιδιού,ενημερώνεται, σύμφωνα με τις ιδιότητες που του προσδίδει το εκάστοτε κτίριο που εισάγεται στον κόσμο και αποθηκεύεται με τις αλλαγές που έχει υποστεί έτσι ώστε να μπορούν να είναι διαθέσιμες ανα πάσα στιγμή οι τιμές των παραμέτρων του κόσμου. Αμέσως μετά και αφού προστεθούν οι ιδιότητες που αποδίδει το αντικείμενο στο χώρο,γίνεται έλεγχος για την αναβάθμιση των levels. Γίνεται δηλαδή έλεγχος για το αν ικανοποιούνται οι απαιτήσεις για την αλλαγή επιπέδου. Αυτό γίνεται εφικτό διαβάζοντας το αρχείο w_status. Στο αρχείο w_status εκτός από τις τιμές των παραμέτρων της εφαρμογής βρίσκονται και οι τιμές των παραμέτρων που καθορίζουν το επίπεδα της εφαρμογής. Στο σημείο αυτό λοιπόν καλείται η μέθοδος checklevel η οποία είναι υπεύθυνη για τον έλεγχο των επιπέδων της εφαρμογής και για την μεταβίβαση από το ένα επίπεδο στο άλλο όταν είναι εφικτό. Η μέθοδος αυτή αφού έχει ήδη διαβαστεί το αρχείο w_status συγκρίνει τις τιμές των παραμέτρων της εφαρμογής με τα απαιτούμενα για την αλλαγή του επιπέδου. Εάν ικανοποιούνται οι απαιτήσεις του εκάστοτε επιπέδου ο χρήστης ενημερώνεται για την αλλαγή του επιπέδου με ένα μήνυμα που εμφανίζεται στην οθόνη και στη συνέχεια ενημερώνεται από το σύστημα το αρχείο w_status καθώς αποθηκεύεται μαζί με τα νέα δεδομένα δηλαδή την αλλαγή το επιπέδου και την αλλαγή των τιμών των παραμέτρων που προκάλεσε η εισαγωγή του νέου αντικειμένου στο χώρο.

Μόλις εισαχθεί ένα αντικείμενο στο χώρο δίνεται η δυνατότητα στο χρήστη να το μετακινήσει. Για την υλοποίηση αυτής της λειτουργίας κατά την εισαγωγή του αντικειμένου στον κόσμο ενεργοποιείται ένας plane sensor που είναι υπεύθυνος για την μετακίνηση του αντικειμένου στο χώρο. Ένας plane sensor έχει τη δυνατότητα να μετακινεί ένα αντικείμενο κατά μήκος των δύο εκ των τριών διαστάσεών.

Η κλάση WorldManager παρατίθεται στο σύνολό της στο παράρτημα του εγγράφου.

3.6 Λειτουργίες Save/Load Build

Στα πλαίσια της υλοποίησης του σεναρίου του παιχνιδιού υλοποιούνται οι λειτουργίες Load Save και Build οι οποίες διαδραματίζουν σημαντικό ρόλο στην εξέλιξη του παιχνιδιού. Στη συνέχεια εξηγείται αναλυτικά ο τρόπος λειτουργίας τους.

3.6.1 Save

Όταν ο χρήστης επιλέξει να αποθηκεύσει την τρέχουσα κατάσταση του παιχνιδιού η κλάση BuildingPanelSC καλεί την κλάση WorldManager και συγκεκριμένα τη μέθοδο SaveWorldStatus η οποία καλεί την κλάση SaveWorld πίνακας 3

```
public void saveWorldStatus() {
    SaveWorld status=new SaveWorld(conf,nodeToNodeID,nodeToName);
    status.save();
}
```

Πίνακας 3

Η μέθοδος αυτή αρχικοποιεί την κλάση SaveWorld στέλνοντας της τις απαραίτητες πληροφορίες που χρειάζεται για να λειτουργήσει. Έτσι στέλνει τις μεταβλητές conf, nodeToNodeID,nodeToName. Η μεταβλητή conf έχει τη διαδρομή στην οποία βρίσκεται το αρχείο configuration. Το οποίο χρησιμοποιείται από την κλάση configuration (Παράρτημα) για να ενημερώσει την πλατφόρμα σχετικά με τις διαδρομές των αρχείων στα οποία θα αποθηκευτεί και τα φορτωθεί το περιβάλλον αντίστοιχα. Η μεταβλητή nodeToNodeID είναι ένα hashmap ,που κρατάει τα Nodes των κτιρίων που έχουν δημιουργηθεί και τον κωδικό που αντιστοιχεί στο καθένα. Η μεταβλητή nodeToName είναι ένα hashmap με τα ονόματα των κτιρίων που έχουν δημιουργηθεί καθώς και τα Nodes με τα ίδια τα κτίρια .

Στη συνέχεια καλείται η μέθοδος save της κλάσης SaveWorld (Πίνακας4). Η μέθοδος αυτή, αρχικά διαβάζει το αρχείο background.x3d στην xml μορφή του και το αποθηκεύει στη μεταβλητή original. Αμέσως μετά και μέχρι να σαρωθεί ολόκληρος ο hashmap ,καλείται η μέθοδος getCachedBuilding. Η μέθοδος αυτή αρχικά ελέγχει αν υπάρχει ήδη ο Node που μας ενδιαφέρει. Ο έλεγχος αυτός γίνεται σαρώνοντας τη μεταβλητή nodeNameToNodeXML,ένα βοηθητικό hashmap που δημιουργείται και αρχικά είναι κενό. Αν κατά τον έλεγχο εντοπιστεί, επιστρέφεται ένα αντίγραφό του αλλιώς φορτώνεται από την αρχή. Σε αυτή την περίπτωση ο Node εισάγεταί στο Document που δημιουργήσαμε νωρίτερα και ονομάζεται original και ενημερώνεται η μεταβλητή nodeNameToNodeXML με τον ίδιο τον Node και το όνομά του.

Στη συνέχεια γίνεται η απενεργοποίηση τού PlaneSensor αφαιρώντας το node του sensor από το αντικείμενο. Αυτή η αλλαγή της συνθήκης του sensor γίνεται μέσα στο XML αρχείο, τελικά επιστρέφεται ένα αντίγραφό του Node με τις αλλαγές που έχει υποστεί ώστε αργότερα κατά την αποθήκευση του αρχείου να ενημερωθεί ο κόσμος για τις αλλαγές.

Στο επόμενο βήμα καλείται η μέθοδος renameDefUse.Ο ρόλος της μεθόδου είναι να προσθέσει σε κάθε πεδίο DEF και USE την μοναδική ταυτότητα που έχει αποδοθεί στο κάθε αντικείμενο-κτίριο. Το Xj3D δεν μπορεί να υποστηρίξει αρχεία τα οποία αποτελούνται από nodes με ίδιο όνομα δηλαδή ίδιο DEF. Κάθε X3D αρχείο αποτελείται από πολλά Nodes τύπου transform.

Ένα transform node χρησιμοποιείται όποτε θέλουμε να δημιουργήσουμε ένα νέο γεωμετρικό σχήμα. Το τελικό αντικείμενο αποτελείται από την χρήση πολλών γεωμετρικών σχημάτων που έχουν τοποθετηθεί και συνδεθεί σωστά ώστε να αποδώσουν την απεικόνιση ολόκληρου του αντικειμένου.

Στη συνέχεια το αρχείο original ενημερώνεται για τις νέες θέσεις των κτιρίων στον κόσμο. Για να γίνει αυτό αρχικά επιλέγεται το πεδίο translation του κάθε κτιρίου. Στη συνέχεια και αφού εντοπιστεί και αποθηκευτεί το πεδίο translation με τις αρχικές τιμές γίνεται η ενημέρωση του με τις τρέχουσες τιμές που του αντιστοιχούν και που έχουν προηγουμένως αποθηκευτεί διαβάζοντάς το X3D αρχείο στην τρέχουσα κατάστασή του. Τελικά ο νέος τροποποιημένος Node εισάγεται στο αρχείο original το οποίο τελικά αποθηκεύεται στο αρχείο conf.getSaveFile(). Σε αυτό το σημείο καλείται η κλάση configuration η οποία μας επιστρέφει το αρχείο στο οποίο θα αποθηκευτεί ο κόσμος και το οποίο είναι το savegame.x3d Ακολουθεί η κλάση SaveWorld

```
public class SaveWorld {
   protected Configuration conf;
   protected Map<X3DNode,String> nodeToNodeID;
   protected Map<X3DNode,String> nodeToName;
   protected Map<String,Node> nodeNameToNodeXML;
   protected Logger log;

   private static Logger log1=Logger.getLogger(SaveWorld.class);

   /** Creates a new instance of SaveWorld */
   public SaveWorld(Map nodeToNodeID,Map nodeToName) {
      this(new Configuration(),nodeToNodeID,nodeToName);
   }

   public SaveWorld(Configuration conf,Map nodeToNodeID,Map nodeToName) {
      this.conf=conf;
      this.nodeToNodeID=nodeToNodeID;
      this.nodeToName=nodeToName;
      this.nodeNameToNodeXML=new HashMap(20);
      this.log=Logger.getLogger(this.getClass());
   }

   public void save() {

      Document original=loadXML("models/"+conf.getBaseFile(),false);

      for(X3DNode curNode:nodeToName.keySet()) {
         /* Load the file */
         Node curXML=getCachedBuilding(nodeToName.get(curNode),original);

         /* Rename the DEF and USE */
         renameDefUse(curXML,nodeToNodeID.get(curNode));

         /* Get the new position */
         SFVec3f transVector = (SFVec3f)curNode.getField("translation");
```

```
            /* Put it in the building we're saving
             *        |-> TRANSFORM (item 3) <--curXML)
             *           |-> Text (item 0)
             *           |-> TRANSFORM (**item 1**) DEF,rotation,translation
             */

Node trans=curXML.getChildNodes().item(1).getAttributes().getNamedItem("translation");
            float[] vector={0,0,0};
            transVector.getValue(vector);
            trans.setNodeValue(vector[0]+" "+vector[1]+" "+vector[2]);

             /* Add to the main document
             * Document
             * |-> X3D (**item 0**)
             *     |-> Text (item 0)
             *     |-> SCENE (**item 1**)
             */
            original.getChildNodes().item(0).getChildNodes().item(1).appendChild(curXML);
        }

        original.normalizeDocument();

        /* Save it to a file */
        try {
            TransformerFactory transfac = TransformerFactory.newInstance();
            transfac.setAttribute("indent-number", new Integer(4));
            Transformer trans = transfac.newTransformer();
            trans.setOutputProperty(OutputKeys.INDENT, "yes");


            String Path=null;
            File currentDir=new File(".");
            try{
               Path=currentDir.getCanonicalPath();
               System.out.println(currentDir.getCanonicalPath());
            }
            catch(IOException e){
               log.error("IOExceprion"+ e);
            }

            String sv="models/"+conf.getSaveFile();
            File output=new File(Path+"/Xj3DP2P/"+sv);

            /* Due to a bug in some releases of java, we need a writer here.
             * So it makes this line really long
             */
            StreamResult result = new StreamResult(
                  new OutputStreamWriter(
                  new FileOutputStream(output),"utf-8"));
            DOMSource source = new DOMSource(original);
            trans.transform(source,result);
        }catch(Exception e) {
            log.error("Couldn't save file. "+e);
        }
```

```
    }

    protected Node getCachedBuilding (String name,Document doc) {

        /* Do we have it cached? */
        Node cachedBuilding=nodeNameToNodeXML.get(name);
        if (cachedBuilding!=null) {
            return cachedBuilding.cloneNode(true);
        }

        /* If we're here, we don't have the node */
        Document buildingDocument=loadXML("models/"+name+".x3d",false);

        /* Document
         * |-> X3D (**item 0**)
         *     |-> Text (item 0)
         *     |-> SCENE (**item 1**)
         *         |-> Text (item 0)
         *         |-> WORLDINFO (item 1)
         *         |-> Text (item 2)
         *         |-> TRANSFORM (**item 3**)
         */

        Node
buildingNode=buildingDocument.getChildNodes().item(0).getChildNodes().item(1).getChild
Nodes().item(3);

        /* Add to the cach? */
        buildingNode=doc.importNode(buildingNode,true);
        nodeNameToNodeXML.put(name,buildingNode);

        /* Remove the PlaneSensor added to move the object, we leave the rest

        /* Remove the PlaneSensor added to move the object, we leave the rest
         * Starting from the last item:
         *         |-> TRANSFORM (item 3)
         *             |-> Text (item 0)
         *             |-> TRANSFORM (item 1)
         *             |-> Text (item 2)
         *             |-> PLANESENSOR (**item 3**)
         */
        buildingNode.removeChild(buildingNode.getChildNodes().item(3));

        return buildingNode.cloneNode(true);
    }

    private void renameDefUse(Node curNode, String nodeID) {
        if (curNode==null) return;

        if (curNode.getAttributes()!=null) {

            Node tmp=curNode.getAttributes().getNamedItem("DEF");
            if (tmp!=null) {
                tmp.setNodeValue(tmp.getNodeValue()+"-"+nodeID);
            }
```

```
            tmp=curNode.getAttributes().getNamedItem("USE");
            if(tmp!=null) {
               tmp.setNodeValue(tmp.getNodeValue()+"-"+nodeID);
            }
         }

         for(int i=0;i<curNode.getChildNodes().getLength();i++) {
            renameDefUse(curNode.getChildNodes().item(i),nodeID);
         }
      }

      public static Document loadXML(String fileName,boolean check) {
         String Path=null;
         File currentDir=new File(".");
         try{
             Path=currentDir.getCanonicalPath();
             System.out.println(currentDir.getCanonicalPath());
         }
         catch(IOException e){}

         Document doc=null;
         File file=new File(Path+"/Xj3DP2P/"+fileName);
         DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();


         try {
            /* Try to load an parse the document */
            DocumentBuilder builder = factory.newDocumentBuilder();
            doc = builder.parse(file);
         }  catch (SAXException sxe) {
            /* Parsing exception */
            Exception  x = sxe;
            if (sxe.getException() != null) x = sxe.getException();
            log1.error("Could not parse the XML questions file: "+x);

         } catch (ParserConfigurationException pce) {
            log1.error("Parser with specified options can't be built: "+pce);
         } catch (IOException ioe) {
            log1.error("IO Error while parsing: "+ioe);
         }

         return doc;

   }
}
```

