ΑΛΕΞΑΝΔΡΕΙΟ Τ.Ε.Ι ΘΕΣΣΑΛΟΝΙΚΗΣ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

Τμήμα Μηχανικών
Πληροφορικής ΑΤΕΙΘ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

WEB ΒΙΒΛΙΟΘΗΚΕΣ ΓΡΑΦΙΚΩΝ

Του φοιτητή

Μανατάκη Ιωάννη

Αριθμός Μητρώου: 10/3680

Επιβλέπων καθηγητής

Πασχάλης Ράπτης

ΠΡΟΛΟΓΟΣ

Στο κείμενο που παρατίθεται, γίνεται καταγραφή και ανάπτυξη της διαδικασίας που ακολουθήθηκε κατά την εκπόνηση της πτυχιακής εργασίας που μου ανατέθηκε από το τμήμα Μηχανικών Πληροφορικής του Ανώτατου Τεχνολογικού Εκπαιδευτικού Ιδρύματος Θεσσαλονίκης.

Στη συγκεκριμένη εργασία με τίτλο Web Βιβλιοθήκες γραφικών που ανέλαβα, αναλύονται οι βασικές Web βιβλιοθήκες γραφικών όπου χρησιμοποιώντας την γλώσσα προγραμματισμού JavaScript, μια γλώσσα σεναρίου (Script Language), και από κώδικα (shader), για την δημιουργία διαδραστικών ιστοσελίδων, όπου συνήθως ενσωματώνεται σε HTML κώδικα. Και εκτελείται στην μονάδα επεξεργασίας γραφικών του υπολογιστή (GPU).

Επομένως γίνεται κατανοητό, ότι με της Web βιβλιοθήκες γραφικών μπορούμε να έχουμε διαδραστικά 2D και 3D γραφικά σε οποιοδήποτε συμβατό πρόγραμμα οδήγησης, επιτρέποντας την επιτάχυνση υλικού (μέσω της κάρτας γραφικών του υπολογιστή του χρήστη), δηλαδή τη μεταφορά της απόδοση όλων των γραφικών και των κειμένων από την κεντρική μονάδα επεξεργασίας (CPU) στη μονάδα επεξεργασίας γραφικών (GPU) και στην συνέχεια στην οθόνη μας.

Τα 2D αλλά ακόμα ποιο πολύ τον τελευταίο καιρό τα 3D γραφικά, έχουν γίνει πολύ δημοφιλής, ιδιαίτερα σε βιντεοπαιχνίδια. Το σημαντικότερο όμως είναι ότι είναι συμβατά με του περισσότερους web browser χωρίς την χρήση plug-ins. Κάνοντας κατά πολύ ευκολότερη την δουλειά τον προγραμματιστών που με την σειρά τους μπορούν να δώσουν περισσότερη διασκέδαση στου χρήστες.

ΠΕΡΙΛΗΨΗ

Τα τελευταία χρόνια, έχουν ωριμάσει οι τεχνολογίες δημιουργίας γραφικών πραγματικού χρόνου σε τρεις διαστάσεις, επιτρέποντας τη δημιουργία λογισμικού κατάλληλου για την προσομοίωση και οπτικοποίηση φαινομένων. Σαφώς ανάλογη εξέλιξη παρουσιάζει και το υλικό, καθώς αν δώσουμε έμφαση στις κάρτες γραφικών, το κόστος τους δεν είναι πλέον απαγορευτικό.

Η εργασία αυτή επιχειρεί να συνοψίσει τις διαθέσιμες δυνατότητες που εμπλουτίζονται από βιβλιοθήκες ρουτινών γραφικών, για τη δημιουργία λογισμικού εκτελέσιμου τόσο σε τοπικό όσο και σε διαδικτυακό επίπεδο. Παρουσιάζονται οι βιβλιοθήκες γραφικών ιστού, από την ιστορία τους και το σχεδιασμό τους, μέχρι τη δομή και τον τρόπο που μπορούν να αποδώσουν στους σύγχρονους φυλλομετρητές.

Επιπροσθέτως, αναφέρονται διάφορα χαρακτηριστικά τους, καθώς και παραδείγματα διαφόρων συναρτήσεων μέσα σε εφαρμογές, που αποτελούν και τη βάση των απεριόριστων δυνατοτήτων που μας προσφέρουν. Τέλος, θα αναφερθούν και άλλες παρόμοιες τεχνολογίες γραφικών ιστού αναλύοντας τις δυνατότητες τους.

Η τεχνολογία της **HTML5** σε συνδυασμό με την **WebGL**, έρχεται για να φέρει μια μεγάλη ανατροπή. Ίσως μια από τις πιο σημαντικές είναι η απεξάρτηση από τον Flash της Adobe και ουσιαστικά θα μπορούμε δικτυακά να απολαμβάνουμε πράγματα που παλιότερα είχαμε προβλήματα, θέλαμε ισχυρό hardware και είχαμε άπειρους περιορισμούς.

Για τους περισσότερους, ένα από τα πιο ευχάριστα αυτών των τεχνολογιών είναι τα παιχνίδια στο cloud (αποθηκεύονται σε διακομιστές και καλούνται και διαδικτυακά παιχνίδια), εννοώντας βέβαια την τεχνολογία εκείνη που επιτρέπει στο χρήστη να χρησιμοποιεί λογισμικό, υπηρεσίες και δεδομένα τα οποία δεν είναι αποθηκευμένα σε δικό του υπολογιστή.

ABSTRACT

In recent years, the real time graphics technologies in three dimensions, has reached a stage of evolution, appropriate for the purposes of software suitable for simulating and visualizing phenomena. Clearly similar development presents the hardware as if emphasize on graphics cards, their cost is no longer prohibitive.

This paper attempts to summarize the options available enriched by routines graphics libraries available for software executed either locally or online. Presented by Web Graphics Libraries, from their history and design, to the structure and how can performance in modern browsers.

Additionally, mentioned various characteristics, as well as examples of various functions within some applications, which are the base of the unlimited possibilities that offer us. Finally, I mention other similar web graphics technologies and their abilities.

The combination of technologies HTML5 and WebGL, comes to bring a major upset. Perhaps one of the most important is the independence from the Adobe's Flash and the essence for us all that we can network to enjoy things that previously had problems, like powerful hardware and anyway we were inexperienced restrictions.

Perhaps for many users, one of the most enjoyable things of these technologies is the cloud gaming (stored on servers and used to call online games), basically refers to the technology that allows the user to use software, services and data that is not stored in the computer.

ΕΥΧΑΡΙΣΤΙΕΣ

Αρχικά θα ήθελα να ευχαριστήσω την οικογένεια μου που με στήριξε ηθικά και οικονομικά κατά την διάρκεια των σπουδών μου και κατά την διάρκεια εκπόνησης της εργασίας αυτής.

Επίσης, ευχαριστώ τον εισηγητή καθηγητή μου Πασχάλη Ράπτη για την καθοριστική καθοδήγηση και τις εύστοχες συμβουλές του που οδήγησαν στο πέρας της πτυχιακής μου εργασίας.

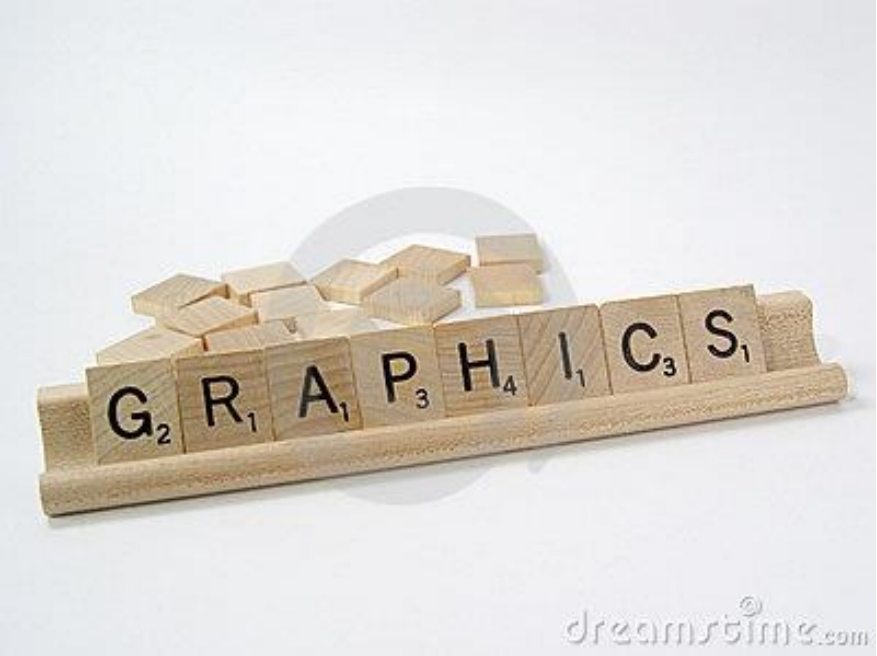
Τέλος, ευχαριστώ όλους τους καθηγητές του τμήματος Μηχανικών Πληροφορικής του ΑΤΕΙ Θεσσαλονίκης για τις γνώσεις που μου παρέδωσαν στα τέσσερα χρόνια των σπουδών μου.

ΠΕΡΙΕΧΟΜΕΝΑ

Ευρετήριο εικόνων

ΕΙΣΑΓΩΓΗ

Ο Παγκόσμιος Ιστός είναι κάτι περισσότερο από κείμενο και πληροφορίες, είναι επίσης ένα μέσο για την έκφραση της καλλιτεχνικής δημιουργικότητας, την οπτικοποίηση δεδομένων και τη βελτιστοποίηση της παρουσίασης των πληροφοριών για διαφορετικά ακροατήρια με διαφορετικές ανάγκες και προσδοκίες. Η χρήση των γραφικών για ιστοσελίδες βελτιώνει την εμπειρία των χρηστών, έχει πολλές διαφορετικές και συμπληρωματικές τεχνολογίες που συνεργάζονται με την HTML και scripting, όπου παρέχονται στους δημιουργούς ιστοσελίδων και web εφαρμογών, σε συνδυασμό με τα εργαλεία που χρειάζονται για να παραδώσουν την καλύτερη δυνατή εκπροσώπηση των περιεχόμενων τους.



Εικόνα 1 - Γραφικά

Τώρα τα γραφικά web είναι οπτικές αναπαραστάσεις που χρησιμοποιούνται σε μια τοποθεσία Web για να ενισχύσουν ή να ενεργοποιήσουν την αναπαράσταση μιας ιδέας προκειμένου να καταλήξουν στον χρήστη (ιστοσελίδα). Τα γραφικά μπορούν να ψυχαγωγήσουν, να εκπαιδεύσουν, ή να επηρεάσουν συναισθηματικά τον

χρήστη, και είναι ζωτικής σημασίας για τη δύναμη του branding, τη σαφήνεια της απεικόνισης, και την ευκολία χρήσης για τις διεπαφές.

Παραδείγματα των γραφικών περιλαμβάνουν χάρτες, φωτογραφίες, σχέδια και μοτίβα, οικογενειακά δέντρα, διαγράμματα, αρχιτεκτονική ή σχεδιαγράμματα μηχανικού, ιστογράμματα και διαγράμματα πίτας, την τυπογραφία, σχηματικές παραστάσεις, διαγράμματα ροής, και πολλές άλλες μορφές εικόνας.

Η υλοποίηση ενός διαδικτύου 3D δεν ήταν εύκολο, καθώς είχαν καταδειχθεί πολυάριθμες αποτυχημένες προσεγγίσεις. Από την πρώτη εμφάνιση του γραφικού ιστού, υπήρχε έντονο ενδιαφέρον για την οικοδόμηση και την απεικόνιση 3D περιεχομένου στο διαδίκτυο.

Ξεκινώντας με τη VRML (Virtual Reality Modeling Language), σχεδιαστές και προγραμματιστές προσπάθησαν να φέρουν την πλοήγηση στους 3D κόσμους, στο οικείο περιβάλλον των προγραμμάτων περιήγησης. Ήταν μια προσπάθεια για την προσομοίωση εικονικών κόσμων που θα τρέχουν στον browser, χρησιμοποιώντας μια γλώσσα βασισμένη στην XML, για να περιγράψει 3D μοντέλα και κινούμενα σχέδια.

Πολύ γρήγορα αναπτύχθηκαν εξειδικευμένες μορφές αυτού που αποκαλείται add-on, τα πρόσθετα λογισμικά (plugins) και εργαλεία επεξεργασίας για να βοηθήσουν στην «επανάσταση της VRML», όμως δεν ήταν όλοι διατεθειμένοι να εγκαταστήσουν επιπλέον λογισμικά και έτσι γρήγορα άρχισε να ξεθωριάζει. Ωστόσο η ανάπτυξη της VRML δε σταμάτησε εκεί, αλλά συνέχισε τις εργασίες για την επόμενη γενιά των 3D γλωσσών βασισμένων στην XML.

Η τελευταία έκδοση είναι η X3D, που είναι πλέον ένα διεθνές πρότυπο με εργαλεία για ανθρωποειδές animation. Υπάρχει υποστήριξη της X3D στην open source Blender 3D εφαρμογή και plugins για τους πιο κοινούς διαδικτυακούς περιηγητές.

Άλλες εναλλακτικές για την παροχή 3D γραφικών στο διαδίχτυο, προήλθαν από τον "κόσμο των plugin" με την JAVA 3D , η οποία έδωσε δυνατότητες σε βοηθητικές εφαρμογές της JAVA (Java applets).

ΚΕΦΑΛΑΙΟ 1

Η ΙΣΤΟΡΙΑ ΤΩΝ WEB GRAPHICS LANGUAGES

1.1 Εισαγωγή

Τα τελευταία 20 χρόνια περίπου έχουμε εμπειρία τρισδιάστατων γραφικών στον υπολογιστή, κυρίως μέσω των παιχνιδιών (Ultima Underworld, Doom, system shock). Ωστόσο τα τρισδιάστατα γραφικά στον browser είναι κάτι σχετικά πρόσφατο.

Κάποια plugin (π.χ. Flash της Adobe) επέτρεπαν τρισδιάστατα γραφικά. Η Web βιβλιοθήκες γραφικών είναι μια νέα τεχνολογία που επιτρέπει την απεικόνιση 3D γραφικών χωρίς την απαίτηση προσθήκης κάποιου πρόσθετο λογισμικού πακέτου. Είναι μέρος του κώδικα της ιστοσελίδας γραμμένο σε JavaScript και απαιτεί φυλλομετρητή που υποστηρίζει HTML5. Οι Web βιβλιοθήκες γραφικών υποστηρίζονται πλέον από όλους τους browsers (ο Internet Explorer άρχισε να παρέχει κάποια λειτουργικότητα από την έκδοση 11).

1.2 Σχεδιασμός

Η WebGL είναι μια διεπαφή λογισμικού (low – level rendering API) βασισμένη στην OpenGL ES 2.0, για την πρόσβαση σε υλικό γραφικών μέσα από έναν web browser, χωρίς την εγκατάσταση επιπλέον plugins. Σχεδιάστηκε ως ένα γενικό πλαίσιο σχεδιασμού γραφικών για το <canvas> element της HTML5. Επιτρέπει σε έναν προγραμματιστή να καθορίσει τα αντικείμενα και τις λειτουργίες που εμπλέκονται στην παραγωγή υψηλής ποιότητας γραφικών, ειδικά 3D αντικειμένων.

Η WebGL (Web Graphics Library) είναι μια JavaScript API (Application Programming Interface) για την απεικόνιση 3D και 2D γραφικών. Μπορεί να τρέξει σε πολλές διαφορετικές συσκευές όπως tablets, τηλεοράσεις, υπολογιστές και κινητά τηλέφωνα. Επίσης η χρησιμοποίηση του δε περιορίζεται μόνο στα παιχνίδια, αλλά και σε 3D μοντελοποίηση, σχεδιαστικά εργαλεία, εξομοιώσεις, διαδραστική τέχνη, αστικοποίηση δεδομένων και άλλα.

Όπως και η OpenGL ES 2.0, η WebGL δεν έχει τα fixed-function APIs που εισήχθησαν στην OpenGL 1.0 και καταργήθηκαν στην OpenGL 3.0. Αυτή η λειτουργικότητα μπορεί να παρέχεται από τον χρήστη στην περιοχή του κώδικα JavaScript.

1.3 Ιστορία της Web Graphics Library

Η OpenGL, στην οποία βασίζεται η WebGL, είναι λογισμικό που επιτρέπει σε προγράμματα πολυμέσων (όπως είναι τα παιχνίδια, οι ταινίες και το λογισμικό σχεδίασης γραφικών) να επικοινωνούν με το υλικό που αποστέλλει τα κινούμενα γραφικά στην οθόνη του υπολογιστή. Επιπροσθέτως επιταχύνει την επικοινωνία ανάμεσα στην προέλευση των κινήσεων και στην οθόνη, βοηθώντας στη δημιουργία ρεαλιστικών κινήσεων.

Το OpenGL είναι ήδη εγκατεστημένο στον υπολογιστή. Οι περισσότερες κάρτες γραφικών λειτουργούν με το OpenGL αρκεί να έχουμε εγκατεστημένα τα πιο πρόσφατα προγράμματα οδήγησης της κάρτας γραφικών. Αυτό μπορεί να διαπιστωθεί αν μεταβούμε στην τοποθεσία Web του κατασκευαστή της κάρτας γραφικών για να βεβαιωθούμε ότι διαθέτουμε τα ενημερωμένα drivers της OpenGL. Όλοι οι κατασκευαστές των καρτών γραφικών κατασκευάζουν τα δικά τους προ-γράμματα οδήγησης OpenGL.

Η WebGL δημιουργήθηκε από τα πειράματα Canvas 3D (c3DL, είναι μια JavaScript βιβλιοθήκη του Canvas 3D, που εμπεριέχει μαθηματικά και τρισδιάστατες τάξεις αντικειμένων για να κάνουν τη WebGL πιο προσιτή στον προγραμματιστή) τα οποία ξεκίνησαν από τον Vladimir Vukićević στο Mozilla.

Ο Vladimir Vukićević επέδειξε ένα πρωτότυπο του Canvas 3D το 2006. Μέχρι το τέλος του 2007 και ο Mozilla και η Opera είχαν τις δικές τους ξεχωριστές εφαρμογές. Στις αρχές του 2009 η μη κερδοσκοπική κοινοπραξία Khronos, ξεκίνησε την ομάδα εργασίας WebGL με την αρχική συμμετοχή της Apple, Google, Mozilla, Opera και άλλες.

Η WebGL version 1.0 κυκλοφόρησε τον Μάρτιο του 2011 και από τον Μάρτιο του 2012 ο πρόεδρος της ομάδας εργασίας είναι ο Ken Russell. Η ανάπτυξη της WebGL 2 ξεκίνησε το 2013 και βασίζεται στην OpenGL ES 3.0 .

