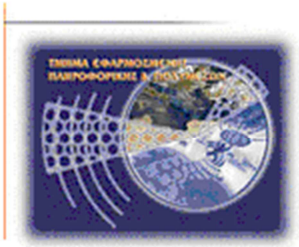
Πτυχιακή εργασία τμήματος Μηχανικών Πληροφορικής

Τεχνολογικό Εκπαιδευτικό Ίδρυμα Κρήτης

Σχολή Τεχνολογικών Εφαρμογών

τμήμα μηχανικών πληροφορικής



Πτυχιακή Εργασία

Τίτλος :

SEGMENTATION ALGORITHM ΓΙΑ 3D MESHES DATA ΚΑΙ ΥΛΟΠΟΙΗΣΗ ΣΕ WEB ΤΕΧΝΟΛΟΓΙΕΣ

Φώτης Μηλιώνης (ΑΜ:1264)

Επιβλέπων Καθηγητής : Αθανάσιος Μαλάμος

Ευχαριστίες

Ο συντάκτης της εργασίας αυτής θα ήθελε να ευχαριστήσει για την πολύτιμη καθοδήγηση και συνεργασία τους: Αθανάσιο Μαλάμο, Κωνσταντίνο Καπετανάκη, Μάρκο Ζάμπογλου τον Ιωάννη Κορναράκη για την παραχώρηση ενός τρισδιάστατου μοντέλου καθώς και όλη την ομάδα του εργαστηρίου Πολυμέσων του Τμήματος Μηχανικών Πληροφορικής του ΤΕΙ Κρήτης.

Abstract

With current days advancement in Web3D through WebGL and X3Dom using both modern and standard Web Transportation technologies it is possible to enable the viewing of complex 3D scenes in great detail by embedding them directly into HTML pages for anyone to see without the use of any specialized software besides a modern browser from any capable device, being either a personal computer, a tablet or even a smart phone.

However delivering such detailed objects and scenes that consist of large volumes of data is troublesome at best and it can lead to a very poor or inconvenient user experience. There has been a number of approaches that aim at progressive streaming of such complex models in the past, most of them demanding a restructuring the 3D information. This final year thesis implements a platform for streaming and delivering declarative X3Dom formatted scenes through the use of the industry standard http delivery mechanism implemented with the RESTful approach by enabling us almost real time restructuring and delivery.

This final year thesis was developed in a parallel research attempt by the Multimedia Content Lab of the Informatics Engineering Dept. TEI of CRETE that aimed in the use of newer web transportation technologies which lead to a conference publication on this topic and parts of the work described here are common. Furthermore in this thesis there is also an experimental State Of The Art predated implementation of the same web technologies described in our publication.

This final year thesis expands our work through the support of modern rendering techniques and streaming multiple 3D objects into one scene, we present our conclusions and the advantages in comparison with standardized transport means of delivery and more advanced ones, in the end the author proposes possible ways to move forward.

Σύνοψη

Με τις τελευταίες εξελίξεις στην περιοχή του Web3D μέσω της διασύνδεσης της OpenGL με τους φυλλομετρητές μέσω JavaScript και την δημιουργία ολόκληρων framework όπως το X3DOM και την χρήση όλων των τελευταίων εξελίξεων στην μετάδοση δεδομένων στο internet έχει καταστεί δυνατή ή θέαση πολύπλοκών τρισδιάστατών σκηνών με μεγάλη λεπτομέρεια απ' ευθείας εντός ενός HTML εγγράφου που μπορεί να δει οποιοσδήποτε χρήστης χώρις την χρήση εξιδεικευμένων εργαλείων ή πρόσθετου λογισμικού.

Όμως η μετάδοση τρισδιάστατων δεδομένων που χαρακτηρίζονται από μεγάλους όγκους μη δομημένης πληροφορίας ειδικότερα για μεγαλύτερες γεωμετρίες ή ολόκληρες σκηνές μπορεί να οδηγήσει σε φτωχή εμπειρία από την μεριά του παρατηρητή. Υπάρχουν αρκετές μεθόδοι που στοχεύουν στην τμηματική αποστόλη και ταυτόχρονη αναπαράσταση (Streaming) των τρισδιάστατων μοντέλων που όμως απαιτούν κάποιου είδους επεξεργασία και αναδιάταξη της πληροφορίας πριν την αποστολή. Αυτή η πτυχιακή εργασία παραδίδει μία πλατφόρμα για την προοδευτική επεξεργασία, μετάδοση γεωμετρών υπό την μορφή X3DOM μέσω της ραχοκοκαλιάς του διαδικτύου HTTP, την χρήση εξελιγμένων αρχιτεκτονικών και την αναπαράσταση των γεωμετριών σε πραγματικό χρόνο.

Η εργασία αυτή περιλαμβάνει πειραματικές προσεγγίσεις προς την επίτευξη του σκοπού της μέσω των τελευταίων τεχνολογικών λύσεων. Η τελική υλοποίηση αναπτύχθηκε παράλληλα με μία ερευνητική εργασία του Εργαστήριου Πολυμέσων (MCLAB) του Τμήματος Μηχανικών Πληροφορικής του ΤΕΙ Κρήτης που είχε ως σκοπό την χρησιμοποίηση ακόμα πιο εξελιγμένων τεχνικών στο επίπεδο της μεταφοράς μέσω διαδικτύου, που οδήγησε σε μία δημοσίευση/ανακοίνωση σε συνέδριο και για αυτό τον λόγο υπάρχουν κοινά τμήματα στην εργασία αυτή.

Η πτυχιακή αυτή εργασία επεκτείνεται στην υποστήριξη μοντέρνων τεχνικών φωτοσκίασης καθώς και στον αυτόματο εμπλουτισμό ενός ολόκληρου τρισδιάστατου κόσμου με μεγάλα σε μέγεθος αντικείμενα. Τέλος παρουσιάζονται τα συμπεράσματα πού βγήκαν και σε σύγκριση με τις τυπικές αλλά και πιο εξελιγμένες διαδικασίες μεταφοράς και προτείνονται πιθανές προοπτικές για την μελλοντική εξέλιξη της εργασίας.

Πίνακας Περιεχόμενων

Πίνακας εικόνων

Εισαγωγή

Η ολοένα και αυξανόμενη χρήση του Internet καθώς και η πρόοδος των χρησιμοποιούμενων τεχνολογιών τα τελευταία χρόνια μας έχουν επιτρέψει να μεταφέρουμε ολοένα και μεγαλύτερο μέρος παραδοσιακών εφαρμογών που χρησιμοποιούνταν αποκλειστικά σε “κλειστά” συστήματα και χωρίς την δυνατότητα της απρόσκοπτης λειτουργίας ανεξάρτητα την πλατφόρμα ή το λειτουργικού συστήματος που χρησιμοποιούμε .

Ένας τομές τέτοιων εφαρμογών που μόλις πρόσφατα έχει μεταπηδήσει σε τεχνολογία ιστού είναι και αυτός των τρισδιάστατων γραφικών από την στιγμή που έγινε εφικτή η διασύνδεση των καρτών γραφικών με τους φυλλομετρητές (Browsers) μέσω της βιβλιοθήκης και σετ εντολών της WebGL.

Παρόλες τις προόδους που πραγματοποιούνται υπάρχει μία βασική δυσκολία ,οι τρισδιάστατες εφαρμογές είναι από τις πιο “ακριβές” υπολογιστικά και χαρακτηρίζονται από μεγάλο όγκο δεδομένων ο οποίος πρέπει να μεταφερθεί στον τελικό χρήστη απρόσκοπτα , με διαφανή διαδικασία , και σε επιτρεπτά χρονικά όρια.

Υπάρχουν αρκετές προσεγγίσεις ώστε να επιτευχθεί αυτό το αποτέλεσμα και όμως πραγματικό Streaming τρισδιάστατων αντικειμένων και εν συνεχεία ολόκληρων σκηνών και ου το καθεξής χώρις να γίνει η διαδικασία δυσάρεστη για τον απλό χρήστη είναι ελάχιστες

Οι προσεγγίσεις αυτές βασίζονται στην μεταφορά ενός αρχικά πολύ βασικού αντικειμένου και εν συνεχεία όσο καταφθάνουν και αποκωδικοποιούνται δεδομένα προστίθενται στο αντικείμενο αυξάνοντας την ποιότητα της απεικόνισης έως ότου στο τέλος θα έχουμε το αρχικό αντικείμενο με την πλήρη λεπτομέρεια του καλύπτοντας έτσι τα κενά άπω τις καθυστερήσεις που προκύπτουν λόγω δικτύου και επεξεργασίας των δεδομένων, ο χρήστης μπορεί να βλέπει καθόλη την διάρκεια της μεταφοράς τα κομμάτια του μοντέλου και μπορεί να τα περιεργάζεται χωρίς να βλέπει κενές οθόνες ή loading bars ,και έτσι είναι λιγότερο πιθανό να αποφασίσει να σταματήσει την όλη διαδικασία ή ακόμα χειρότερα να την αρχίσει από την αρχή με refresh.

Σε γενικές γραμμές οι παραπάνω προσεγγίσεις βασίζονται στην προ- επεξεργασία των εν λόγω τρισδιάστατων μοντέλων παράγοντας τα ενδιάμεσα επίπεδα λεπτομέρειας, Level Of Detail LOD και στην συνέχεια αποστέλλονται σειριακά από την γενική εικόνα προς την πλήρη.

1.1 Περίληψη

Ο σκοπός αυτής της πτυχιακής εργασίας ήταν η μελέτη ή σχεδίαση και ή ανάπτυξη ενός αλγορίθμου για την επιτόπου επεξεργασία και μεταφορά τρισδιάστατων μοντέλων και σκηνών απρόσκοπτα στον χρήστη χρησιμοποιώντας τις τελευταίες διαθέσιμες τεχνολογικά λύσεις .

Ένα μεγάλο πρόβλημα που παρουσιάζεται βρίσκεται στο μέγεθός της πληροφορίας που πρέπει να μεταδοθεί , οι τρισδιάστατες γεωμετρίες χαρακτηρίζονται από πλήθος μη δομημένης πληροφορίας που είναι δύσκολο να κωδικοποιηθεί και να μεταφερθεί στο διαδίκτυο.

Εικόνα 1 Μη δομημένη πληροφορία σε ένα έγγραφο Html κατά το πρότυπο x3Dom.

Προς την επίτευξη αυτού του στόχου μελετήθηκαν οι διαθέσιμοι ανοιχτοί αλγόριθμοι τμηματοποίησης μεγάλου όγκου δεδομένων , οι διαθέσιμες τεχνολογίες μεταφοράς στο διαδίκτυο, οι διαθέσιμες λύσεις συμπίεσης δεδομένων ,οι διαφορετικές λύσεις σε επίπεδο διακομιστών και οι τεχνολογίες που υποστηρίζονται από αυτούς τέλος διερευνήθηκαν τα τεχνολογικά όρια των παρόντων browser σε συνδυασμό με όλα τα παραπάνω και το επίπεδο υλικού και ταχύτητας μετάδοσης δεδομένων.

Πιο συγκεκριμένα σε πρώιμα στάδια χρησιμοποιήθηκε διακομιστής βασισμένος στην πλατφόρμα Node.js (JavaScript) και μεταφορά βασισμένη σε Websocket ένω σε τελικά στάδια χρησιμοποιήθηκε το Spring Framework με μεταφορά χρησιμοποιώντας HTTP GET ώστε να γίνει ή ευκολότερη ενσωμάτωση με τα ευρέως διαδεδομένα Standard στην βιομηχανία Java - tomcat Server , ως τεχνολογία τρισδιάστατης απεικόνισης χρησιμοποιήθηκε το αναπτυσσόμενο πρότυπο X3DOM το όπιο συνδυάζει δύο παλαιότερες τεχνολογίες την OpenGL και τις X3D -VRML συγκεντρωτικά έχουμε Java ,JavaScript, X3D ,X3dom ,JSON ,Html5, xhtml.

Η πλατφόρμα αναπτύχθηκε με βασική επιδίωξη την όσο το δυνατόν απρόσκοπτη παρουσίαση στον τελικό χρήστη της σκηνής που μας ζητάει με διάφανες σε αυτόν διαδικασίες χρησιμοποιώντας στον διακομιστή επί τόπου επεξεργασία προγενεστέρων μοντέλων επιτρέποντας έτσι την επαναχρησιμοποίηση των εν λόγω μοντέλων, με αυτόν τον τρόπο παρέχεται και στον δημιουργό μια ευελιξία στην ανάπτυξη μίας σκηνής, επιπλέον έχει ληφθεί μια βασική μέριμνα έτσι ώστε να απαλλαχτεί και ο δημιουργός από περιττές και κοπιαστικές προγραμματιστικές διαδικασίες.

1.2 Κίνητρο για την διεξαγωγή της Εργασίας

Η παρουσίαση τρισδιάστατων γραφικών χωρίς την εγκατάσταση ειδικού λογισμικού σε έναν browser είναι ένα ζητούμενο τής βιομηχανίας τόσο παλιό όσο το διαδίκτυο ,είκοσι χρόνια πριν ήταν μόνο επιστημονική φαντασία σε διάφορες ταινίες του κινηματογράφου, στα πρόσφατα χρόνια μπορούσαμε να επιτύχουμε ψευδό-τρισδιάστατα γραφικά με την χρησιμοποίηση τρίτων εφαρμογών plug-in σε browser, εργαλεία όπως το X3D ή το Adobe Flash , πρόσφατα με την εισαγωγή τής WebGL JavaScript API μπορούμε να προβάλουμε τρισδιάστατα γραφικά σε browser ενδημικά χρησιμοποιώντας την κάρτα γραφικών απευθείας και χωρίς την παρεμβολή του επεξεργαστή επιτρέποντας έτσι την αναπαράσταση ακόμα ρεαλιστικότερων αναπαραστάσεων.

Με την παράλληλη επιτάχυνση της ανάπτυξης πρότυπων όπως το X3DOM και το XML3D λόγω τής εισαγωγής τής WebGL, τα οποία παρέχουν μια δομημένη μορφή της πληροφορίας σύμφωνα με το πρότυπο XML μπορούμε να παρουσιάσουμε τρισδιάστατα γραφικά σε μία ευρεία γκάμα συσκευών όχι μόνο προσωπικών υπολογιστών , συσκευές όπως Smartphones, tablets, Smart TV , και γενικότερα σε όσες συσκευές διαθέτουν τους κατάλληλους φυλλομετρητές και την υπολογιστική ισχύ.

Το X3Dom πρότυπο μας παρέχει την βασική XML δομή του εγγράφου που φέρει την πληροφορία και με την επεξεργασία μέσω JavaScript μπορούν να λειτουργήσουν εντός όλων των μοντέρνων browser, κάτι το οποίο θέλουμε να διατηρήσουμε και για την αναπαράσταση τεράστιου όγκου δεδομένων σε πολυπλοκότερες σκηνές.

1.3 Σκοπός και στόχοι εργασίας

Ο σκοπός την πτυχιακής εργασίας είναι η επίτευξη της επεξεργασίας μίας τρισδιάστατης γεωμετρίας σε πραγματικό χρόνο ΄ώστε αυτή να μεταφερθεί προοδευτικά με την χρήση μοντέρνων τεχνικών μεταφοράς και την αναπαράσταση πολύπλοκων τρισδιάστατων μοντέλων και σκηνών σε διαδικτυακό περιβάλλον χρησιμοποιώντας τις τελευταίες διαθέσιμες τεχνολογίες.

1.4 Δομή Εργασίας

- Στο δεύτερο κεφάλαιο αναλύεται η μεθοδολογία της υλοποίησης καθώς και ο τρόπος που θα αναπτυχθεί η πτυχιακή εργασία και ποιοι αλγόριθμοι θα χρησιμοποιηθούν.
- Στο τρίτο κεφάλαιο θα εκπονηθεί το σχέδιο δράσης της πτυχιακής εργασίας και η έρευνα State of The Art.
- Στο τέταρτο κεφάλαιο γίνεται η αναλυτική ανάλυση του προβλήματος, θα εκπονηθούν οι απαιτήσεις του συστήματος και θα σχεδιαστεί η υλοποίηση. Εν συνεχεία θα ακολουθήσει ή υλοποίηση που έχει σχεδιαστεί.
- Στο πέμπτο κεφάλαιο ακολουθούν τα συμπεράσματα της εργασίας καθώς και προτάσεις για την μελλοντική επέκταση.
- Στο έκτο κεφάλαιο απαριθμούνται οι βιβλιογραφικές πήγες που έχουν χρησιμοποιηθεί.

2 Μεθοδολογία υλοποίησης

Θεωρούμε ότι δεδομένο ότι ή χρήση των γλωσσών HTML5 JavaScript και java είναι επιβεβλημένη σύμφωνα με τα τελευταία τεχνολογικά πρότυπα στον τομέα των Web εφαρμογών επιπλέον η χρήση των προτύπων WebGL X3D και X3Dom εν συνεχεία καλύπτουν τις βασικές ανάγκες για την απεικόνιση τρισδιάστατων γραφικών επιπλέον το πρωτόκολλο http καλύπτει τις ανάγκες για την μεταφορά δεδομένων μέσω διαδικτύου.

Θα διερευνηθούν όμως και άλλες αναπτυσσόμενες τεχνολογίες όπως το πρωτόκολλο web socket καθώς και οι δυνατότητες αποτελεσματικής αποκλειστικής χρήσης JavaScript και στο κομμάτι του server ενώ θα εξεταστούν και διάφοροι αλγόριθμοι συμπίεσης .

Θεωρούμε ότι ο αναγνώστης έχει μια βασική ιδέα για τον τρόπο λειτουργίας των αναφερόμενων τεχνολογιών καθώς και τις θεμελιώδεις αρχές λειτουργίας των τρισδιάστατων γραφικών.