Πίνακας 4

3.6.2 Load

Στην περίπτωση που ο χρήστης επιλέξει να φορτώσει τον αποθηκευμένο κόσμο καλείται η μέθοδος loadWorldStatus. Αρχικά εντοπίζει την τρέχουσα διαδρομή στην οποία εκτελείται η εφαρμογή. Στη συνέχεια με τη βοήθεια της

κλάσης configuration ελέγχει σε ποιό αρχείο βρισκόμαστε δηλαδή ελέγχει ποιό αρχείο χρησιμοποιήθηκε την τελευταία φορά που καλέστηκε, και ποιό αρχείο τελικά θα φορτωθεί. Τελικά επιστρέφεται η διαδρομή του αρχείου που έχει αποθηκευτεί η τελευταία κατάσταση του κόσμου και το οποίο στην περίπτωση μας είναι το savegame.x3d. Το αρχείο που επιστράφηκε χρησιμοποιείται τελικά για να δημιουργηθεί μια νέα X3D σκηνή. Τελικά η νέα αυτή σκηνή αντικαθιστά την προηγούμενη και έτσι η αποθηκευμένη σκηνή γίνεται τώρα η τρέχουσα, πάνω στην οποία ο χρήστης μπορεί να συνεχίσει τη δημιουργία του φανταστικού του κόσμου.

```
public void loadWorldStatus() {
    /* Get the current Path to append to the load world URL */
    File currentDir=new File(".");
    String currentPath=".";

    try {
       currentPath=currentDir.getCanonicalPath();
    }catch(Exception e) {
       log.error("Couldnt get current path");
    }

    /* The next time we will save the world using this saved world */
    if(!conf.getBaseFile().equals(conf.getSaveFile())){
       System.out.println("base !== save");
       conf.setBaseFile(conf.getSaveFile());
    }
    /* Load the saved world scene, and replace the current world */

    X3DScene loadedWorld=x3dBrowser.createX3DFromURL(
          new String[] {conf.getSaveFile()});
    x3dBrowser.replaceWorld(loadedWorld);

  }
```

Πίνακας 5

3.6.3 Build

Όταν ο χρήστης εισάγει ένα αντικείμενο στον τρισδιάστατο κόσμο,έχει τη δυνατότητα να το μετακινήσει στο χώρο. Για να μπορέσει να αποδοθεί σωστά η λειτουργία του "χτισίματος" του κτιρίου,έπρεπε η εφαρμογή να απενεργοποιεί τη δυνατότητα μετακίνησης του. Έτσι με την υλοποίηση του Build δίνεται η δυνατότητα στο χρήστη αφού μετακινήσει το αντικείμενο να το χτίσει απενεργοποιώντας ουσιαστικά τον PlaneSensor. Αυτό γίνεται κατά την αποθήκευση του αρχείου με τον τρόπο που αναφέρθηκε παραπάνω, θα έπρεπε λοιπόν μια απλή ανανέωση της σκηνής του Xj3D να είναι ικανή να αποδώσει τη νέα κατάσταση. Το Xj3D όμως δεν υποστηρίζει την ανανέωση της X3D σκηνής δυναμικά, στην περίπτωση που έχει τροποποιηθεί το ίδιο το αρχείο του X3D. Έτσι όποτε χρειάζεται να γίνει ανανέωσή στον κόσμο και έχουν προηγηθεί αλλαγές στον κώδικα του X3D πρέπει να αποθηκευτεί η

σκηνή, με τις αλλαγές που έχει υποστεί και να φορτωθεί ξανά από την αρχή έτσι ώστε να τεθούν σε λειτουργία οι αλλαγές. Όταν λοιπόν ο χρήστης επιλέξει να χτίσει ένα κτίριο από το κουμπί Build, καλείται η μέθοδος saveWorldStatus() και αμέσως μετά καλείται η loadWorldStatus() όπως φαίνεται στον πίνακα 6. Η μέθοδος saveWorldStatus έκτος από την αποθήκευση του αρχείου απενεργοποιεί την λειτουργία της μετακίνησης του αντικείμενου. Αμέσως μετά η loadWorldStatus επαναφέρει τον κόσμο στην προηγούμενη κατάστασή του χωρίς να μπορούν να μετακινηθούν τα αντικείμενα. Ουσιαστικά σε αυτό το σημείο της εφαρμογής είναι σαν να επιλεχθούν το save και αμέσως μετά το load button.

```
else if (evt.getSource() == buildClicked) {
    log.debug("Deactivate sensors.");
    myWorldManager.saveWorldStatus();
    myWorldManager.loadWorldStatus();
```

Πίνακας 6

3.7 Λειτουργίες Properties/Status

Οι λειτουργίες properties και status λειτουργούν με παρόμοιο τρόπο γι' αυτό το λόγο παρουσιάζονται μαζί. Ουσιαστικά αφορούν στον τρόπο ανάγνωσης και εμφάνισης των πεδίων των XML αρχείων properties και w_status στην οθόνη.

3.7.1 Properties

Στην εφαρμογή έχει υλοποιηθεί η δυνατότητα ελέγχου των ιδιοτήτων του κάθε αντικειμένου. Έτσι ο χρήστης μπορεί οποιαδήποτε στιγμή να ελέγξει τι κερδίζει από την εισαγωγή του κάθε κτιρίου καθώς επίσης δίνεται η δυνατότητα να ελέγξει τα προαπαιτούμενα κτίρια που χρειάζονται για την ανέγερση του εκάστοτε κτιρίου. Στο μενού των κτιρίων δίπλα σε κάθε κτίριο βρίσκεται το κουμπί properties.Όταν ο χρήστης επιλέξει τη συγκεκριμένη δυνατότητα δηλαδή να δει τις ιδιότητες του αντικειμένου καλείται από την κλάση BuildingPanelSC η κλάση properties.Η κλάση αυτή καλεί την μέθοδο houseproperties της κλάσης housemanager η οποία χρησιμοποιεί την XMLHelper για να διαβάσει το αρχείο με τις ιδιότητες του εκάστοτε αντικειμένου. Εάν αυτό συμβεί επιτυχώς δημιουργείται ένα παράθυρο στο οποίο αριστερά παρουσιάζεται μία φωτογραφία του αντικειμένου και δεξιά μια λίστα με τις ιδιότητες του. Συγκεκριμένα αρχικά καλείται η μέθοδος houseproperties η οποία αφού διαβάσει το αρχείο με τις ιδιότητες του αντικειμένου επιστρέφει στο σημείο που καλέστηκε ένα nodelist με τις ιδιότητες. Στη συνέχεια δημιουργείται ένα παράθυρο διαλόγου στο οποίο εμφανίζονται οι εικόνες του αντικειμένου χρησιμοποιώντας έτσι ένα ακόμη τμήμα από το οποίο αποτελείται το αντικείμενο όπως φαίνεται και στο σχήμα

1. Στη συνέχεια με τη χρήση της for σαρώνονται όλα τα στοιχεία και προβάλλονται στο παράθυρο που δημιουργήθηκε. Στην εικόνα 4 φαίνεται το παράθυρο που εμφανίζεται στην οθόνη στην περίπτωση εκτέλεσης αυτής της λειτουργίας.