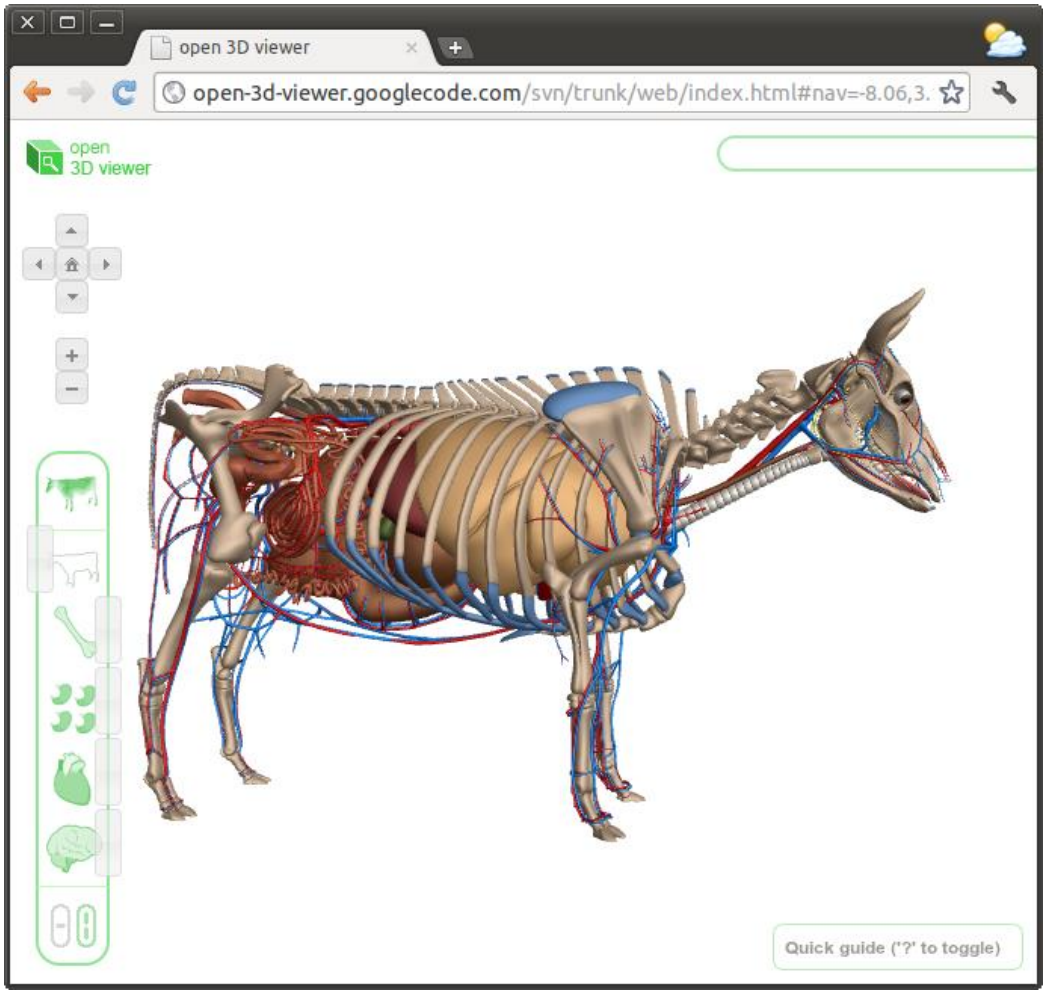
Η WebGL, είναι λογισμικό υπεύθυνο για να δώσει επιτάχυνση υλικού (hardware acceleration) και έτσι να μην χρειάζεται το X3D πρόσθετα λογισμικά πακέτα. Χρησιμοποιώντας την κάρτα γραφικών του υπολογιστή μας και τους οδηγούς της, παράγεται η σκηνή όπου μέσα της βρίσκονται τα X3D αντικείμενά μας.

1.4 Εφαρμογές Web Graphics Library

- **Google Maps** (δημοφιλή υπηρεσία χαρτών της Google, όπου είναι δυνατό να δούμε τον καιρό σε διάφορα μέρη του πλανήτη, τη μορφολογία του εδάφους, την κίνηση στους δρόμους κ.α).

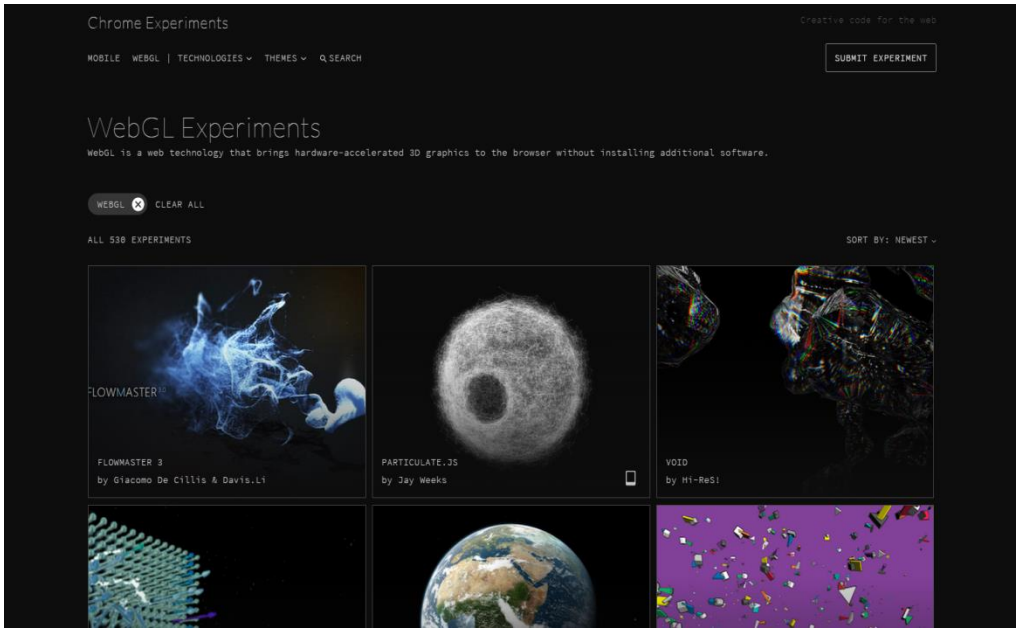
Εικόνα 2 - Google Maps

- **Zygote Body** (προηγουμένως Google body) είναι μια διαδικτυακή εφαρμογή όπου καθίστανται διαχειρίσιμα 3D ανατομικά μοντέλα του ανθρώπινου σώματος. Χρησιμοποιεί JavaScript και την τεχνολογία WebGL για να απεικόνιση 3D γραφικών στον web browser χωρίς την απαίτηση κάποιου ειδικού plugin.



Εικόνα 3 - Zygote Body

Επιπλέον μία μεγάλη γκάμα εφαρμογών μπορούν να βρουν οι χρήστες στην ιστοσελίδα http://www.chromeexperiments.com/webgl/ , όπου το Google Chrome Experiments είναι ένας online εκθεσιακός χώρος πειραμάτων, καλλιτεχνικών έργων και διαδραστικών προγραμμάτων. Ξεκίνησε το Μάρτιο του 2009 και είναι μια επίσημη σελίδα της Google που προοριζόταν αρχικά να δοκιμάσει τα όρια της JavaScript και τις ικανότητες του προγράμματος περιήγησης Google Chrome.

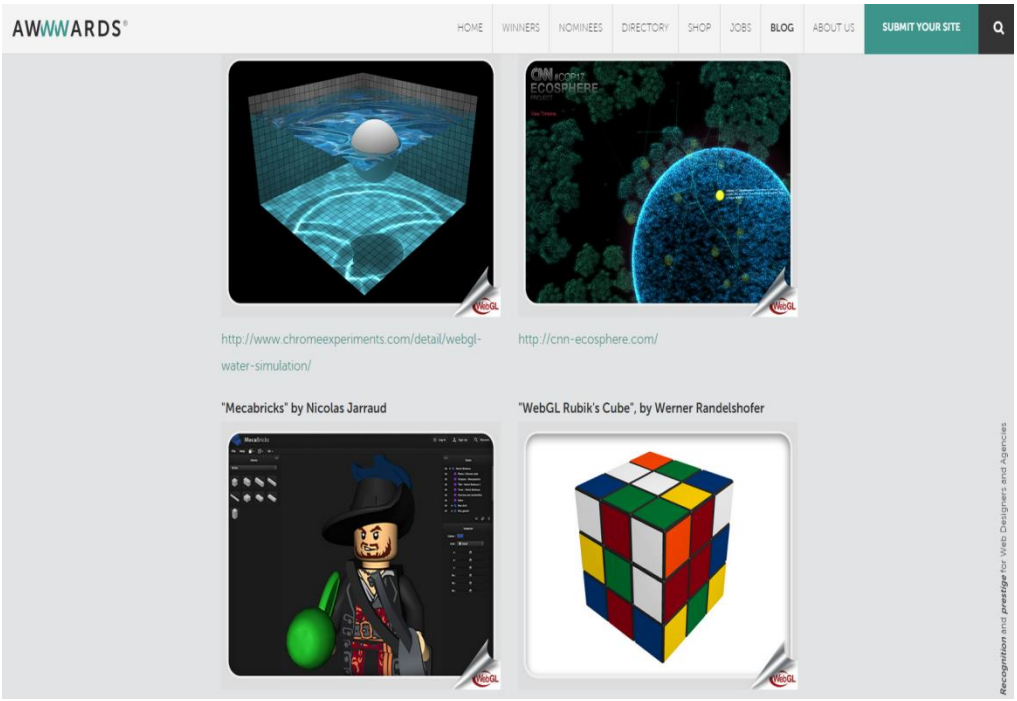


Εικόνα 4 - WebGL Experiments

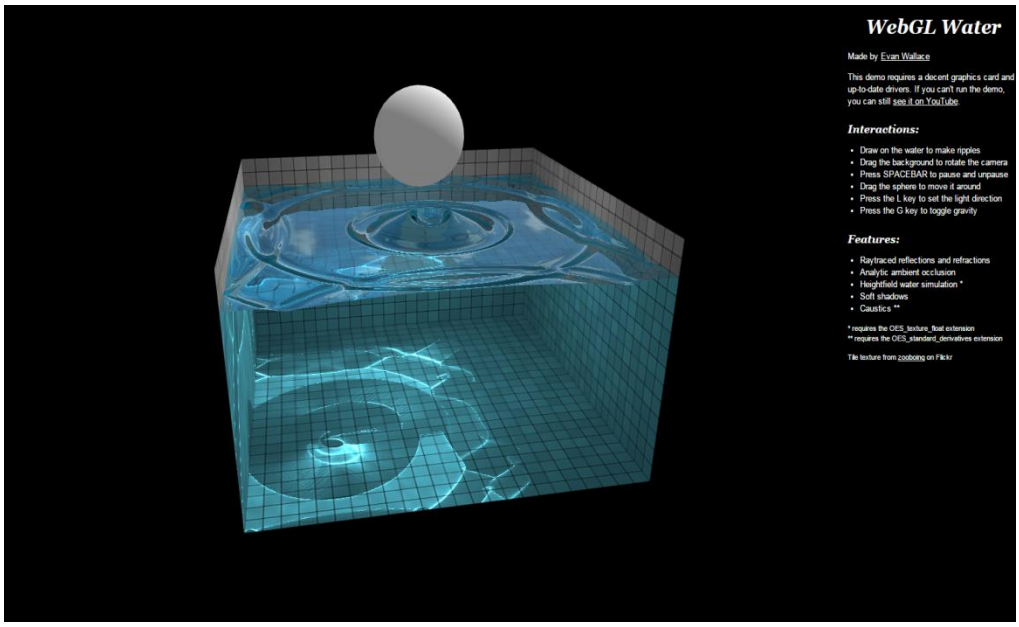
Καθώς το έργο προχώρησε, πήρε το ρόλο της ανάδειξης των τεχνολογιών ανοικτού πηγαίου κώδικα βασισμένων στο διαδίκτυο, όπως JavaScript, HTML5, WebGL, Canvas, SVG, CSS, και άλλα. Όλα τα έργα έχουν υποβληθεί από τους χρήστες και γίνονται με τη χρήση τεχνολογιών ανοιχτού κώδικα.

Από την 1 Αυγούστου του 2013, υπάρχουν περίπου 800 διαφορετικά έργα αναρτημένα στην ιστοσελίδα, όπου κάθε χρήστης του διαδικτύου μπορεί να βρει εντυπωσιακές εφαρμογές της τεχνολογίας όπως βουνά, πλανήτες, μέχρι και τα ανάκτορα των Βερσαλλιών να ξεδιπλώνονται μπροστά στα μάτια του.

Τέλος εφαρμογές της WebGL ,οι οποίες σε αντίθεση με το Google Chrome Experiments, που μπορεί να απεικονιστεί παρέχοντας την πλήρη λειτουργικότητα στον Google chrome, λειτουργούν σε όλους τους φυλλομετρητές, μπορεί να βρει κανείς και στην ιστοσελίδα http://www.awwwards.com/22-experimental-webgl-demo-examples.html . Υπάρχει πληθώρα εφαρμογών, από μαθηματικού τύπου μέχρι εξερεύνηση στα βάθη των ωκεανών, παροχή καινοτόμων τρόπων διερεύνησης του ανθρώπινου σώματος και απλών παιχνιδιών.



Εικόνα 5 - WebGL Demo Examples



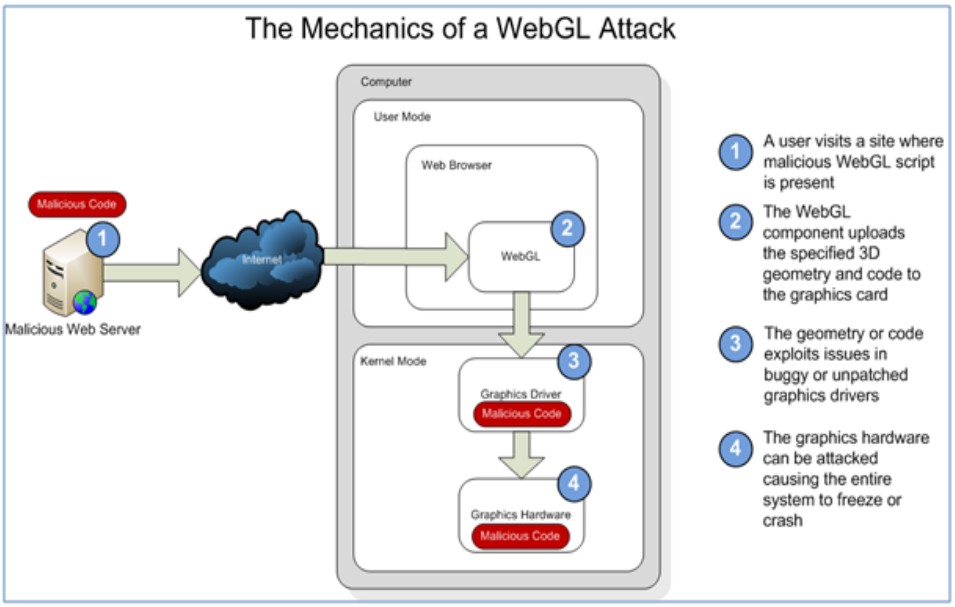
Εικόνα 6 - WebGL Water Παράδειγμα

1.5 Πιθανά μειονεκτήματα

Έχουν δημιουργηθεί ανησυχίες σχετικά με την ασφάλεια της πλατφόρμας και την ακεραιότητά της σε πιθανές μαζικές επιθέσεις. Κατά κύριο λόγο, το αποτέλεσμα μιας τέτοιας επίθεσης θα είναι η υπέρβαση των ορίων της κάρτας γραφικών, με άμεσο αποτέλεσμα την ολική αστάθεια του συστήματος και ενδεχομένως το σβήσιμο του υπολογιστή.

Με λίγα λόγια, θα μπορούσε μια κακόβουλη τοποθεσία, να παράσχει κώδικα μέσω ενός web browser απ' ευθείας στην GPU του υπολογιστή και των προγραμμάτων οδήγησης, καθιστώντας το μηχάνημα άχρηστο.

Από την άλλη η Khronos, η ομάδα που οργανώνει αυτό το πρότυπο, έχει πάρει θέση για το θέμα της ασφάλειας, επισημαίνοντας ότι υπάρχει μια επέκταση (extension) στη διάθεση των κατασκευαστών καρτών γραφικών που μπορεί να εντοπίσει και να αποτρέψει τις εν λόγω επιθέσεις. Σε μια νέα τεχνολογία, υπάρχουν πάντα θέματα τα οποία χρίζουν βελτίωσης, ωστόσο πρέπει να αρχίσει να χρησιμοποιείται ευρέως για να εντοπιστεί μια τυχόν τέτοια, αρκετά σοβαρού επιπέδου, αδυναμία.