Το αντικείμενο της πτυχιακής αυτής εργασίας αποτελεί ανοικτό πεδίο ερευνάς από πλήθος ερευνητικών ιδρυμάτων, Πανεπιστημίων, ανοικτών consortium εταιρειών που αποτελούνται από πολυπληθής ομάδες έμπειρων επιστημόνων με χρόνια εμπειρία σε αυτόν τον τομέα, Ο συγγραφέας αποδέχεται ότι οι όποιες γνώσεις έχει αποκτήσει κατά την διάρκεια των σπουδών του δεν επαρκούν για την δημιουργία πολύ εξεζητημένων τεχνικών από μηδενικό σημείο, ελπίζει μόνο ή όρεξη για μάθηση επάνω σε αυτό το αντικείμενο με αυτήν την εργασία να θέσει γερά θεμέλια για την μετέπειτα εξέλιξη του.

2.1 Μέθοδος ανάλυσης & ανάπτυξης πτυχιακής

Σε πρώτο στάδιο θα αναπτυχθεί ένας web Server βασισμένος στην πλατφόρμα node.js υλοποιημένος όμως ώστε δομικό του στοιχείο να είναι η μεταφορά μέσω web socket πληροφορίας χρησιμοποιώντας ένα module, το binary.js (Binary.js n.d.) όπου υποστηρίζει την αυτόματη τμηματοποίηση σε επίπεδο δικτύου μεγάλων αρχείων και την δημιουργία ενός stream 3d ,παράλληλα θα διερευνηθούν τα “όρια ανοχής” των browser στην εισαγωγή και αναπαράσταση μεγάλων γεωμετριών .

Θα εξεταστούν επίσης κατά πόσο είναι δυνατόν η επί τόπου επεξεργασία τής γεωμετρίας σε πραγματικό χρόνο εντός του εν λόγω διακομιστή για την αποστολή και κατά πόσο είναι δυνατόν να γίνει εύκολη ή διαδικασία για τον διαχειριστή ενός τέτοιου συστήματος.

Στην περίπτωση πού η αποκλειστική χρήση JavaScript και web socket στον server δεν καταστεί προτιμητέα λόγω πολυπλοκότητας ή και λόγω μη επίτευξης των προσδοκώμενων “διαφημιζομένων” αποτελεσμάτων θα αναπτυχθεί αντίστοιχη πλατφόρμα με την χρήση του Java Spring Framework ώστε να διασφαλιστεί ή καλύτερη ενσωμάτωση με τα βιομηχανικά standard με την χρήση του παραδοσιακότερο πρωτοκόλλου μεταφοράς HTTP.

Για τον αλγόριθμό τμηματοποίησης τής γεωμετρίας αρχικά θα αναπτυχθεί μία σειριακή προσέγγιση κομματιάσματος σε αυτοδύναμα πακέτα ώστε να καλύπτει τις βασικές ανάγκες έλεγχού και εν συνεχεία θα διερευνηθεί η δυνατότητα κατασκευής πιο εξελιγμένων τεχνικών.

Στο επίπεδό τής προβολής στον browser είναι προτιμητέο να κινηθούμε ανεξάρτητα από τα επίπεδα ολοκλήρωσης του framework x3Dom μίας και το τελευταίο βρίσκεται ακόμα σε πειραματικό στάδιο και αλλαγές στην λειτουργία του μπορούν να επηρεάσουν την έκβαση τής εργασίας, οπότε θα αρκεστούμε στην τοποθέτηση της πληροφορίας εντός του HTML document με κατάλληλο τρόπο ώστε μετά να αναλάβει ανεξάρτητα το framework την αναπαράσταση και τις λειτουργίες τής WebGL με την όσο το δυνατόν μικρότερη παρέμβαση από μέρους μας.

Για την συμπίεση θα διερευνηθούν οι διαθέσιμοι standard αλγόριθμοι και θα υλοποιηθεί ή ενσωμάτωση αυτών .

2.1.1 Τεχνικές.

Κατά την διάρκεια της εργασίας θα εξεταστούν οι κύριες τεχνικές 3D stream

- Progressive mesh encoding.
- Progressive compression.
- Sequencal Image Geometry
- Rapid Progressive Clustering

2.1.2 Μοντέλα 3D Γεωμετρίας.

Στα πλαίσια της εργασίας θα χρησιμοποιηθούν τρισδιάστατά μοντέλα γεωμετρίας που έχουν κατασκευαστεί στο εργαστήριο πολυμέσων MCLAB του τμήματος μηχανικών Πληροφορικής του ΤΕΙ Κρήτης, τα μοντέλα αυτά προέρχονται από τρισδιάστατα scanners εσωτερικών και εξωτερικών χώρων και απεικονίζουν μία εκκλησιά, και διάφορα άλλα αντικείμενα πολιτιστικού ενδιαφέροντας όπως αμφορείς και μουσικά όργανα που διαθέτουν πληροφορίες γεωμετρίας και color άνα σημείο , χρησιμοποιήθηκε επίσης ένα μοντέλο ύδρας που έχει κατασκευάσει με την ενσωμάτωση μοντέρνων τεχνικών shading όπως Normal Mapping, τα μοντέλα αυτά έχουν κωδικοποιηθεί κατά το declarative πρότυπό x3Dom.

Εικόνα 2 Τα τρισδιάστατά μοντέλα που θα χρησιμοποιήσουμε.

3 Σχέδιο δράσης για την εκπόνηση της εργασίας

Το κεφάλαιο αυτό ασχολείται με την βιβλιογραφική αναζήτηση τής τεχνολογίας αιχμής (state of the Art) και την υλοποίηση του σχεδίου δράσης για την εκπόνηση της πτυχιακής εργασίας .

3.1 State of the Art

Το προοδευτικό πλέγμα (Progressive mesh) εισάχθηκε ως έννοια από τον (Hoppe H. 1996)που πρότεινε την τεχνική της προ επεξεργασίας μιας τρισδιάστατης γεωμετρίας όταν θελήσουμε να την μεταδώσουμε μέσω διαδικτύου, σε αυτήν την προ επεξεργασία κατασκευάζονται προοδευτικά μία σειρά από απλούστερες γεωμετρίες όπου περιέχουν λιγότερη πληροφορία, όταν ένας χρήστης ζητήσει να λάβει την εν λόγο γεωμετρία, κατ' αρχήν του αποστέλλεται ή γεωμετρία χαμηλότερης ποιότητας – ανάλυσης Level Of Detail ώστε να μπορέσει αυτός να αρχίσει να την βλέπει γρήγορα, όσο του αποστέλνονται προοδευτικά οι επόμενες γεωμετρίες με υψηλότερη ακρίβεια μέχρι να φτάσουμε στην πλήρη γεωμετρία.

Η προοδευτική συμπίεση (progressive compression) που εισάχθηκε ως πρακτική πάλι από τον HOPPE βασίζεται στην εσωτερική αναδιάταξη των σημείων ενός επίπεδου λεπτομέρειας και την συσχέτιση τους με τις μεταβολές που τους συμβαίνουν σε ένα επόμενο επίπεδο λεπτομέρειας μίας γεωμετρίας, έτσι δεν χρειάζεται να αποθηκευτούν όλα τα στιγμιότυπα που έχουν επεξεργαστείτε άλλα ή γεωμετρία της χαμηλότερης ανάλυσης κράτα όλες τις αλλαγές στις συσχετίσεις που πρέπει να προκύψουν μειώνοντας έτσι κατά πολύ τον όγκο πού πρέπει να μεταδοθεί, επιπλέον και μπορεί μέσω αλγορίθμων να εισαχθεί και lossless συμπίεση για ακόμα καλύτερα αποτελέσματα όλη η λειτουργία της δημιουργίας λιγότερο λεπτομερών στιγμιότυπων βασίζεται στην κατάρρευση των άκμων μιας γεωμετρίας προοδευτικά, κατά την διάρκεια της ανακατασκευής συμβαίνει η αντίστροφη διαδικασία. Επάνω στην εργασία του HOPPE έχουν αναπτυχθεί πολλαπλές παραλλαγές και υλοποιήσεις, με ποιο πρόσφατες τίς:

(G. Lavoue 2013)όπου πραγματεύεται την επέκταση του progressive compression και στις νεότερες Web 3D τεχνολογίες μέσω της WebGL, η εργασία αυτή μάλιστα υπόσχεται πολύ γρήγορα οπτικά αποτελέσματα με αρκετά μεγάλη ακρίβεια στα αρχικά στάδια μετάδοσης, χρησιμοποιώντας Ajax για την μετάδοση της πληροφορίας, ενώ υποστηρίζει και χρώμα ανά σημείο, σε αντίθεση με προηγούμενες τεχνικές ένα μειονέκτημα της είναι η υψηλή χρησιμοποίηση τής CPU κατά την διάρκεια της αποκωδικοποίησης καθώς και ή κλειστή φύση της υλοποίησης.

Η προσέγγιση των (M. Limper, Using images and explicit binary container for efficient and incremental delivery of declarative 3D scenes on the web 2012)που έχει ήδη ενσωματωθεί στο πρότυπό x3Dom προτείνει στην πραγματικότητα τρεις προσεγγίσεις τίς: Binary Geometry (BG), Sequential Image Geometry (SIG),Binary Level of detail (BLD), κατά την binary Geometry κωδικοποιούνται οι γεωμετρίες σε δυαδικούς πίνακες typed array (KHRONOS 2012) αντί για απλό κείμενο που χρησιμοποιείται για την αποθήκευση στο x3Dom μειώνοντας κατά ένα μεγάλο βαθμό το μέγεθος τους.

Στην SIG προσέγγιση χρησιμοποιείται μια σειρά από εικόνες για την αποθήκευση της πληροφορίας οι οποίες είναι συμπιεσμένες με lossless αλγόριθμο png χρησιμοποιώντας την υπάρχουσα υποδομή και διαμόρφωση στους διακομιστές που είναι optimized για την αποστολή εικόνων, ενώ στην τρίτη προσέγγιση έχουμε αντίστοιχα μια πληροφορία κωδικοποιημένη δυαδικά σε

πακέτα που κάθε πακέτο περιέχει περισσότερα αρχεία με ολοένα καλύτερη λεπτομέρεια της γεωμετρίας.

Μία άλλη πρόσφατη εργασία επάνω στο θέμα είναι η (M. Limper, The POP buffer: Rapid progressive clustering by geometry quantization, 2013) που πραγματεύεται την πραγματοποίηση του σταδίου της αποκωδικοποίησης κατευθείαν στην GPU γλυτώνοντας έτσι πολύτιμους υπολογιστικούς πόρους, βασίζεται σε μια ιεράρχηση και κβαντοποίηση του πλήθους των τρίγωνων σε πολλαπλά επίπεδα λεπτομέρειας που κάποιο περιέχει τα επόμενα.

Η μεταφορά του σετ εντολών της OpenGL και η διασύνδεση με την JavaScript έχει επιτρέψει την απρόσκοπτη λειτουργία τρισδιάστατων εξελιγμένων γραφικών σε όλους τους μοντέρνους browser η OpenGL υποστηρίζεται από σχεδόν όλο το πλήθος των καρτών γραφικών που υπάρχουν σήμερα διαθέσιμες αλλά είναι και προς τα πίσω συμβατή σε προηγούμενες εκδώσεις της OpenGL καθώς και κάρτες γραφικών, εισάχθηκε το 1992 και πλέον είναι ένα Standard τής βιομηχανίας για την αναπαράσταση 2D και 3D γραφικών , πλέον όσο γίνεται καλύτερη και η υποστήριξη από τους browser τείνει να αντικαταστήσει άλλες ξεπερασμένες τεχνολογικά λύσεις όπως το flash και shockwave και στην περιοχή του Web3D ή περαιτέρω ανάπτυξη καθορίζεται άπο το Khronos Group (Khronos n.d.) ένα μη κερδοσκοπικό consortium εταιρειών με σκοπό την προτυποποίηση και την περαιτέρω ανάπτυξη της OpenGL.

Η WebGL, η παραλλαγή για την χρήση στο Web βασίζεται σε JavaScript το οποίο αναλαμβάνει να προετοιμάζει τα δεδομένα πού θα στείλουμε κατευθείαν στην κάρτα γραφικών προς επεξεργασία κάνοντας την πολύ πιο γρήγορα από ότι μία CPU μίας και οι GPU είναι αποκλειστικά φτιαγμένες για τον σκοπό αυτόν, αφήνοντας και πόρους στον επεξεργαστή για άλλες εργασίες εδώ βρίσκεται και το κύριο ατού σε σχέση με την Flash και άλλα παλαιότερα πρότυπα που πρέπει να πρόεπεξεργαστουν στην CPU πριν αναλάβει το σύστημα γραφικών ,το επιπλέον ατού βρίσκεται στην γλώσσα που χρησιμοποιούμε, την JavaScript οπού μπορεί υπό προϋποθέσεις να τρέξει σε οποιονδήποτε browser χωρίς καμιά απολύτως εγκατάσταση κάπου player ή plugin το οποίο είναι και πολύ επιθυμητό στα πλαίσια της εργασίας

Το X3D είναι μία μορφή αρχείου και εκτελέσιμη αρχιτεκτονική το όποιο είναι ανοιχτού προτύπου και χρησιμοποιείται για την αναπαράσταση άλλα και την μετάδοση τρισδιάστατου περιεχομένου χρησιμοποιώντας XML και είναι απόγονος της Virtual Reality Modeling Language (VRML), υποστηρίζει όλες τις λειτουργίες τού XML ,2D και 3D αναπαράσταση, Animation, τοποθέτηση ήχου και video εντός αντικείμενων, events, είναι δυνατόν με την χρησιμοποίηση κάπου player να αναπαρασταθούν σκηνές βασισμένες σε X3D και εντός browser ενώ υποστηρίζεται από το μη κερδοσκοπικό (WEB3D CONSORTIUM n.d.).

Επάνω στην WebGL βασίζεται το Framework X3Dom (BEHR J. 2009) αναφέρεται και ως X-Freedom είναι μία προσπάθεια του ινστιτούτου Fraunhofer όπου συνδυάζει την WebGL και το X3D δημιουργώντας έτσι μία γέφυρα ανάμεσα στις δυο αυτές τεχνολογίες απεικόνισης ,χρησιμοποιώντας όλα τα τελευταία πρότυπά στον τομέα των ιστοσελίδων HTML5 (HICKSON 2012), CSS, τοποθετώντας X3D το οποίο είναι XML στην βασική δόμη DOM μίας ιστοσελίδας και αναλαμβάνει άπο εκεί ή JavaScript οπού διαβάζει το έγγραφο και διασυνδέει μέσω WebGL ώστε να επιτύχει την αναπαράσταση τής σκηνής μας, στη παρούσα φάση τού βρίσκεται ακόμα υπό ανάπτυξη και δεν υποστηρίζει όλες τις λειτουργίες του X3D ενώ έχει να αντιμετωπίσει και κάποιους από τους περιορισμούς της OpenGL.

Το 2011 προτυποποιήθηκε το πρωτόκολλο επικοινωνίας των websocket, σε αντίθεση με την κλασσική επικοινωνία στο INTERNET μέσω του πρωτοκόλλου http , υποστηρίζει αμφίπλευρη (full-duplex) επικοινωνία μεταξύ δυο μερών, είναι σχεδιασμένο για επικοινωνία μεταξύ client - Server βασισμένο στο TCP και η μόνη του σύνδεση με το http είναι το αρχικό handshake και μετά γίνεται upgrade σε tcp παρέχοντας έτσι την δυνατότητα να μεταφέρουμε με οριοθετημένο τρόπο δεδομένα κρατώντας το ίδιο κανάλι επικοινωνίας ανοιχτό, ενώ χρησιμοποιεί τις Standard ports επικοινωνίας (80) μας επιτρέπεται να δημιουργήσουμε συνδέσεις χωρίς να λαμβάνουμε υπόψιν firewall και υποστηρίζεται και από όλους τους μοντέρνους browser και διαθέτει υποστήριξη κρυπτογραφημένης σύνδεσης.

Το 2011 επίσης ή Google εισήγαγε το WebRTC (WebRTC.com 2011) api που υποστηρίζει μεταφορά βίντεο, ήχου, real time Video conference, chat, και P2P file sharing, στον browser χωρίς την χρήση plugins παρόλο που για P2P εφαρμογές θέμα που αρχικά δεν ενδιαφέρει την εργασία αυτή σε πρωταρχικά στάδια μπορεί σε βάθος χρόνου να συν εφαρμοστεί για κοινά περιβάλλοντα θέασης.

Το Representational State Transfer (REST) είναι ένας τρόπος, μια αρχιτεκτονική λογισμικού ώστε να δημιουργηθεί, σβηστεί, ανανεωθεί και διαβαστεί πληροφορία σε έναν server χρησιμοποιώντας απλές HTTP κλήσεις. Εισάχθηκε το 2000 στην διδακτορική διατριβή του (Fielding 2000) και αποτελείται από Components Connectors και Data περιορισμένα στις σχέσεις τους ώστε να επιτευχθεί ο στόχος της αρχιτεκτονικής. Το βασικό του πλεονέκτημα είναι η απλότητα του σε σχέση με άλλες υλοποιήσεις κατανεμημένων συστημάτων καθώς και ή ευκολία στην επέκταση.

Το Spring Framework (Spring n.d.) παρέχει στους προγραμματιστές τα κατάλληλα εργαλεία ώστε αυτοί να κατασκευάσουν γρήγορα ένα μεγάλο πλήθος εφαρμογών χρησιμοποιώντας JAVA και το Eclipse IDE (Eclipse n.d.) ,σε αυτές τις εφαρμογές συμπεριλαμβάνονται και διάφορα webService restful , SOAP , καθώς παρέχονται και websocket υλοποιήσεις , ενώ μπορούν να αποσταλούν δεδομένα μέσω των υπαρχόντων προτύπων XML και JSON, εδώ αξίζει να σημειωθεί ότι μέσω Java μπορούμε να χρησιμοποιήσουμε δύο ειδών υλοποιήσεις ,την κλασσική όπου αποστέλλουμε στον client το βασικό HTML και τα JavaScript - css και αναλαμβάνει αυτός να τα επεξεργαστεί δήλ. να εκτελέσει κώδικα και να τα αναπαραστήσει, και ή δεύτερη περιλαμβάνει όλη την επεξεργασία στον Server και την Αποστόλη μόνο του τελικού αποτελέσματος.