3.7.2 Status

Η επιλογή Status δίνει τη δυνατότητα στο χρήστη να ελέγξει την κατάσταση των παραμέτρων του συστήματος οποιαδήποτε στιγμή, να ελέγξει σε ποιό επίπεδο βρίσκεται και να δει πόσα χρήματα έχει μαζέψει στην περίπτωση που θέλει να εισάγει κάποιο κτίριο και τα χρήματα πού έχει δεν επαρκούνε. Επιλέγοντας την προβολή της τρέχουσας κατάστασης του συστήματος δηλαδή του status button, καλείται η κλάση statusmanager. Ή κλάση αυτή εντοπίζει μέσα από το X3D το πεδίο name,μέσα από το node meta.Το node αυτό έχει σαν πληροφορία τη διαδρομή που βρίσκεται το αρχείου w_status. Στη συνέχεια με τη βοήθεια της κλάσης XMLHelper διαβάζεται το αρχείο που εντοπίστηκε στο προηγούμενο βήμα. Από αυτό επιλέγονται τα πεδία με τις παραμέτρους του συστήματος και το πεδίο που κρατάει τα επίπεδα (levels).Στη συνέχεια δημιουργείται το παράθυρο που θα προβάλει τα επίπεδα αυτά. Έτσι με τη χρήση της εντολής for προβάλλονται όλα τα στοιχεία που έχουν επιλεχθεί. Τελικά εμφανίζεται στην οθόνη ένα παράθυρο που περιέχει όλες τις παραμέτρους της εφαρμογής και την τρέχουσα τιμή τους καθώς και ένα κουμπί που όταν επιλεγεί κλείνει το παράθυρο διαλόγου που δημιουργήθηκε. Η λειτουργία αυτή είναι σημαντική για τη λειτουργία του παιχνιδιού γιατί παρέχει πληροφορίες στο χρήστη για την κατάσταση του κόσμου οποιαδήποτε στιγμή κριθεί αναγκαίο από το χρήστη με δυναμικό τρόπο. Στην εικόνα 5 φαίνεται η συγκεκριμένη κατάσταση.

3.8 Λειτουργιά εισαγωγής νέου αντικειμένου

Η λειτουργία εισαγωγής νέου αντικειμένου επιτρέπει στο χρήστη να εισάγει ένα αντικείμενο στο χώρο το οποίο δεν υπάρχει στο μενού με τα κτίρια. Έτσι ο χρήστης μπορεί να εισάγει ένα αντικείμενο και να δημιουργήσει τον κόσμο του όπως εκείνος το επιθυμεί. Η λειτουργία ατή ενεργοποιείται μόλις ο χρήστης επιλέξει το κουμπί AddNewBuilding όπως φαίνεται στη εικόνα 1. Από τη στιγμή που θα επιλεχθεί η συγκεκριμένη λειτουργία δίνεται η εντολή από τη βασική κλάση της εφαρμογής να δημιουργηθεί ένα παράθυρο διαλόγου χρησιμοποιώντας τη μέθοδο openfile(). Η μέθοδος αυτή δημιουργεί το παράθυρο μέσα από το οποίο θα επιλεχθεί το αρχείο που θα εισαχθεί στον κόσμο. Αφού επιλεχθεί το αρχείο επιστρέφεται εκεί που αρχικά καλέστηκε. Στη συνέχεια με τη χρήση της απόλυτης διαδρομής του αρχείου απομονώνεται το όνομα το αρχείου και άρα του αντικειμένου που θα εισάγουμε στον κόσμο,και καλείται η μέθοδος popUpQuestion με όρισμα το όνομα του αντικειμένου που θα εισαχθεί. Από αυτό το σημείο και μετά η εισαγωγή του

αντικειμένου γίνεται όπως και στις άλλες περιπτώσεις που εξηγήθηκαν παραπάνω. Αρχικά γίνεται έλεγχος των προαπαιτούμενων του κτιρίου και του κόστους ανέγερσης του, με τη χρήση της κλάσης housemanager στη συνέχεια γίνεται η διαχείριση των ερωτήσεων και εμφανίζεται η ερώτηση που πρέπει να απαντηθεί από τον χρήστη, με τη βοήθεια των κλάσεων JQuestion και QuestionManager, και τελικά γίνεται η εισαγωγή του αντικειμένου στο χώρο με τη κλάση WorldManager.

```
File file_to_open=openfile();
        String FullName=file_to_open.getName();
        String name=FullName.substring(0,FullName.indexOf("."));
        popUpQuestion(name);
        log.debug("Open File Clicked");
```

```
public File openfile(){

      JFileChooser chooser = new JFileChooser();
      FileNameExtensionFilter filter = new FileNameExtensionFilter("X3D
file","x3d");
      chooser.addChoosableFileFilter(filter);

      int returnVal = chooser.showOpenDialog(chooser);
      if(returnVal == JFileChooser.APPROVE_OPTION) {
          System.out.println("You chose to open this file: " +
          chooser.getSelectedFile().getName());
          System.out.println("You chose to open this file: " +
          chooser.getSelectedFile().getAbsolutePath() );
       }
   return chooser.getSelectedFile();

}
```

Πίνακας 7

3.9 Catastrophic Events

Στα πλαίσια της υλοποίησης της πτυχιακής εργασίας δημιουργήθηκαν καταστροφικές καταστάσεις οι οποίες προσομοιώνουν καταστροφές από την καθημερινότητα. Στον X3D κόσμο έχει ενσωματωθεί ένας time sensor ο οποίος δημιουργεί γεγονότα καθώς περνάεί ο χρόνος.

```
<TimeSensor DEF="CS" cycleInterval="1000" enabled="true" loop="true"/>
```

Η εντολή αυτή είναι η εντολή ενεργοποίησης του time sensor μέσα στον κώδικα τύπου XML του X3D και ορίζει μετά από πόσο χρόνο θα δημιουργηθεί το event που θα διαχειριστεί από τη Java (Ο χρόνος μετριέται σε milliseconds).Μόλις υπάρξει υπερχείλιση του timer δημιουργείται ένα event και καλείται από την κλάση BuildingPanelSC η κλάση

catastrophymanager.Αρχικά δημιουργείται μια μεταβλητή random η οποία με τυχαίο τρόπο επιστρέφει έναν αριθμό από το ένα ως το τρία. Κάθε αριθμός αντιστοιχεί σε μία καταστροφή,το ένα στο σεισμό το δύο στην φωτιά και το τρία στη πλημμύρα. Στη συνέχεια και ανάλογα με την καταστροφή που τυχαία πρόκειται να προσομοιωθεί καλείται η αντίστοιχη μέθοδος.

```
Random generator = new Random();
        int t=generator.nextInt(3)+1;
        if(t==1){
           earthquake("earthquake");
           JOptionPane.showMessageDialog(null, "An Earhtquake Just Occured As A
Result "+total_building_loss+
                   " Buildings And "+total_loss+" Money Have Been Lost","
INFORMATION ", JOptionPane.WARNING_MESSAGE);
        }else if (t==2){
           fire("fire");
           JOptionPane.showMessageDialog(null, "A Fire Took Place As A Result
"+total_building_loss+
                   " Buildings And "+total_loss+" Money Have Been Lost","
INFORMATION ", JOptionPane.WARNING_MESSAGE);
        }else if (t==3){
           flood("flood");
           JOptionPane.showMessageDialog(null, "A Flood Happpened As A Result
"+total_building_loss+
                   " Buildings And "+total_loss+" Money Have Been Lost","
INFORMATION ", JOptionPane.WARNING_MESSAGE);
        }
```

Πίνακας 8

Αρχικά εντοπίζεται και διαβάζεται με τη χρήση της κλάσης XMLHelper το XML αρχείο της εκάστοτε καταστροφής. Το αρχείο αυτό περιέχει πληροφορίες όπως το όνομα της καταστροφής , το μεγαλύτερο μέγεθός με το οποίο μπορεί να εκδηλωθεί,το μεγαλύτερο ποσοστό κτιρίων που μπορεί να επηρεάσει και έναν μετρητή που αποθηκεύει τον αριθμό που έχει εκδηλωθεί η εκάστοτε καταστροφή. Μόλις διαβαστεί το αρχείο επιλέγεται το πεδίο με την ένταση του φαινομένου και με τη χρήση της συνάρτησης random υπολογίζεται μια τυχαία τιμή ανάμεσα στον αριθμό ένα και την τιμή που πάρθηκε από το XML αρχείο. Στη συνέχεια επιλέγεται το πεδίο με το ποσοστό της καταστροφής και με τη χρήση της random υπολογίζεται ένα τυχαίο ποσοστό ανάμεσα στο 10 και την τιμή του ποσοστού που πάρθηκε από το αρχείο. Έπειτα αυξάνεται ο μετρητής που μετράει τις φορές που έχει εκδηλωθεί η καταστροφή και η αλλαγή αποθηκεύεται στο αρχείο XML χρησιμοποιώντας τη μέθοδο savefile.

Στη συνέχεια εντοπίζεται και φορτώνεται το αρχείο savegame.x3d που έχει αποθηκευμένα όλα τα κτίρια που έχουν εισαχθεί στον κόσμο. Δημιουργείται ένας Vector που θα χρησιμοποιηθεί αργότερα και μέσα από το αρχείο savegame εντοπίζονταί τα nodes των κτιρίων και σαρώνονται όλα μέχρι η λίστα με τα αποθηκευμένα nodes να τελειώσει.

Για κάθε Node κτιρίου που εντοπίζεται φορτώνεται το αρχείο catastrophy που του αντιστοιχεί. Το αρχείο αυτό περιέχει πληροφορίες για το είδος των καταστροφών που επηρεάζουν το κάθε αντικείμενο,το μέγεθος του φαινομένου πάνω από το οποίο το αντικείμενο καταστρέφεται, και το κόστος συντήρησης του κάθε αντικειμένου στην περίπτωση που το φαινόμενο είναι μικρότερο από το όριο καταστροφής οπότε και μπορεί να επισκευαστεί η ζημιά που έχει προκληθεί από την εκάστοτε καταστροφή.

Μόλις διαβαστεί αυτό το αρχείο ελέγχεται αν το συγκεκριμένο κτίριο επηρεάζεται από τη συγκεκριμένη καταστροφή. Στην περίπτωσή αυτή η θέση του αντικειμένου στη λίστα με τα υπόλοιπα αντικείμενα αποθηκεύεται στο Vector που είχε δημιουργηθεί νωρίτερα μαζί με το κόστος συντήρησης του κτιρίου και το όριο καταστροφής που του αντιστοιχεί και πάνω από το οποίο καταστρέφεται. Σε διαφορετική περίπτωση ο συγκεκριμένος Node προσπερνάτε. Το αποτέλεσμα αυτής της διαδικασίας είναι η δημιουργία ενός Vector που έχει τις θέσεις το κόστος και το όριο κατεδάφισης όλων των κτιρίων που επηρεάζονται από τη συγκεκριμένη καταστροφή.

Στη συνέχεια υπολογίζεταί το μέγεθος του Vector για να ξέρουμε το σύνολο των αντικείμενων που επηρεάζονται. Με τη χρήση του ποσοστού που έχει υπολογιστεί νωρίτερα από την κλήση του αρχείου της καταστροφής και τη χρήση του μεγέθους του πίνακα, υπολογίζεται ο αριθμός των αντικειμένων που θα δεχθούν τις συνέπειες του καταστροφικού γεγονότος. Έπειτα με τη χρήση της συνάρτησης random επιλέγονται τυχαία μέσα από το vector οι θέσεις των αντικειμένων που θα δεχθούν τις συνέπειες της καταστροφής.