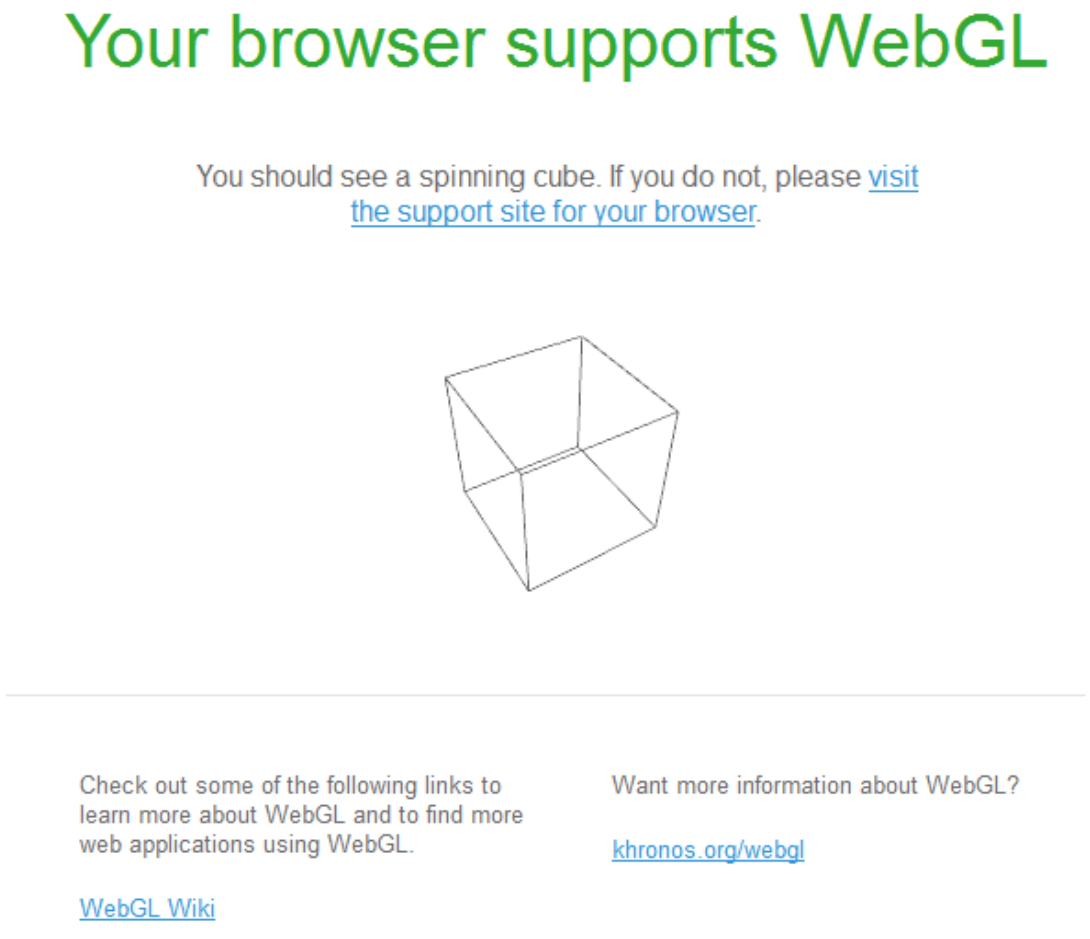


Εικόνα 7 - WebGL Επιθέσεις

ΚΕΦΑΛΑΙΟ 2

ΔΟΜΗ ΤΟΥ WEBGL

Η WebGL για να μπορέσει να λειτουργήσει και να αποδώσει σ' έναν υπολογιστή απαιτεί έναν φυλλομετρητή ιστοσελίδων που να υποστηρίζει WebGL και τις πιο ενημερωμένες εκδόσεις των drivers της κάρτας γραφικών.
Ο πιο εύκολος τρόπος να ελέγξουμε αν ο φυλλομετρητής μας υποστηρίζει τη WebGL, είναι να μεταβούμε στην ιστοσελίδα https://get.webgl.org. Σε περίπτωση που δεν υποστηρίζεται παρέχονται οδηγίες για την εύκολη ενεργοποίηση της.



Εικόνα 8 - Υποστήριξη της WebGL από τον Browser

Δε χρειάζεται να καταλάβουμε πλήρως τις εσωτερικές λειτουργίες του WebGL. Υπάρχουν αρκετές βιβλιοθήκες WebGL, που διατίθενται να

πάρουν μεγάλο μέρος της πολυπλοκότητας από τα χέρια μας. Ωστόσο η κατανόηση τους είναι χρήσιμη στην περίπτωση που θελήσουμε να εμπλουτίσουμε την εφαρμογή μας με επιπλέον χαρακτηριστικά, τα οποία μπορεί να μην παρέχονται στη βιβλιοθήκη της επιλογής μας.

Οι περισσότερες βιβλιοθήκες παρέχουν μια ποικιλία από έτοιμα μοντέλα, σκιές και άλλα, το οποίο σαφώς μειώνει δραστικά την ποσότητα του κώδικα που θα χρειαστεί να γράψουμε.

2.1 Ξεκινώντας με την WebGL

Υπάρχουν πολλές βιβλιοθήκες WebGL. Αυτό που κάνουν περισσότερο είναι να οικοδομούν πάνω από τη WebGL και να δημιουργούν τα αισθητικά στοιχεία σε ένα 3D περιβάλλον, όπως μια σκηνή, μια φωτογραφική μηχανή, μια πηγή φωτός, το φως του περιβάλλοντος, έτοιμα σχήματα, υλικά, υφές, σκιές, ομίχλη και αιωρούμενα σωματίδια και πολλές άλλες λειτουργίες. Η ιδέα αυτών των στοιχείων παραμένει λίγο πολύ η ίδια σε όλες τις βιβλιοθήκες. Για το πώς χρησιμοποιούνται, ωστόσο, αυτό εξαρτάται από την αρχιτεκτονική της βιβλιοθήκης.

2.2 Μερικές Web βιβλιοθήκες γραφικών

Επειδή η WebGL είναι low-level, η δημιουργία τρισδιάστατων χώρων, θα λέγαμε ότι απαιτεί σκληρή δουλειά. Γι' αυτό έχουν αναπτυχθεί 3D βιβλιοθήκες για υψηλού επιπέδου δυνατότητες. Κάποιες από αυτές παρατίθενται παρακάτω:

- Three.js : Είναι μια ελαφριά 3D μηχανή με πολύ χαμηλό επίπεδο πολυπλοκότητας. Η μηχανή χρησιμοποιεί το <canvas>, <svg> και WebGL. Είναι η πιο δημοφιλής βιβλιοθήκη, όσον αφορά τον αριθμό των χρηστών
- PhiloGL : Είναι χτισμένη με επίκεντρο τις καλές πρακτικές της JavaScript
- GLGE: Έχει κάποιες πιο σύνθετες λειτουργίες, όπως η σκελετική κίνηση και κινούμενα υλικά.

- J3D : Επιτρέπει όχι μόνο τη δημιουργία σκηνών, αλλά και η εξάγωγή αυτών των σκηνών από το Unity σε WebGL.

2.3 Βασική δομή HTML Document

Το πρώτο πράγμα που χρειάζεται για να χρησιμοποιηθεί η WebGL και να γίνει rendering (φωτορεαλιστική απόδοση) 3D είναι ένας καμβάς, χρησιμοποιώντας τη βιβλιοθήκη three.js που βρίσκεται στην κορυφή της WebGL και μας παρέχει όλες τις δυνατότητες για την κατασκευή ενός 3D κόσμου.

Το παρακάτω κομμάτι HTML δημιουργεί έναν καμβά:

```
<html>
    <head>
        <title> Πτυχιακή εργασία (Μανατάκης Ιωάννης) </title>
        <style>
            body { margin: 0; }
            canvas { width: 100%; height: 100% }
        </style>
    </head>
    <body>
        <script src="three.min.js"></script>
        <script></script>
    </body>
</html>
```

Για να μπορέσουμε να χρησιμοποιήσουμε το Three.js, θα πρέπει να το εμφανίσουμε στον κώδικα μας όπως φαίνετε παραπάνω(<script src="three.min.js"></script>). Αποθηκεύουμε το παραπάνω HTML σε ένα αρχείο στον υπολογιστεί μας, μαζί με ένα αντίγραφο της http://threejs.org/build/three.min.js (three.min.js) στον JS/ κατάλογο, και το ανοίγουμε στο πρόγραμμα περιήγησης μας.

2.4 Παράδειγμα κώδικα με την χρήση της Βιβλιοθήκης Three.js

Για να χρησιμοποιήσουμε την βιβλιοθήκη Three.js, η οποία είναι ένα αρχείο JavaScript , υπάρχουν 2 τρόποι :

- Κατεβάζουμε στον υπολογιστή μας ένα αντίγραφο της βιβλιοθήκης και αποθηκεύοντάς το με το όνομα three.js, το ενσωματώνουμε ως εξής :

<script src="three.min.js"></script>

- Εναλλακτικά μπορούμε να ενσωματώσουμε στη σελίδα μας ένα απομακρυσμένο αντίγραφο με το εξής τρόπο:

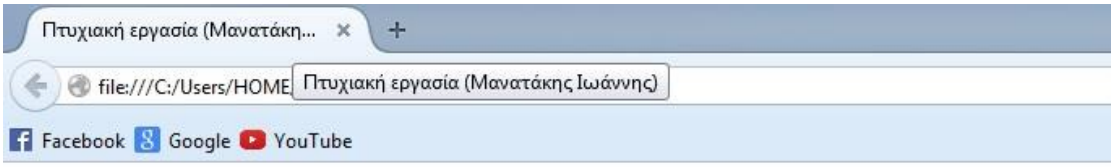
<script src="http://www.html5canvastutorials.com/libraries/three.min.js"> </script>

<u>Βασική δομή κώδικα της Three.js βιβλιοθήκης</u>

```
<!DOCTYPE html >

<html >
    <head>

        <meta charset=iso-8859-1" />
        <title>Πτυχιακή εργασία (Μανατάκης Ιωάννης) </title>

    </head>
    <body>

        <script
        src="http://www.html5canvastutorials.com/libraries/three.mi
        n.js"> </script>

        <script src="OrbitControls.js"></script>
        <script> // Our JavaScript will go here. </script>
```

```
    </body>
</html>
```

Στο <head>, δηλαδή στην κεφαλίδα του προγράμματός μας, τοποθετούμε τη σήμανση <meta>, όπου εμπεριέχονται οδηγίες για την σελίδα μας. Στη συγκεκριμένη, ορίζεται το σετ χαρακτήρων που θα χρησιμοποιηθεί για τις ελληνικές ιστοσελίδες. Η σήμανση <title>, ορίζει τον τίτλο που θα φαίνεται στον browser.



Εικόνα 9 - Σήμανση "title"

Τέλος, στο <body> ενσωματώνουμε τις τα εξής:

- Τη JavaScript 3D βιβλιοθήκη **three.js.**

- Το αρχείο **OrbitControls** , το οποίο ήδη έχουμε κατεβάσει στον υπολογιστή μας και χειρίζεται θέματα που αφορούν τον θεατή και την αλληλεπίδραση με το ποντίκι. Μας επιτρέπει να χρησιμοποιήσουμε το ποντίκι κάνοντας zoom-in και zoom-out με το mouse wheel, σύροντας κατά μήκους της οθόνης κ.α.

Ξεκινώντας το κώδικά μας είναι απαραίτητο να αναφέρουμε ότι τα βασικά στοιχεία για να μπορέσουμε να απεικονίσουμε κάτι με τη χρήση της three.js, είναι τα παρακάτω :

- Μια σκηνή - scene

- Μια κάμερα - camera
- Φωτορεαλιστική Απεικόνιση - renderer

Όμως εμείς θα το προχωρήσουμε λίγο πιο βαθειά. Θα χρησιμοποιήσουμε φωτισμό στα αντικείμενα καθώς και επιπλέον συναρτήσεις για το χειρισμό των αντικειμένων.
Αρχικά ορίζουμε μέσα στον <script> μας τις μεταβλητές που θα χρησιμοποιήσουμε στο πρόγραμμά μας και τις λειτουργίες των οποίων σαφώς θα επεξηγήσουμε αργότερα, όταν χρησιμοποιηθούν.

2.4.1 Προσθήκη σκηνής

Δημιουργούμε την σκηνή έτσι ώστε να μπορούμε να δώσουμε μια οπτική μορφή του αρχείου, Ουσιαστικά η σκηνή είναι ο χώρος πάνω στο οποίο θα βάλουμε όλα τα αντικείμενα μας και διάφορες άλλες λειτουργιές όπως φωτισμός, κάμερα.

var scene = new THREE.Scene();

2.4.2 Προσθήκη κάμερας

Για να μπορεί να προβληθεί ο τρισδιάστατος κόσμος που θα δημιουργηθεί, πρέπει να προστεθεί στη σκηνή μια κάμερα. Στη three.js υπάρχουν 2 ειδών κάμερες η ορθογραφικής (orthographic projection) και η προοπτικής (perspective projection) προβολής.

Όταν καθορίζουμε ένα είδος προβολής εξυπηρετούμε 2 σκοπούς :

- Θέτουμε στον κόσμο μας κάποια όρια και ότι είναι έξω από αυτά δεν θα κάνει render
- Θέτουμε το τρόπο με τον οποίο θα φαίνονται τα αντικείμενα μας

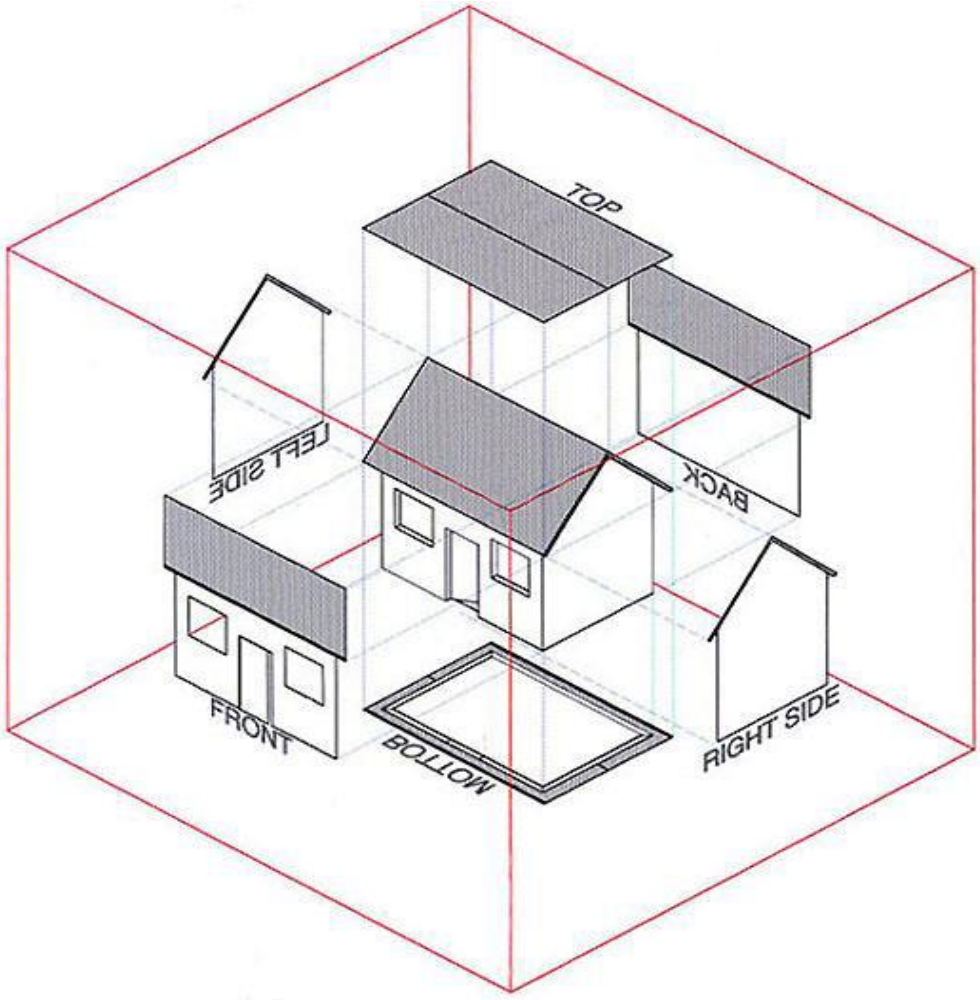
Orthographic Projection

- Όταν λέμε Ορθογραφική προβολή, στην ουσία εννοούμε ότι δεν έχουμε διαφορά ανάλογη της απόστασης από την κάμερα (Ουσιαστικά ισοδυναμεί με τη διαγραφή της 3ης διάστασης).

- Παρόλο που δεν είναι τόσο ρεαλιστικό όσο το άλλο είδος προβολής η Orthographic Projection έχει της χρήσεις της.
- Χρησιμοποιείτε συνήθως σε CAD εφαρμογές, δηλαδή αρχιτεκτονικού τύπου και 2D games.

OrthographicCamera (left, right, top, bottom, near, far)

Ορίζεται το οπτικό πεδίο ανάλογα με τη θέση πάνω στο επίπεδο



Εικόνα 10 - Orthographic Projection

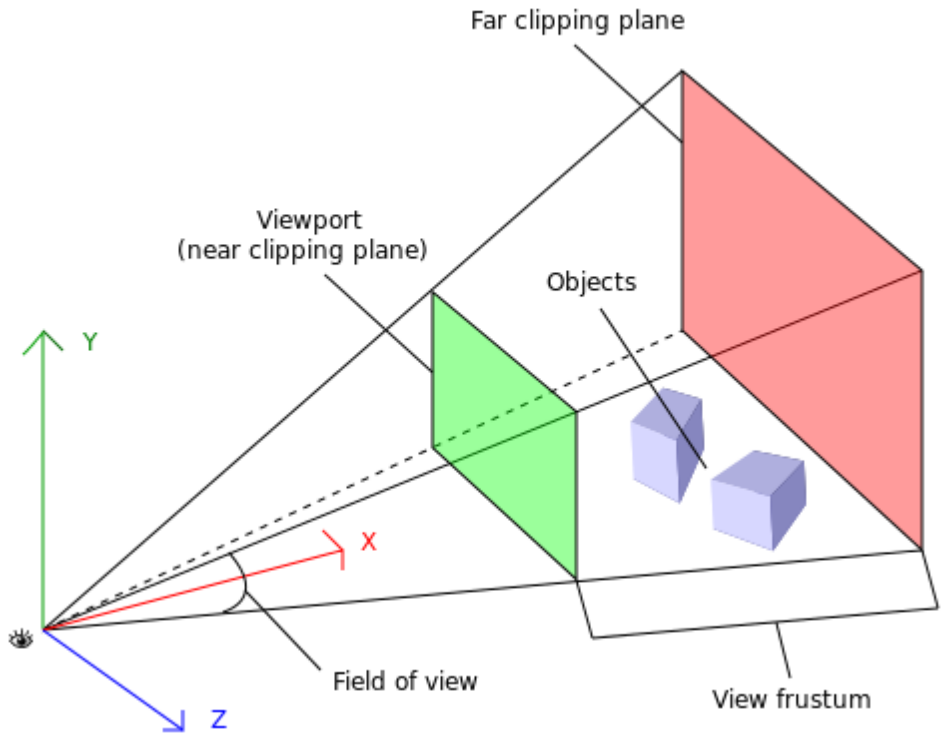
Perspective Projection

- Στην προοπτική προβολή έχουμε αλλαγή στα αντικείμενα μας, ανάλογη της απόστασης από την κάμερα (Ο τρόπος που βλέπει ο άνθρωπος και οι γάτες).

- Το πόσο μικρά ή μεγάλα φαίνονται τα αντικείμενα είναι ανάλογο της απόστασης.
- Το frustum (fov) που χρησιμοποιείται, στην ουσία είναι μια πυραμίδα με κομμένη κορυφή. Η κομμένη κορυφή είναι προς την κάμερα. Αυτό μας βοηθάει γιατί ανάλογα την θέση στο frustum τόση παραμόρφωση θα έχουν τα αντικείμενα.

PerspectiveCamera (fov, aspect, near, far)

Fov: Ορίζει το οπτικό πεδίο, από κάτω προς τα πάνω, σε μοίρες (πόσο μεγάλο θα φαίνεται το αντικείμενο).
Aspect: Ορίζει το πλάτος του παραθύρου, διαιρούμενο με το ύψος του παραθύρου.
Near: Τα αντικείμενα δε θα απεικονίζονται πιο κοντά από το near.
Far: Τα αντικείμενα δε θα απεικονίζονται πιο μακριά από το far.