Τέλος υπάρχει και το node.js (node.js n.d.) το οποίο είναι μία πλατφόρμα βασισμένη στην runtime JavaScript μηχανή που χρησιμοποιεί ο Chrome της Google χρησιμοποιείται στην μεριά του server για την ανάπτυξη εφαρμογών δικτύου και χρησιμοποιεί αποκλειστικά JavaScript και εκμεταλλεύεται τα event που μπορούν να παραχθούν ενώ είναι και non-blocking όπου περιμένουμε I/O και είναι ιδανική για ανάπτυξη εφαρμογών σε κατανεμημένα συστήματα.

Είναι σύνηθες να χρησιμοποιείται ώς middleware σε μεγάλα συστήματα για να διαχειρίζεται τα data request αλλά μπορεί να σταθεί από μόνος του και ως Webserver με βασικό πλεονέκτημα το ότι χρησιμοποιούμε παντού JavaScript στον server και τον client. Είναι απόλυτα επεκτάσιμη καθώς επιδέχεται δικές μας εφαρμογές JavaScript ως modules και διαθέτει μία μεγάλη και ενεργή κοινότητα που την υποστηρίζει.

Πρωτόκολλο - Τεχνική	Υπέρ	Κατά
HTTP – Ajax-rest	Εύχρηστο και αξιόπιστο, stateless,υποστήριξη native συμπίεσης από τους browsers, backwards compatible	Υψηλό υπολογιστικό κόστος για πολλαπλές κλήσεις, single channel
web socket	Full -duplexed κανάλι επικοινωνίας όπως το tcp, χαμηλότερο υπολογιστικό κόστος, μεταφοράς text ή byte	Δεν υπάρχει υποστήριξή σε παλαιότερους browser, πρέπει να υλοποιηθεί συμπίεση από τον κατασκευαστή.
WebRTC	P2P orientated, supports both TCP and UDP like, lower latency	P2P, δεν υπάρχει υποστήριξη από όλους τους browser

Πίνακας 1: Σύγκριση τεχνικών μεταφοράς

Αλγόριθμος Streaming γεωμετρίας	Τύπος	Υπέρ	Κατά
Lavue 2013	Progressive Compression	Fast decoding	JavaScript decoding – high CPU usage, Render through WebGL, not publicly available, geometry needs to be preprocessed.
Behr 2012 - BG	Data encoding	Typed array /GPU mapping, No JavaScript decoding, Supports Quantization X3Dom integration	Binary only Nneeds prepossessing
Behr 2012 - SIG	Data encoding	Same as BG but with use of multiple images as a medium, Servers are optimized for image transmission	Needs prepossessing
Behr 2012 - BLD	Progressive Compression	Same as BG	Binary, needs prepossessing
Behr 2013 – POP buffer	Progressive Compression	Novel approach Nested & quantized Level of detail	No support for animations, needs prepossessing.

Πίνακας 2: Σύγχρονοι Αλγόριθμοι streaming 3D

3.2 Σημαντικοί στόχοι για την ολοκλήρωση της πτυχιακής.

- Ολοκλήρωση της έρευνας State of the art.
- Ολοκλήρωση της ανάλυσης του προβλήματος.
- Ολοκλήρωση του σχεδιασμού ανάπτυξης της πτυχιακής.
- Υλοποίηση της πρώτης υλοποίησης του τεχνικού μέρους της πτυχιακής εργασίας.
- Υλοποίηση της δεύτερης υλοποίησης του τεχνικού μέρους της πτυχιακής εργασίας.
- Ολοκλήρωση της υλοποίησης του τεχνικού μέρους της εργασίας.
- Έλεγχος της λειτουργίας του τεχνικού μέρους της υλοποίησης.
- Συγγραφή της Αναφοράς Εργασίας.
- Υποβολή αίτησης αξιολόγησης εργασίας.
- Προετοιμασία παρουσίασης αναφοράς.
- Παρουσίαση αναφοράς.

3.2.1 Προτεινόμενο χρονοδιάγραμμά.

4 Κύριο μέρος πτυχιακής

Το κεφάλαιο αυτό ασχολείται με το κύριο μέρος της πτυχιακής δηλαδή με την ανάλυση του προβλήματος την σχεδίαση της λύσης και την υλοποίηση της.

4.1 Ανάλυση προβλήματος

Η μεταφορά και αναπαράσταση τρισδιάστατων γεωμετριών μεγάλου όγκου στο διαδίκτυο και η προβολή αυτών σε μορφή stream είναι το πεδίο χρόνιων ερευνών μεταξύ των πρωτοπόρων τού ίντερνετ και των τρισδιάστατων γραφικών.

Χρειαζόμαστε μία αξιόπιστη μέθοδο ώστε να μεταφέρουμε τμηματικά μεγάλες γεωμετρίες και ολόκληρες σκηνές με διαφανή ως προς τον χρήστη τρόπο μετάδοσης όπου θα ελαχιστοποιεί την αναμονή του πρώτου , ενώ θα πρέπει να διασφαλιστεί και η ευκολία στην διαχείριση ενός τέτοιου συστήματος από τους κατασκευαστές ή διαχειριστές.

Επιπλέον λόγω των διαθέσιμων σε εμάς σήμερα αλγορίθμων 3D stream πρέπει να προσθέσουμε στην επιστήμη μία διαφορετικής λογικής μέθοδο από τις προ υπάρχουσες, τέλος πρέπει να απαντήσουμε με ποιου είδους δεδομένα θα δουλέψουμε textual ή binary, Θα προσθέσουμε μία ακόμα τεχνική παραγωγής progressive Compression ή data encoding και αν θα προσπαθήσουμε να επιτρέψουμε επεξεργασία σε πραγματικό χρόνο.

4.1.2 Απαιτήσεις συστήματος

1. Απαιτήσεις του τελικού χρήστη – θεατή
    - Γρήγορη μεταφορά .
    - Αξιοπιστία μεταφοράς .
    - Γρήγορη επίτευξη αρχικής αναπαράστασης σκηνής .
    - Διαφάνεια ως προς τις διαδικασίες του συστήματος .
    - Ανεκτός χρόνος πλήρης αναπαράστασης της σκηνής .
    - Ενδιαφέρον οπτικά αποτελέσματα χωρίς ασυνέχειες .
    - System – Browser agnostic
2. Απαιτήσεις διαχειριστή
    - Ευελιξία στην ενσωμάτωση σε υπάρχουσες υποδομές διακομιστών.
    - Δυνατότητα επαναχρησιμοποίησης υπαρχόντων σκηνών.
    - Μελλοντικά επεκτάσιμη .
    - Μικρή κατανάλωση πόρων συστήματος .

4.2.1 Σχεδιασμός Υλοποίησης με την χρήση node.js

Έχοντας υπόψιν τους διαθέσιμους σήμερα αλγορίθμους 3D stream παρατηρούμε ότι υπάρχει ένα κενό στην μετάδοση textual πληροφορίας οι περισσότερες λύσεις που έχουν αναπτυχθεί επικεντρώνονται στην καλύτερη μετάδοση binary πληροφορίας μέσω typed arrays και την απόδοση της στην GPU προς αποκωδικοποίηση, σχεδόν σε όλες τις περιπτώσεις αν όχι σε όλες η τρισδιάστατη γεωμετρία πρέπει να επεξεργαστεί καταλλήλως και να αποθηκευτεί στον διακομιστή.

Θέλουμε να απαλείψουμε το τελευταίο αυτό στάδιο όσο είναι δυνατόν ώστε να προσφέρουμε μία καινούρια τεχνική με δυνατότητες real time επεξεργασίας ενός κανονικού αρχείου που κρατάει text 3d πληροφορίας, υπάρχουν αρκετά τέτοια format όπως το x3d και το x3dom στην web μορφή, JSON ή BSON για WebGL Loader.

Απατώντας και στο ερώτημα αν θα χρησιμοποιήσουμε το Framework x3dom ή την WebGL καθορίζει και την διαδικασία που πρέπει να κινηθούμε, χρησιμοποιώντας το x3dom framework ανοίγουμε τις προοπτικές της εργασίας εκθετικά αφού μας επιτρέπεται να δουλέψουμε σε ανώτερο επίπεδό πλεονέκτημα επίσης αποτελεί το υπόβαθρό που μας παρέχει το παλιότερο άλλα άμεσα συνδεδεμένο format X3D.

Η text πληροφορία σύμφωνα με τα specs που υπάρχει νόημα να επεξεργαστούμε βρίσκεται στα X3D nodes:

- <IndexedFaceSet>
- <Coordinate>
- <Color>
- <Normal>
- <TextureCoordinate>

καθώς και άλλα παραπλήσιου τύπου nodes όλα αυτά τα nodes , ανάλογα με τον τύπο της γεωμετρίας χαρακτηρίζονται από μεγάλο πλήθος τιμών.

Το Node <IndexedFaceSet> περιέχει τους απαραίτητους συσχετισμούς μεταξύ των σημείων μίας γεωμετρίας που χρειάζεται ή GPU ώστε να σχηματίσει τρίγωνα χρησιμοποιεί μόνο ακεραίους αριθμούς δημιουργούνται αντιστοιχίες με την σειρά σημείων που έχουμε δώσει πχ ένα στοιχείο του IndexedFaceSet με την τιμή 1 αναφέρεται στο πρώτο στοιχείο του Node Coordinate που θα βρει κ.ο.κ. σημείωση: όλα τα nodes που αναφέρονται σε γεωμετρία και τύπου Index είναι πιθανό να διαθέτουν το μεγαλύτερο πλήθος τιμών σε όλο το έγγραφο καθώς για να δημιουργηθούν τρίγωνα επαναχρησιμοποιούνται τα διάφορα <points> που διαθέτουμε. Για την δημιουργία ενός τριγώνου χρειαζόμαστε τρία σημεία τα οποία διαχωρίζονται από τα επόμενα με -1 .

<IndexedFaceSet coordIndex=”1 2 3 -1 2 3 4 -1”>

τα τρίγωνα αυτά αναφέρονται συχνά και ως επιφάνειες (Faces)

Το Node <Coordinate> περιγράφει με μία τριάδα float τις συντεταγμένες ενός σημείου στον χώρο και αυτά είναι το δομικό στοιχείο των τρισδιάστατων γραφικών, κάθε τριάδα διαχωρίζετε από την επόμενη της με ένα κόμμα ”,”. <Coordinate point=”0.1 0.2 0.3, 0.6,0.7 0.8,”> με τις τιμές να

αντιστοιχούν στους x,y,z άξονες συντεταγμένων και είναι “παιδί του IndexedFaceSet”.

Το Node <Color> περιγράφει και αυτό με μία τριάδα float μεταβλητών τις τιμές RGB για ένα point ακόμα και για ένα ολόκληρο face και ακολουθεί την ίδια δομή γραφής με τις συντεταγμένες ενός σημείου και είναι “παιδί του IndexedFaceSet”.

Το Node <Normal> περιγράφει το κάθετο διάνυσμα επάνω σε ένα ολόκληρο Face ή και αλλά αντίστοιχου είδους nodes και χρησιμοποιούνται στον υπολογισμό των αντανακλάσεων του φωτισμού εντός μιας σκηνής και αυτά περιγράφονται από τριάδες float και είναι “παιδί του IndexedFaceSet”.

<Normal vector=”'0.000000 0.441300 0.897300, 0.000000 -0.311800 0.950100,”>

Το Node <TextureCoordinate> χρησιμοποιείται για την αντιστοίχηση σετ δισδιάστατων συντεταγμένων σε Faces αυτού του είδους το Node χωρίζει τις τιμές του ανά 2 σετ τιμών καθώς χρειάζεται μόνο τις 2 κάτω γωνίες για την αντιστοίχηση του σε ένα face και είναι “παιδί του IndexedFaceSet”.

<TextureCoordinate point=”'0.296500 0.161300, 0.298700 0.154300,”>

Τα δεδομένα των παραπάνω nodes μας παρέχουν όλες τις βασικές δυνατότητες που χρειαζόμαστε για την αναπαράσταση ενός περίπλοκου γεωμετρικού σχήματος ,όλες πληροφορίες δομημένες κατά xml είναι παιδιά ενός <shape> που είναι με την σειρά του ένα παιδί του <transform> Node το οποίο είναι υπεύθυνο για την τοποθέτηση της γεωμετρίας μας σε σχέση με το κέντρο του κόσμου μας ή σε σχέση με το κέντρο του Πατρικού Node πολλά <shape> μπορούνε να ενωθούν μαζί υπό ένα <group> και αντιμετωπίζονται εξωτερικά ως ένα αντικείμενο για παράδειγμα μπορούμε να έχουμε τα λειτουργικά κομμάτια ενός αυτοκινήτου όλα ενωμένα υπό το <group DEF=”car1”>.

Έχοντας στοχεύσει από που θέλουμε να αντλήσουμε την πληροφορία εντός του X3Dom document μας πρώτο μέλημα μας είναι να δούμε με τι είδους διακομιστή θα δουλέψουμε, το πρωτόκολλό websocket παρουσιάζει εξαιρετικό ενδιαφέρον καθώς και η υλοποίηση του με την πλατφόρμα node.js στοχεύοντας αρχικά στην μετάδοση της μη δομημένης πληροφορίας που περιγραφικέ παραπάνω σε plaintext – String μορφή σε μια άδεια από γεωμετρία σελίδα x3dom.

Εφόσον επιτευχθεί αυτός ο στόχος λογικό βήμα είναι να βρούμε πως ή αποστολή της πληροφορίας μπορεί να βελτιωθεί, έχοντας επιλέξει να την μεταδώσουμε ως κείμενο αυτό υποθέτει ότι μπορούμε να αναπαράγουμε στον πελάτη μέρος της πληροφορίας χωρίς να παράγονται λάθη ,πρέπει δοκιμαστεί ένα είδος κατάτμησης για να εξεταστούν τα όρια της πλατφόρμας υπό αυτό το σενάριο.

Ανάλογα με τα συμπεράσματα που θα εξαχθούν μίας και το αντικείμενο ενδιαφέρει ερευνητικά το εργαστήριο Πολυμέσων του Τμήματος Μηχανικών πληροφορικής θα επανεξεταστεί από κοινού ή χρήση των πλατφορμών.

4.2.1 Σχεδιασμός υλοποίησης με την χρήση Spring Rest.

Σύμφωνα με τα συμπεράσματα της πρώτης υλοποίησης η χρήση της πλατφόρμας node.js δεν επαρκεί για τους στόχους που έχουμε θέσει χρειαζόμαστε ένα αποδοτικότερο σύστημα διακομιστή με μεγαλύτερες και δοκιμασμένες ικανότητες, τελευταία το framework Spring χρησιμοποιείται ευρέως για την ανάπτυξη robust web συστημάτων, αποφασιστικέ να συνεχίσει αυτή η εργασία με την ανάπτυξη σε αρχιτεκτονική Rest web service στο επίπεδο δικτύωσης ενώ θα αναπτυχθεί παράλληλα με τις ίδιες κατευθυντήριες γραμμές και framework από την ερευνητική ομάδα του εργαστηρίου πολυμέσων μια αρχιτεκτονική βασισμένη σε web sockets.

Ξεκινώντας θα υλοποιηθεί μία restful αρχιτεκτονική σύμφωνα τα παραδείγματα που παρέχει το framework αποστέλλοντας την βασική πληροφορία IndexedFaceSet,points,Normal και color μέσω ενός αντικείμενου JSON που στην τελική του μορφή θα περιέχει τις τιμές για τα Nodes που περιγράψαμε παραπάνω.

Θα δημιουργηθούν δύο βασικοί request handlers ο πρώτος θα ενεργοποιεί τον αναγνώστη εγγράφων και τον κατατμητή της γεωμετρίας που θα επιστρέφει το πλήθος των κομματιών και ένας δεύτερος για την κλήση των εν λόγω κομματιών από μία λίστα.

Σημαντικό ρόλο στην υλοποίηση κατέχει ο αναγνώστης XML εγγράφων μέσω αυτού θα είναι δυνατή ή εύρεση των τιμών των Nodes που μας ενδιαφέρουν, στην πρώτη υλοποίηση άπλα προσημειώσαμε τα αποτελέσματα του είναι εξαιρετικής σημασίας μία αποδοτική σε χρόνο λύση, και στην συνέχεια να συνδεθεί με έναν κατατμητής γεωμετρίας σταθερής απόδοσης που λειτουργεί σειριακά για αρχή με δοθέν μέγεθος κομματιών.

Χτίζοντας την εφαρμογή του χρήστη χρειαζόμαστε ένα minimal interface ώστε να παρέχουμε την δυνατότητα να εκκινήσει ο χρήστης την διαδικασία αφού έχει επιλέξει τις βασικές ρυθμίσεις που επιθυμεί καθώς και την δυνατότητα επιλογής από πλήθος γεωμετριών επιθυμητές λειτουργίες είναι η επιπλέον υποστήριξη συμπίεσης.

Με την ολοκλήρωση της βασικής εφαρμογής που θα υποστηρίζει color και normal ζητούμενο είναι η επέκταση της υποστήριξης για περισσότερους τύπους X3D Node άμεση προτεραιότητα αποτελεί η υποστήριξη των texture Node και ενδεχομένως διαφόρων μεθόδων shading βασισμένων σε αυτό όπως normal mapping , bump mapping.

Όπως και στην πρώτη υλοποίηση θα εγγράψουμε τα δεδομένα μας απείθειας στο DOM τις ιστοσελίδας άλλα όπως αποδείχτηκε στην πρώτη υλοποίηση η συχνές αχρείαστες αλλαγές στο DOM μπορούν αν επηρεάσουν σοβαρά τις επιδόσεις, τώρα θα γίνει μία προσπάθεια να εγγράφονται τα νέα στοιχειά σε ένα πέρασμα κάθε φορά που είναι ένα πακέτο έτοιμο να μπει στην σκηνή.

4.3 Υλοποίηση

Ακολουθεί η υλοποίηση της πρώτης πειραματικής εφαρμογής βασισμένης στην πλατφόρμα node.js με την δημιουργία ενός webserver πού θα έχει την κύρια ευθύνη διανομής τής στατικής πληροφορίας δηλαδή την αρχική σελίδα σε xhtml και τα αρχεία css και JavaScript που θα είναι υπεύθυνα για την λειτουργία της πλατφόρμας x3Dom.