Στη συνέχεια ελέγχεται αν το μέγεθος της καταστροφής είναι μεγαλύτερο από το όριο αντοχής του κάθε αντικειμένου. Αν ισχύει η συνθήκη τότε το συγκεκριμένο αντικείμενο διαγράφεται και το αρχείο savegame αποθηκεύεται για να κρατήσει τις αλλαγές. Επίσης αυξάνεται ένας μετρητής που υπολογίζει το συνολικό αριθμό των κτιρίων που καταστράφηκαν.

Στην περίπτωση που η προηγούμενη συνθήκη δεν αληθεύει και άρα το μέγεθος της καταστροφής δεν είναι ικανό να καταστρέψει ένα κτίριο,καλείται η μέθοδος removecost η οποία αφαιρεί το κόστος από τους συνολικούς πόντους που έχει μαζέψει ο παίκτης. Αυτό επιτυγχάνεται καθώς η συγκεκριμένη μέθοδος με τη χρήση των meta data του X3D αρχείου εντοπίζει το αρχείο status του κόσμου και αφαιρεί το κόστος επισκευή του εκάστοτε κτιρίου από τους συνολικούς πόντους. Στη συνέχεια αυξάνεται ο μετρητής που υπολογίζει το συνολικό κόστος ανέγερσης για κάθε κτίριο που είχε επιβαρυνθεί από την καταστροφή. Τελικά ο χρήστης ενημερώνεται με ένα dialog window πόσα κτίρια επηρεάστηκαν από την καταστροφή και πόσα χρήματα έχασε για την επισκευή των κτιρίων που δε καταστράφηκαν ολοσχερώς.

Ενδεικτικά παρατίθεται η μέθοδος earthquake με ανάλογο τρόπο λειτουργούν και οι μέθοδοι flood , fire.

```
public void earthquake(String name){
        String xml_url="models/"+name+".xml";
```

```
        Document tmp = XMLHelper.loadXML("models/"+name+".xml");

        /** diarkia**/
        int
duration=Integer.valueOf(tmp.getChildNodes().item(0).getChildNodes().item(1).getC
hildNodes().item(5).getChildNodes().item(0).getNodeValue());
        Random generator=new Random();
        /* megethos katastrofis*/
        int dur=generator.nextInt(duration)+1;

        /*** pososto katastrofis ****/
        int
percent=Integer.valueOf(tmp.getChildNodes().item(0).getChildNodes().item(1).getCh
ildNodes().item(7).getChildNodes().item(0).getNodeValue());
        int perc=generator.nextInt(percent)+10;

        /**set the counter */
        int
counter=Integer.valueOf(tmp.getChildNodes().item(0).getChildNodes().item(1).getC
hildNodes().item(9).getChildNodes().item(0).getNodeValue());

tmp.getChildNodes().item(0).getChildNodes().item(1).getChildNodes().item(9).getCh
ildNodes().item(0).setNodeValue(String.valueOf(counter+1));

        savefile(xml_url,tmp);

        /** load savegame**/
        Document original=loadXML("models/"+"savegame.x3d",false);

        /* catch buildings- transform */
        /* check for transform nodes */
        int x=271;
        Vector v=new Vector();

        while
(original.getChildNodes().item(0).getChildNodes().item(1).getChildNodes().item(x).g
etNextSibling().getNextSibling()!=null){
            /* catch building's names */
            Node
transf=original.getChildNodes().item(0).getChildNodes().item(1).getChildNodes().ite
m(x+2).getAttributes().getNamedItem("DEF");
            String buildingname= transf.getNodeValue();
            String name1=buildingname.substring(0,buildingname.indexOf("-"));
            /* load catastrophy xml - check if current catastrophy affects building */
            Document dom=loadXML("models/"+name1+"/catastrophy.xml",false);

            int y=1;
            while
(dom.getChildNodes().item(1).getChildNodes().item(1).getChildNodes().item(y).get
NextSibling().getNextSibling()!=null){
```

```
                Node
s=dom.getChildNodes().item(1).getChildNodes().item(1).getChildNodes().item(y).get
Attributes().getNamedItem("name");
                String temp=s.getNodeValue();

            /* if true => catastrophy effect building so... get pos,cost,limit */
                if(temp.equals("earthquake")){
                   NodeList
earthqua=dom.getChildNodes().item(1).getChildNodes().item(1).getChildNodes().ite
m(y).getChildNodes();
                   int
cost=Integer.valueOf(earthqua.item(1).getChildNodes().item(0).getNodeValue());
                   int
limit=Integer.valueOf(earthqua.item(3).getChildNodes().item(0).getNodeValue());
                   v.add(new catastrophymanager(x+2,cost,limit));
                }
                y=y+2;
             }
             x=x+2;

         }
         /* calculate the percentage of destruction */
         int vsize=v.size();
         double per=Math.round(((double)perc/(double)100)*vsize);
         /*catch with random ,buildings */
         for(int z=0;z<per;z++){
            generator.nextInt(vsize);
            catastrophymanager nodetodel=(catastrophymanager)
v.get(generator.nextInt(vsize));
            if(nodetodel.limit<dur){

               /* delete nodes */
               Node
del=original.getChildNodes().item(0).getChildNodes().item(1).getChildNodes().item(
nodetodel.pos);

original.getChildNodes().item(0).getChildNodes().item(1).removeChild(del);

               /*save file */
               try {
                   TransformerFactory transfac = TransformerFactory.newInstance();
                   transfac.setAttribute("indent-number", new Integer(4));
                   Transformer trans = transfac.newTransformer();
                   trans.setOutputProperty(OutputKeys.INDENT, "yes");


                   String Path=null;
                   File currentDir=new File(".");
                   try{
                      Path=currentDir.getCanonicalPath();
```

```
                System.out.println(currentDir.getCanonicalPath());
            }
            catch(IOException e){
                log.error("IOExceprion"+ e);
            }

            File output=new File(Path+"/Xj3DP2P/models/savegame.x3d");

            /* Due to a bug in some releases of java, we need a writer here.
             * So it makes this line really long
             */
            StreamResult result = new StreamResult(
                    new OutputStreamWriter(
                    new FileOutputStream(output),"utf-8"));
            DOMSource source = new DOMSource(original);
            trans.transform(source,result);
        }catch(Exception e) {
            log.error("Couldn't save file. "+e);
        }

        total_building_loss=total_building_loss+1;
    }else if(nodetodel.limit>=dur){
        removecost(nodetodel.cost);
        total_loss=total_loss+nodetodel.cost;
    }
}
```

Πίνακας 9

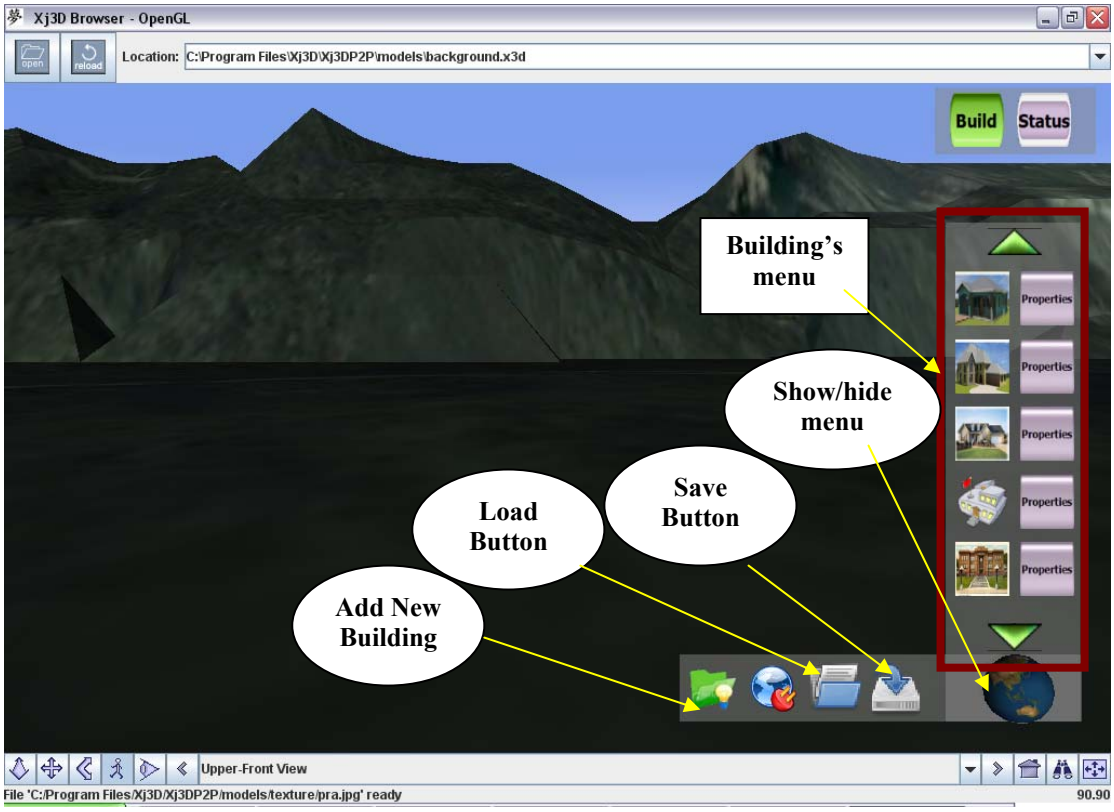
4. Οδηγίες χρήσης της εφαρμογής

Σε αυτό το σημείο κρίθηκε σκόπιμο να γίνει πλήρης αναφορά του τρόπου λειτουργίας της εφαρμογής έτσι ώστε να κατανοήσει ο χρήστης τον τρόπο που παίζεται το παιχνίδι.

Για να ξεκινήσει το παιχνίδι θα πρέπει να εκτελεστεί ο browser του xj3d. Στη συνέχεια θα πρέπει να φορτωθεί το αρχικό αρχείο της εφαρμογής που είναι το background.x3d. Θα πρέπει εδώ να σημειωθεί ότι για να εκτελεστεί σωστά η εφαρμογή θα πρέπει ο φάκελος XJ3DP2P καθώς και όλα τα περιεχόμενά του να εγκατασταθεί μέσα στο φάκελο του Xj3D.Μόλις ανοίξει το αρχείο background.x3d το οποίο δεν παίρνει περισσότερο από μερικά δευτερόλεπτά εμφανίζεται το περιβάλλον της εφαρμογής.