Εικόνα 11 - Perspective Projection

Στην εφαρμογή που παρουσιάζεται, χρησιμοποιείται η προοπτική προβολή, όπου θέτουμε στην κάμερα μια θέση πάνω στον xyz άξονα, «γυρνάμε» την κάμερα να κοιτάει προς την σκηνή και την προσθέτουμε

σε αυτή. Παρακάτω βλέπουμε μια γραμμή κώδικα όπου δημιουργούμε την κάμερα και της δίνουμε τιμές έτσι ώστε να ρυθμίσουμε πως θα βλέπουμε την σκηνή μας.

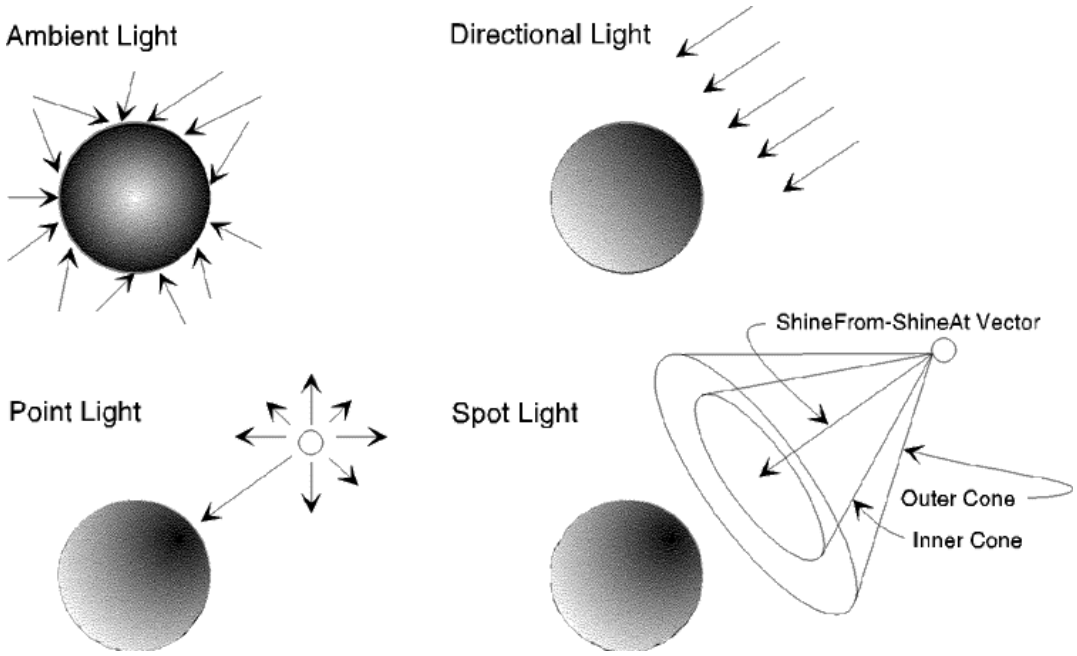
```
var camera = new THREE.PerspectiveCamera( 100, window.innerWidth / window.innerHeight, 0.1, 1000 );
```

2.4.3 Προσθήκη Φωτισμού

Το επόμενο σημείο στο οποίο πρέπει να σταθούμε , είναι ο φωτισμός και οι σκιές πάνω στο αντικείμενο μας. Σαφώς υπάρχουν διαφορετικοί τρόποι φωτισμού.

Κάποια είδη φωτισμού είναι τα εξής :

- **DirectionalLight**, το οποίο επηρεάζει μόνο MeshLambertMaterial ή MeshPhongMaterial και αν θέλουμε να βάλουμε σκιές θα πρέπει να τοποθετηθεί και μια orthographic camera.
- **SpotLight**, ένα φως που λάμπει από ένα δεδομένο σημείο σε μια κατεύθυνση και σχηματίζει ένα σχήμα κώνου.
- **Ambient Light,** το χρώμα από το φως εφαρμόζεται σε όλα τα αντικείμενα της σκηνής συνολικά.
- **PointLight**, δημιουργεί ένα φως σε μία συγκεκριμένη θέση στη σκηνή. Το φως λάμπει προς όλες τις κατευθύνσεις (κατά προσέγγιση παρόμοια με μια λάμπα).



Εικόνα 12 - Είδη φωτισμών

Η δομή των φωτισμών έχει ως εξής :

AmbientLight (hex)

hex: Αριθμητική τιμή του στοιχείο RGB του χρώματος.

Αυτό το φως παίρνει ένα χρώμα και εφαρμόζεται σε όλα τα αντικείμενα της σκηνής.

DirectionalLight (hex, intensity)

hex: Η αριθμητική τιμή του πρότυπου χρώματος RGB.

intensity: Η αριθμητική τιμή της δύναμης/έντασης του φωτός.

Δημιουργείται ένας φωτισμός που λάμπει από μια συγκεκριμένη κατεύθυνση και όχι από μια συγκεκριμένη θέση. Συμπεριφέρεται σαν να είναι απείρως μακριά και οι ακτίνες που παράγονται από αυτό είναι παράλληλες. Είναι μια πηγή φωτός που λειτουργεί σαν τον ήλιο: ο ήλιος είναι τόσο μακριά ώστε όλο το ηλιακό φως προέρχεται από την ίδια οπτική γωνία.

PointLight(hex, intensity, distance)

hex: Αριθμητική τιμή του στοιχείο RGB του χρώματος.

intensity: Αριθμητική τιμή του φωτός αντοχή / ένταση.

distance: Η απόσταση του φωτός.

SpotLight (hex, intensity, distance, angle, exponent)

hex: Η αριθμητική τιμή του πρότυπου χρώματος RGB.

intensity: Η αριθμητική τιμή της δύναμης/έντασης του φωτός.

Distance: Μέγιστη απόσταση από την πηγή, όπου η ένταση του φωτός θα εξασθενεί γραμμικά ανάλογα με την απόσταση από την πηγή.

angle: Ο μέγιστος βαθμός σε ακτίνια από την κατεύθυνση , του οποίου το ανώτερο όριο είναι Math.PI / 2. Η default τιμή του είναι Math.PI / 3.

exponent: Η ταχύτητα της πτώσης του φωτός με default τιμή 10.

2.4.4 Δημιουργία Αντικειμένων

Τώρα για να δημιουργήσουμε αντικείμενα στην σκηνή μας προσθέτουμε τον παρακάτω κώδικα, αυτός ο κώδικας προσθέτει έναν κύβο. Το BoxGeometry είναι μια τετράπλευρη κατηγορία γεωμετρίας, συνήθως χρησιμοποιείται για τη δημιουργία ενός κύβου ή ακανόνιστου τετράπλευρου, με τις διαστάσεις που δίνονται, δηλαδή το «πλάτος», «ύψος», και «βάθος». Όπως βλέπουμε στα δεξιά της πρώτης γραμμής του κώδικα που βρίσκεται παρακάτω, δίνουμε της τρείς αυτές διαστάσεις που θέλουμε να έχει ο κύβος.

- Πλάτος: Πλάτος των πλευρών επί του X άξονα.
- ύψος: Ύψος των πλευρών επί του Y άξονα.
- βάθος: Βάθος των πλευρών στον άξονα Z.

Στην δεύτερη γραμμή δίνουμε το χρώμα το οποίο θα έχει το σχήμα μας, χρησιμοποιώντας την MeshBasicMaterial, στην δική μας περίπτωση είναι ανοικτό πράσινο. Εάν θέλουμε μπορούμε να χρησιμοποιήσουμε την ιδιότητα (wireframe: true), έτσι ώστε να βλέπουμε το σκελετό του κύβου.

```
var geometry = new THREE.BoxGeometry( 2, 2, 2 );
var material = new THREE.MeshBasicMaterial( { color: 0x00FF00 } );
var cube = new THREE.Mesh( geometry, material );
scene.add( cube );
```

2.4.5 Προσθήκη Φωτορεαλιστικής Απόδοσης (renderer)

Παρακάτω βλέπουμε μια απλή συνάρτηση η οποία δίνει ζωή στην σκηνή μας, έχοντας δώσει τα κατάλληλα στοιχεία ο κύβος περιστρέφετε. Με του άξονες x, y, z όπου έχουμε μπορούμε να περιστρέψουμε τον κύβο μας προς όποια κατεύθυνση θέλουμε. Επίσης με τους αριθμούς στα δεξιά του κώδικα της συνάρτησης (cube.rotation.x += **0.01**;) βάζουμε την ταχύτητα όπου θέλουμε να περιστρέφετε ο κύβος μας στον κάθε άξονα.

```
function render()
    {
        requestAnimationFrame( render );
         cube.rotation.x += 0.01;
         cube.rotation.y += 0.01;
        cube.rotation.z += 0.01;
        renderer.render( scene, camera );
    }
```

Στην συνεχεία καλούμε την συνάρτηση μας:

render();

2.4.6 Προσθήκη των Controls της OrBitControls

Όπως προαναφερθήκαμε, για να μπορέσουμε να έχουμε αλληλεπίδραση με το ποντίκι, πρέπει να τοποθετηθούν τα controls που έχουμε ενσωματώσει, επίσης καθορίζουμε και την ταχύτητα που θέλουμε να κινείται η κάμερα καθώς περιστρέφουμε τη σκηνή μας μέσω του ποντικιού. Για να δουλέψει η ιδιότητα αυτή πρέπει να φορτώσουμε το κατάλληλο αρχείο (<script src="OrbitControls.js"></script>).

```
controls = new THREE.OrbitControls(camera);

controls.userPanSpeed = 0.1;

controls.noPan = false;

controls.keyPanSpeed = 7.0;
```

2.5 Παραδείγματα προγραμμάτων σε Three.js

2.5.1 Πρώτο παράδειγμα προγράμματος

Έχοντας αναλύσει όλα τα βασικά στοιχεία όπου χρειαζόμαστε, είμαστε έτοιμοι να δούμε ολοκληρωμένο ένα μικρό πρόγραμμα. Καταρχήν δημιουργούμε την σκηνή μας και μετά προσθέτουμε την κάμερα. Στην συνέχεια αρχίζουμε να προσθέτουμε τα αντικείμενα που θέλουμε, στο παράδειγμα αυτό θα προσθέσουμε έναν κύβο ο οποίος περιστρέφετε γύρο από τους άξονες του.

Κάποιες επιπλέον επιλογές όπου δεν έχουμε αναλύσει έως τώρα είναι οι ιδιότητες Window Resize, και η FullScreen όπου θα της χρησιμοποιήσουμε στο παρακάτω παράδειγμα. Για να λειτουργήσουν πρέπει να δηλώσουμε στο Head του προγράμματος τα κατάλληλα αρχεία

όπου παρέχουν επιπλέον κώδικα όπου χρειαζόμαστε χωρίς να είναι ανάγκη να τον γράφουμε εμείς.

```
<script src="vendor/threex/THREEx.FullScreen.js"></script>

<script src="vendor/threex/THREEx.WindowResize.js"></script>
```

Window Resize:

Η Window Resize αυτό που κάνει είναι να αλλάζει το μέγεθος της εικόνας του προγράμματος καθώς τρέχει, όταν ο χρήστης αλλάζει το μέγεθος του φυλλομετρητή.

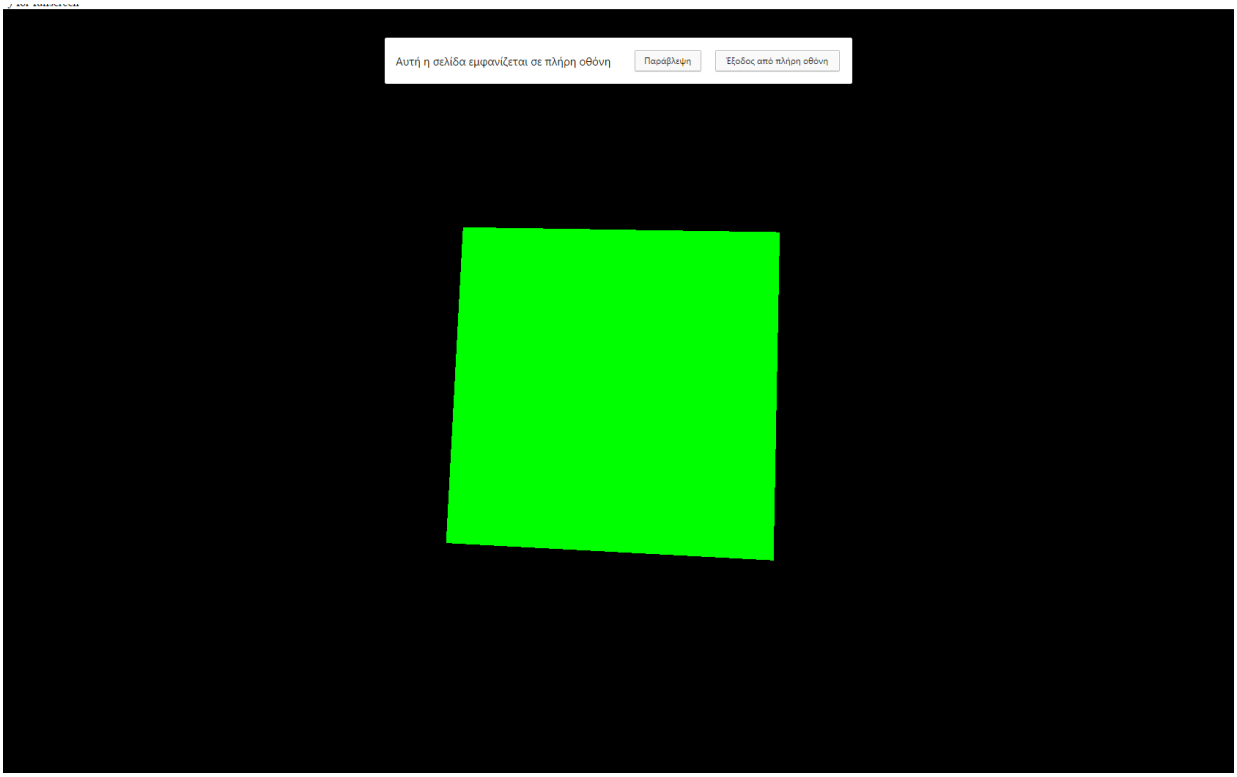
```
THREEx.WindowResize.bind(renderer, camera);
```

FullScreen:

Η FullScreen κατά την διάρκεια όπου εκτελείτε το πρόγραμμα μας εάν πατήσουμε “f” τότε η εφαρμογή μπαίνει σε λειτουργία πλήρης οθόνης.

```
if( THREEx.FullScreen.available() )
        {
          THREEx.FullScreen.bindKey();
          document.getElementById('inlineDoc').innerHTML  += "-<i>f </i> for
               fullscreen";
         }
```

Στην παρακάτω εικόνα φαίνεται το πρώτο μας πρόγραμμα.



Εικόνα 13 - Πρώτο παράδειγμα

Όλος ο κώδικας του πρώτου προγράμματος.

```
<!doctype html>
<html>
	<head>
		<title>Ptyxiakh Ergasia Manatakh Iwannh</title>


		<style>
			body { margin: 0; }
			canvas { width: 100%; height: 100% }
		</style>

		<script src="vendor/threex/THREEx.FullScreen.js"></script>
		<script src="vendor/threex/THREEx.WindowResize.js"></script>
```

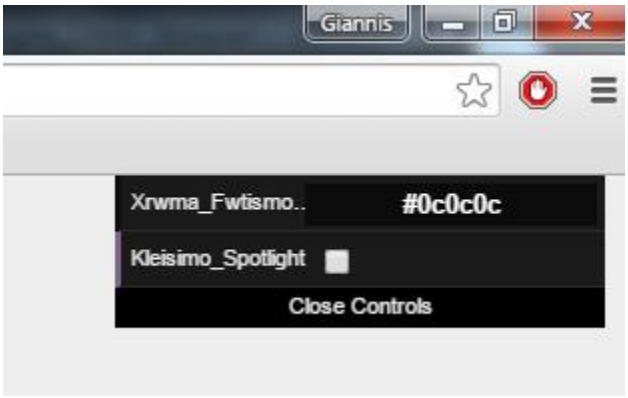
```
        <link  href="css/main.css" rel="stylesheet"/>

</head>
<body>

        <div class="top" id="inlineDoc"></div>

   <script src="three.min.js"></script>

   <script>
                    var scene = new THREE.Scene();
                    var camera = new THREE.PerspectiveCamera( 100,
                   window.innerWidth / window.innerHeight, 0.1, 1000 );

                   var renderer = new THREE.WebGLRenderer();
                    renderer.setSize( window.innerWidth, window.innerHeight );
                    document.body.appendChild( renderer.domElement );

                   var geometry = new THREE.BoxGeometry( 2, 2, 2 );
                   var material = new THREE.MeshBasicMaterial( { color:
                   0x00FF00} );
                    var cube = new THREE.Mesh( geometry, material );

                    scene.add( cube );

                   camera.position.z = 5;

                   function render()
                            {
                                    requestAnimationFrame( render )
                                    cube.rotation.x += 0.01;
                                    cube.rotation.y += 0.01;
                                    cube.rotation.z += 0.01;

                                   renderer.render( scene, camera );
                            }
                    render();

                    //Window Resize
                   THREEx.WindowResize.bind(renderer, camera);

                   //FullScreen
                   if( THREEx.FullScreen.available() )
                           {
                                   THREEx.FullScreen.bindKey();
                                document.getElementById('inlineDoc').innerHTML
                               += "- <i> f </i> for fullscreen";
                           }
```

```
        </script>
    </body>
</html>
```

2.5.2 Δεύτερο παράδειγμα προγράμματος

Στο δεύτερο παράδειγμα μας θα δημιουργήσουμε αρχικά την σκηνή μας και στην συνέχεια θα τοποθετήσουμε δυο αντικείμενα, έναν κύβο και μια σφαίρα. Όπου ο κύβος περιστρέφετε γύρο από τους άξονες του, και η σφαίρα όπου κινείται δεξιά και αριστερά στον άξονα x και πηγαίνει πάνω και κάτω στον άξονα y. Οι φωτισμοί όπου θα χρησιμοποιήσουμε είναι ο φωτισμός (Ambient Light) περιβάλλοντος όπου δίνει ένα απαλό χρώμα σε όλη την σκηνή μας, και ο δεύτερος είναι ο φωτισμός (Spotlight) όπου φωτίζει στο σημείο στο όποιο τον έχουμε στρέψει εμείς να κοιτάζει, στο παράδειγμα μας κοιτάζει την σκηνή μας από τα αριστερά.