4.3.1 Node web server

Στον webserver που δημιουργήσαμε θα επισυνάψουμε ένα Binary Server ο οποίος θα είναι υπεύθυνος για την εφαρμογή του πρωτοκόλλου websocket και την αποστολή των δεδομένων της γεωμετρίας.

Λόγω του περιορισμού τής JavaScript ή οποία τρέχει σε ένα νήμα πρέπει να χρησιμοποιήσουμε τον ορισμό event driven asynchronous callbacks μέσω event loop στην ουσία χρησιμοποιούμε functional programming και ανώνυμες functions εντός άλλων με την χρήση event για να μην περιμένουμε να ολοκληρωθούν αργές εργασίες πριν προχωρήσει περαιτέρω το βασικό μας πρόγραμμα βλ. περισσότερα (Kiessling 2013) The Node Beginner Book , www.node.js .

Ξεκινώντας χρειαζόμαστε την δημιουργία του βασικού Server για αυτόν τον σκοπό ενσωματώνουμε στο server.js δύο βασικά modules τής πλατφόρμας node.js , το http όπου μας προσφέρει τις βασικές λειτουργίες του πρωτοκόλλου http και μας επιτρέπει να αρχίσουμε τον server μας μέσω της εντολής http.createServer(onRequest).listen(80); και μάλιστα τού ορίζουμε και την πόρτα πού θα ακούει.

Το module “url” αναλαμβάνει με τις μεθόδους πού μας δίνει να μας δώσει τα διαφορετικά

```
var http = require('http');
var url = require("url");
function start(route, handle){
   function onRequest(req, res){
      var pathname=url.parse(req.url).pathname;
      //console.log("Request for " + pathname + " recived.");
      route (handle, pathname, res, req);
   }
   http.createServer(onRequest).listen(80);
   console.log("Server has started");
}
exports.start=start;
```

κομμάτια ενός url που λαμβάνει ώστε να προβούμε σε περαιτέρω επεξεργασία τού query string .

Όταν λαμβάνουμε οπουδήποτε http request πρέπει να απαντάμε και αναλόγως, αυτήν την ευθύνη την αναλαμβάνει ο router.js πού ανταπαντά με τα κατάλληλα http codes ένα δεν

```
function route(handle, pathname, res, req){
   //console.log("About to route a request for " + pathname);
   if (typeof handle[pathname] === 'function'){
      handle[pathname](res, req);
   }else {
      console.log("No request handler found for" +pathname);
      res.writeHead(404, {"Content-Type": "text/plain"});
      res.write("404 Not found");
      res.end();
   }
}
exports.route=route;
```

υποστηρίζεται το url μας ζήτησαν ή άμα υπάρχει ως συνάρτηση κάπου στον server μας την καλεί.

Οι συναρτήσεις αυτές περιλαμβάνονται στο request Handlers.js ,οπού από εδώ μπορούμε να αποστείλουμε μία συνάρτηση για κάθε διαφορετικό url που θα υποστηρίξουμε, για παράδειγμα η συνάρτηση file:

Αναλαμβάνει να διαβάσει μέσω του module fs ένα αρχείο , στην προκειμένη περίπτωση το 1.html εάν επιτύχει στο διάβασμά θα αποστείλει το αρχείο αυτό με την κατάλληλη κωδικοποίηση, αλλιώς θα γυρίσει πίσω μόνο ένα error code, και έτσι μπορούμε να αποστείλουμε στατικό περιεχόμενο .

```
var querystring=require("querystring"),
 fs = require("fs");

function bss(res ,req){
   fs.readFile("./public/1.html", "utf8", function(error, file){
     if (error){
         res.writeHead(500, {"Content-Type": "text/plain"});
         res.write(error + "\n");
         res.end();
      }else {
         res.writeHead(200, {"Content-Type": "utf8"});
         res.write(file, "utf8");
         res.end();
         console.log("sending static data");
      }} );
}
exports.bss=bss;
```

index.js

```
var server=require("./server");
var router=require("./router");
var requestHandlers=require("./requestHandlers");
var binaryserver=require("./binaryStreamServer");
var handle={}
handle["/"] = requestHandlers.bss;
handle["/bss"]=requestHandlers.bss;
handle["/favicon.ico"]=requestHandlers.favicon;
handle["/start"]= requestHandlers.x3d2;
handle["/x3d2"]=requestHandlers.x3d2;
handle["/box"]=requestHandlers.box;
handle["/x3dom.js"]=requestHandlers.x3domjs;
handle["/x3dom.css"]=requestHandlers.x3domcss;
handle["/binary.js"]=requestHandlers.binaryjs;
handle["/troulos"]=requestHandlers.troulos;

server.start(router.route, handle);
binaryserver.startb(router.route, handle);
```

Τέλος χρειαζόμαστε το index.js πού ενοποιεί τα παραπάνω αρχεία και τα χρησιμοποιεί ως εσωτερικά modules, αρχίζοντας μέσω της γραμμής εντολών το index με την εντολή: node index.js ο webserver βρίσκεται σε λειτουργία.

4.3.2 Binary Server

Ο υπό server που αναλαμβάνει την υλοποίηση του πρωτοκόλλου web socket είναι βασισμένος στο module binary.js http://binaryjs.com/ , ξεκινώντας τον βασικό μας server, παράλληλα ξεκινά και ο binary server ο οποίος όμως ακούει σε διαφορετικό port ώστε να μην έχουμε προβλήματα, Όταν έχουμε κλήση από τον client στην κατάλληλη πόρτα πραγματοποιείται μία “αναβάθμιση” του πρωτοκόλλου από http σε tcp και αφού έχει εγκαθιδρυθεί ή επικοινωνία, περιμένει event μέχρι να λάβει ορίσματα για τα δεδομένα που θα αποστείλει.

Εδώ έχει δημιουργηθεί ένα εσωτερικό πρωτόκολλό επικοινωνίας της εφαρμογής που ξεκινά με την κλήση από τον client τής εντολής: 'req IndexedFaceSet' , ο server αναλαμβάνει να “διαβάσει” από ένα άλλο JavaScript αρχείο το μέσω getter συναρτήσεων όλο το πλήθος των τιμών για χρώμα και την γεωμετρία των σημείων που είναι αποθηκεμένες σε μορφή string.

Και τα αποστέλλει εξ -ολόκληρου στον client ενώ διαβάζει όλο το πλήθος των τιμών τού Coordinate Index και το κομματιάζει σε ίσα μέρη σύμφωνα με ένα πλήθος που του έχουμε ορίσει προσέχοντας ώστε να γίνεται πάντοτε ακέραια διαίρεση με το 4, εν συνεχεία τα κομμάτια αυτά αποθηκεύονται σε ένα δυναμικού μεγέθους πίνακα και αποστέλλονται πίσω στον client .

Splitter function

Εδώ ολοκληρώνεται ή εργασία τού server .

4.3.3 Binary Client

Όπως έχει προαναφερθεί ό server μας δίνει στατικό περιεχόμενο όποτε αποστέλλουμε στον χρήστη ένα έγγραφο html το όποιο περιγράφει την σκηνή μας , και περιλαμβάνει links στην επικεφαλίδα του κειμένου για την τοποθεσία των απαραίτητων αρχείων για την λειτουργία του x3dom καθώς και για την τοποθεσία τού binary.js που είναι υπεύθυνο για την λειτουργιά του πρωτοκόλλου websocket και στον client.

Τα παραπάνω αρχεία παρότι μπορούμε να τα λάβουμε κατευθείαν από τους δημιουργούς τους έτσι ώστε να διαθέτουμε τις τελευταίες εκδώσεις έχουμε επιλέξει να κρατάμε τοπικά αντίγραφα για καλύτερη αξιοπιστία στις παρατηρήσεις μας.

Η δομή του έγγραφου μας πρέπει να είναι όσον το δυνατόν ποιο αυστηρή συντακτικά (xhtml) λόγω των περιορισμών που μας επιβάλει το x3dom framework (αν και σε νεότερες εκδώσεις υποστηρίζεται και ή χαλαρότερη σύνταξης html).

Εντός της σκηνής μας έχουμε επιλέξει να αρχικοποιήσουμε ένα απλό αντικείμενο , ένα κύβο ο οποίος διαθέτει μια βασικότατη γεωμετρία.

Ο λόγος πού επιλέξαμε να αρχικοποιήσουμε την σκηνή είναι για να αποφύγουμε διαφορές ασυνέχειες στην προβολή αυτής που παρατηρούνταν κατά την διάρκεια της υλοποίησης.

Εν συνεχεία όταν το framework μας παράγει event ότι είναι έτοιμο , καλούμε την συνάρτηση sceneInit() ; η όποια εγκαθιδρύει σύνδεση μέσω websocket και ζητάει την γεωμετρία από τον binary server.

```
function sceneInit(){
   var client = new BinaryClient('ws://localhost:9000/binary-endpoint');
   console.log("binary endpoint started");
   client.on('open',function(req){
   req="req " + "IndexedFaceSet";
   var res=JSON.stringify(req);
   res.replace(/"/g,");
   client.send(req);
   });
```

Τα δεδομένα που μας αποστέλλει ο server ανάλογα με το τι είδους είναι αντικαθιστούν τα αντίστοιχα δεδομένα εντός τού αρχικού μας αντικειμένου , πρώτα αντικαθίστανται και εγγράφονται εντός του εγγράφου μας οι πληροφορίες για τα point και το color του μοντέλου και στην συνέχεια με

την χρήση μίας χρόνο καθυστερημένης function τοποθετούνται και τα δεδομένα του coordIndex κολλώντας τα chunks που λαμβάνουμε.

Καθυστερούμε την τελική προβολή του μοντέλου μας για να πετύχουμε συνοχή , διότι σε τοπικό περιβάλλον καθώς και για μικρά σε μέγεθος μοντέλα όλη η διαδικασία γίνεται πολύ γρήγορα και έτσι ο χρήστης δεν βλέπει καν τα ενδιάμεσα στάδια του παραγόμενου αποτελέσματος.

Εικόνα 3 Τα στάδια της αναπαράστασης της αποσπασματικής γεωμετρίας.

4.4 Η μεγαλύτερή εικόνα της εργασίας.

Από την πρώτη εφαρμογή της εργασίας εξήχθησαν πολύτιμα συμπεράσματα τα όποια απαντούν στα βασικά ερωτήματα που έχουμε ήδη θέσει σε προηγούμενο κεφ. θα βρείτε τα συμπεράσματα στο κεφ. 5 .

Το ερευνητικό πεδίο αυτής της εργασίας απασχολεί σημαντικά το Εργαστήριο πολυμέσων Multimedia content Lab (mclab) του Τμήματος Μηχανικών Πολυμέσων του ΤΕΙ Κρήτης ,για αυτόν τον λόγο αποφασίστηκε από τον συγγραφέα της εργασίας και από κοινού τους ερευνητές και υπεύθυνους του εργαστηρίου σε επί μέρους εργασίες να προχωρήσουν οι εργασίες με κοινές συνισταμένες (common ground)

4.5 Spring Framework

Με την χρήση του spring framework μπορούμε να κατασκευάσουμε πολύ γρήγορα web service με την χρήση java web και να πακετάρουμε όλη την εφαρμογή μας σε ένα super jar το όποιο θα μπορεί να αναπτυχθεί σε οποιονδήποτε tomcat server.

Βασικό ρόλο για την ανάπτυξη του webService παίζει ο μηχανισμός dependency injection που έχει ενσωματωθεί στο framework άλλα πρώτα πρέπει να απαντήσουμε σε ένα βασικό ερώτημα, τι θα μεταφέρουμε με το εν λόγω service και σε τι μορφή.

Θα μεταφέρουμε σετ αυτοδύναμων συντεταγμένων points ,pointIndex, color, normal,texturecoordinates και ενδημικά θα πρέπει να υποστηρίξουμε όλες τις βασικές απαραίτητες πληροφορίες πού διαθέτει το πρότυπο του x3dom ώστε να αποδώσει τα μοντέλα και χαρακτηρίζονται από μεγάλο πλήθος τιμών μία ιδανική μορφή για να μεταφέρουμε αυτού του είδους τής πληροφορίες είναι μέσω αρχείων JSON τα όποια χρησιμοποιούνται πλέον κατά κόρων σε web εφαρμογές.

Το spring framework υποστηρίζει την μετάδοση μέσω web service JSON όποτε είναι ιδανικό για αυτήν την εργασία, ή λογική πού ακολουθούμε είναι ότι όταν λάβουμε μία αίτηση GET με συγκεκριμένο query string, θα απαντήσουμε με τον κωδικό 200 ok και στο ΄σώμα της απάντησης θα

```
{
   "id": 1,
   "content": "Hello,
World!"
```

περιέχεται το JSON που περιγράφει τα δεδομένα μας π.χ.

Ξεκινώντας στην java χρειαζόμαστε μία κλάση πού αναπαριστά την πληροφορία , ή οποία θα περιέχει δίνει τις βασικές πληροφορίες κάθε πακέτου τού μοντέλου πού του περιγράφουμε παραπάνω, στην τελική τής μορφή αυτή η κλάση μέσω setter και getter μεθόδων αναθέτει για κάθε τύπο τιμής επί τόπου τις τρέχον τιμές και επιστρέφεται ως ένα γενικό object που περιέχει String , ο λόγος πού καταλήξαμε σε αυτήν την τυποποίηση είναι για να αποφύγουμε ασυνέχειες μεταξύ των διαφορετικών υλοποιήσεών σε επίπεδο μετάδοσης.

```
package spring;

public class deliverMessage {

	private String mes1, mes2, mes3, mes4, mes5;


	public String getMes1() {
		return mes1;
	}

	public void setMes1(String mes1) {
		this.mes1 = mes1;
	}
    ...
}
```

Σύμφωνα με τον μηχανισμό τού spring τις αιτήσεις διαχειρίζεται ένας resource controller στον όποιο χρησιμοποιώντας διάφορα annotations μπορούμε να υλοποιήσουμε ένα πλήθος διαφορετικών service για την περίπτωση της εργασίας αυτής όπου υλοποιείται το Restful με το annotation @RequestMapping ο μηχανισμός του Spring θα χρησιμοποιήσει τα αντίστοιχα bean ώστε να αποστείλει μέσω rest webService τα δεδομένα σε μορφή JSON, αμέσως μετά το annotation δηλώνουμε οποίο URL δέχεται ό μηχανισμός ώς έναρξη για το service ενώ μπορούμε να

```
@RequestMapping(value ="/act1" , method=RequestMethod.GET)
```

περιορίσουμε και την Http μέθοδό που θα το δεχτούμε αυτό.

Στην συνέχεια μπορούμε να ορίσουμε και ποια θα είναι τα ορίσματα που δεχόμαστε εντός του url query όπου εδώ έχουμε ενσωματώσει τα βασικά ορίσματα που θα μας αποστέλλει ένας client αυτά τα ορίσματα στην υλοποίηση μας , λένε ποιο μοντέλο ζητάει ο χρήστης, το μέγεθος των κομματιών που ζητάει να το χωρίσουμε, τον αλγόριθμο με τον όποιο θέλει να το χωρίσουμε και το αν θέλει ένα επιπλέον επίπεδο συμπίεσης εκτός από την native συμπίεση πού μας παρέχεται από το επίπεδο δικτύου στον server.

```
@RequestMapping(value ="/act1" , method=RequestMethod.GET)
	public @ResponseBody  int sQ(

			@RequestParam(value="modelName", required=true,
defaultValue="lyra.x3d")String modelName,
			@RequestParam(value="orderingMethod", required=true,
defaultValue="Serial")String orderingMethod,
			@RequestParam(value="chunkNum", required=true)String
chunkNum,
			@RequestParam(value="compression",required=false)String
compression)
```

Αυτό το query μας παρέχει όλες τις βασικές παραμέτρους πού χρειάζεται ή εφαρμογή για να

εκκινήσει τον packeTeeser που θα αναλάβει να κατατμήσει το μοντέλο και να το επιστρέψει σε αυτοδύναμα κομμάτια του όποιου την λειτουργιά θα αναλύσουμε σε επόμενο κεφάλαιο .

Στην συνέχεια κρατάμε εντός μίας global μεταβλητής την τιμή του ορίσματος τής συμπίεσης, και δημιουργούνται τα αντικείμενα sendthismessage και packeTeeser και καλείται ή συνάρτηση αρχικοποίησης του τελευταίου με τα ορίσματα που έχουμε λάβει ο όποιος μας επιστρέφει το πλήθος των πακέτων αυτών μόλις ολοκληρώσει την διαδικασία και επιστρέφουμε την τιμή αυτή πίσω στον client.

Σε αυτό το σημείο όμως ο client το μόνο που έχει λάβει είναι το πλήθος των κομματιών, που παράγονται από τον αλγόριθμο τμηματοποίησης, θα πρέπει να τα ζητήσει σύμφωνα με την σειρά που θέλει αυτός, πρέπει να δημιουργήσουμε λοιπόν τον κατάλληλο handler για την δουλειά αυτή που θα

```
@RequestMapping(value = "/act2", method = RequestMethod.GET)
	public @ResponseBody
	deliverMessage sQ1(@RequestParam(value = "counter", required = false)
int counter)
```

δέχεται ως όρισμα τον αριθμό του κομματιού.

Παραπάνω βλέπουμε ότι έχουμε αλλάξει τον τύπο επιστροφής της μεθόδου σε delivermessage όποτε θα επιστραφεί πίσω στον client ένα αντικείμενο της μορφής deliverMessage, για σύγκριση στην αρχική κλήση επιστρέφεται μόνο μια μεταβλητή τύπου int, εδώ θα λάβουμε τον αριθμό κομματιού που επιθυμεί ο χρήστης, μέσω τής μεθόδου “e.BuildNumberedChunk(counter);” μας επιστρέφεται το κομμάτι αυτό και εν συνεχεία ανατίθενται μέσω των setter εντός του deliverMessage, αν έχει επιλεχθεί ή έξτρα συμπίεση τα σετ δεδομένων του κομματιού περνάνε πρώτα από τον αλγόριθμο συμπίεσης LZW.