Δεξιά υπάρχει το μενού με τα αντικείμενα που μπορούν να εισαχθούν στον κόσμο. Από τα βέλη (πάνω ,κάτω) αποκτάται πρόσβαση σε περισσότερα αντικείμενα που αρχικά δε φαίνονται. Επιλέγοντας κάποιο από τα κουμπιά properties εμφανίζονται οι ιδιότητες του αντικείμενου που βρίσκεται δίπλα του. Οι ιδιότητες αυτές περιέχουν πληροφορίες σχετικά με τους πόντους και τα

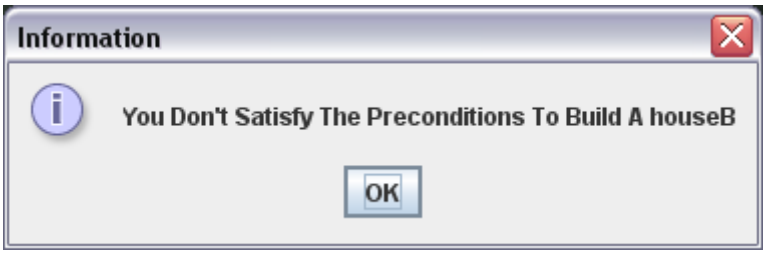
προαπαιτούμενα που έχει το κάθε αντικείμενο. Επιλέγοντας το εικονίδιο με την περιστρεφόμενη αποκρύπτεται ή εμφανίζεται ολόκληρο το μενού. Η δυνατότητα αποθήκευσης της τρέχουσας κατάστασης του κόσμου γίνεται εφικτή επιλέγοντας το κουμπί Save. Για να φορτωθεί μια προηγούμενη κατάσταση που προηγούμενος είχε αποθηκευτεί θα πρέπει να επιλεχθεί το κουμπί Load. Το κουμπί status που βρίσκεται πάνω δεξιά εμφανίζει την τρέχουσα κατάσταση του παιχνιδιού. Τους πόντους τα επίπεδα και τις τιμές των παραμέτρων που έχει ο κόσμος. Η περιήγηση στον κόσμο είναι εφικτή με διάφορους τρόπους. Ο browser του Xj3D παρέχει ένα μενού με διαφορετικές επιλογές για τον τρόπο περιήγησης στον κόσμο. Εκτός όμως από τον Xj3D browser υπάρχει η δυνατότητα της άμεσης αλλαγής του οπτικού πεδίου αλλάζοντας το View Point. Η επιλογή αυτή καθώς και αυτές που παρέχει το Xj3D βρίσκονται στο κάτω μέρος του παραθύρου. Συγκεκριμένα έχουν δημιουργηθεί πέντε διαφορετικές οπτικές γωνίες ή αλλιώς View Points. Η πρώτη δείχνει τον κόσμο από μπροστά,η δεύτερη από το πλάι αριστερά,η Τρίτη από το πλάι δεξιά,η τέταρτη από το πίσω μέρος και τέλος η Πέμπτη από πάνω. Στόχος του χρήστη είναι να δημιουργήσει μία σωστά δομημένη πόλη φτάνοντας το επίπεδο 3.Για να συμβεί αυτό θα πρέπει να δημιουργηθεί μία φανταστική πόλη αντιμετωπίζοντας κάθε δυσκολία που παρουσιάζεται.



Εικόνα 1

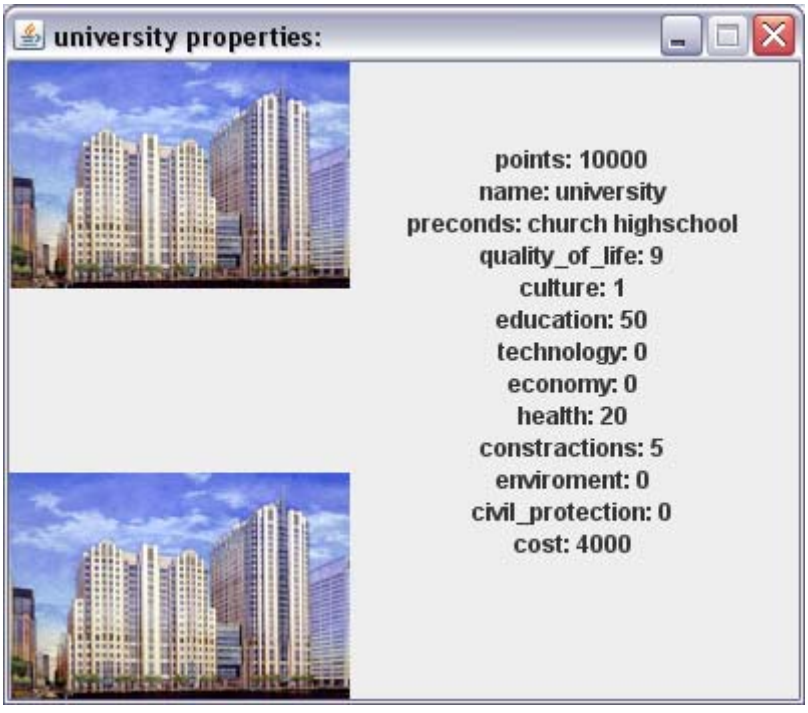
Σε αυτό το σημείο ο χρήστης είναι έτοιμος να αρχίσει τη δημιουργία της δίκής του φανταστικής πόλης. Επιλέγοντας ένα κτίριο για να εισαχθεί στον κόσμο θα πρέπει να ικανοποιούνται οι προαπαιτούμενες συνθήκες, δηλαδή αν

φτάνουν τα χρήματα για την ανέγερσή του κτιρίου και αν τα προαπαιτούμενα κτίρια έχουν ήδη χτιστεί. Εάν οι συνθήκες δεν ικανοποιούνται εμφανίζεται ένα παράθυρο που ενημερώνει ότι η ανέγερση του κτιρίου δε μπορεί να προχωρήσει.



Εικόνα 2

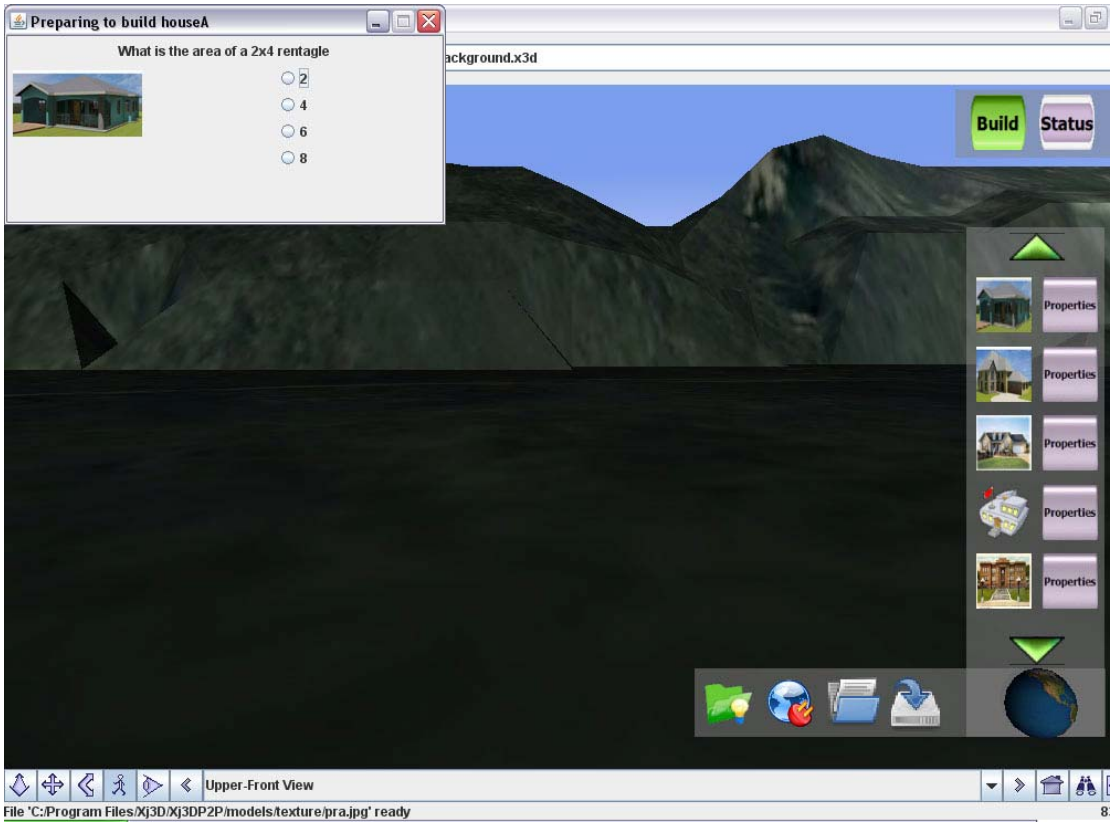
Ο χρήστης μπορεί να αποφύγει την πιθανότητα ενημέρωσής του από το σύστημα για την ανέφικτη εισαγωγή αντικειμένου στο χώρο χρησιμοποιώντας την επιλογή properties. Με αυτόν τον τρόπο μπορεί να ελέγξει τα προαπαιτούμενα κτίρια και το κόστος ανέγερσής κάθε κτιρίου πριν να επιλέξει να το εισάγει στον κόσμο. Στην εικόνα 3 φαίνονται τα προαπαιτούμενα του κτιρίου university καθώς και το κόστος ανέγερσής του.



Εικόνα 3

Η επιλογή properties εκτός απο τον έλεγχο των προαπαιτούμενων μπορεί να χρησιμοποιηθεί και για τον γενικότερο έλεγχο των παραμέτρων του αντικειμένου. Έτσι ο χρήστης μπορεί να ελέγξει τι κερδίζει εισάγοντας κάποιο αντικείμενο στο χώρο ανα πάσα στιγμή.

Εάν ικανοποιούνται οι προαπαιτούμενες συνθήκες για την εισαγωγή ενός αντικειμένου στο χώρο, κάθε φορά που ο χρήστης επιλέξει να εισάγει ένα αντικείμενο στον κόσμο, εμφανίζεται ένα παράθυρο με μία απλή μαθηματική ερώτηση. Η ερώτηση αυτή θα πρέπει να απαντηθεί σωστά για να μπορέσει να εισαχθεί το κτίριο στον εικονικής πραγματικότητας κόσμο που έχει δημιουργήσει ο χρηστής. Θα πρέπει να σημειωθεί ότι η ερώτηση αυτή είναι κάθε φορά διαφορετική. Στην εικόνα 4 απεικονίζεται η συγκεκριμένη κατάσταση κατά την οποία στην οθόνη εμφανίζεται ένα παράθυρο διαλόγου με μία ερώτηση και με μία σειρά απαντήσεων στο κάτω μέρος του.



Εικόνα 4

Στην περίπτωση που απαντηθεί σωστά το κτίριο εισάγεται στον κόσμο και ο χρήστης μπορεί να το μετακινήσει στο χώρο απλώς επιλέγοντας το και μετακινώντας το με το ποντίκι. Διαφορετικά το παράθυρο με την ερώτηση κλείνει χωρίς να εισαχθεί το κτίριο.