Επίσης έχουμε πρόσθεση κάποια Controls στα δεξιά της οθόνης από τα οποία μπορούμε να αλλάξουμε τον χρώμα του φωτισμού περιβάλλοντος, και να απενεργοποιήσουμε το SpotLight.



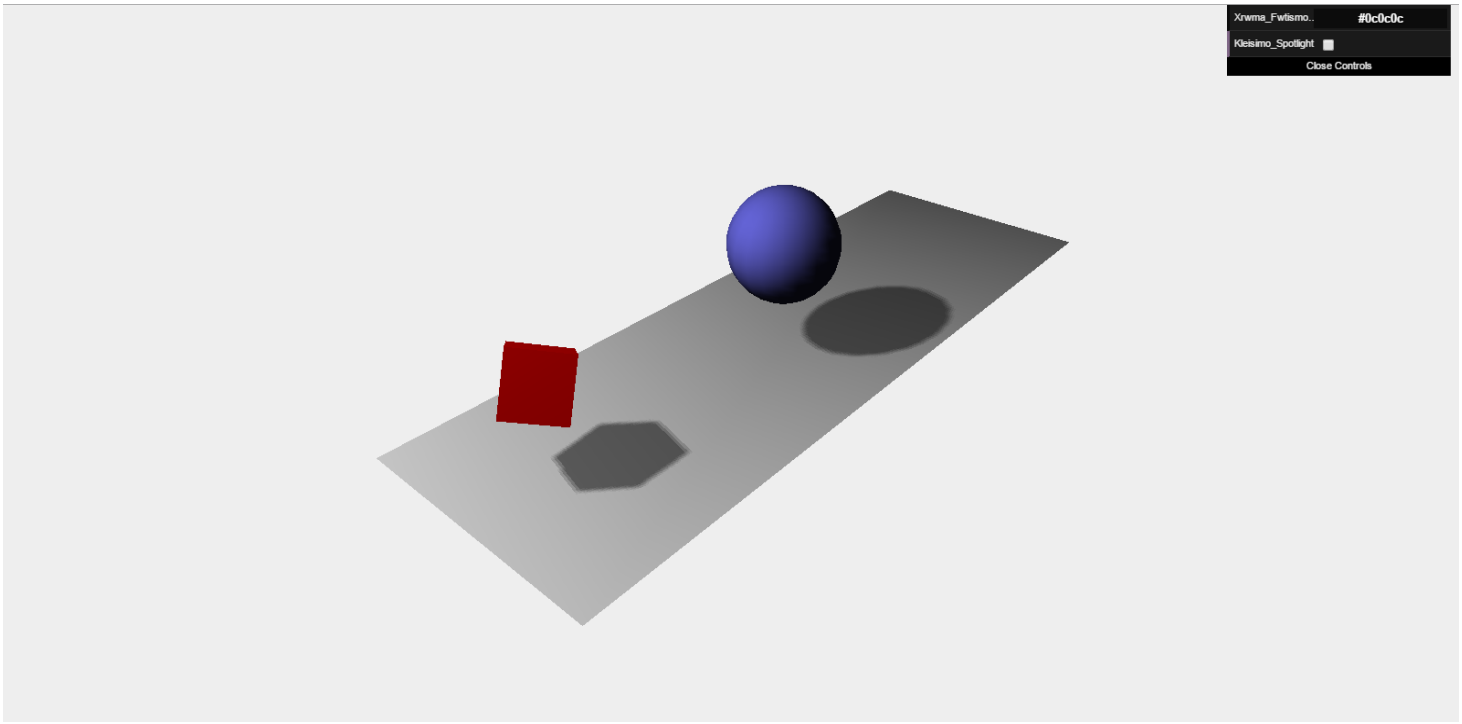
Εικόνα 14 - Controls

Για να προσθέσουμε αυτά τα Controls στην εφαρμογή μας πρέπει να φορτώσουμε πρώτα το αρχείο “dat.gui.js” όπου έχει διάφορα τέτοια Controls, εμείς βέβαια χρησιμοποιούμε αυτά που χρειαζόμαστε.

Κώδικας για την φόρτωση του αρχείου.

```
<script type="text/javascript" src="dat.gui.js"></script>
```

Στην παρακάτω εικόνα βλέπουμε το πρόγραμμα ολοκληρωμένο.



Εικόνα 15 - Δεύτερο Παράδειγμα

Όλος ο κώδικας του δεύτερου προγράμματος.

```
<!DOCTYPE html>
<html>
<head>
        <title>Ptyxiakh Ergasia Manatakh Iwannh</title>

        <script type="text/javascript" src="three.min.js"></script>

        <script type="text/javascript" src="dat.gui.js"></script>
        <style>
```

```
    body
        {
            margin: 0;
            overflow: hidden;
        }
        </style>
</head>

<body>


        <div id="WebGL-output">
        </div>

        <script type="text/javascript">

        function init()
            {

            // Dhmhourgia skhnhs opou tha topothetithoun
            antikeimena, kameres kai fwta
            var scene = new THREE.Scene();

            // Dhmiourgia cameras
```

```
var camera = new THREE.PerspectiveCamera(45,
window.innerWidth / window.innerHeight, 0.1, 1000);


// Dhmhourgia render

var renderer = new THREE.WebGLRenderer();

renderer.setClearColor(new
THREE.Color(0xEEEEEE, 1.0));

renderer.setSize(window.innerWidth,
window.innerHeight);

renderer.shadowMapEnabled = true;


// Dhmhourgia edafous

var planeGeometry = new THREE.PlaneGeometry(60,
20, 1, 1);

var planeMaterial = new
THREE.MeshLambertMaterial({color: 0xffffff});

var plane = new THREE.Mesh(planeGeometry,
planeMaterial);

plane.receiveShadow = true;


// Peristrofh kai topothethsh edafous

plane.rotation.x = -0.5 * Math.PI;

plane.position.x = 17;

plane.position.y = 0;

plane.position.z = 0;


// add edafous sthn skhnh
```

```
scene.add(plane);


// Dhmiourgia kuvou

var cubeGeometry = new THREE.BoxGeometry(4, 4,
4);

var cubeMaterial = new
THREE.MeshLambertMaterial({color: 0xff0000});

var cube = new THREE.Mesh(cubeGeometry,
cubeMaterial);

cube.castShadow = true;


// Thesh tou kuvou

cube.position.x = -7;

cube.position.y = 7;

cube.position.z = 0;


// add the cube

scene.add(cube);


//Dhmhourgia sferas.

var sphereGeometry = new
THREE.SphereGeometry(4, 20, 20);

var sphereMaterial = new
THREE.MeshLambertMaterial({color: 0x7777ff});

var sphere = new THREE.Mesh(sphereGeometry,
sphereMaterial);
```

```
// Thesh sfairas
sphere.position.x = 20;
sphere.position.y = 0;
sphere.position.z = 2;
sphere.castShadow = true;

// add sfaira
scene.add(sphere);

// Thesh ths kamera sthn skhnh
camera.position.x = -30;
camera.position.y = 34;
camera.position.z = 35;
camera.lookAt(new THREE.Vector3(10, 0, 0));

//Prosthikh apalou fwtismou peribalontos
var ambiColor = "#0c0c0c";
var ambientLight = new
THREE.AmbientLight(ambiColor);
scene.add(ambientLight);

// Prosthikh spotlight
var spotLight = new THREE.SpotLight(0xffffff);
spotLight.position.set(-40, 60, -10);
spotLight.castShadow = true;
```

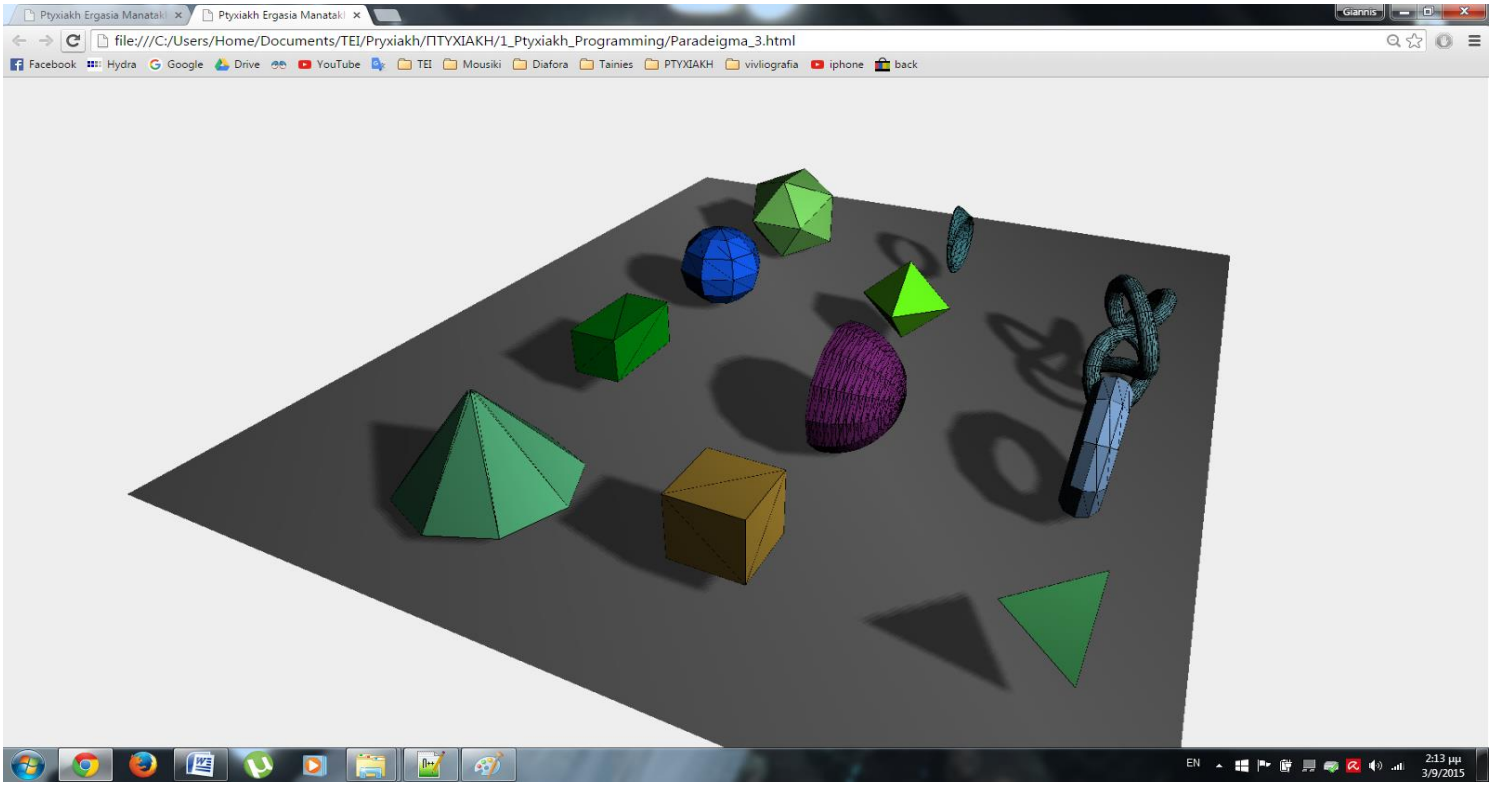
```
scene.add(spotLight);

document.getElementById("WebGL-
output").appendChild(renderer.domElement);



var step = 0;

var controls = new function ()

    {

    this.rotationSpeed = 0.02;

    this.bouncingSpeed = 0.03;

    this. Xrwma_Fwtismou_Perivalontos =
    ambiColor;

    this. Kleisimo_Spotlight = false;

    };

 //Dhmhourgia Controls sthn othoni

var gui = new dat.GUI();

gui.addColor(controls,
'Xrwma_Fwtismou_Perivalontos').onChange(function
(e) {

ambientLight.color = new THREE.Color(e);

});

gui.add(controls,
'Kleisimo_Spotlight).onChange(function (e) {

spotLight.visible = !e;

});

//Kalesma Render synarthshs
```

```
            render();


            //Synarthsh Render
            function render()
                {


                // Peristrofh tou kuvou gyro apo tous aksones
                cube.rotation.x += controls.rotationSpeed;
                cube.rotation.y += controls.rotationSpeed;
                cube.rotation.z += controls.rotationSpeed;
                // Anaphdhsh ths sferas panw katw
                step += controls.bouncingSpeed;
                sphere.position.x = 20 + ( 10 *
                (Math.cos(step)));
                sphere.position.y = 2 + ( 10 *
                Math.abs(Math.sin(step)));
                // render using requestAnimationFrame
                requestAnimationFrame(render);
                renderer.render(scene, camera);
                }
            }
            window.onload = init
        </script>
    </body>
</html>
```

2.5.3 Τρίτο παράδειγμα προγράμματος

Στο τρίτο παράδειγμα μας θα δημιουργήσουμε αρχικά την σκηνή μας και στην συνέχεια θα τοποθετήσουμε έντεκα αντικείμενα. Μια πυραμίδα, ορθογώνιο, σφαίρα, πολύγωνο, κύβο, μισή σφαίρα, εξάγωνο, κύκλο, τρίγωνο, τόρο, και έναν μεγάλο τορο. Οι φωτισμοί όπου θα χρησιμοποιήσουμε είναι ο φωτισμός (Ambient Light) περιβάλλοντος όπου δίνει ένα απαλό χρώμα σε όλη την σκηνή μας, και ο δεύτερος είναι ο φωτισμός (Spotlight) όπου φωτίζει στο σημείο στο όποιο τον έχουμε στρέψει εμείς να κοιτάζει.

Στην παρακάτω εικόνα βλέπουμε το πρόγραμμα.



Εικόνα 16 - Τρίτο παράδειγμα

Όλος ο κώδικας του τρίτου προγράμματος.

```
<!DOCTYPE html>
<html>
	<head>
		<title>Ptyxiakh Ergasia Manatakh Iwannh</title>

		<script type="text/javascript" src="three.min.js"></script>
		<script type="text/javascript" src="ParametricGeometries.js"></script>
		<script type="text/javascript" src="ConvexGeometry.js"></script>
		<script type="text/javascript" src="jquery-1.9.0.js"></script>
		<script type="text/javascript" src="dat.gui.js"></script>
		<style>
			body{
				margin: 0;
				overflow: hidden;
				}
		</style>

	</head>
	<body>

		<div id="WebGL-output">
		</div>
```

```
<script type="text/javascript">

    $(function ()
        {

        // Dhmhourgia skhnhs opou tha topothetithoun antikeimena, kameres kai fwta.
        var scene = new THREE.Scene();

        // Dhmiourgia cameras.
        var camera = new THREE.PerspectiveCamera(45, window.innerWidth / window.innerHeight, 0.1, 1000);

        // Dhmhourgia render.
        var renderer = new THREE.WebGLRenderer();
        renderer.setClearColor(new THREE.Color(0xEEEEEE, 1.0));
        renderer.setSize(window.innerWidth, window.innerHeight);
        renderer.shadowMapEnabled = true;

        // Dhmhourgia edafous
        var planeGeometry = new THREE.PlaneGeometry(65,52,1,1);
        var planeMaterial = new THREE.MeshLambertMaterial({color: 0xffffff});
```

```
var plane = new THREE.Mesh(planeGeometry,planeMaterial);

plane.receiveShadow  = true;


// Peristrofh kai topothethsh edafous.

plane.rotation.x=-0.5*Math.PI;

plane.position.x=0

plane.position.y=0

plane.position.z=0


// add edafous sthn skhnh.

scene.add(plane);


// Thesh ths kamera sthn skhnh

camera.position.x = -50;

camera.position.y = 30;

camera.position.z = 20;

camera.lookAt(new THREE.Vector3(-10,0,0));


//Prosthikh apalou fwtismou peribalontos

var ambientLight = new THREE.AmbientLight(0x090909);

scene.add(ambientLight);


// Prosthikh spotlight fwtismou

var spotLight = new THREE.SpotLight( 0xffffff );

spotLight.position.set( -40, 50, 60 );
```

```
spotLight.castShadow = true;

scene.add( spotLight );


// add Gewmetries

addGeometries(scene);


// add the output of the renderer to the html element

$("#WebGL-output").append(renderer.domElement);


//Kalesma Render synarthshs

var step=0;

render();


//Synarthsh Opou Dhmiourgh olla ta antikeimena sthn skhnh mas.

function addGeometries(scene)

        {

        var geoms = [];


        //dhmhourgia polugwnhs puramhdas.

        geoms.push(new THREE.CylinderGeometry(0.2,5,6));


        //Dhmhourgia orthogwniou parallhlogramou.
```

```
geoms.push(new
THREE.CubeGeometry(6,3,3));


//Dhmhourgia sferas

geoms.push(new
THREE.SphereGeometry(3));


//Dhmhourgia polugwnou

geoms.push(new
THREE.IcosahedronGeometry(4)
);


//Dhmhourgia kuvou

var points = [

        new  THREE.Vector3(  2,
        2, 2 ),

        new  THREE.Vector3(  2,
        2, -2 ),

        new  THREE.Vector3(  -2,
        2, -2 ),

        new  THREE.Vector3(  -2,
        2, 2 ),

        new THREE.Vector3( 2, -
        2, 2 ),

        new THREE.Vector3( 2, -
        2, -2 ),

        new  THREE.Vector3(  -2,
        -2, -2 ),

        new  THREE.Vector3(  -2,
        -2, 2 )

        ];
```

```
geoms.push(new
THREE.ConvexGeometry(points
));


// Dhmhourgia mishs sferas.

var pts = [];

//Leptomeria kampulhs.

var detail = .1;

//Megethos.

var radius = 4;

//Gwnia.

for(var angle = 0.0; angle <
Math.PI ; angle+= detail)

pts.push(new
THREE.Vector3(Math.cos(angle)
* radius,0,Math.sin(angle) *
radius));

geoms.push(new
THREE.LatheGeometry( pts, 12
));


//Dhmhourgia 6gwnou.

geoms.push(new
THREE.OctahedronGeometry(3)
);


//Dhmhourgia kuklou.

geoms.push(new
THREE.ParametricGeometry(
THREE.ParametricGeometries.m
obius3d, 20, 10 ));
```