Τέλος πρέπει να γίνει εκτελέσιμη η εφαρμογή όποτε χρειαζόμαστε μία main() ,πλέον μπορούμε να δημιουργήσουμε ένα εκτελέσιμο jar αρχείο που θα περιέχει όλη την εφαρμογή και άλλα υπό – jar καθώς και τις στατικές ιστοσελίδες κτλ. το οποίο μπορεί να μπει αυτόνομα εντός ενός tomcat server.

4.5.1 Κατάτμηση πακέτων

Ο μηχανισμός τμηματοποίησης τρισδιάστατων μοντέλων χρειάζεται κατ' αρχήν να διαβάσει ένα αρχείο τύπου xml που θα βρει εντός μίας διαδρομής, στην συνέχεια εντοπίζει ένα node τύπου X3D και μπορεί εν συνεχεία να κάνει αντίστοιχα ερωτήματα ώστε να βρει τα subNodes που μας ενδιαφέρουν και να ελέγξει αν περιέχουν πληροφορία.

Εδώ χρησιμοποιήθηκε μία βιβλιοθήκη ή scireum.open για αυτόν τον σκοπό που είναι ικανή να επεξεργαστεί αρχεία xml μεγάλου μεγέθους (ίσως και αρκετών εκατοντάδων megabyte) η οποία δεν χρειάζεται να τα φορτώνει εξ' ολοκλήρου στην μνήμη κρατώντας έτσι μόνο ένα μερικό κομμάτι που μας ενδιαφέρει και ενσωματώνοντας τις λειτουργίες της στον κατατμητή μοντέλων διαβάζουμε τιμές για τα παρακάτω Nodes.

- CoordinateIndex & Points
- ColorIndex & Colors
- NormalIndex & Normals
- TextureCoordinateIndex & TextureCoordinates

Στην συνέχεια ανάλογα με τον αριθμό chunk που εισάγεται χωρίζονται τα String των τιμών τους έτσι ώστε να έχουμε πλήρη δεδομένα ανά chunk δηλαδή: οι συντεταγμένες ενός point είναι τρεις (3) αριθμοί (x,y,z) συντεταγμένες, ενώ οι τιμές των CoordinateIndex είναι 4 αριθμοί αντίστοιχα για ένα πλήρες σετ textureCoordinate χρειαζόμαστε μόνο 2 αριθμούς, βλ. Περισσότερα X3D API. Μόλις ολοκληρωθεί ή κατάτμηση επιστρέφεται ένα μήνυμα “Done”

Στο τελικό αποτέλεσμα διαθέτουμε σωστά κατακτημένα string ανά μέγεθός chunk και μπορούμε να καλέσουμε μέσω δύο μεθόδων, η πρώτη επιστρέφει πάντα το επόμενο chunk ενώ ή δεύτερη μέσω ορίσματος επιστρέφει το συγκεκριμένο chunk που του ζητάμε, αξίζει να ειπωθεί ότι ή μορφή τής γεωμετρίας με αυτές τις “Σειριακές” μεθόδους εξαρτάται αποκλειστικά από τον τρόπο με τον όποιο έχει κατασκευαστεί το μοντέλο μας κάθε φορά.

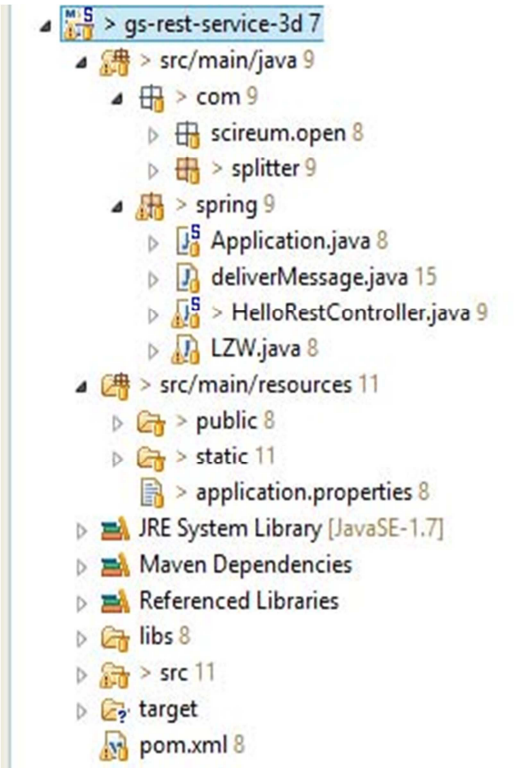
Δημιουργήθηκε επιπλέον μία μέθοδός “Sort By Areas”ή όποια ανακατατάσσει τα πακέτα κατά μέγεθος επιφανείας (Face) από το μεγαλύτερο προς το μικρότερο ή όποια εάν κληθεί εκτελείται μετά τις προηγούμενες εργασίες και ενδέχεται να καθυστερήσει κάπως την ολοκλήρωση τής μετάδοσης ή όποια όμως παράγει εξαιρετικά οπτικά αποτελέσματα και προσδίδει μία ποιο ολοκληρωμένη αίσθηση του μοντέλου στον παρατηρητή.

4.5.2 Συμπίεση.

Στα πλαίσια της εργασίας θεωρήθηκε ενδιαφέρον να διερευνηθεί ή υποστήριξη ενός επιπλέον επιπέδου συμπίεσης, αποφασίστηκε αυτό το επίπεδο να εφαρμοστεί εντός του αρχείου JSON που θα αποστείλουμε, αποφασίστηκε να χρησιμοποιηθεί ο μη απολεστικός αλγόριθμος συμπίεσης Lempel-Ziv-Welch (LZW) πού είναι πλέον ελεύθερος στο ευρύτερο κοινό και δοκιμασμένος, ενσωματώθηκε η java εκδοχή του .

4.5.3 Λοιπές εργασίες.

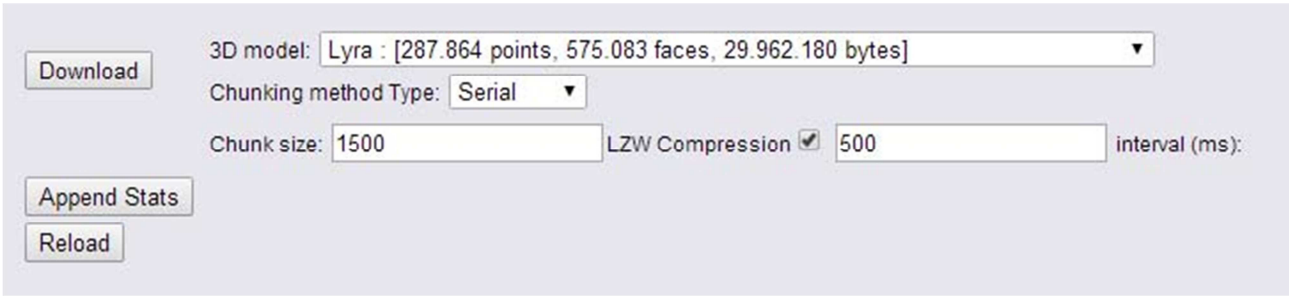
Για την αποφυγή πιθανών προβλημάτων στην ανάγνωση των X3D αρχείων πρέπει να σιγουρευτούμε ότι τα αρχεία μας εντός των node δεν έχουν απομείνει διαφορά “tokens” πού μπορεί να προκύψουν από διαδικασίες export όποτε τα string τιμών χρειάζεται να έχουν μόνα κενά μεταξύ των τιμών και να μην διαθέτουν κόμματα.



Σε περίπτωση πού θέλουμε να αναπτύξουμε την εφαρμογή μας μαζί με άλλες χρειάζεται να λάβουμε μέριμνα σε ποίο port θα δουλεύει το framework μπορεί να διαχειριστεί και αυτήν την περίπτωση , το μόνο που χρειάζεται είναι να δημιουργήσουμε ένα αρχείο “application.properties” εντός τής διαδρομής στατικού περιεχόμενου, στην όποια αποθηκεύουμε και τις στατικές σελίδες, τα μοντέλα που χρησιμοποιούμε, αρχεία JavaScript και λοιπό στατικό περιεχόμενο, τέλος θα πρέπει να σιγουρευτούμε ότι θα προσθέσουμε στο build path και την επιπλέον βιβλιοθήκη που έχουμε προσθέσει.

4.5.4 Client page.

Όπως και στην πρώτη δοκιμαστική λειτουργία ή σελίδα που θα μεταδώσουμε στον παρατηρητή θα περιέχει όλο το περιεχόμενο του εγγράφου πλην των δεδομένων της γεωμετρίας, έχει εδώ δημιουργηθεί ένα βασικό interface από το όποιο μπορεί ο χρήστης να επιλέξει από διαφορά διαθέσιμα μοντέλα πού έχουν δημιουργηθεί εντός του εργαστηρίου πολυμέσων και να ορίσει με ποιον αλγόριθμο θέλει να κατατμηθεί το μοντέλο που θα επιλέξει με τι μέγεθός chunk θα γίνει ή εργασία αυτή και αν επιθυμεί επιπλέον επίπεδο συμπίεσης και ανά πόσα ms θα καλούνται τα κομμάτια αυτά, τέλος δημιουργήθηκαν 3 κουμπιά το ένα εκκινεί την όλη διαδικασία , ένα κουμπί μεταφόρτωσης όλης της σελίδας και ένα κουμπί για την προβολή στατιστικών στοιχείων σχετικά με τα πόσα πολύγωνα εμφανίζονται στην σκηνή ανά δευτερόλεπτο.



Εικόνα 4 Το βασικό interface του χρήστη.

Όταν ο χρήστης επιλέξει να “κατεβάσει” το μοντέλο που επιθυμεί λαμβάνονται από την συνάρτηση connect() οι τιμές ΄των επιλογών του και δημιουργείται το querystring πού θα αποσταλεί στον server μας , για την αρχική τμηματοποίηση του μοντέλου, το query αυτό το έχουμε ορίσει να έχει την εξής μορφή : URL/act1?modelName="filename”&orderingMethod=”METHOD”&chunkNum=”NUMBER”

&compression=”true”
αξίζει να σημειωθεί ότι άμα γράψουμε και σκέτο το querystring με σωστά ορίσματα στην γραμμή διεύθυνσης ενός browser θα εκκινήσει κανονικά ή διαδικασία τμηματοποίησης και θα λάβουμε απάντηση από τον διακομιστή πχ.

http://localhost:8081/act1?modelName=lyra.x3d&orderingMethod=Serial&chunkNum=1500&compression=true θα γυρίσει ως response ο αριθμός των πακέτων που έχει χωριστεί το μοντέλο .

Η connect συνάρτηση θα εκκινήσει την συνάρτηση httpClient() η όποια αναλαμβάνει να πραγματοποιήσει την σύνδεση με τον διακομιστή και όταν λάβει περιεχόμενο απάντησης με κωδικό 200 θα επιστρέψει το περιεχόμενο της στην connect() .

Στα αρχικά στάδια της ανάπτυξης τής εργασίας είχε χρησιμοποιηθεί ή βιβλιοθήκη jQuery για την πραγματοποίησης της σύνδεσης, για λόγους απλότητας στην τελική έκδοση απαλείφθηκε τελείως η χρήση αυτής καθώς την χρειαζόμασταν μόνο για μία συνάρτηση η όποια που μπορεί να δημιουργηθεί με απλή χρήση JavaScript, παρ' όλα αυτά αν κάποιος θέλει να την χρησιμοποιήσει ώστε να ενσωματώσει και άλλα features μπορεί άνετα να αντικαταστήσει τα αντίστοιχα τμήματα κώδικα.

Μόλις έχουμε την αρχική απάντησή από τον διακομιστή αναλαμβάνει ή συνάρτηση Download(), η όποια πάλι με την χρήση της httpClient() εκκινεί μία συνάρτηση η όποια επαναλαμβάνεται ανά τακτά χρονικά διαστήματα που έχει ορίσει ο χρήστης (interval) μέχρι ένας μετρητής φτάσει την αριθμητική τιμή που λάβαμε στο προηγούμενο response τής δώσει εντολή να σταματήσει, ο λόγος που προτιμήθηκε αυτή η ψευδό-επανάληψη έναντι μίας κλασσικής επαναληπτικής δομής, βρίσκεται στην μη ύπαρξη υποστήριξης παράλληλης επεξεργασίας από την JavaScript, παρατηρήθηκε σε πρώιμα στάδια όπως και κατά την διάρκεια ανάπτυξης τής πρώτης υλοποίησης ότι "μπουκώνει" η εκτέλεση όλων των εργασιών της εφαρμογής χρησιμοποιώντας blocking δομές.

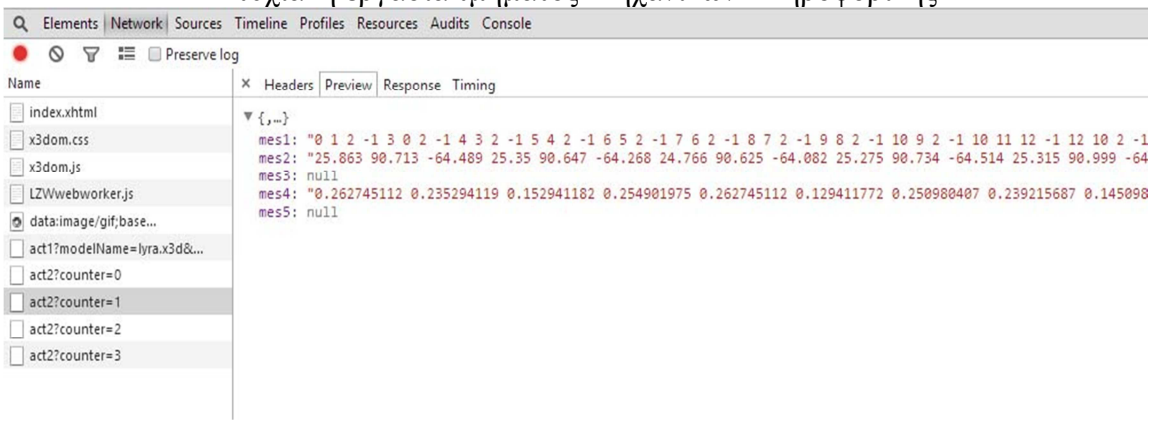
```
function connect() {

	Client = new HttpClient();
	//change localhost with server URL
	req="http://localhost:8081/act1?modelName="+modelName.value.toString(
)+
		"&orderingMethod="+actlist.value.toString()+
		"&chunkNum="+chunkNum.value.toString()+
		"&compression="+compression.checked.toString();
	Client.get(req, function(res) {
	    // do something with answer
		size=res;
		console.log("success");
		Download();
	});
}
```

Στην συνέχεια ανά κάθε επανάληψη τής συνάρτησης δημιουργείται το querystring που χρειαζόμαστε ώστε να ανασύρουμε κάθε διαφορετικό chunk, αυτήν την φορά το Spring πρέπει να είναι της μορφής :

URL/act2?counter="+N όπου 0<= N < Number of Chunks
πάλι άμα για παράδειγμα γράψουμε στην γραμμή διεύθυνσης (αφού έχει ενεργοποιηθεί ο κατατμητής πρώτα) το querystring : http://localhost:8081/act2?counter=0

παίρνουμε το παρακάτω response που περιλαμβάνει την πληροφορία της γεωμετρίας του μοντέλου μας.

(τα mes3 και mes5 είναι κενά γιατί το μοντέλο δεν διαθέτει τιμές για normal και textureCoordinates)



Εικόνα 5 Τα περιεχόμενα δεδομένα ενός αρχείου JSON

Το response που λαμβάνουμε βρίσκεται σε μορφή JSON το όποιο όμως δεν είναι valid περιέχει κώδικα JavaScript με άλλα JSON πρέπει να περάσει από την συνάρτηση eval() ή οποία κάνει extract τα δεδομένα μας και τα αναθέτουμε σε μία προσωρινή μεταβλητή, εν συνεχεία άμα είχαμε εξ αρχής επιλέξει επιπλέον συμπίεση παίρνανε από ένα web worker ο οποίος αναλαμβάνει αυτήν την εργασία, αλλιώς εκκινούμε την συνάρτηση πού θα τα τοποθετήσει εντός της σκηνής μας, τέλος αυξάνουμε τον μετρητή.

Ο web worker που έχει δημιουργηθεί αναλαμβάνει να “τρέξει” την αποσυμπίεση σε ένα ξεχωριστό tread εξ ορισμού ώστε να μην μπλοκάρεται όλη η εφαρμογή μέχρι την ολοκλήρωση της διαδικασίας, δουλεύει περνώντας μηνύματα από την κύρια εφαρμογή στον worker και αυτός με την σειρά του τα περνάει πίσω με την ολοκλήρωση της διεργασίας του έτσι τοποθετούμε την συνάρτηση αποσυμπίεσης των string εντός του και μας επιστρέφεται το αρχικό string.

Η συνάρτηση τοποθέτησης των chunk στην σκηνή δέχεται ως ορίσματα τα αποκωδικοποιημένα string που περιέχουν τις τιμές για κάθε κατηγορία Node μετά συνθέτουμε ένα μεγάλο string array που έχει μορφοποιηθεί κατάλληλα και Valid ως ένα Node <shape> και περιέχει τις τιμές που χρειαζόμαστε στην συνέχεια δημιουργούμε ένα νέο transform node στο έγγραφο μας και θέτουμε το εσωτερικό HTML ίσο με το string Array αφού το ενοποιήσουμε, που δημιουργήσαμε και τέλος τοποθετούμε το transform node ως child node του X3D node, όλο σε ένα πέρασμα επάνω στο έγγραφο.

Δημιουργήθηκαν και μερικές βοηθητικές συναρτήσεις συνοπτικά :

- setCameraPos(), αλλάζει την περιοχή θέασης των μοντέλων ανάλογα με το ποιο έχει επιλεχθεί.
- myStopFunction(), Σταματά την επανάληψη της συνάρτησης Download.
- StartTimer(), Λαμβάνει ανά τακτά χρονικά διαστήματα στατιστικά στοιχειά μέσω του run-time environment του x3dom framework της σκηνής.

- DispalyShots(), Εμφανίζει τα στατιστικά στοιχεία εντός της ιστοσελίδας.
- MyWorker. Αρχικοποιεί τον LZW web worker και τού περνάει τα μηνύματα.