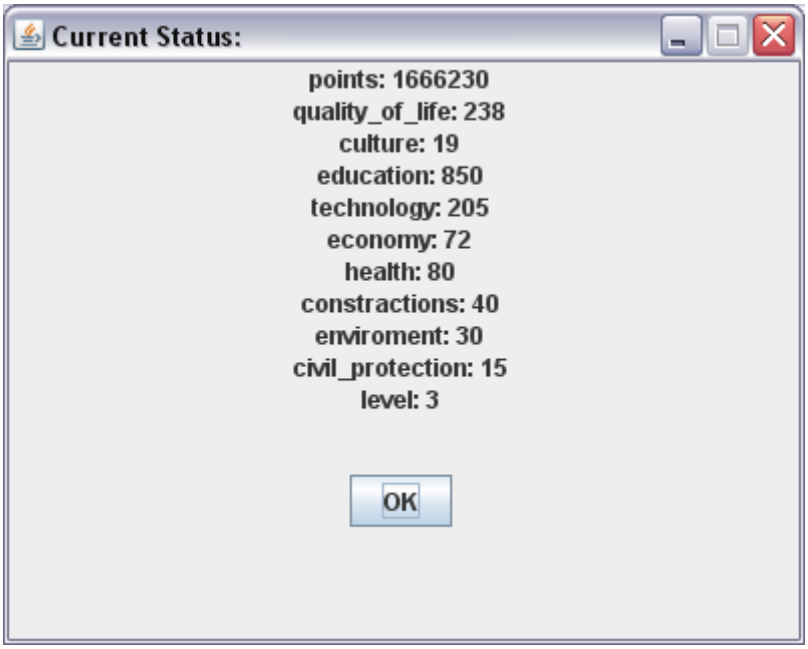
Έπειτα και αφού αποφασίσει το σημείο που θέλει να το κτίσει επιλέγοντάς το κουμπί Build το κτίριο κτίζεται καθώς απενεργοποιείται η δυνατότητα μετακίνησης του και από αυτό το σημείο και έπειτα δεν ενεργοποιείται ποτέ ξανά. Έτσι καλό είναι να επιλέγεται μόνο όταν έχει αποφασιστεί το σημείο που θα χτιστεί το κτίριο. Πρέπει να σημειωθεί ότι επιλέγοντας τη λειτουργία build, όσα κτίρια έχουν τη δυνατότητα να μετακινούνται στο χώρο και δεν τα έχει χτίσει ο χρήστης σε προηγούμενη κατάσταση χτίζονται επίσης χάνοντας έτσι για πάντα τη δυνατότητα μετακίνησης τους στο χώρο.

Καθώς εξελίσσεται το παιχνίδι και εισάγονται νέα κτίρια, οι παράμετροι του συστήματος αυξάνονται και έτσι κάποια στιγμή το επίπεδο αναβαθμίζεται απο ένα που είναι αρχικά σε δυο και στη συνέχεια τρία. Όταν επιτυγχάνεται αυτή η αλλαγή επιπέδου ο χρήστης ενημερώνεται με ένα παράθυρο που εμφανίζεται στην οθόνη.

Εικόνα 5

Η επιλογή save επιτρέπει την αποθήκευση της εκάστοτε τρέχουσας κατάστασης του παιχνιδιού. Η επαναφορά της τελευταίας αποθηκευμένης κατάστασης γίνεται με την επιλογή του button load.

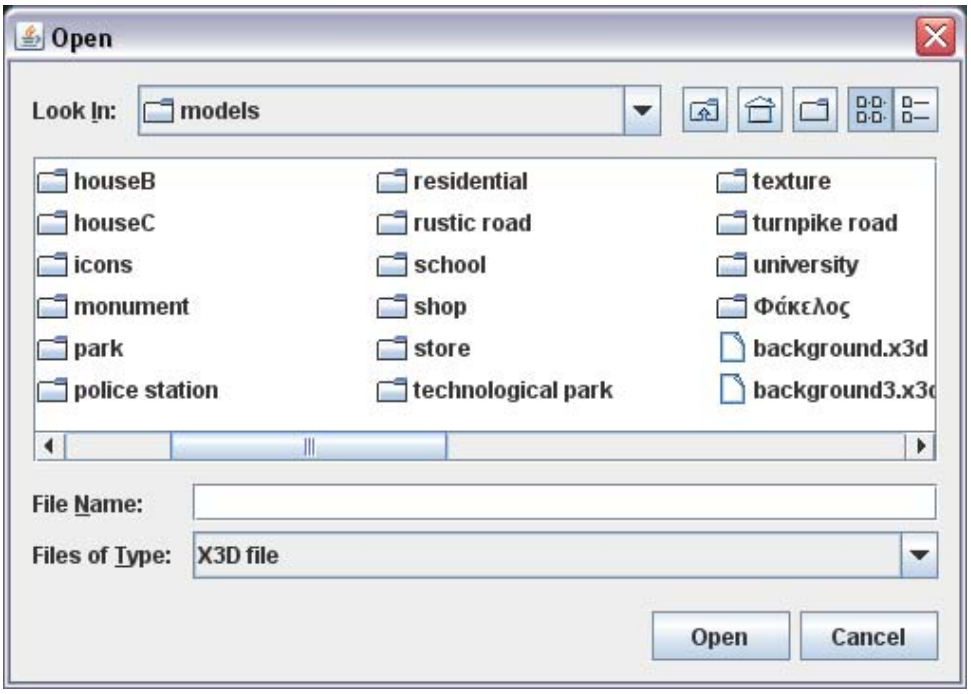
Το Button Status παρέχει τη δυνατότητα της επισκόπησης της τρέχουσας κατάστασης του κόσμου. Με τη χρήση του μπορεί να γίνει επισκόπηση ης κατάστασης των παραμέτρων της εφαρμογής ανα πάσα στιγμή.



Εικόνα 6

Τέλος η επιλογή του κουμπιού AddNewBuilding όπως φαίνεται στην εικόνα 1, δίνει τη δυνατότητα εισαγωγής ενός αντικείμενου στον κόσμο το οποίο δεν βρίσκεται στο μενού των κτιρίων αλλά ο χρήστης επιθυμεί να το εισάγει στον κόσμο του. Για να μπορέσει να συμβεί αυτό ο χρήστης επιλέγει το x3d αρχείο που επιθυμεί και το σύστημα το ενσωματώνει στον κόσμο.

Σημειώνεται εδώ ότι το κτίριο θα πρέπει να είναι αποθηκευμένο στον φάκελο models με τον ίδιο τρόπο όπως είναι και τα υπόλοιπα αντικείμενα δηλαδή θα πρέπει να αποτελείται από το x3d αρχείο, και ένα φάκελο που περιέχει τα xml αρχεία properties και catastrophy καθώς επίσης και τη φωτογραφία του αντικειμένου. Όταν εντοπιστεί το αρχείο και γίνει η εισαγωγή του στον κόσμο γίνεται κανονικά έλεγχος για την ικανοποίηση των προαπαιτούμενων και εμφανίζεται η ερώτηση που θα πρέπει να απαντηθεί για να εισαχθεί το αντικείμενο στον κόσμο.



Εικόνα 7

Παράρτημα

1.Περιορισμοί /Τεχνικές δυσκολίες

Κατά την υλοποίηση της εφαρμογής που παρουσιάστηκε παρουσιάστηκαν διάφορα προβλήματα σχετικά με το xj3d και τον τρόπο που επικοινωνεί με τη java .Έτσι κρίθηκε σκόπιμο να αναφερθούν σε αυτό το παράρτημα.

a) PlaneSensors

Το X3D δεν επιτρέπει την ενσωμάτωση δύο ή περισσότερων sensors στο ίδιο αντικείμενο οι οποίοι θα λειτουργούν την ίδια στιγμή. Έτσι εγκαταστάθηκε ένας sensor στο κάθε αντικείμενο με αποτέλεσμα να μπορεί να

γίνει μετακίνηση μόνο στις δύο από τις τρεις διαστάσεις υποστηρίζεται έτσι από πάνω προς τα κάτω και από δεξιά προς τα δεξιά κίνηση.

b) Proto instances

Με τη χρήση του proto instance έγινε δυνατό να εγκατασταθούν περισσότεροι από ένας sensors και επιτεύχθηκε η δυνατότητα μετακίνησης του εκάστοτε αντικειμένου και στις τρεις διαστάσεις του κόσμου. Τα proto instances είναι κατά κάποιο τρόπο ορισμοί αντικειμένων που καλούνται και αποδίδονται στη σκηνή, είναι πρότυπα βάση των οποίων μπορούν να σχεδιαστούν αντικείμενα με τα ίδια χαρακτηριστικά. Τα proto instances βολεύουν στη δημιουργία αντικειμένων αλλά όχι στην διαχείριση τους μέσα από τη java Το πρόβλημα που παρουσιάστηκε σε αυτή την περίπτωση είναι ότι από τη java δεν υπάρχει κανένας τρόπος επικοινωνίας με το εσωτερικό του proto και άρα στην περίπτωσή της εφαρμογής αυτής ήταν αδύνατο να αποκτηθεί πρόσβαση στο πεδίο translation του αντικειμένου που μετακινούνταν για να αποκτηθούν οι εκάστοτε συντεταγμένες του στον κόσμο. Έτσι η ιδέα χρήσης proto instances εγκαταλείφθηκε.

c) ECMA Script

Υπάρχει αδυναμία εκτέλεσης του ecma Script στην περίπτωση που χρειάζεται να "σηκωθεί" μία σκηνή μέσα από μία άλλη X3D σκηνή με τη χρήση του Xj3D browser.Αυτό γίνεται γιατί το xj3d αντιλαμβάνεται πρώτα τα αντικείμενα που υποστηρίζονται από το opengl και τα οποία σχεδιάζει, σε δεύτερο χρόνο αντιλαμβάνεται τους sensors και τέλος τα scripts. Σε μία αυτόνομη σκηνή δεν αντιμετωπίζεται κανένα πρόβλημα στην εκτέλεση του ecma Script καθώς γίνεται update, μόνο στο αντικείμενο που σχεδιάζεται και όχι σε ολόκληρη τη σκηνή. Όταν το αντικείμενο "φορτωθεί" από μια άλλη σκηνή γίνεται update σε ολόκληρη τη σκηνή με αποτέλεσμα να μη μπορούν να εκτελεστούν τα Scripts.

d) Sensor Manager

Ένα ακόμη από τα προβλήματα που παρατηρήθηκαν κατά την υλοποίηση της εφαρμογής είναι ότι ο Sensor manager αντιμετωπίζει προβλήματα στη διαχείριση δυναμικών sensors και ιδιαίτερα στη δημιουργία και διαχείριση των routes των sensors. Το αποτέλεσμα είναι να μην είναι εφικτή η απενεργοποίηση του planesensor του κάθε κτιρίου ,αφαιρώντας το route που τον έχει ενεργοποιήσει. Έτσι για την απενεργοποίηση του είναι αναγκαστική η διαγραφή του node του Planesensor μέσα από το XML αρχείο.

Το συγκεκριμένο πρόβλημα έχει αναφερθεί και έχει κατατεθεί στον επίσημο bugzilla του x3d από άτομο της ομάδας του εργαστηρίου ανάπτυξης πολυμέσων του ΤΕΙ Κρήτης.

e) Omni lights

Κατά την απεικόνιση του x3d κόσμου χρειάστηκε να εγκατασταθούν omni lights για την βελτιστοποίηση του φωτισμού στον κόσμο. Το πρόβλημα που παρατηρήθηκε είναι ότι ενώ τα αντικείμενα που εισάγονται στον κόσμο φωτίζονται το terrain που έχει δημιουργηθεί και ο γενικότερος φωτισμός του ουρανού δεν φωτίζονται.