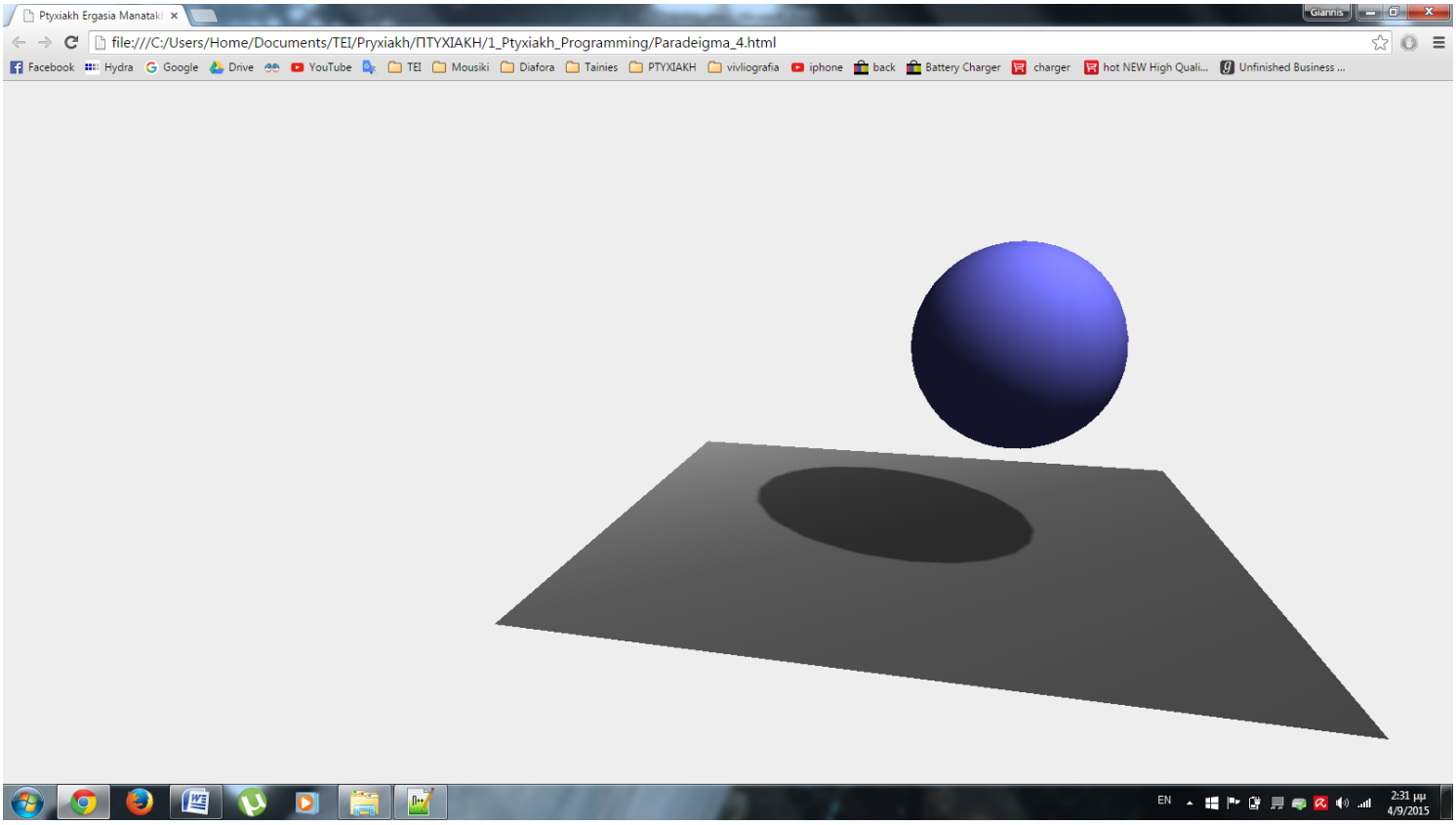
```
//Dhmhourgia Trigwnou.

geoms.push(new
THREE.TetrahedronGeometry(3)
);


//Dhmiourgia torou

geoms.push(new
THREE.TorusGeometry(3,1,10,1
0));


//Dhmhourgia megalou torou

geoms.push(new
THREE.TorusKnotGeometry(3,0
.5,50,20));


var j=0;

//Topothetish anteikeimenon sthn
skhnh mas.

for (var i = 0 ; i < geoms.length ;
i++)

        {

var cubeMaterial = new
THREE.MeshLambertMaterial({
wireframe: true, color:
Math.random() * 0xffffff });

        var materials = [

                new
THREE.MeshLambertMaterial( {
color:  Math.random() * 0xffffff,
shading: THREE.FlatShading } ),

                new
THREE.MeshBasicMaterial(    {
color: 0x000000, wireframe: true
} )
```

```
                                                    ];
                                                    var mesh = THREE.SceneUtils.createMultiMaterialObject(geoms[i],materials);
                                                    mesh.traverse(function(e) {e.castShadow=true});
                                                    mesh.position.x=-24+((i%4)*13);
                                                    mesh.position.y=4.3;
                                                    mesh.position.z=-8+(j*14);
                                                    if ((i+1)%4==0) j++;
                                                    scene.add(mesh);
                                                    }
                                            }
                                    // Render Synarthsh
                                    function render()
                                            {

                                            requestAnimationFrame(render);
                                            renderer.render(scene, camera);
                                            }

                                    });
                            </script>
                    </body>
            </html>
```

2.5.4 Τέταρτο παράδειγμα προγράμματος

Στο τέταρτο παράδειγμα μας έχουμε δημιουργήσει μια σκηνή όπου τοποθετήσαμε σε αυτήν την κάμερα, το έδαφος, και μια σφαίρα. Η σφαίρα κινείτε δεξιά και αριστερά, καθώς πέφτει πάνω τους ο φωτισμός SpotLight και AmbientLight δημιουργώντας της αντίστοιχες σκιές πάνω στο έδαφος. Όλα αυτά όμως τα έχουμε δει στα προηγούμενα παραδείγματα μας, αυτό που θα δούμε εδώ είναι την περιστροφή της κάμερας σε συνδυασμό με την αναπήδηση της σφαίρας δημιουργεί μια φανταστική εικόνα.

Στην παρακάτω εικόνα βλέπουμε το πρόγραμμα.



Εικόνα 17 - τέταρτο παράδειγμα

Όλος ο κώδικας του τέταρτου προγράμματος.

```
<!DOCTYPE html>
<html>
	<head>
		<title>Ptyxiakh Ergasia Manatakh Iwannh</title>
		<script type="text/javascript" src="three.min.js"></script>
		<script type="text/javascript" src="jquery-1.9.0.js"></script>
		<style>
			body{
				margin: 0;
				overflow: hidden;
				}
		</style>
	</head>
	<body>

		<div id="WebGL-output">
		</div>
		<script type="text/javascript">
		$(function ()
			{

				// Dhmhourgia skhnhs opou tha topothetithoun antikeimena, kameres kai fwta.
				var scene = new THREE.Scene();
```

```
// Dhmhourgia cameras.

var camera = new THREE.PerspectiveCamera(45, window.innerWidth / window.innerHeight, 0.1, 1000);

camera.position.x = 10;

camera.position.y = 80;

camera.position.z = 280;


// Dhmhourgia render.

var renderer = new THREE.WebGLRenderer();

renderer.setClearColor(new THREE.Color(0xEEEEEE, 1.0));

renderer.setSize(window.innerWidth, window.innerHeight);

renderer.shadowMapEnabled = true;


// Dhmhourgia edafous.

var planeGeometry = new THREE.PlaneGeometry(200,200);

var planeMaterial = new THREE.MeshLambertMaterial({color: 0xffffff});

var plane = new THREE.Mesh(planeGeometry,planeMaterial);

plane.receiveShadow = true;


// Peristrofh kai topothetish edafous.

plane.rotation.x=-0.5*Math.PI;

plane.position.x=0
```

```
plane.position.y=-20

plane.position.z=0


// Prosthikh edafous sthn skhnh.

scene.add(plane);


//Dhmhourgia sferas.

var sphereGeometry = new THREE.SphereGeometry(30, 20, 20);

var sphereMaterial = new THREE.MeshLambertMaterial({color: 0x7777ff});

var sphere = new THREE.Mesh(sphereGeometry, sphereMaterial);


// Thesh sfairas

sphere.position.x = 2;

sphere.position.y = 20;

sphere.position.z = 20;

sphere.castShadow = true;


// add sfaira

scene.add(sphere);



// Prosthikh apalou fwtismou perivalontos.

var ambientLight = new THREE.AmbientLight(0x292929);
```

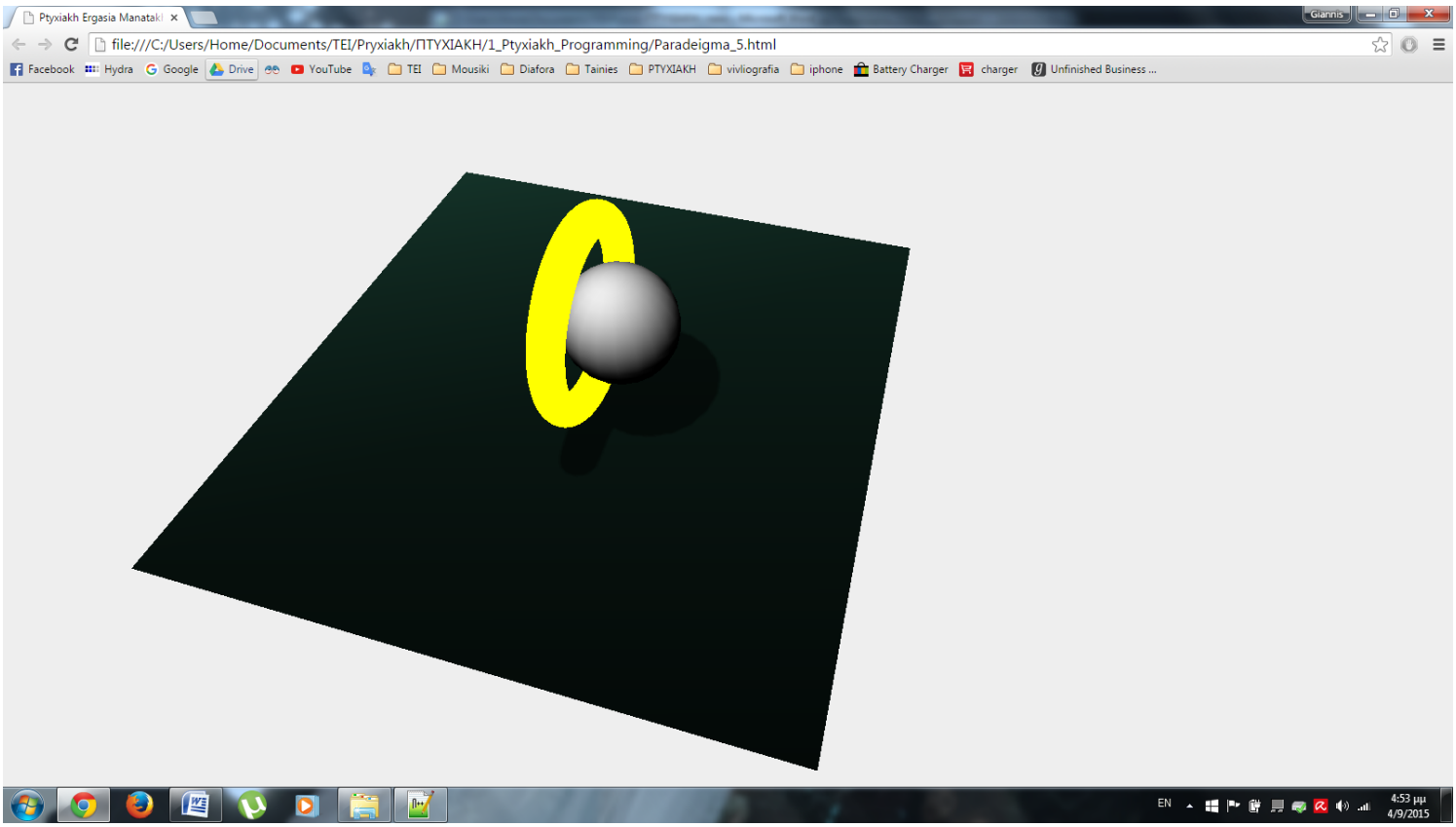
```
scene.add(ambientLight);


// Prosthikh spotlight fwtismou

var spotLight = new THREE.SpotLight(0xffffff);

spotLight.position.set(100, 190, 100);

spotLight.castShadow = true;

scene.add(spotLight);

document.getElementById("WebGL-output").appendChild(renderer.domElement);



$("#WebGL-output").append(renderer.domElement);

var step=0;

render();

var step=0;


//Synarthsh Render.

var step = 0;

function render()

        {


                // Anaphdhsh ths sferas deksia - aristera.

                sphere.position.x = 10 + ( 40 * (Math.cos(step)));

                sphere.position.y = 2 + ( 50 * Math.abs(Math.sin(step)));
```

```
                                        // Peristrofh cameras deksia - aristera.
                                        step+=0.03;
                                        if (camera instanceof THREE.PerspectiveCamera)
                                            {
                                            var x = 6+( 80*(Math.sin(step)));
                                            camera.lookAt(new THREE.Vector3(x,10,0));
                                            sphere.position=new THREE.Vector3(x,10,0);
                                            }
                                        requestAnimationFrame(render);
                                        renderer.render(scene, camera);
                                    }

                            });
                        </script>
                </body>
        </html>
```

2.5.5 Πέμπτο παράδειγμα προγράμματος

Στο πέμπτο παράδειγμα μας, μετά τα απαραίτητα αντικείμενα που χρειαζόμαστε, δημιουργήσαμε μια σφαίρα και έναν κύκλο (Torus) όπου η σφαίρα αναπήδα δεξιά από τον κύκλο, στην συνέχεια περνά από μέσα του και αναπήδα από την αριστερή μεριά του.

Στην παρακάτω εικόνα βλέπουμε το πρόγραμμα.



Εικόνα 18 - Πέμπτο παράδειγμα

Όλος ο κώδικας του πέμπτου προγράμματος.

```
<!DOCTYPE html>
<html>
	<head>
		<title>Ptyxiakh Ergasia Manatakh Iwannh</title>
		<script type="text/javascript" src="three.min.js"></script>
		<script type="text/javascript" src="jquery-1.9.0.js"></script>
		<style>
			body{
				margin: 0;
				overflow: hidden;
				}
		</style>
	</head>
	<body>

		<div id="WebGL-output">
		</div>

		<script type="text/javascript">

		$(function ()
			{

				// Dhmhourgia skhnhs opou tha topothetithoun antikeimena, kameres kai fwta.
```

```
var scene = new THREE.Scene();


// Dhmhourgia cameras.

var camera = new THREE.PerspectiveCamera(55, window.innerWidth / window.innerHeight, 0.1, 1000);

camera.position.x = 130;

camera.position.y = 250;

camera.position.z = 200;

camera.lookAt(new THREE.Vector3(100, 90,90));


// Dhmhourgia render.

var renderer = new THREE.WebGLRenderer();

renderer.setClearColor(new THREE.Color(0xEEEEEE, 1.0));

renderer.setSize(window.innerWidth, window.innerHeight);

renderer.shadowMapEnabled = true;


// Dhmhourgia edafous.

var planeGeometry = new THREE.PlaneGeometry(300,300);

var planeMaterial = new THREE.MeshLambertMaterial({color: 0x66FFCC});

var plane = new THREE.Mesh(planeGeometry,planeMaterial);

plane.receiveShadow = true;
```

```
// Peristrofh kai topothetish edafous.
plane.rotation.x=-0.5*Math.PI;
plane.position.x=0
plane.position.y=-20
plane.position.z=0

// Prosthikh edafous sthn skhnh.
scene.add(plane);

//Dhmhourgia sferas.
var sphereGeometry = new THREE.SphereGeometry(25, 20, 20);
var sphereMaterial = new THREE.MeshLambertMaterial({color: 0xffffff});
var sphere = new THREE.Mesh(sphereGeometry, sphereMaterial);

// Thesh sfairas
sphere.position.x = 2;
sphere.position.y = 20;
sphere.position.z = 20;
sphere.castShadow = true;

// add sfaira
scene.add(sphere);

//Dhmhourgia torou.
```

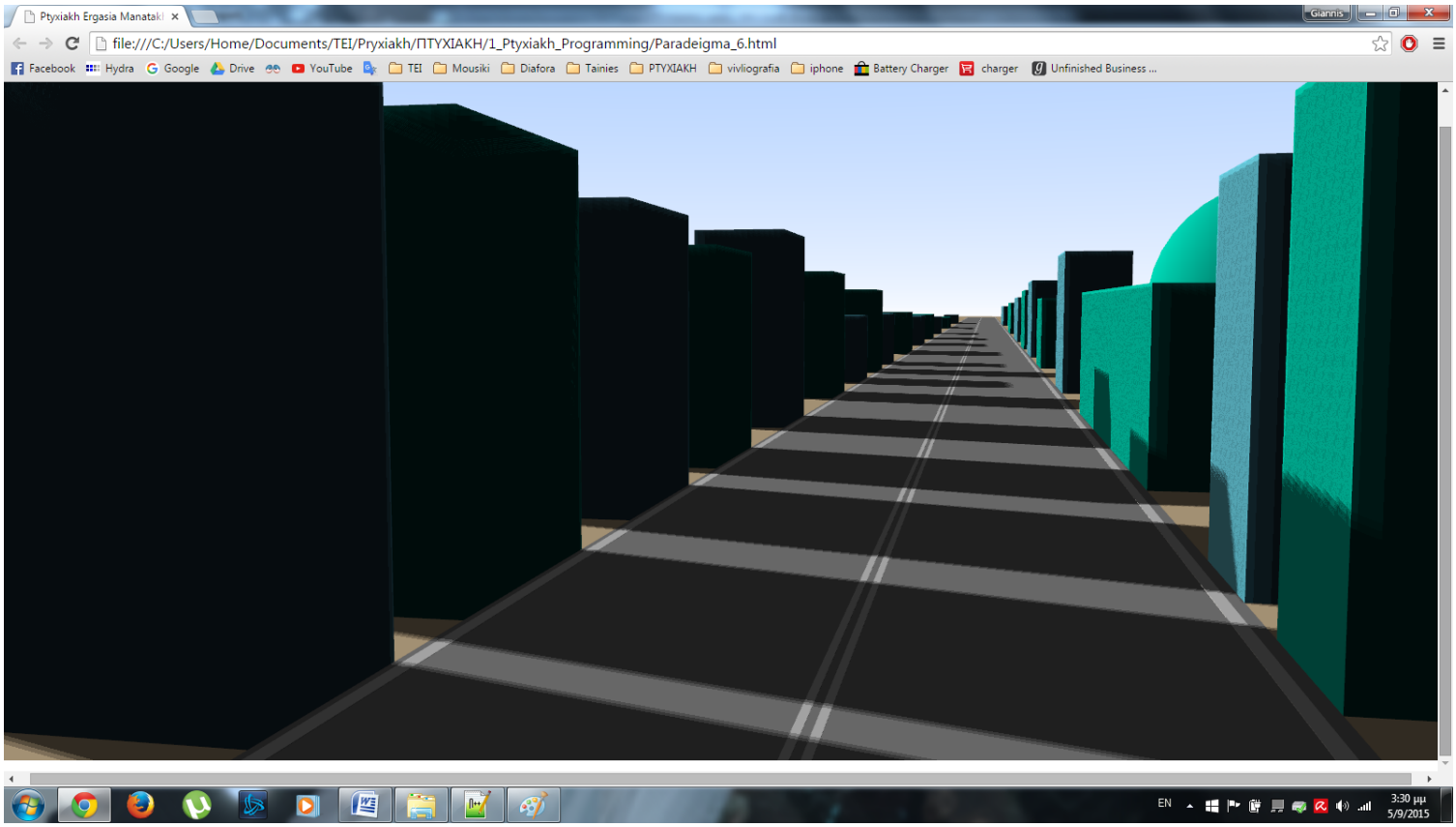
```
var geometry = new THREE.TorusGeometry( 40, 7, 60, 60);

var material = new THREE.MeshBasicMaterial( { color: 0xffff00 } );

var torus = new THREE.Mesh( geometry, material );


//Thesh torou.

torus.position.x = 20;

torus.position.y = 50;

torus.position.z = 20;

torus.lookAt(new THREE.Vector4(900, 0,0));

torus.castShadow = true;


//prosthikh torou sthn skhnh.

scene.add( torus );


// Prosthikh spotlight fwtismou

var spotLight = new THREE.SpotLight(0xffffff);

spotLight.position.set(40, 250, 100);

spotLight.castShadow = true;

scene.add(spotLight);

document.getElementById("WebGL-output").appendChild(renderer.domElement);



$("#WebGL-output").append(renderer.domElement);
```

```
var step=0;
render();
var step=0;


//Synarthsh Render.
var step = 0;
function render()
        {


            // Anaphdhsh ths sferas deksia - aristera.
            sphere.position.x = 10 + ( 90 * (Math.cos(step)));
            sphere.position.y = 2 + ( 50 * Math.abs(Math.sin(step)));


            step+=0.03;


            requestAnimationFrame(render);
            renderer.render(scene, camera);
        }


    });
</script>
</body>
</html>
```