4.5.5 Σημειώσεις.

Με τον τρόπο που αναπτύχθηκε η εργασία είναι αδύνατον να χρησιμοποιηθούν με ασύγχρονο οι κλήσεις get και να έχουμε απρόσκοπτα αποτελέσματα γρήγορα, μπορεί πάντοτε να χρησιμοποιηθεί ένας ενδιάμεσος Buffer στον διακομιστή που θα επιτρέψει αυτού του είδους την λειτουργία επίσης στο παρών επίπεδο υλοποίησης δεν επιτρέπεται ή ταυτόχρονη κλήση πολλαπλών μοντέλων καθώς κάθε "/act1" δημιουργεί νέα chunks,όπου χρειάζεται κρατάμε global μεταβλητές και τέλος έχει επιλεχθεί προσωρινά η τοποθέτηση στην σκηνή μόνο τού υπάρχοντος σετ πληροφορίας γεωμετρίας και χρώματος (coordinateIndex , points , color, ή και normals).

Εικόνα 6 Στιγμιότυπα της λήψης με σειριακό αλγόριθμο

Εικόνα 7 Στιγμιότυπα της λήψης με χρήση του αλγορίθμου Area Sort

4.5.6 Επεκτάσεις - textures - normal mapping.

Από την στιγμή που επιτεύχθηκε η απρόσκοπτη μεταφορά του μοντέλου μας πρέπει να με αυτήν την βάση να υποστηρίξουμε ακόμα περισσότερες τεχνικές ή πρακτικές καλύτερα, μια από αυτές είναι το normal mapping που καταρχάς υποστηρίζεται από x3dom framework ή συγκεκριμένη τεχνική απαιτεί την χρήση δυο αρχείων image το ένα περιέχει όλη την πληροφορία χρώματος και το δεύτερο την πληροφορία των normal είναι δυνατόν χρησιμοποιώντας αυτήν την τεχνική να χρησιμοποιήσουμε ένα μοντέλο μικρότερης ανάλυσης και πολυγώνων με το normal map ενός μεγαλύτερου, το παραγόμενο αποτέλεσμα είναι εφάμιλλο με αυτό του αρχικού μοντέλου .

Για αρχή πρέπει να χρησιμοποιήσουμε ένα διαφορετικό μοντέλο από αυτά που έχουμε χρησιμοποιήσει έως τώρα τα όποια παρείχαν μόνο πληροφορία για points normal και color, χρησιμοποιήθηκε ένα μοντέλο Ύδρας που έχει δημιουργηθεί στο εργαστήριο.

Εικόνα 8 Το μοντέλο της ύδρας.

Μετά από την ανάλυση τού x3d export του συγκεκριμένου μοντέλου προκύπτει ότι διαθέτει επιπλέον nodes που μάλιστα περιέχουν μεγάλο πλήθος τιμών καθώς και ίσως παρουσιαστεί ένα ζήτημα με μεγάλων πλήθος inline κλήσεων των texture files διότι για κάθε chunk έως τώρα τοποθετούμε ένα νέο transform με το shape εντός του κεντρικού group και με την συγκεκριμένη λογική θα πρέπει να τοποθετήσουμε πολλαπλά texture , επιπλέον δεν είναι προφανές το κατά πόσο θα

```
<Transform >
        <Shape>
        <Appearance>
                <CommonSurfaceShader diffuseFactor= ... specularFactor=... shininessFactor='0.145'
alphaFactor='1.000' tangentTextureCoordinatesId='1' binormalTextureCoordinatesId='2' >
                        <SurfaceShaderTexture containerField='diffuseTexture' >
                                <ImageTexture url=URL />
                        </SurfaceShaderTexture>
                        <SurfaceShaderTexture containerField='normalTexture' >
                                <ImageTexture url=URL />
                        </SurfaceShaderTexture>
                </CommonSurfaceShader>
</Appearance>
<IndexedFaceSet ...>
<Coordinate point=.. >
<Normal vector= … >
<TextureCoordinate point= ...>
<FloatVertexAttribute name='tangent' numComponents='3' value= … >
<FloatVertexAttribute name='binormal' numComponents='3' value= ...>
</IndexedFaceSet>
```

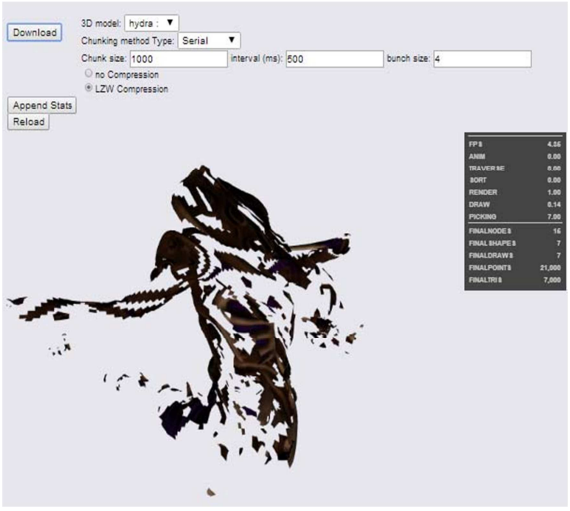
επιτραπεί από το framework μία τέτοιου είδους κατάτμηση της γεωμετρίας και ενιαίου texture.

Όπως φαίνεται και στην παραπάνω εικόνα έχουμε την εισαγωγή των FloatVertexAttribute κανονικά θα έπρεπε να τροποποιήσουμε τον κατατμητή πακέτων ώστε να επεξεργάζεται και αυτά τα Nodes με λίγη έρευνα στα specifications του X3D αποδείχτηκε ότι ή ενσωμάτωση δεν είναι απαραίτητη, καθώς τα πεδία αυτά είναι μόνο βοηθητικά για την επιτάχυνση του rendering.

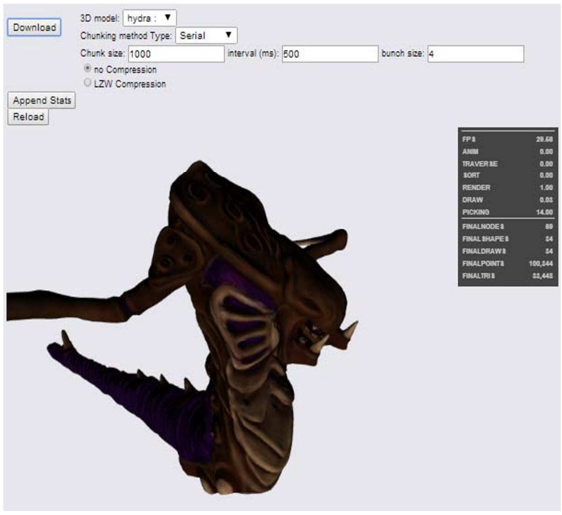
Σύμφωνα με τα παραπάνω επεκτάθηκε πλήρως η υποδομή που έχει ήδη δημιουργηθεί για την πλήρη υποστήριξη τής μεταφοράς των Node τύπου TextureCoordinatePoint στην εφαρμογή του διακομιστή καθώς και στο client page δημιουργήθηκε επιπλέον μία νέα συνάρτηση η showgreeting2() ή όποια τοποθετεί εντός της σκηνής την διαφορετική δομή των νέων nodes.

Παράλληλα αναπτύχθηκε μία εναλλακτική σειρά μεταφοράς των chunks ελεγχόμενη αποκλειστικά από τον client, στην ουσία ο χρήστης επιλέγει ανά ποίο N κομμάτι θέλει να λάβει το μοντέλο και με επαναληπτικό αλγόριθμο εντός της συνάρτησης ψευδό επανάληψης γίνονται κλήσεις στον διακομιστή μέχρι να καταφτάσουν όλα τα chunks, όλη η διαδικασία επιφέρει μια ποίο ολοκληρωμένη εικόνα του μοντέλου σε αρχικά στάδια από ότι ή σειριακή μετάδοση.

Αναπτύχθηκε επίσης ένας επιπρόσθετος requestHandler ο /act3 για λόγους ευκολίας που δεν υποστηρίζει κανένα επιπλέον επίπεδο συμπίεσης άλλα μπορεί να χρησιμοποιήσει διαφορετική υποστήριξη native συμπίεσης επάνω στο HTTP frame ενώ προστέθηκε υποστήριξη για τους αλγορίθμους συμπίεσης inflate και gzip καθώς δοκιμάστηκε και ή ενσωμάτωση τής βιβλιοθήκης zlib για Client JavaScript decompression.



Εικόνα 9 Στιγμιότυπο κατά την διάρκεια της λήψης.



Εικόνα 9 Το ολοκληρωμένο μοντέλο της Υδρας.

Με την ολοκλήρωση των προαναφερθέντων ενεργειών και με τα συμπεράσματα που βγήκαν ή εργασία μπορεί να καλύψει ένα μεγάλο πλήθος υποστηριζόμενων πρακτικών και στον τρόπο κατασκευής των μοντέλων και στις καλύτερες πρακτικές σχετικά με την συμπίεση και την αποσυμπίεση του JSON με την χρήση JavaScript βλ. περισσότερα στα συμπεράσματα.

4.5.6 Διαδραστική μετάδοση εντός σκηνής.

Για να μεταφέρουμε σε μια ολόκληρη σκηνή ένα πλήθος από τα διαθέσιμα μοντέλα μας χωρίς να επέμβουμε στην κατάτμηση χρειαζόμαστε τις τελικές θέσεις εντός της σκηνής για τα μοντέλα αυτά χωρίς όμως να επέμβουμε στο string εγγραφής των Node που θα εγγράψουμε σε αυτήν θα χρειαστούμε ένα κύριο transform με τις κατάλληλες συντεταγμένες για κάθε group σχήματος, επιπλέον αποφασίστηκε να προχωρήσει περαιτέρω ή υλοποίηση έτσι ώστε να εκκινείται αυτόματα όλη η διαδικασία όταν ή περιοχή πού θέλουμε να τοποθετήσουμε την γεωμετρία μας “μπει” εντός τής περιοχής προβολής της κάμερας μας.

Για λόγους διευκόλυνσης επίσης μεταφέρθηκαν όλες οι συναρτήσεις μας σε ένα εξωτερικό αρχείο .js κρατώντας μόνο τις global μεταβλητές στην κεντρική σελίδα μας, λόγω του τρόπου εκτέλεσης ενός JavaScript αρχείου χρειαζόμαστε έναν event listener πού θα μεταφέρει την εκτέλεση του αρχείου εντός μίας συνάρτησης και δεν θα τρέξει έτσι όλο το αρχείο με την φόρτωση του χωρίς τις απαραίτητες αρχικές μας μεταβλητές αποφεύγοντας έτι λάθη κατά την εκτέλεση.

```
window.addEventListener("load", function load(event){
    window.removeEventListener("load", load, false); //remove listener, no
longer needed
    myLoad();
},false);
```

Όλη ή λογική για την μετάδοσης τής γεωμετρίας μας όταν την χρειαζόμαστε και μόνο χρειάζεται μια αντιστροφή προσέγγιση από την συνήθη πρακτική όπου βρίσκουμε ποίο είναι το Node στο έγγραφο που μας ενδιαφέρει και έπειτα να επέμβουμε σε αυτό με συναρτήσεις τύπου getElementById, χρειαζόμαστε event που όταν παράγονται θα ενεργοποιούνται οι κατάλληλες συναρτήσεις.

Το x3dom framework μας παρέχει ένα πλήθος συναρτήσεων για την εξαγωγή/ εισαγωγή

στοιχείων, μεταβλητών κ.α. κατά την εκτέλεση επίσης παράγονται διάφορα event που μπορούμε να εκμεταλλευτούμε για να πετύχουμε τον σκοπό μας, χρειαζόμαστε το event που παράγεται όταν

```
document.onload = function()
{
document.getElementById('viewpoint').addEventListener('viewpointChanged',vi
ewpointChangedHandler,false);
};
```

αλλάξουν οι συντεταγμένες της κάμερας όποτε τοποθετούμε το κατάλληλο event στο viewpoint μας.

Οπότε κάθε φορά πού υπάρχει αλλαγή στο viewpoint μας ακόμα και κατά την εκκίνηση της σκηνής μεταβαίνουμε στην συνάρτηση viewpointChangedHandler , από εδώ εκμεταλλευόμενοι την συνάρτηση pickRect του runtime environment του x3dom πού μας επιστρέφει έναν πίνακα με τα shape που βρίσκονται στο πεδίο θέασης μας μπορούμε σχεδόν αν εκκινήσουμε τις γνώστες έως τώρα διαδικασίες.

Είναι προφανές ότι αφού η συνάρτηση pickRect μας επιστρέφει ένα πίνακα με shape πρέπει να τοποθετήσουμε στις αρχικές θέσεις μας κάποια προσωρινά αντικείμενα , για λόγους δοκιμών τοποθετήθηκαν διάφοροι κύβοι αντιπροσωπευτικοί περίπου του όγκου των σχημάτων, μπορούμε μάλιστα να τους ορίσουμε και ως διάφανη ή ημιδιαφανή και έτσι ο χρήστης δεν θα καταλάβει την ύπαρξή τους για τον απειροελάχιστο χρόνο που θα παραμείνουν εντός του οπτικού του πεδίου.

Η συνάρτηση pickRect λαμβάνει ως ορίσματα εισόδου τις διαστάσεις του canvas object του εγγράφου μίας και αυτές οι διαστάσεις δεν θα είναι πάντοτε οι ίδιες σε κάθε σκηνή ή μπορεί και να μεταβληθούν με δυναμικό τρόπο βρίσκουμε μέσω των συναρτήσεων runtime.getWidth() και runtime.getHeight() το πλάτος και το ύψος του canvas και αναθέτουμε σε ένα πίνακα όλη την συνάρτηση.

Σέ αυτό το σημείο άμα “κατεβάσουμε” μία γεωμετρία και αντικαταστήσουμε το πρώτο προσωρινό αντικείμενο μας θα παρουσιαστεί πρόβλημα καθώς στο επόμενο event θα εκκινήσει όλη η διαδικασία από την αρχή, πρέπει οπότε να εφεύρουμε ένα τρόπο πού θα διαχωρίζει τα είδη “κατεβασμένα” σχήματα ή κομμάτια τους από τα άλλα που θέλουμε να αντικαταστήσουμε ή και ακόμα από στοιχειά της σκηνής που θέλουμε να παραμείνουν ως έχει.

Για αυτόν τον σκοπό χρησιμοποιούμε ένα επιπλέον πίνακα που κρατάει πιά αντικείμενα έχουμε ήδη τοποθετήσει στην σκηνή, μιας και η συνάρτηση pickPect δεν είναι απαραίτητο ότι σε κάθε αλλαγή θα επιστρέφει μη μηδενικό αποτέλεσμα, τοποθετώντας ένα id σε κάθε βασικό transform που θέλουμε να αντικαταστήσουμε στην σκηνή μας μπορούμε με ένα απλό έλεγχο αν υπάρχει τιμή για αυτό το id να προχωρήσουμε παρακάτω την διαδικασία.

```
<Group DEF="1">
<transform id="marker1"  translation="-150 0 0"  >
      <shape>
      <box size="50 50 50" ></box>
      <appearance>
      <material diffuseColor="green" transparency="0.0"></material>
      </appearance>
      </shape>
</transform>
</Group>
```

Οπότε τελικά έχουμε σε κάθε event έναν έλεγχο αν περιέχονται shapes μέσα στο viewpoint μας αν υπάρχουν τότε για κάθε ένα από αυτά τα shape και μόνο :

- Βρίσκουμε το parentNode.
- Ελέγχουμε εάν το Node που βρισκόμαστε τώρα διαθέτει τιμή για το attribute id.
- Αν υπάρχει τότε σβήνουμε όλα τα "παιδιά" του node και μεταφέρουμε όλο το Node στον πίνακα που κρατάμε τα στοιχειά που θα μπουν εντός σκηνής.
- Θέτουμε στην μεταβλητή whereto την τρέχουσα θέση του πίνακα shapelistMat.
- Καλούμε την συνάρτηση connect.
- Εάν δεν υπάρχει το attribute id καθαρίζουμε τον activeViewMat.

Παρατήρηση έχουμε εισάγει έναν επιπλέον μετρητή πού ώστε για την πρόσβαση στα επί τόπου στοιχειά του shapeListMat τον οποίο αυξάνουμε εντός της συνάρτησης showGreeting ενώ θέτουμε τιμή στην μεταβλητή whereto της showGreeting είδη πριν καλέσουμε την connect εντός του event handler.

Κατά την διάρκεια των δοκιμών έχει παρατηρηθεί ολοένα και αυξανόμενη κατανάλωση μνήμης σε διαδοχικές κλήσεις μοντέλων στον διακομιστή, μίας και ή Java έχει αυτόματη διαχείριση μνήμης η καλύτερη προσωρινή λύση για την αντιμετώπιση αυτού του προβλήματος είναι να δημιουργηθεί ένας requesthandler που θα καλείται στο τέλος της μετάδοσης και εντός αυτού αφού θέσουμε όλο το στιγμιότυπου του κατατμητή πακέτων ίσο με null θα κάνουμε κλήση του garbage collector.

Οπότε πρέπει τώρα από τον client να καλέσουμε αυτόν τον request handler όποτε δημιουργούμε μια συνάρτηση disconnect και την καλούμε εντός της βοηθητικής συνάρτησης που σταματάει την κλήση των chunks ενώ στο ίδιο σημείο για να είμαστε σίγουροι ότι έχουμε τοποθετήσει όλα τα αντικείμενα που μπορεί να υπάρχουν εντός του viewpoint καλούμε και την συνάρτηση viewpointChangedHandler.

```
@RequestMapping(value = "/end")
        public @ResponseBody
        deliverMessage sQend(){
                if(e!= null){
                        e=null;
                System.gc();
                }else{
                        //do nothing
                }
        String mms1="ok";
        sendthismessage.setMes1(mms1);
        return sendthismessage;
}
```

```
function disconnect() {
      Client = new HttpClient();
      req="http://localhost:8081/end"
      Client.get(req,function(res){});
      console.log("Disconnected");
}

function myStopFunction(){
      clearInterval(myvar);
      disconnect();
      viewpointChangedHandler();
}
```

Εικόνα 10 Στιγμιότυπα κατά την διαδικασίας της μεταφοράς των γεωμετριών εντός της σκηνής.