2. Κώδικας X3D (αρχείου background)

Παρατίθεται τμήμα του κώδικα του αρχείου background :

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<X3D profile="Immersive">
<Scene>

 <Background groundAngle="    .03491    1.5708" groundColor="    .10588 .08627
.06275    .12549 .12549 .08235    .12941 .17647 .10196" skyAngle="   1.4   1.5"
skyColor="   .39216 .65098 .93333   .4902 .61961 .92549   .90196 .90196 1"/>

<Viewpoint DEF="A_vp_vp_vp_vp" description="Upper-Front View"
fieldOfView="0.785" jump="true" orientation="1 0 0 -.1" position="0 10 28"/>

<ProximitySensor DEF="HereIAm" center="0 0 0" containerField="children"
size="1000 1000 1000"/>
 <Transform DEF="dad_Omni01" containerField="children" translation="2.638 500
6.796">
  <PointLight DEF="Omni01" ambientIntensity="0.000" attenuation="1 0 0" color="1
1 1" containerField="children" intensity="1.000" on="true" radius="10290.000"/>
 </Transform>

 <Transform DEF="HUD" containerField="children">
  <Transform DEF="MenuPosition" containerField="children" translation="1.7 0 -3">

   <Transform DEF="BuildingPanel" containerField="children" translation="-.01 -.2
0">
    <Shape containerField="children">
     <Appearance containerField="appearance">
      <Material ambientIntensity="0.200" containerField="material" diffuseColor=".5
.5 .5" shininess="0.200" transparency="0.500"/>
     </Appearance>
     <Box containerField="geometry" size=".5 1.85 .0001"/>
    </Shape>
    <Transform containerField="children" translation="0 .85 .01">
     <Transform containerField="children">
      <Group containerField="children">
       <Shape containerField="children">
        <Appearance containerField="appearance">
         <ImageTexture containerField="texture" url="
&amp;quot;./icons/arrowUp.gif&amp;quot;"/>
         <Material ambientIntensity="0.200" containerField="material" diffuseColor="1
0 1" shininess="0.200"/>
        </Appearance>
        <Box containerField="geometry" size=".2 .1 .0001"/>
       </Shape>
       <TouchSensor DEF="arrow_up" containerField="children"/>
      </Group>
```

```
    </Transform>
   </Transform>
   <Transform DEF="Build1" containerField="children" translation="-.12 .64 .01">
    <Transform containerField="children">
     <Group containerField="children">
      <Shape containerField="children">
       <Appearance containerField="appearance">
        <ImageTexture containerField="texture" url="
&amp;quot;./icons/houseA.jpg&amp;quot;"/>
        <Material ambientIntensity="0.200" containerField="material" diffuseColor="1
0 1" shininess="0.200"/>
       </Appearance>
       <Box containerField="geometry" size=".2 .2 .0001"/>
      </Shape>
      <TouchSensor DEF="Build1TS" containerField="children"/>
     </Group>
    </Transform>
   </Transform>
   <Transform DEF="Build17P" containerField="children" translation=".12 .64 .01">
    <Transform containerField="children">
     <Group containerField="children">
      <Shape containerField="children">
       <Appearance containerField="appearance">
        <ImageTexture containerField="texture" url="
&amp;quot;./icons/properties.gif&amp;quot;"/>
        <Material ambientIntensity="0.200" containerField="material" diffuseColor="1
0 1" shininess="0.200"/>
       </Appearance>
       <Box containerField="geometry" size=".2 .2 .0001"/>
      </Shape>
      <TouchSensor DEF="Build1PTS" containerField="children"/>
     </Group>
    </Transform>
   </Transform>
   <Transform containerField="children" translation="0 -.61 .01">
    <Transform containerField="children">
     <Group containerField="children">
      <Shape containerField="children">
       <Appearance containerField="appearance">
        <ImageTexture containerField="texture" url="
&amp;quot;./icons/arrowDown.gif&amp;quot;"/>
        <Material ambientIntensity="0.200" containerField="material" diffuseColor="1
0 1" shininess="0.200"/>
       </Appearance>
       <Box containerField="geometry" size=".2 .1 .00001"/>
      </Shape>
      <TouchSensor DEF="arrow_down" containerField="children"/>
     </Group>
    </Transform>
   </Transform>
```

```
  </Transform>
  </Transform>
 </Transform>
 <MetadataString DEF="meta" containerField="metadata"
value="models/w_status.xml"/>
</Transform>

<ROUTE fromNode='HereIAm' fromField='orientation_changed' toNode='HUD'
toField='rotation'/>
<ROUTE fromNode='HereIAm' fromField='position_changed' toNode='HUD'
toField='translation'/>

 <Script DEF="SC" directOutput="TRUE"  mustEvaluate="TRUE"  url="/Program
Files/Xj3D/Xj3DP2P/build/classes/BuildingPanelSC.class">

  <field accessType='inputOnly' name='houseAClicked' type='SFTime'/>
  <field accessType='inputOnly' name='houseAPClicked' type='SFTime'/>

  <field accessType='outputOnly' name='b1_visibility' type='SFVec3f'/>
  <field accessType='outputOnly' name='b2_visibility' type='SFVec3f'/>
</Script>

<!-- Routes used for the build buttons. -->

<ROUTE fromNode='Build1TS' fromField='touchTime' toNode='SC'
toField='houseAClicked'/>
<ROUTE fromNode='SC' fromField='b1_visibility' toNode='Build1'
toField='set_scale'/>
<ROUTE fromNode='SC' fromField='b1_translation' toNode='Build1'
toField='set_translation'/>

<TimeSensor DEF='CS' cycleInterval='10'  loop='true' enabled='true'/>

</Scene>
</X3D>
```

3. Κλάση configuration

```
package xj3dp2p.configuration;
import java.io.File;
import java.io.FileOutputStream;
import java.io.OutputStreamWriter;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.SAXParser;
import javax.xml.parsers.SAXParserFactory;
import javax.xml.transform.OutputKeys;
import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.dom.DOMSource;
```

```
import javax.xml.transform.stream.StreamResult;
import org.w3c.dom.Document;
import org.apache.log4j.Logger;
import org.w3c.dom.Element;
import org.xml.sax.Attributes;
import org.xml.sax.helpers.DefaultHandler;
import org.xml.sax.Locator;
import org.xml.sax.SAXException;
import java.io.IOException;

public class Configuration {
    protected static String defaultFile="C:/Program
    Files/Xj3D/Xj3DP2P/configuration.xml";
    protected Logger log;
    protected static String baseFile;//="background.x3d";
    protected static String saveFile;//="background.x3d";

   /** Creates a new instance of Configuration */
    public Configuration() {
        this(defaultFile);
    }

    public Configuration(String filename) {
        log=Logger.getLogger(this.getClass());

        SAXParserFactory factory = SAXParserFactory.newInstance();
        try {
            // Parse the input
            SAXParser saxParser = factory.newSAXParser();
            saxParser.parse(new File(filename),new ConfigurationReadHandler(this));

        } catch (Throwable t) {
            log.error("Parsing configuration file");
        }
    }

    public void saveConfiguration() {
        DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
        try {
            /* Create the document */
            DocumentBuilder builder = factory.newDocumentBuilder();
            Document doc=builder.newDocument();

            /* Create the root node */
            Element elm=doc.createElement("config");
            doc.appendChild(elm);

            /* Append the savefile element */
            Element file=doc.createElement("savefile");
            file.appendChild(doc.createTextNode(this.getSaveFile()));
```

```
            elm.appendChild(file);
            /* Append the basefile element */
            file=doc.createElement("basefile");
            file.appendChild(doc.createTextNode(this.getBaseFile()));
            elm.appendChild(file);

            /* Save in a file */
            TransformerFactory transfac = TransformerFactory.newInstance();
            transfac.setAttribute("indent-number", new Integer(4));
            Transformer trans = transfac.newTransformer();
            trans.setOutputProperty(OutputKeys.INDENT, "yes");

trans.setOutputProperty(OutputKeys.DOCTYPE_SYSTEM,"configuration.dtd");


            File output=new File(this.defaultFile);

            /* Due to a bug in some releases of java, we need a writer here.
             * So it makes this line really long
             */
            StreamResult result = new StreamResult(
                  new OutputStreamWriter(
                  new FileOutputStream(output),"utf-8"));
            DOMSource source = new DOMSource(doc);
            trans.transform(source,result);

         } catch (Exception e) {
            log.error("Error saving: "+e);
         }
      }

      public String getBaseFile() {
         return baseFile;
      }

      public void setBaseFile(String baseFile) {
         this.baseFile = baseFile;
      }

      public String getSaveFile() {
         return saveFile;
      }

      public void setSaveFile(String saveFile) {
         this.saveFile = saveFile;
      }
}
```

4.Κλάση ConfigurationReadHandler

```
package xj3dp2p.configuration;

import org.xml.sax.Attributes;
import org.xml.sax.helpers.DefaultHandler;

public class ConfigurationReadHandler extends DefaultHandler {
private final int UNKNOWN=0;
private final int BASEFILE=1;
private final int SAVEFILE=2;
private int status=this.UNKNOWN;
private Configuration conf;

/** Creates a new instance of ConfigurationReadHandler */
public ConfigurationReadHandler(Configuration conf) {
this.conf=conf;
}


public void startElement(String uri, String localName, String qName, Attributes atts)
{
if(qName=="basefile") this.status=this.BASEFILE;
else if(qName=="savefile") this.status=this.SAVEFILE;
else this.status=this.UNKNOWN;
}

public void endElement(String uri, String localName, String qName) {
this.status=UNKNOWN;
}

public void characters(char[] ch, int start, int length) {
switch(this.status) {
case BASEFILE:
conf.setBaseFile(new String(ch,start,length).trim());
break;
case SAVEFILE:
conf.setSaveFile(new String(ch,start,length).trim());
break;
}
}
}
```

5. Κλάση WorldManager

```
package xj3dp2p;

import java.io.File;
import java.util.HashMap;
import java.util.LinkedList;
import java.util.List;
import java.util.Map;
```

```
import org.apache.log4j.Logger;
import java.rmi.dgc.VMID;
import xj3dp2p.configuration.Configuration;
import xj3dp2p.save.SaveWorld;
import xj3dp2p.save.SaveTimeWorld;
import org.web3d.x3d.sai.*;
import xj3dp2p.xml.XMLHelper;
import org.w3c.dom.*;
/**update world status **/
import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.stream.StreamResult;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.TransformerException;
import javax.xml.transform.OutputKeys;
import java.io.IOException;
import xj3dp2p.housemanager.housemanager;
import javax.swing.*;

public class WorldManager  {
protected Browser x3dBrowser;
protected Map fields;
protected X3DScene mainScene;
protected Map<X3DNode,String> nodeToNodeID=null;
protected Map<String,X3DNode> nodeIDToNode=null;
protected Map<X3DNode,String> nodeToName=null;
protected static Configuration conf=null;
protected Logger log;
protected int i=1;
private String Path;
housemanager hm;
private SFTime res_Clicked;
X3DTimeDependentNode t;
SFTime ti;

/** Creates a new instance of WorldManager */
public WorldManager(Browser browser,Map fields){
this.x3dBrowser=browser;
this.fields=fields;
this.mainScene=(X3DScene)browser.getExecutionContext();
this.nodeIDToNode=new HashMap(100);
this.nodeToNodeID=new HashMap(100);
this.nodeToName=new HashMap(100);
if(WorldManager.conf==null)
WorldManager.conf=new Configuration();
this.log=Logger.getLogger(this.getClass());
}

public void saveWorldStatus() {
SaveWorld status=new SaveWorld(conf,nodeToNodeID,nodeToName);
```