2.5.6 Έκτο παράδειγμα προγράμματος

Στο έκτο παράδειγμα μας προσθέσαμε στην σκηνή μας διάφορα αντικείμενα, για αρχή προσθέσαμε τον ουρανό και το έδαφος. Έπειτα δημιουργήσαμε έναν δρόμο όπου δεξιά και στα αριστερά του βάλαμε πολυκατοικίες. Το βασικότερο όμως είναι ο φωτισμός ο όποιος βάλαμε, δίνοντας του της κατάλληλες τιμές είναι σαν ήλιος, όπου βγαίνει από την ανατολή και πηγαίνει στην δύση, και πάλι το ίδιο. Ρίχνοντας της ακτίνες του πάνω στην σκηνή μας κατά την διάρκεια της περιστροφής του, δημιουργώντας φανταστικά εφέ.

Στην παρακάτω εικόνα βλέπουμε το πρόγραμμα.



Εικόνα 19 - Έκτο παράδειγμα

Όλος ο κώδικας του έκτου προγράμματος.

```
<!DOCTYPE html>
   <html>
   <head>
            <title>Ptyxiakh Ergasia Manatakh Iwannh</title>
         <script src="three.min.js"></script>
          <script type="x-shader/x-vertex" id="vertexShader">
        varying vec3 worldPosition;
        void main() {
            vec4 mPosition = modelMatrix * vec4( position, 1.0 );
         gl_Position = projectionMatrix * modelViewMatrix * vec4( position, 1.0 );
            worldPosition = mPosition.xyz;
        }
      </script>

      <script type="x-shader/x-fragment" id="fragmentShader">
        uniform vec3 topColor;
        uniform vec3 bottomColor;
        uniform float offset;
        uniform float exponent;
        varying vec3 worldPosition;
        void main() {
            float h = normalize( worldPosition + offset ).y;
                  gl_FragColor = vec4( mix( bottomColor, topColor, max( pow( h,
      exponent ), 0.0 ) ), 1.0 );
        }
      </script>
```

```
<script>
window.onload = function()
                        {
   renderer = new THREE.WebGLRenderer( { antialias: true } );
   renderer.setPixelRatio( window.devicePixelRatio );
   renderer.setSize( window.innerWidth, window.innerHeight );
   document.body.appendChild( renderer.domElement );
   renderer.gammaInput = true;
   renderer.gammaOutput = true;
   renderer.shadowMapEnabled = true;
   renderer.shadowMapCullFace = THREE.CullFaceBack;
   var scene = new THREE.Scene();
    camera = new THREE.PerspectiveCamera( 30, window.innerWidth /
   window.innerHeight, 1, 5000 );
   camera.position.set( -25, 2, 5 );
   camera.lookAt( scene.position );

                        //Antikeimena sta aristera tou dromou.
   var geometry1 = new THREE.BoxGeometry( 3, 12, 3 );
    var material1 = new THREE.MeshLambertMaterial( { color: 0x33CCFF} );
   var mesh1 = new THREE.Mesh( geometry1, material1 );
   mesh1.position.x = -13
   mesh1.position.y = -1.5
   mesh1.position.z = -3
   mesh1.castShadow = true;
```

```
mesh1.receiveShadow = true;
scene.add( mesh1 );

        var geometry2 = new THREE.BoxGeometry( 5, 12, 3 );
var material2 = new THREE.MeshLambertMaterial( { color: 0x00CC99 } );
var mesh2 = new THREE.Mesh( geometry2, material2 );
mesh2.castShadow = true;
mesh2.receiveShadow = true;
mesh2.position.x = -7
mesh2.position.y = -1.5
mesh2.position.z = -3
scene.add( mesh2 );

var geometry3 = new THREE.BoxGeometry( 5, 11, 3 );
var material3 = new THREE.MeshLambertMaterial( { color: 0x33CCFF } );
var mesh3 = new THREE.Mesh( geometry3, material3 );
mesh3.castShadow = true;
mesh3.receiveShadow = true;
        mesh3.receiveShadow = true;
mesh3.position.x = 1
mesh3.position.y = -1.5
mesh3.position.z = -3
scene.add( mesh3 );

        var geometry4 = new THREE.BoxGeometry( 5, 10, 3 );
var material4 = new THREE.MeshLambertMaterial( { color: 0x00CC99 } );
var mesh4 = new THREE.Mesh( geometry4, material4 );
```

```
mesh4.position.x = 9
mesh4.position.y = -1.5
mesh4.position.z = -3
mesh4.castShadow = true;
mesh4.receiveShadow = true;
scene.add(mesh4);

                var geometry5 = new THREE.BoxGeometry( 5, 12, 3 );
var material5 = new THREE.MeshLambertMaterial( { color: 0x33CCFF } );
var mesh5 = new THREE.Mesh( geometry5, material5 );
mesh5.position.x = 20
mesh5.position.y = -1.5
mesh5.position.z = -3
mesh5.castShadow = true;
mesh5.receiveShadow = true;
scene.add(mesh5);

                var geometry6 = new THREE.BoxGeometry( 5, 10, 3 );
var material6 = new THREE.MeshLambertMaterial( { color: 0x00CC99 } );
var mesh6 = new THREE.Mesh( geometry6, material6 );
mesh6.position.x = 34
mesh6.position.y = -1.5
mesh6.position.z = -3
mesh6.castShadow = true;
mesh6.receiveShadow = true;
scene.add( mesh6 );
```

```
var geometry7 = new THREE.BoxGeometry( 5, 6, 3 );
var material7 = new THREE.MeshLambertMaterial( { color: 0x33CCFF } );
var mesh7 = new THREE.Mesh( geometry7, material7 );
mesh7.position.x = 46
mesh7.position.y = -1.5
mesh7.position.z = -3
mesh7.castShadow = true;
mesh7.receiveShadow = true;
scene.add( mesh7 );

var geometry8 = new THREE.BoxGeometry( 5, 9, 3 );
var material8 = new THREE.MeshLambertMaterial( { color: 0x00CC99 } );
var mesh8 = new THREE.Mesh( geometry8, material8 );
mesh8.position.x = 57
mesh8.position.y = -1.5
mesh8.position.z = -3
mesh8.castShadow = true;
mesh8.receiveShadow = true;
scene.add( mesh8 );

var geometry9 = new THREE.BoxGeometry( 5, 6, 3 );
var material9 = new THREE.MeshLambertMaterial( { color: 0x33CCFF } );
var mesh9 = new THREE.Mesh( geometry9, material9 );
mesh9.position.x = 68
mesh9.position.y = -1.5
mesh9.position.z = -3
mesh9.castShadow = true;
```

```
mesh9.receiveShadow = true;
scene.add( mesh9 );

        var geometry10 = new THREE.BoxGeometry( 9, 6, 3 );
var material10 = new THREE.MeshLambertMaterial( { color: 0x00CC99 } );
var mesh10 = new THREE.Mesh( geometry10, material10 );
mesh10.position.x = 90
mesh10.position.y = -1.5
mesh10.position.z = -3
mesh10.castShadow = true;
mesh10.receiveShadow = true;
scene.add( mesh10 );

        var geometry11 = new THREE.BoxGeometry( 9, 5, 3 );
var material11 = new THREE.MeshLambertMaterial( { color: 0x33CCFF } );
var mesh11 = new THREE.Mesh( geometry11, material11 );
mesh11.position.x = 115
mesh11.position.y = -1.5
mesh11.position.z = -3
mesh11.castShadow = true;
mesh11.receiveShadow = true;
scene.add( mesh11 );

        var geometry111 = new THREE.BoxGeometry( 9, 5, 3 );
var material111 = new THREE.MeshLambertMaterial( { color: 0x00CC99 } );
var mesh111 = new THREE.Mesh( geometry111, material111 );
mesh111.position.x = 140
```

```
mesh111.position.y = -1.5
mesh111.position.z = -3
mesh111.castShadow = true;
mesh111.receiveShadow = true;
scene.add( mesh111 );

                var geometry112 = new THREE.BoxGeometry( 9, 4, 3 );
var material112 = new THREE.MeshLambertMaterial( { color: 0x33CCFF }
);
var mesh112 = new THREE.Mesh( geometry112, material112 );
mesh112.position.x = 170
mesh112.position.y = -1.5
mesh112.position.z = -3
mesh112.castShadow = true;
mesh112.receiveShadow = true;
scene.add( mesh112 );

                var geometry113 = new THREE.BoxGeometry( 19, 3, 3 );
var material113 = new THREE.MeshLambertMaterial( { color: 0x00CC99 } );
var mesh113 = new THREE.Mesh( geometry113, material113 );
mesh113.position.x = 220
mesh113.position.y = -1.5
mesh113.position.z = -3
mesh113.castShadow = true;
mesh113.receiveShadow = true;
scene.add( mesh113 );
```

```
                var geometry114 = new THREE.BoxGeometry( 19, 3, 3 );
        var material114 = new THREE.MeshLambertMaterial( { color: 0x33CCFF }
);
        var mesh114 = new THREE.Mesh( geometry114, material114 );
        mesh114.position.x = 290
        mesh114.position.y = -1.5
        mesh114.position.z = -3
        mesh114.castShadow = true;
        mesh114.receiveShadow = true;
        scene.add( mesh114 );



                //Anteikeimena sta deksia tou dromou.
                var geometry12 = new THREE.BoxGeometry( 3, 12, 3 );
        var material12 = new THREE.MeshLambertMaterial( { color: 0x00CC99 } );
        var mesh12 = new THREE.Mesh( geometry12, material12 );
        mesh12.position.x = -10
        mesh12.position.y = -1.5
        mesh12.position.z = 10
        mesh12.castShadow = true;
        mesh12.receiveShadow = true;
        scene.add( mesh12 );

                var geometry13 = new THREE.BoxGeometry( 3, 11, 3 );
        var material13 = new THREE.MeshLambertMaterial( { color: 0x33CCFF } );
        var mesh13 = new THREE.Mesh( geometry13, material13 );
        mesh13.position.x = -5
```

```
mesh13.position.y = -1.5
mesh13.position.z = 10
mesh13.castShadow = true;
mesh13.receiveShadow = true;
scene.add( mesh13 );


                var geometry14 = new THREE.BoxGeometry( 21, 8, 3 );
var material14 = new THREE.MeshLambertMaterial( { color: 0x00CC99 } );
var mesh14 = new THREE.Mesh( geometry14, material14 );
mesh14.position.x = 15
mesh14.position.y = -1.5
mesh14.position.z = 10
mesh14.castShadow = true;
mesh14.receiveShadow = true;
scene.add( mesh14 );


                var sphereGeometry = new THREE.SphereGeometry(3.5, 40,
20);
                var sphereMaterial = new
THREE.MeshLambertMaterial({color: 0x00CC99});
                var sphere = new THREE.Mesh(sphereGeometry,
sphereMaterial);
                sphere.position.x = 14;
                sphere.position.y = 2;
                sphere.position.z = 13;
                sphere.castShadow = true;
                scene.add(sphere);
```

```
var geometry15 = new THREE.BoxGeometry( 6, 12, 3 );
var material15 = new THREE.MeshLambertMaterial( { color: 0x33CCFF } );
var mesh15 = new THREE.Mesh( geometry15, material15 );
mesh15.position.x = 40
mesh15.position.y = -1.5
mesh15.position.z = 10
mesh15.castShadow = true;
mesh15.receiveShadow = true;
scene.add( mesh15 );

var geometry16 = new THREE.BoxGeometry( 7, 8, 3 );
var material16 = new THREE.MeshLambertMaterial( { color: 0x00CC99 } );
var mesh16 = new THREE.Mesh( geometry16, material16 );
mesh16.position.x = 65
mesh16.position.y = -1.5
mesh16.position.z = 10
mesh16.castShadow = true;
mesh16.receiveShadow = true;
scene.add( mesh16 );

var geometry17 = new THREE.BoxGeometry( 9, 11, 3 );
var material17 = new THREE.MeshLambertMaterial( { color: 0x33CCFF } );
var mesh17 = new THREE.Mesh( geometry17, material17 );
mesh17.position.x = 85
mesh17.position.y = -1.5
mesh17.position.z = 10
mesh17.castShadow = true;
```

```
mesh17.receiveShadow = true;
scene.add( mesh17 );

        var geometry18 = new THREE.BoxGeometry( 11, 10, 3 );
var material18 = new THREE.MeshLambertMaterial( { color: 0x00CC99 } );
var mesh18 = new THREE.Mesh( geometry18, material18 );
mesh18.position.x = 105
mesh18.position.y = -1.5
mesh18.position.z = 10
mesh18.castShadow = true;
mesh18.receiveShadow = true;
scene.add( mesh18 );

        var geometry19 = new THREE.BoxGeometry( 18, 9, 3 );
var material19 = new THREE.MeshLambertMaterial( { color: 0x33CCFF } );
var mesh19 = new THREE.Mesh( geometry19, material19 );
mesh19.position.x = 135
mesh19.position.y = -1.5
mesh19.position.z = 10
mesh19.castShadow = true;
mesh19.receiveShadow = true;
scene.add( mesh19 );

        var geometry20 = new THREE.BoxGeometry( 21, 8, 3 );
var material20 = new THREE.MeshLambertMaterial( { color: 0x00CC99 } );
var mesh20 = new THREE.Mesh( geometry20, material20 );
mesh20.position.x = 180
```

```
mesh20.position.y = -1.5
mesh20.position.z = 10
mesh20.castShadow = true;
mesh20.receiveShadow = true;
scene.add( mesh20 );


                var geometry21 = new THREE.BoxGeometry( 35, 7, 3 );
var material21 = new THREE.MeshLambertMaterial( { color: 0x33CCFF } );
var mesh21 = new THREE.Mesh( geometry21, material21 );
mesh21.position.x = 270
mesh21.position.y = -1.5
mesh21.position.z = 10
mesh21.castShadow = true;
mesh21.receiveShadow = true;
scene.add( mesh21 );



                //Dhmhourgia dromou.
var geometry30 = new THREE.BoxGeometry( 900, 0.01, 10 );
var material30 = new THREE.MeshLambertMaterial( { color: 0x666666 } );
var mesh30 = new THREE.Mesh( geometry30, material30 );
mesh30.position.x = 0
mesh30.position.y = -2
mesh30.position.z = 3.5
mesh30.receiveShadow = true;
scene.add( mesh30 );
```

```
                var geometry301 = new THREE.BoxGeometry( 900, 0.02, 0.1
);
        var material301 = new THREE.MeshLambertMaterial( { color: 0xFFFFFF } );
        var mesh301 = new THREE.Mesh( geometry301, material301 );
        mesh301.position.x = 0
        mesh301.position.y = -2
        mesh301.position.z = 3.5
        mesh301.receiveShadow = true;
        scene.add( mesh301 );

                var geometry3011 = new THREE.BoxGeometry( 900, 0.02, 0.1
);
        var material3011 = new THREE.MeshLambertMaterial( { color: 0xFFFFFF }
);
        var mesh3011 = new THREE.Mesh( geometry3011, material3011 );
        mesh3011.position.x = 0
        mesh3011.position.y = -2
        mesh3011.position.z = 3.3
        mesh3011.receiveShadow = true;
        scene.add( mesh3011 );

                var geometry302 = new THREE.BoxGeometry( 900, 0.02, 0.2
);
        var material302 = new THREE.MeshLambertMaterial( { color: 0xFFFFFF } );
        var mesh302 = new THREE.Mesh( geometry302, material302 );
        mesh302.position.x = 0
        mesh302.position.y = -2
        mesh302.position.z = -1.3
```

```
mesh302.receiveShadow = true;
scene.add( mesh302 );

                var geometry303 = new THREE.BoxGeometry( 900, 0.02, 0.2
);
var material303 = new THREE.MeshLambertMaterial( { color: 0xFFFFFF } );
var mesh303 = new THREE.Mesh( geometry303, material303 );
mesh303.position.x = 0
mesh303.position.y = -2
mesh303.position.z = 8.2
mesh303.receiveShadow = true;
scene.add( mesh303 );

        //Dhmhourgia prasinou edafous sta aristera tou dromou.
         var geometry31 = new THREE.BoxGeometry( 900, 0.01, 451 );
var material31 = new THREE.MeshLambertMaterial( { color: 0x00CC00 } );
var mesh31 = new THREE.Mesh( geometry31, material31 );
mesh31.position.x = 0
mesh31.position.y = -2
mesh31.position.z = -230
mesh31.receiveShadow = true;
scene.add( mesh31 );

        //Dhmhourgia prasinou edafous sta deksia tou dromou.
         var geometry32 = new THREE.BoxGeometry( 900, 0.01, 451 );
var material32 = new THREE.MeshLambertMaterial( { color: 0x00CC00 } );
var mesh32 = new THREE.Mesh( geometry32, material32 );
```

```
mesh32.position.x = 0
mesh32.position.y = -2
mesh32.position.z = 240
mesh32.receiveShadow = true;
scene.add( mesh32 );



var hemiLight = new THREE.HemisphereLight( 0xffffff, 0xffffff, 0.05 );
hemiLight.color.setHSL( 0.6, 1, 0.6 );
hemiLight.groundColor.setHSL( 0.095, 1, 0.75 );
hemiLight.position.set( 0, 100, 0 );
scene.add( hemiLight );

// this is the Sun
dirLight = new THREE.DirectionalLight( 0xffffff, 1 );
dirLight.color.setHSL( 0.1, 1, 0.95 );
dirLight.position.set( -1, 0.75, 1 );
dirLight.position.multiplyScalar( 50 );
scene.add( dirLight );

 dirLight.shadowCameraVisible = false;
dirLight.castShadow = true;
dirLight.shadowMapWidth = dirLight.shadowMapHeight = 1024*10;
var d = 300;
dirLight.shadowCameraLeft = -d;
dirLight.shadowCameraRight = d;
dirLight.shadowCameraTop = d;
```

```
dirLight.shadowCameraBottom = -d;
dirLight.shadowCameraFar = 3500;
dirLight.shadowBias = -0.000001;
dirLight.shadowDarkness = 0.35;
scene.add( dirLight );
scene.fog = new THREE.Fog(0x222233, 0, 20000);
renderer.setClearColor( scene.fog.color, 1 );

// Prosthikh edafous.
var groundGeo = new THREE.PlaneBufferGeometry( 1000, 1000 );
var groundMat = new THREE.MeshPhongMaterial( { color: 0xffffff, specular:
0x050505 } );
groundMat.color.setHSL( 0.095, 1, 0.75 );
ground = new THREE.Mesh( groundGeo, groundMat );
ground.rotation.x = -Math.PI/2;
ground.position.y = -2;
scene.add( ground );
ground.receiveShadow = true;

// prosthikh ouranou.
var vertexShader = document.getElementById( 'vertexShader' ).textContent;
var fragmentShader = document.getElementById( 'fragmentShader'
).textContent;
var uniforms = {
   topColor:    { type: "c", value: new THREE.Color( 0x0077ff ) },
   bottomColor: { type: "c", value: new THREE.Color( 0xffffff ) },
   offset:      { type: "f", value: 33 },
   exponent:    { type: "f", value: 0.6 }
```

```
}
uniforms.topColor.value.copy( hemiLight.color );
scene.fog.color.copy( uniforms.bottomColor.value );
var skyGeo = new THREE.SphereGeometry( 4000, 32, 15 );
var skyMat = new THREE.ShaderMaterial( { vertexShader: vertexShader, fragmentShader: fragmentShader, uniforms: uniforms, side: THREE.BackSide } );
var sky = new THREE.Mesh( skyGeo, skyMat );
scene.add( sky );
var clock = new THREE.Clock();
animate();
function animate() {
   requestAnimationFrame( animate );
   render();
}
function render() {
   var delta = clock.getDelta();
   renderer.render( scene, camera );
   var time = new Date().getTime() * 0.0002;
   var nsin = Math.sin(time);
   var ncos = Math.cos(time);

   //Hlios.
   dirLight.position.set( 1500*nsin, 2000*nsin, 2000*ncos);
   if (nsin > 0.2 )   // day
   {
      sky.material.uniforms.topColor.value.setRGB(0.25,0.55,1);
      sky.material.uniforms.bottomColor.value.setRGB(1,1,1);
```

```
            var f = 1;
            dirLight.intensity = f;
            dirLight.shadowDarkness = f*0.7;
          }
          else if (nsin < 0.2 && nsin > 0.0 )
          {
            var f = nsin/0.2;
            dirLight.intensity = f;
            dirLight.shadowDarkness = f*0.7;
            sky.material.uniforms.topColor.value.setRGB(0.25*f,0.55*f,1*f);
            sky.material.uniforms.bottomColor.value.setRGB(1*f,   1*f,1*f);
          }
          else
          {
            var f = 0;
            dirLight.intensity = f;
            dirLight.shadowDarkness = f*0.7;
            sky.material.uniforms.topColor.value.setRGB(0,0,0);
            sky.material.uniforms.bottomColor.value.setRGB(0,0,0);
          }
        }
      };
      </script>
  </head>
  <body></body>
</html>
```