5 Αποτελέσματα.

Το κεφάλαιο αυτό ασχολείται με τα αποτελέσματα της ΠΕ, ποιο ωφελούνται από αυτή και ποιους σκοπούς εξυπηρετεί.

Ως κύριο αποτέλεσμα έχουμε την real time επεξεργασία, μεταφορά και αναπαράσταση των δεδομένων ενός 3D mesh μέσω web χωρίς ασυνέχειες προς τον παρατηρητή χωρίς την χρήση plugin και με την υποστήριξη ενός μεγάλου εύρους state of the art τεχνικών αναπαράστασης και προβολής μιας τρισδιάστατης σκηνής.

Η πτυχιακή εργασία μπορεί να ωφελήσει στην καλύτερη προβολή μέσω διαδικτύου διαφόρων οργανισμών όπως μουσεία, πινακοθήκες, πολιτιστικών ιδρυμάτων, επίσης μπορεί να συμβάλει στην ανάπτυξη τρισδιάστατων παιχνιδιών μέσω internet, ενδεχομένως και να μπορέσει να βρει τόπο εφαρμογής στην τρισδιάστατη απεικόνιση μοντέλων για ιατρικούς σκοπούς.

Η χρησιμότητα της εργασίας στην έρευνα είναι σημαντική καθώς βρίσκεται στα τελικά στάδια δημοσίευσης μια εργασία βασισμένη σε αυτήν: “State of the art web technologies for progressive presentation of synthetic cultural heritage scenes”, στόχος της εργασίας είναι να βοηθήσει την ερευνητική κοινότητα στην εύρεση των best practices για να επιτευχθεί real time 3D mesh streaming.

Η εργασία στοχεύει στην επιτάχυνση της ανάπτυξης του τομέα των web 3D γραφικών που αυτήν την περίοδο βρίσκεται στην απαρχή της επίλυσης παιδικών ασθενειών.

Η εκπαιδευτική αξία για τον συγγραφέα είναι καθοριστικής σημασίας για την περαιτέρω εμβάθυνση του στον τομέα των τρισδιάστατων γραφικών και την ανάπτυξη web τεχνολογιών και την προετοιμασία του για την αγορά εργασίας.

Εικόνα 11 Διαφορά των οπτίκων αποτελεσμάτων serial επάνω, Area Sort κάτω.

5.1 Συμπεράσματα.

Ακολουθούν τα συμπεράσματα που βγαίνουν από τις δυο διαφορετικές υλοποιήσεις.

5.1.1 Συμπεράσματα υλοποίησης με node.js

Κατά την διάρκεια της πρώτης διερευνητικής υλοποίησης με την χρήση node.js και websocket αποκλειστικά σε JavaScript και την χρήση x3dom μπορούμε να θεωρήσουμε σχετικά ασφαλή την τεχνική που επιλέξαμε για να εισάγουμε την γεωμετρία στην σκηνή μας ή επέμβαση στο DOM και όχι απευθείας σε επίπεδο WebGL μπορεί να επιφέρει αυξημένη χρήση μνήμης στον χρήση άλλα κάνουμε σίγουρο ότι ή τεχνική αυτή θα δουλεύει σε βάθος χρόνου και είναι ανεξάρτητη από μεγάλες αλλαγές στο framework, είδαμε ότι όταν πραγματοποιούνται αλλαγές στις τιμές μιας γεωμετρίας μέσα στο DOM αυτές εμφανίζονται κατευθείαν στο επόμενο render pass αλλά παρατηρείται αυξημένη χρήση μνήμης.

Επειδή το x3dom δεν παρέχει πλήρη υποστήριξη για την ανάγνωση απλής html θα είναι καλύτερο να χρησιμοποιηθεί παντού η ποιο αυστηρή σύνταξη της xhtml, η συνεκτικότητα των δεδομένων που θα εγγράψουμε στο DOM είναι εξαιρετικής σημασίας, δηλαδή άν εγγράψουμε λιγότερες τιμές για ένα coordinate Index από αυτές που έχουν οριστεί στα specification (3 αντί για 4) τότε υπάρχει ακόμα και η περίπτωση πλήρους αποτυχίας της εφαρμογής με την εμφάνιση μηνύματος από τον browser για unresponsive script με αυτό το δεδομένο ως οδηγό πρέπει να σιγουρευτούμε ότι παράγονται πάντα σωστά αποτελέσματα κατά την κατάτμηση μιας γεωμετρίας.

Σε επίπεδο webserver πάρα το γεγονός ότι όλη η βασική υποδομή που μας παρέχει το node.js είναι αρκετά lightweight παρουσιάζεται ένα μεγάλο μειονέκτημα, δεν παρέχεται καμία ευκολία στην συντήρηση και στις αλλαγές, πρέπει να καταβληθεί μεγάλη προσπάθεια στο να αυτοματοποιηθούν όλες οι λειτουργίες και αυτό υπερβαίνει τον σκοπό αυτής της πτυχιακής εργασίας, επίσης ένας τυχών μελλοντικός διαχειριστής θα χρειαστεί ειδική εκπαίδευση για την λειτουργία της εφαρμογής.

Όσον αφορά το binaryjs αποδείχθηκε αρκετά ενδιαφέρουσα η υλοποίηση μεσώ websocket καθώς η ύπαρξή ενός σταθερού αμφίπλευρου καναλιού επικοινωνίας παρέχει πολλές ευκολίες, ή βασική ιδέα για την χρησιμοποίηση του που ήταν η αυτόματη κατάτμηση πακέτων που παρέχει η όποια όμως λειτουργεί αποκλειστικά στο επίπεδο δικτύου, επίσης δεν μας αφήνει το περιθώριο να επιλέξουμε τον τύπο που θα μεταφέρουμε σε low level (blob η array view) ανεξάρτητα του τι τύπου δεδομένα μεταφέρουμε string, JSON ,mpg, κ.α. εδώ έγινε προφανές ότι σε επόμενη υλοποίηση πρέπει να χρησιμοποιηθεί JSON αντί για απλά string.

Ένα βασικό πρόβλημα που παρουσιάστηκε βρίσκεται στο κομμάτι τις κατάτμησης και ανάγνωσης X3D πρέπει να χρησιμοποιηθεί το api του fs module του node.js θα καθυστερεί γενικά όλες τις διαδικασίες, το επόμενο θέμα αφορά όμως τα πιθανά λάθη κατά την διαδικασία τμηματοποίησης, που άμα συμβούν θα χάσουμε κάθε έλεγχο της εφαρμογής χωρίς δυνατότητες ανάκαμψης.

Μερικά ζητήματα ακόμα βρίσκονται στην μη υποστήριξη άλλων Browser πέραν του chrome από το binary.js και η ή μη υποστήριξη native compression από το websocket protocol πρέπει να υλοποιηθεί από εμάς καθώς τα αποτελέσματα σε ασυμπίεστα δεδομένα συγκρίνονται με τους χρόνους μεταφοράς ενός κανονικού x3dom αρχείου.

5.1.2 Συμπεράσματα τελικής υλοποίησης.

Ολοκληρώνοντας και την τελική υλοποίηση , σε συνδυασμό και με την δημοσίευση της ευρύτερης εργασίας (Kapetanakis K. 2014) μπορούμε να καταλήξουμε στα έξης:

Με την χρήση του spring framework είναι δυνατόν να πετύχουμε μία ενιαία πλατφόρμα με πολλαπλές δυνατότητες στην μετάδοση τρισδιάστατων γεωμετριών χρησιμοποιώντας και “παραδοσιακές” μεθόδους μεταφοράς όπως έχουν υλοποιηθεί σε αυτήν εδώ την εργασία και νεότερες όπως το πρωτόκολλο websocket πατώντας επάνω σε κοινά στοιχειά σε όλους τους άλλους τομείς , οπός ο κατατμητής γεωμετρίας και το αντικείμενο JSON που έχουμε επιλέξει να μεταδώσουμε μέσω των διαφορετικών μέσων.

Έχοντας υλοποιήσει ένα restful μοντέλο webService έχουμε επιτρέψει την λειτουργία της πλατφόρμας και σε παλαιότερους browser αρκεί αυτοί να υποστηρίζουν WebGL, βεβαίως η μέθοδος http έχει και κάποια μειονεκτήματα, καθώς υπάρχουν μεγαλύτερες απαίτησες και σε επεξεργαστική ισχύ και μίας και υπάρχουν προσθήκες overhead πληροφορίας που πρέπει αν αποκωδικοποιηθεί.

Από την στιγμή που κατέστη δυνατό να διαβάσουμε ένα μεγάλο αρχείο X3D και να συλλέξουμε πληροφορία μόνο για τα nodes που μας ενδιαφέρουν κρατάμε τις σχετικές απαίτησες σε μνήμη σχετικά χαμηλές μάλιστα, και έχοντας επιτεύξει την έπη τόπου κατάτμηση της πληροφορίας χώρις να προπεξεργαστούμε το αρχικό μας αντικείμενο καταφέραμε να δημιουργήσουμε μία από τις λίγες υλοποιήσεις που έχουν επιτεύξει real-time επεξεργασία.

Όλη μας η πλατφόρμα του server είναι εύκολα επεκτάσιμη μιας και έχουμε χρησιμοποιήσει java για την λειτουργιά της και μπορεί αν ενσωματωθεί σε πολλά συστήματα εξυπηρετητών επεκτάσιμη είναι και οι τύποι γεωμετρίας που μπορούμε να υποστηρίξουμε ενώ υποστηρίζονται και μοντέρνες μέθοδοι shading.

Η εμπειρία που καταφέραμε να παρέχουμε στον χρήστη παρουσιάζει ιδιαίτερο ενδιαφέρον, χρησιμοποιώντας ένα “κανονικό” αρχείο x3Dom με γεωμετρία μεγάλου μεγέθους μία κρητική λύρα (30MB) σε επαναλαμβανόμενες μετρήσεις χρησιμοποιώντας ένα επιτραπέζιο υπολογιστή και σύνδεση στο internet της ταχύτητας 900kb/s το αρχείο αυτό “φόρτωσε” σε 57 -60 sec (58.1 μέση τιμή), καθ' όλη την διάρκεια ο χρήστης δεν μπορεί να έχει καμία αλληλεπίδραση με το αντικείμενο ,ενώ χρησιμοποιώντας την μέθοδό που αναπτύχθηκε χρησιμοποιώντας τον ίδιο υπολογιστή και με διαφορά μεγέθη chunks πετυχαίνουμε συνεχόμενη θέαση από τον χρήστη εντός των 5 πρώτον περίπου δευτερολέπτων ό χρόνος που χρειαζόμαστε για να λάβουμε και το τελευταίο κομμάτι ποικίλλει ανάλογα με το πόσα κομμάτια έχουμε να λάβουμε.

Χρησιμοποιώντας μικρά chunks πετυχαίνουμε πάρα πολύ ωραία αποτελέσματα δημιουργώντας ένα λείο αποτέλεσμα πού μπορεί να καθυστερήσει όμως η ολοκλήρωση του , μπορούμε όμως να επιλέξουμε και την μονοκόμματη μεταφορά όλης της γεωμετρίας αγνοώντας τελείως την τελική εμπειρία του χρήστη που έχει ως αποτέλεσμα ακόμα όλη η διαδικασία να διαρκέσει περίπου 6 - 10 δευτερόλεπτα.

Με την πειραματική απόδειξη βλέπουμε ότι είναι προτιμητέο ακόμα και για την απλή μεταφορά της γεωμετρίας μονοκόμματα χρησιμοποιώντας τις μεθόδους που αναπτύχθηκαν για την εργασία αυτή έχουμε εκπληκτικά βελτιωμένα αποτελέσματα και στον χρόνο, όγκο μεταφοράς και στην τελική εμπειρία του χρήστη σε σχέση με την παραδοσιακή μεταφορά ενός x3Dom html μεγάλου μεγέθους

Έχοντας διαθέσιμο και τον πιο αργό αλγόριθμο ταξινόμησης των faces Area Short παράγονται ακόμα πιο εντυπωσιακά οπτικά αποτελέσματα άλλα εδώ είναι πιθανό να υπάρξει αναμετάδοση κάποιων points οπότε έχουμε αύξηση ΄της μεταδιδόμενη πληροφορίας.

Η χρήση συμπίεσης LZW στο JSON αντικείμενο μπορεί ανά περιπτώσεις να μικρύνει τον τελικό όγκο του αντικειμένου αυτό όμως εξαρτάται αποκλειστικά από το chunk size διότι προστίθεται ένα overhead στο αντικείμενο μας , σε τέτοιο βαθμό μάλιστα που άμα χρησιμοποιήσουμε πολύ μικρά chunks να χρειαστεί να μεταφέρουμε διπλάσιο αριθμό MB από ότι διαθέτει πράγματι το αρχικό αντικείμενο, γενικά με την συμπίεση πετυχαίνουμε καλύτερα αποτελέσματα όσο μεγαλύτερο είναι το μέγεθος ενός chunk.

Μπορούμε να αρκεστούμε στην native συμπίεση μόνο που μας παρέχεται από τους browser και τους webserver που χρησιμοποιούν τους αλγορίθμους gun zip και inflate και μάλιστα σε πολύ ποιο αποδοτικές υλοποιήσεις από ότι μπορούμε να πετύχουμε με την χρήση JavaScript, οι δόκιμες έδειξαν ότι και ο φόρτος στην cpu έπεσε σημαντικά και μεταφέραμε λιγότερη πληροφορία μάλιστα σε μεταφορά του βασικού μοντέλου αναφοράς μας ΄σε ένα chunk μόνο με native συμπίεση επιτυγχάνεται ένας λόγος συμπίεσης της τάξης του 1/20 1,5MB / 30MB και έχει χρόνο μεταφοράς περίπου 7000ms (σε localhost) ενώ επαναλαμβάνοντας την ίδια μέτρηση και με συμπίεση LZW έχουμε λόγω συμπίεσης 1/27 1.1MB /30MB με χρόνο μεταφοράς 15K ms είναι προφανές ότι στην δεύτερη περίπτωση καταναλώνουμε υπερβολικούς πόρους για πολύ μικρό κέρδος, ή γενική υποκειμενική αίσθηση που λαμβάνουμε σε όλες τις μετρήσεις χωρίς συμπίεση είναι ότι έχουμε πολύ γρηγορότερες αποδώσεις και αναδράσεις από το σύστημα.

Η χρήση της cpu μπορεί να ελαττωθεί ακόμα περισσότερο άμα ρυθμίσουμε κατάλληλα το κάθε πότε θα ζητήσουμε ένα chunk μέσω της τιμής interval έχοντας όμως παράλληλο κόστος στην ταχύτητα ολοκλήρωσης της διαδικασίας λήψης, εάν θέσουμε την τιμή αυτή ίση με το μηδέν ή ένα ms στην ουσία θα έχουμε effective κάθε (23 ms) να προστίθεται ή κλήση του επόμενου chunk στην αμέσως επομένη δουλειά που έχει να εκτελέσει το runtime environment της JavaScript engine και επειδή χρησιμοποιούμε συγχρονισμένες κλήσεις first in first out το αποτέλεσμα είναι να μπουκώσουμε το σύστημα με τις κλήσεις κυματίων αυτό θα επηρεάσει και το runtime environment του x3Dom framework, στην πράξη κατάλληλες τίμιες για interval μπορούν να θεωρηθούν από 350 ms μέχρι και 1000 ms (1sec) .

Ανάλογα την εφαρμογή και το αποτέλεσμα που θέλουμε να πετύχουμε πρέπει να βρούμε ένα καλό συνδυασμό στις παραπάνω ρυθμίσεις, σημείωση: είναι δυνατόν με την εισαγωγή ενός buffer που θα περιέχει όλα τα chunks στον κατάλληλο request handler να ενεργοποιήσουμε την ασύγχρονη κλήση chunks.

Μεγάλο ενδιαφέρον παρουσιάζει η ενσωμάτωση των common surface shaders καθώς αφού για κάθε chunk τοποθετούμε στην ουσία ένα καινούριο shape εντός της σκηνής, αναμέναμε ότι θα πραγματοποιούνταν inline κλήσεις για την μεταφόρτωση των texture files σε κάθε shape στην

πραγματικότητα επειδή ζητάμε URI για την τοποθεσία των texture και όχι URL το framework θα ξαναχρησιμοποιήσει τα ίδια texture για κάθε instance τους, ακόμα και σε δεύτερο ίδιο μοντέλο στην ίδια σκηνή.

Χρησιμοποιώντας μοντέλα που εκμεταλλεύονται τους εν λόγω shaders ανοίγουμε τον δρόμο ώστε να μπορούμε να κτίσουμε ακόμα πολυπλοκότερες σκηνές σε web περιβάλλον καθώς μειώνονται οι σχετικές απαιτήσεις από το δίκτυο και από την cpu καθώς μπορούμε να χρησιμοποιήσουμε γεωμετρίες μικρότερης ακρίβειας άλλα εφάμιλλου αποτελέσματος μάλιστα και με την επαναχρησιμοποίηση των resources μας.

Όσον αφορά την εμπλούτιση μιας ολόκληρης σκηνής με μεγάλες γεωμετρίες όπως έχουν αναπτυχθεί στα πλαίσια της εργασίας εδώ παρουσιάζονται αυξημένες προκλήσεις που πρέπει να απαντηθούν από την πλατφόρμα, η λειψή sensors στο x3Dom όπως αυτοί υλοποιούνται στο παλαιότερο X3D μας αναγκάζει να δουλέψουμε με τα διάφορα events που παράγονται στην JavaScript.