```
status.save();
}

public void saveTimeWorldStatus(){
SaveTimeWorld status1=new SaveTimeWorld(conf,nodeToNodeID,nodeToName);
status1.saveTime();
}

/**
* Add a new building to the world and control its status.
* @param name Name of the building. It has to be a valid file inside the
*             <code>models</code> directory.
*/
public synchronized void addNewBuilding(String name)  {
/* Load the building file */
X3DScene buildingScene = x3dBrowser.createX3DFromURL( new String[] { name +
".x3d"} );

/* Get the elements we need from the scene */
/* Element to add to the scene */
X3DNode buildingAndSensor = buildingScene.getNamedNode(name);
/* Sensor to be able to move the building */
X3DNode sensor = buildingScene.getNamedNode("sensor");
/* The building structure */
X3DNode building = buildingScene.getNamedNode("building");

/* Update the maps  */
this.nodeToName.put(building,name);
VMID myID=new VMID();
log.debug("New '"+name+"' with id '"+myID+"'");
this.nodeToNodeID.put(building,myID.toString());
this.nodeIDToNode.put(myID.toString(),building);

mainScene.addRootNode(buildingAndSensor);
mainScene.addRoute(sensor, "translation_changed",building, "set_translation" );
/* get world  meta*/
mainScene = (X3DScene) x3dBrowser.getExecutionContext();
X3DNode meta_world = mainScene.getNamedNode("meta");
MFString meta_w_value=(MFString)meta_world.getField("value");
String xml_world =meta_w_value.get1Value(0);

/** Update World */
updatestatus(xml_world,name);
}

public synchronized void addNewBuilding(String name,String FullPath)  {
/* Load the building file */

String lol=FullPath.substring(FullPath.indexOf("models"),FullPath.indexOf("."));
String path=lol.substring(7);
```

```
X3DScene buildingScene = x3dBrowser.createX3DFromURL(new String[]{path+
".x3d"});

/* Get the elements we need from the scene */
/* Element to add to the scene */
X3DNode buildingAndSensor = buildingScene.getNamedNode(name);
/* Sensor to be able to move the building */
X3DNode sensor = buildingScene.getNamedNode("sensor");
/* The building structure */
X3DNode building = buildingScene.getNamedNode("building");

/* Update the maps  */
this.nodeToName.put(building,name);
VMID myID=new VMID();
log.debug("New '"+name+"' with id '"+myID+"'");
this.nodeToNodeID.put(building,myID.toString());
this.nodeIDToNode.put(myID.toString(),building);

mainScene.addRootNode(buildingAndSensor);
mainScene.addRoute(sensor, "translation_changed",building, "set_translation" );

}

/**
* Load the world status using the default configuration. This will replace
* the current world
*/
public void loadWorldStatus() {
/* Get the current Path to append to the load world URL */
File currentDir=new File(".");
String currentPath=".";

try {
currentPath=currentDir.getCanonicalPath();
}catch(Exception e) {
log.error("Couldnt get current path");
}

/* The next time we will save the world using this saved world */
if(!conf.getBaseFile().equals(conf.getSaveFile())){
System.out.println("base !== save");
conf.setBaseFile(conf.getSaveFile());
}
/* Load the saved world scene, and replace the current world */

X3DScene loadedWorld=x3dBrowser.createX3DFromURL( // new String[]
{currentPath+"/"+conf.getSaveFile()});
new String[] {conf.getSaveFile()});
x3dBrowser.replaceWorld(loadedWorld);
```

```
}

public void updatestatus(String xml_url , String name){
Document tmp = XMLHelper.loadXML(xml_url);
NodeList  properties = tmp.getChildNodes().item(0).getChildNodes();
NodeList propertie=properties.item(1).getChildNodes();

if(name=="houseA" ||name =="houseB" || name=="houseC"){

NodeList hpropertie =houseProperties(name);
String house_name=hpropertie.item(3).getChildNodes().item(0).getNodeValue();

/* Add houses to the xml w_status */
String houses=propertie.item(3).getChildNodes().item(0).getNodeValue();
System.out.println(houses);
propertie.item(3).getChildNodes().item(0).setNodeValue(houses+", "+name);

addpoints(propertie,hpropertie);


}else if(name=="police station"||name=="fire station"){
NodeList hpropertie =houseProperties(name);
String road_points=hpropertie.item(1).getChildNodes().item(0).getNodeValue();
addpoints(propertie,hpropertie);
/*Add everything else*/
for(i=7;i<=23;i=i+2){
String w_attribute = propertie.item(i+4).getChildNodes().item(0).getNodeValue();
String h_attribute=hpropertie.item(i).getChildNodes().item(0).getNodeValue();
propertie.item(i+4).getChildNodes().item(0).setNodeValue(String.valueOf(Integer.val
ueOf(w_attribute).intValue()+Integer.valueOf(h_attribute).intValue()));
}
/* Add stations to the xml w_status */
String stations=propertie.item(7).getChildNodes().item(0).getNodeValue();
propertie.item(7).getChildNodes().item(0).setNodeValue(stations+", "+name);
}else{
NodeList hpropertie =houseProperties(name);
addpoints(propertie,hpropertie);
for(i=7;i<=23;i=i+2){
String w_attribute=propertie.item(i+4).getChildNodes().item(0).getNodeValue();
String h_attribute=hpropertie.item(i).getChildNodes().item(0).getNodeValue();
propertie.item(i+4).getChildNodes().item(0).setNodeValue(String.valueOf(Integer.val
ueOf(w_attribute).intValue()+Integer.valueOf(h_attribute).intValue()));
}
/* Add buildings to the xml w_status */
String buildings=propertie.item(9).getChildNodes().item(0).getNodeValue();
propertie.item(9).getChildNodes().item(0).setNodeValue(buildings+", "+name);
}
checklevel(properties);

try{
```

```
Transformer transformer = TransformerFactory.newInstance().newTransformer();
transformer.setOutputProperty(OutputKeys.INDENT, "yes");

//initialize StreamResult with File object to save to file

File currentDir=new File(".");
try{
Path=currentDir.getCanonicalPath();
System.out.println(currentDir.getCanonicalPath());
}
catch(IOException e){
log.error("IOException" +e);
}

File output=new File(Path +"/Xj3DP2P/"+xml_url);
StreamResult result = new StreamResult(output);
DOMSource source = new DOMSource(tmp);
transformer.transform(source, result);
}
catch(TransformerException te){
log.error("Transformer Exception" +te);
}
}

public void addpoints(NodeList propertie,NodeList hpropertie){
String house_points=hpropertie.item(1).getChildNodes().item(0).getNodeValue();
/* Add point Status */
String points=propertie.item(1).getChildNodes().item(0).getNodeValue();
propertie.item(1).getChildNodes().item(0).setNodeValue(String.valueOf(Integer.valu
eOf(points).intValue()+Integer.valueOf(house_points).intValue()));
String points1=propertie.item(1).getChildNodes().item(0).getNodeValue();

/*remove cost*/
String house_cost=hpropertie.item(29).getChildNodes().item(0).getNodeValue();
int new_points=Integer.valueOf(points1)-Integer.valueOf(house_cost);
propertie.item(1).getChildNodes().item(0).setNodeValue(String.valueOf(new_points)
);
}

public void checklevel(NodeList properties){
NodeList propertie=properties.item(1).getChildNodes();
NodeList lvl_propertie=properties.item(3).getChildNodes();
int index0=0;

/*w_status*/
String points=propertie.item(1).getChildNodes().item(0).getNodeValue();
int pts=Integer.valueOf(points);
String quality=propertie.item(11).getChildNodes().item(0).getNodeValue();
int qt=Integer.valueOf(quality);
String houses=propertie.item(9).getChildNodes().item(0).getNodeValue();
```

```
/******* level pre *******/
/*** lvl 1 ***/
NodeList lev1=lvl_propertie.item(1).getChildNodes();

int
lvl_points=Integer.valueOf(lev1.item(1).getChildNodes().item(0).getNodeValue());
String build_lvl=lev1.item(3).getChildNodes().item(0).getNodeValue();
int qt_lvl=Integer.valueOf(lev1.item(5).getChildNodes().item(0).getNodeValue());
int
level=Integer.valueOf(propertie.item(29).getChildNodes().item(0).getNodeValue());
index0=houses.indexOf(build_lvl);
if((pts>=lvl_points)&& (qt>=qt_lvl) && (index0!=-1) &&(level==1)){
JOptionPane.showMessageDialog(null, "You Have Reached Level 2","Information ",
JOptionPane.WARNING_MESSAGE);
propertie.item(29).getChildNodes().item(0).setNodeValue(String.valueOf(2));
}

/*lvl 3*/
NodeList lev3=lvl_propertie.item(3).getChildNodes();
int curr_values[]=new int[10];
int lvl_values[]=new int[10];
curr_values[0]=Integer.valueOf(propertie.item(1).getChildNodes().item(0).getNodeV
alue());
lvl_values[0]=Integer.valueOf(lev3.item(1).getChildNodes().item(0).getNodeValue())
;
int j=1;

for(i=7;i<=23;i=i+2){
curr_values[j]=Integer.valueOf(propertie.item(i+4).getChildNodes().item(0).getNode
Value());
lvl_values[j]=Integer.valueOf(lev3.item(i).getChildNodes().item(0).getNodeValue());
j++;
}
String curr_values1[]=new String[2];
String lvl_values1[]=new String[2];


curr_values1[0]=propertie.item(5).getChildNodes().item(0).getNodeValue();
lvl_values1[0]=lev3.item(3).getChildNodes().item(0).getNodeValue();
curr_values1[1]=propertie.item(9).getChildNodes().item(0).getNodeValue();
lvl_values1[1]=lev3.item(5).getChildNodes().item(0).getNodeValue();

int tmp[]=new int[11];
for(i=0;i<=9;i++){

if(curr_values[i]>=lvl_values[i]){
tmp[i]=1;
}else{
tmp[i]=-1;
```

```
}
}

tmp[9]=curr_values1[0].indexOf(lvl_values1[0]);
tmp[10]=curr_values1[1].indexOf(lvl_values1[1]);
int x=1;
for(i=0;i<=10;i++){
if(tmp[i]==-1){
break;
}else{

x++;
}
}

int
level2=Integer.valueOf(propertie.item(29).getChildNodes().item(0).getNodeValue());
if((x==12) &&(level2==2)){
JOptionPane.showMessageDialog(null, "You Have Reached Level 3","Information ",
JOptionPane.WARNING_MESSAGE);
propertie.item(29).getChildNodes().item(0).setNodeValue(String.valueOf(3));
}


}

public NodeList houseProperties(String name){

Document tmp = XMLHelper.loadXML("/models/"+name+"/properties.xml");
NodeList  properties = tmp.getChildNodes().item(1).getChildNodes();
NodeList propertie=properties.item(1).getChildNodes();
return propertie;
}
}
```