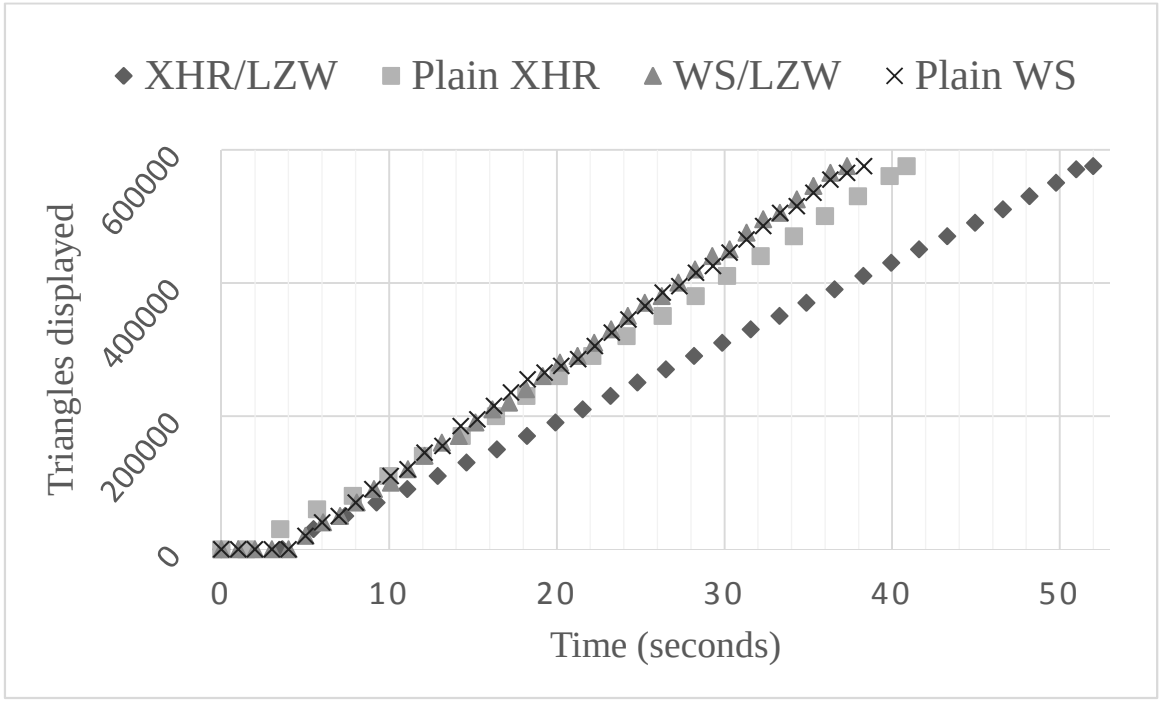
Έχει υλοποιηθεί μια βασική τεχνική βασισμένη σε event που όμως επιβαρύνει την εκτέλεση της πλατφόρμα καθώς σε μία αλλαγή του viewpoint μπορούν να παραχθούν δεκάδες event σε λίγα δευτερόλεπτα που πρέπει με τους κατάλληλους ελέγχους να εκκινήσουμε άλλες διαδικασίες ή όχι.

Επίσης έχει καταστεί προφανές ότι ή αρχική υλοποίηση στον server χρήζει βελτίωσης καθώς λειτουργεί αρκετά ικανοποιητικά σε σενάρια σειριακής αποστολής γεωμετριών αλλά είναι αδύνατη ή ταυτόχρονη εξυπηρέτηση δυο πελατών ή δύο μεταδόσεων διαφορετικών γεωμετριών από τον ίδιο πελάτη.

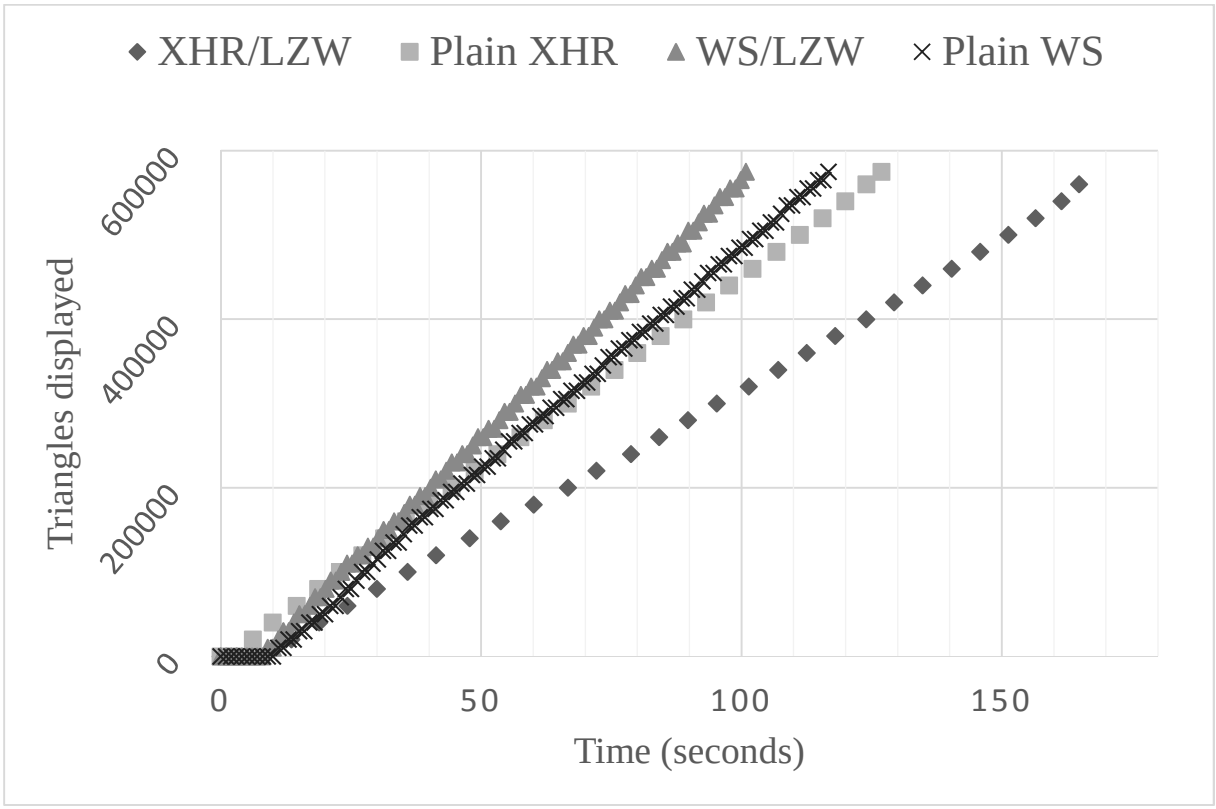
Συγκρίνοντας την βασική υλοποίηση με αυτή των web sockets που έχει αναπτυχθεί στο εργαστήριο πολυμέσων δεν προκύπτει κάποιος ξεκάθαρος νικητής στην εν λόγο υλοποίηση όπου προωθούνται συνεχόμενα τα δεδομένα υπάρχει σίγουρα μικρότερο overhead καθώς και μικρότερος φόρτος στην cpu αφήνοντας μεγαλύτερα περιθώρια για αλληλεπίδραση του χρήστη με την γεωμετρία ένα ξεκάθαρα ανοιχτό θέμα είναι η χρήση της συμπίεσης όπως προ είπαμε.

Χρησιμοποιώντας μικρά με μεσαία chunks η υλοποίηση με web sockets επιτυγχάνει καλύτερες ταχύτητες λόγω μικρότερου overhead όσο όμως μεγαλώνουμε το μέγεθός των chunk η διαφορά μεταξύ των υλοποιήσεων αλλάζει υπέρ της restful μεθόδου καθώς αρχίζουμε να έχουμε μεγαλύτερα οφέλη από την συμπίεση.

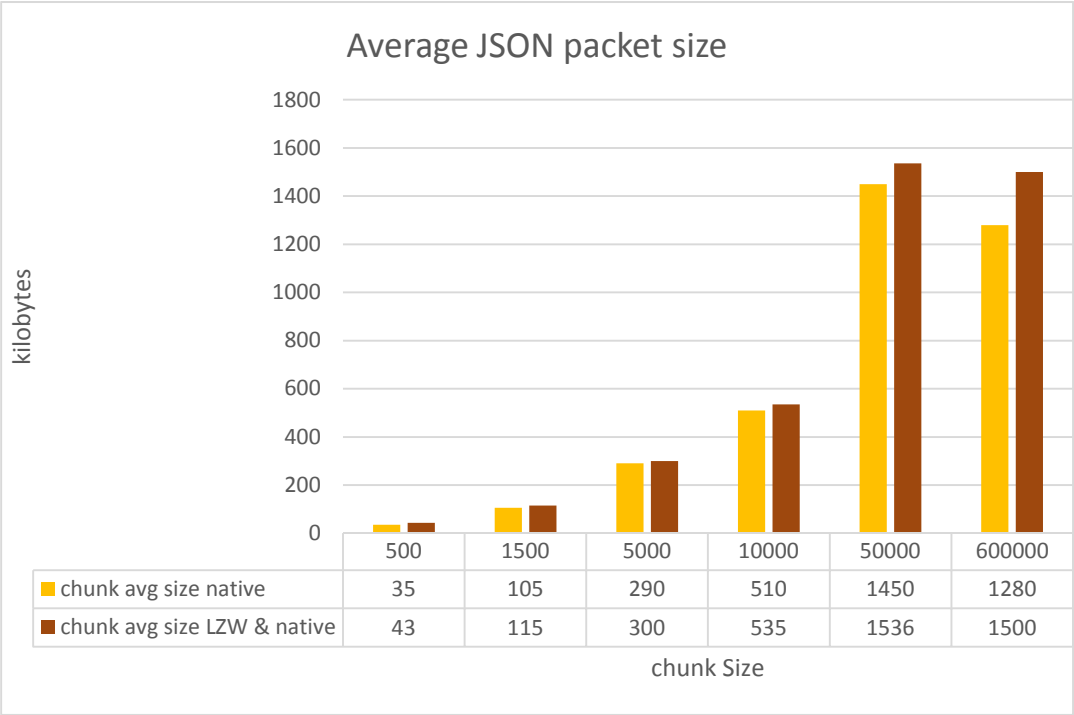
Στα δυο γραφήματα παρακάτω είναι εμφανής η επιβάρυνση της XHR (HTTP) μεθόδου όταν χρησιμοποιούμε JavaScript decompression όπως εξηγήθηκε παραπάνω, η σύγκριση βεβαία δεν είναι μεταξύ δύο τελείως ομοιών τεχνικών λόγω έλλειψης gun zip συμπίεσης από την μεριά της υλοποίησης με web socket οπού στην XHR υλοποιείται αυτόματα από τον tomcat server, μία ακόμα λεπτομέρεια που είναι δύσκολο να μετρηθεί βρίσκεται στο συνολικό μέγεθος της μεταδιδόμενης πληροφορίας μέσω του πρωτοκόλλου web Socket.



Γράφημα 1 Αριθμός τρίγωνων σε βάθος χρόνου με την σειριακή μέθοδο κατάτμησης.

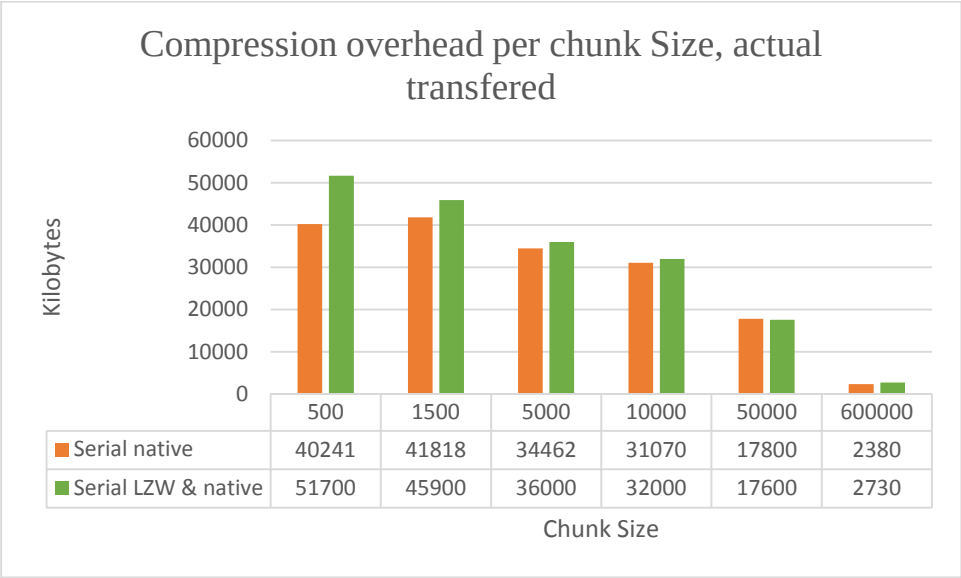


Γράφημα 2 Αριθμός τρίγωνων σε βάθος χρόνου με την μέθοδο κατάτμησης Area Sort.



Γράφημα 3 Μέσο μέγεθος Πακέτων JSON ανά chunk size.

Στο παραπάνω γράφημα 3 παρατηρούμε την επιβάρυνση που επιφέρει το overhead της LZW συμπίεσης στις περιοχές chunk Size που έχουμε ομαλές μεταβολές στην προβολή του μοντέλου αναφοράς μας που φτάνει μέχρι και τα 10 k chunk size, έπειτα από αυτό το σημείο αρχίζουμε να έχουμε κέρδος όγκου παραπάνω από αυτό της native συμπίεσης.



Γράφημα 4 Σύγκριση τελικής μεταδιδόμενης πληροφορίας

Στο γράφημα 4 βλέπουμε την πραγματική αύξηση του όγκου που επιφέρει το αυξημένο overhead καθώς και το γενικό κέρδος πού έχουμε για τα μεγάλα chunk size.

5.2 Μελλοντική Εργασία και επεκτάσεις.

Για την μελλοντική επέκταση της εργασίας έχουμε πολλά ανοικτά θέματα εξ αντικειμένου, έχοντας παρουσιάσει την βασική πλατφόρμα μπορούμε να επεκταθούμε στην πλήρη υποστήριξη όλων των δόμων που μπορούν να χρησιμοποιηθούν στην κατασκευή των βασικών γεωμετρών που υποστηρίζει το x3Dom, όπως για παράδειγμα τα triangle Set, nodes που χαρακτηρίζονται όμως από μεγάλο όγκο δεδομένων.

Πρέπει να βελτιώσουμε την αποδοτικότητα του κατατμητή και να απαλείψουμε τυχών memory leaks καθ' όλη την διάρκεια της λειτουργίας του, ενώ ή ίδια εργασία πρέπει να πραγματοποιηθεί και για όλη την εφαρμογή του εξυπηρετητή.

Μία ενδιαφέρουσα επέκταση για τον κατατμητή πακέτων θα είναι ή δημιουργία αλγορίθμου κατάτμησης χρησιμοποιώντας μη αυτοδύναμα πακέτα που θα περιλαμβάνει ανακατασκευή του περιγράμματος μίας γεωμετρίας κατά τα διάφορα στάδια της αποστολής.

Πρέπει να βελτιωθεί η παράλληλη εξυπηρέτηση πολλαπλών αιτήσεων για γεωμετρίες σε ένα διακομιστή και η κατασκευή μίας περισσότερο κατανεμημένης πλατφόρμας βασισμένης σε πολλαπλούς εξυπηρετητές ίσως με την ενσωμάτωση λειτουργιών κατά τα πρότυπα του πρωτοκόλλου MPEG-Dash.

Έχοντας όλα τα παραπάνω υπόψιν μας σε επίπεδό χρήστη μπορούν να βελτιωθούν διάφορα ζητήματα απόδοσης κάνοντας μεγαλύτερη χρήση των web workers κυρίως στο κομμάτι της διαχείρισης της μεταφοράς, την απελευθέρωση μνήμης όπου χρειάζεται και την εύρεση αποδοτικότερων διαδικασιών στην αλληλεπίδρασης σε μεγάλες σκηνές.

Με την ολοκλήρωση της υλοποίησης των standard του x3dom ή και της WebGL2.0 θα πρέπει επιχειρηθεί όπου είναι δυνατή παράκαμψη του DOM και ή απόδοση της γεωμετρίας απευθείας στην GPU πού θα έχει ως σκοπό να οδηγήσει σε αύξηση των επιδόσεων στον τελικό χρήστη, με την δημιουργία ενός container της γεωμετρίας.

Μία ακόμα ενδιαφέρουσα επέκταση μπορεί να θεωρηθεί η συνεργατική συνθετική σκηνή μέσω της εκμετάλλευσης του WebRTC πρωτοκόλλου και Peer to Peer αλληλεπίδραση και ανανέωση, εισαγωγή γεωμετριών.

Εάν γίνει εφικτή η δημιουργία ενός container με υποστήριξη διαφορετικών level of detail θα μπορεί να γίνει εφικτή ή αυτόματη έπη τόπου απλοποίηση μιας γεωμετρίας και η αναπλήρωση της μέσω εξελιγμένων τεχνικών shading .

Βιβλιογραφία

BEHR J., ESCHLER P., JUNG Y., ZÖLLNER M. 2009. «X3DOM: a DOM-based HTML5/X3D integration model.»

n.d. *Binary.js.* http://binaryjs.com/.

n.d. *Eclipse.* http://www.eclipse.org/.

Fielding, Roy. 2000. «Representational State Transfer (Rest).»

G. Lavoue, L. Chevalier, and F. Dupont,. 2013. «Streaming Compressed 3D Data on the Web using JavaScript and WebGL.»

HICKSON, I. 2012. «HTML5 (W3C Working Draft).»

Hoppe H. 1996. «Progressive meshes.»

Kapetanakis K., Zampoglou M.,Milionis F.Malamos A.,Panagiotakis S.,. 2014. «State of the Art Web technologies for progressive presentation of synthetic caltural heritage scenes.»

Khronos. n.d. *Khronos group.* https://www.khronos.org/.

KHRONOS. 2012. «Typed array specification.»

Kiessling, Manuel. 2013. *The Node Beginner Book.*

M. Limper, Y. Jung, J. Behr, and M. Alexa. 2013. «The POP buffer: Rapid progressive clustering by geometry quantization,.»

—. 2012. «Using images and explicit binary container for efficient and incremental delivery of declarative 3D scenes on the web.»

node.js. n.d. *Node.js.* http://nodejs.org/.

n.d. *Spring.* http://spring.io/.

WEB3D CONSORTIUM. n.d. «Extensible 3d (X3D).»

WebRTC.com. 2011. *WebRTC.* http://www.webrtc.org/.