ΚΕΦΑΛΑΙΟ 3

ΥΠΟΣΤΗΡΙΞΗ ΚΑΙ ΣΥΜΒΑΤΟΤΗΤΑ ΤΗΣ WEBGL

3.1 Υποστήριξη σε σύγχρονα προγράμματα περιήγησης.

- Mozilla Firefox: Η WebGL είναι ενεργοποιημένη σε όλες τις πλατφόρμες που έχουν μια ικανή κάρτα γραφικών με ενημερωμένα προγράμματα οδήγησης από την έκδοση 4.0.

- Google Chrome: Η WebGL είναι ενεργοποιημένη σε όλες τις πλατφόρμες που έχουν μια ικανή κάρτα γραφικών με ενημερωμένα προγράμματα οδήγησης από την έκδοση 9.

- Safari: Ο Safari 6.0 και οι νεότερες εκδόσεις που είναι εγκατεστημένες στον OS X Mountain Lion, Mac OS X Lion και ο Safari 5.1 στον Mac OS X Snow Leopard , η οποία είναι απενεργοποιημένη από προεπιλογή.

- Opera: Η WebGL έχει εφαρμοστεί στην Opera 11 και 12, αν και είναι απενεργοποιημένη από προεπιλογή.

- Internet Explorer: Κάποιο μέρος της λειτουργικότητας της WebGL υποστηρίζεται στον Internet Explorer 11. Μέχρι τον Internet Explorer10, ο μόνος τρόπος για να χρησιμοποιηθεί η WebGL - είναι η εγκατάσταση plugins, όπως το Chrome Frame και IEWebGL . Επιπλέον ο νέος Internet Explorer των Windows 8.1 θα υποστηρίζει τη βιβλιοθήκη WebGL, αν και η Microsoft είχε δηλώσει παλαιότερα ότι δεν προτίθεται να την υποστηρίξει στον browser.

3.2 Υποστήριξη σε Mobile browsers

- BlackBerry Playbook – Είναι ένα mini tablet computer που φτιάχτηκε από τη BlackBerry και το οποίο δόθηκε στην αγορά στις 19 Απριλίου του 2011. Η WebGL είναι διαθέσιμη μέσω του

WebWorks (ένα open-source online σύστημα για μαθήματα μαθηματικών και φυσικής) και του browser του PlayBook OS 2.00.

- Firefox for mobile – Η WebGL είναι διαθέσιμη για τις συσκευές Android από την έκδοση Firefox 4.

- Firefox OS – Γνωστό και ως *B2G,* είναι ένα ελεύθερο λειτουργικό σύστημα, βασισμένο στα Linux για smartphones και tablet computers.

- Google Chrome – Η WebGL είναι διαθέσιμη για τις συσκευές Android από την έκδοση Google Chrome 25 και είναι ενεργοποιημένη από προεπιλογή από την έκδοση 30.

- Maemo – Στο Nokia N900, η WebGL είναι διαθέσιμη στο microB browser.

- Opera Mobile – Ο Opera Mobile 12 υποστηρίζει WebGL (μόνο για Android).

- Tizen 1.0 – Είναι ένα λειτουργικό σύστημα βασισμένο στο Linux για συσκευές όπως smartphones, tablets, smart-TVs, laptops και smart cameras.

- Ubuntu Touch – Είναι μια έκδοση η οποία σχεδιάστηκε αρχικά για οθόνες αφής κινητών , όπως τα tablets και τα smartphones.

- WebOS – Είναι ένα λειτουργικό βασισμένο στον πυρήνα Linux για ‘έξυπνες συσκευές’ όπως οι TVs και τα ‘έξυπνα ρολόγια’ (smartwatches).

3.3 Υποστηριζόμενα λειτουργικά συστήματα

- Windows Vista/ Windows 7/ Windows 8
- Mac OS 10.6 ή νεότερη έκδοση (συνιστάται 10.8 ή νεότερη έκδοση)
- Linux
- Chrome OS

3.4 Κάρτες γραφικών και συμβατότητα

Το WebGL διατίθεται στις περισσότερες πρόσφατες κάρτες γραφικών και στα περισσότερα νεότερα προγράμματα οδήγησης. Ωστόσο, σε ορισμένες περιπτώσεις, συγκεκριμένες ή και όλες οι λειτουργίες του WebGL δεν είναι διαθέσιμες.
Προκειμένου να παρέχουν την καλύτερη εμπειρία στον χρήστη, οι φυλλομετρητές μπορούν επιλεκτικά να ενεργοποιούν ή να απενεργοποιούν την υποστήριξη της WebGL, ή ορισμένα επιμέρους χαρακτηριστικά. Η ειδική αυτή μεταχείριση συνήθως χρησιμοποιείται για να επιλύσει προβλήματα σταθερότητας.

3.5 ANGLE

Το WebGL, μια τεχνολογία η οποία επιτρέπει στους προγραμματιστές να δημιουργήσουν ιστοσελίδες με 3D γραφικά που αξιοποιούν την/τις GPU σε έναν υπολογιστή, προϋποθέτει την ύπαρξη της πλατφόρμας OpenGL, η οποία δε συμπεριλαμβάνεται στη βασική εγκατάσταση των Windows. Ένα νέο project, όμως, έρχεται να δώσει λύση σ' αυτό το πρόβλημα.

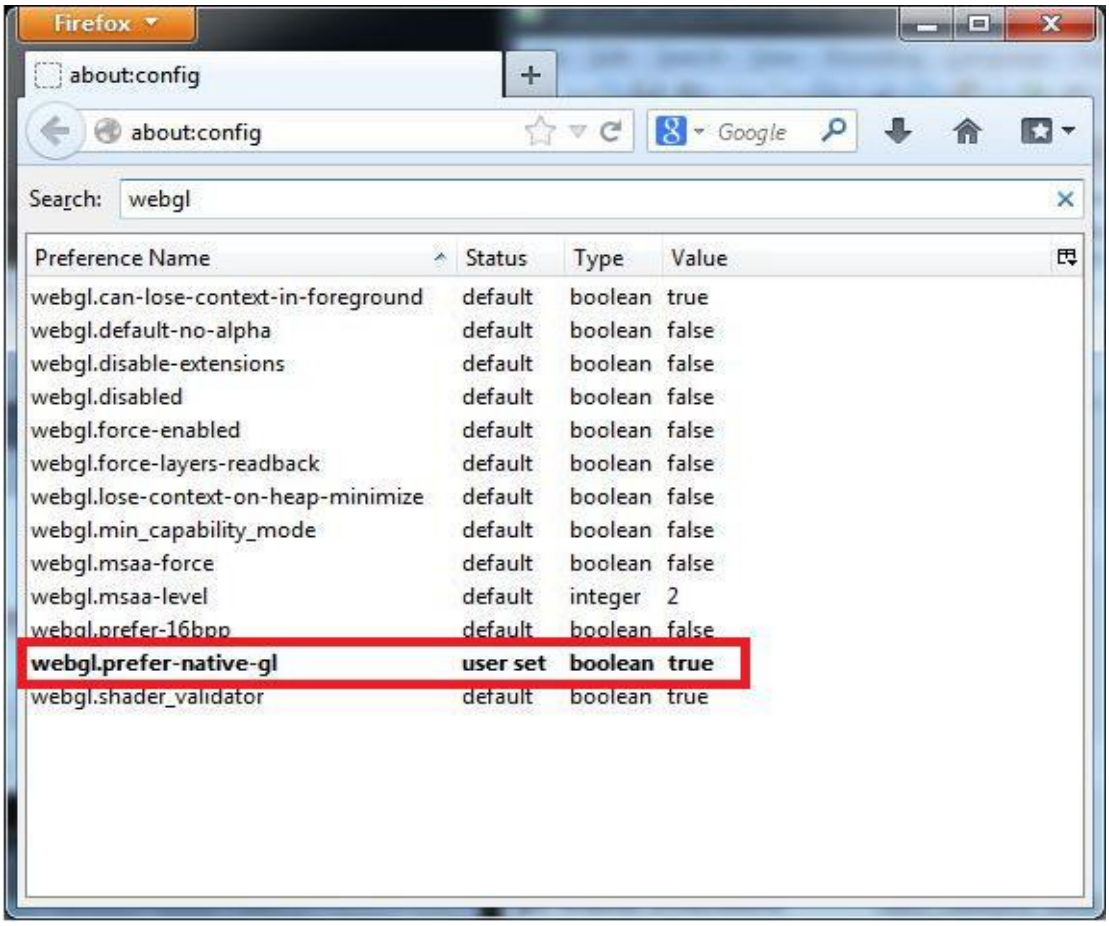
Η Google ανακοίνωσε το **ANGLE** (Almost Native Graphics Layer Engine), μια ανοιχτού κώδικα zlib βιβλιοθήκη λογισμικού για τη συμπίεση δεδομένων, που αναπτύχθηκε από τη Google και είναι γραμμένη σε C++. Αυτή έχει ως αποστολή να μεταφράζει τις εντολές του **OpenGL σε Direct3D**, δηλαδή σε ένα περιβάλλον που αναγνωρίζεται από όλους τους υπολογιστές με Windows. Έτσι όπως είναι τώρα τα πράγματα, τα PCs με το λειτουργικό της Microsoft απαιτούν την εγκατάσταση επιπλέον drivers για να κάνουν render το περιεχόμενο που στηρίζεται στη WebGL. Το ANGLE αναλαμβάνει να αξιοποιήσει το WebGL content χωρίς να προστεθεί νέο software στον υπολογιστή.

Το project αυτό είναι ουσιαστικά η υλοποίηση της ιδέας που είχε η Mozilla, όταν έψαχνε τρόπους για να απευθυνθεί στα Windows-based συστήματα. Η Google το κυκλοφορεί με την open source άδεια BSD,

ανοίγοντας έτσι το δρόμο για την υιοθέτησή του από τη Microsoft και την Opera. Από προεπιλογή, τόσο ο Firefox τόσο και ο Chrome χρησιμοποιούν το ANGLE layer για να αποδώσουν τις κλήσεις WebGL στα Windows.

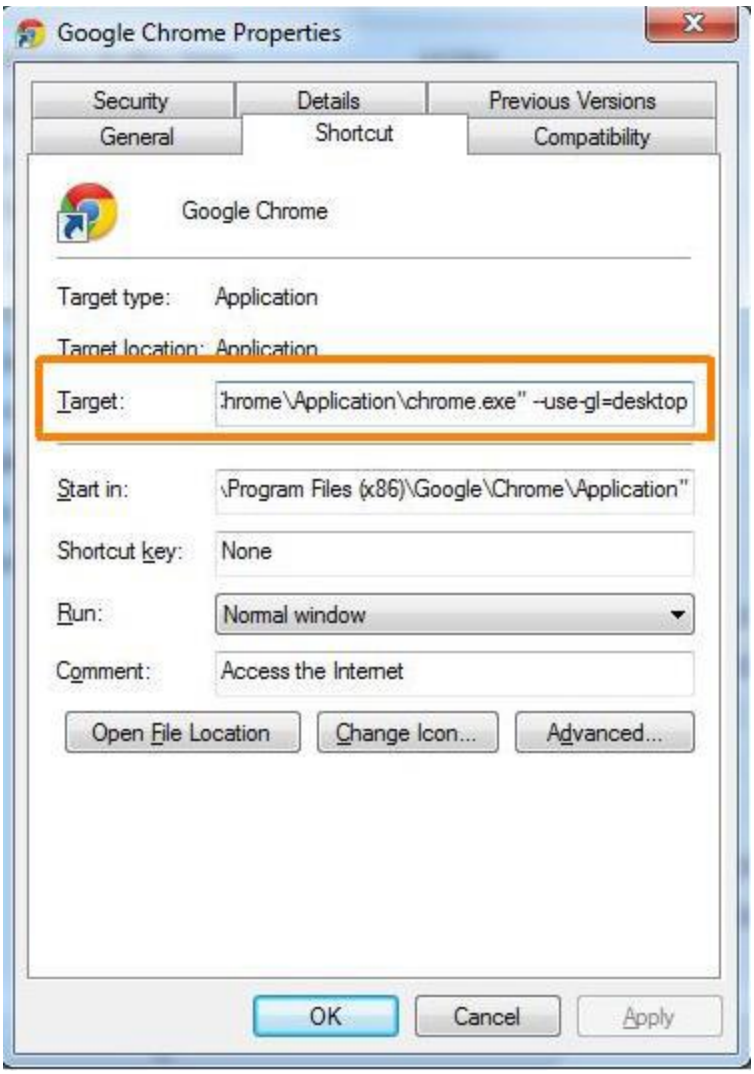
Εάν, για κάποιους λόγους, όπως για κάποια demos Shadertoy για παράδειγμα ή για να μπορούμε να παίζουμε τα νέα και καλά παιχνίδια που υπάρχουν στο *ChromeStore* μα και άλλα που κατά καιρούς παρουσιάζονται στο *osarena*, πρέπει να ενεργοποιήσουμε και το **Native Client** σε Firefox και Chrome.

Στον Firefox, απλά θέτουμε την μεταβλητή **webgl.prefer-native-gl** σε true στην σελίδα **about:config.**



Εικόνα 20 - Σελίδα about:config

Στον Chrome, θα πρέπει να προσθέσουμε την ακόλουθη παράμετρο γραμμής εντολών: **(–use-gl=desktop)**. Απλά επεξεργαζόμαστε τις ιδιότητες και αλλάζουμε το target value.



Εικόνα 21 - Google Chrome properties

3.6 "Μαύρη Λίστα" και "Λευκή Λίστα" για την WebGL

Προκειμένου να παρέχετε η καλύτερη εμπειρία για τον χρήστη, τα προγράμματα περιήγησης μπορεί επιλεκτικά να ενεργοποιούνε ή να απενεργοποιούνε την υποστήριξη WebGL, ή ορισμένα επιμέρους χαρακτηριστικά, σε ορισμένες περιπτώσεις. Η ειδική αυτή μεταχείριση συνήθως χρησιμοποιείται για να επιλυθούν προβλήματα σταθερότητας.

Η μαύρη λίστα του Firefox είναι πλήρως τεκμηριωμένη στο wiki του Mozilla. Παρακάτω παρουσιάζετε μια περίληψη των κύριων κανόνων που επηρεάζουν της WebGL.

https://wiki.mozilla.org/Blocklisting/Blocked_Graphics_Drivers

Firefox on Windows

Για τον Firefox στα Windows, απαιτείται λειτουργικό σύστημα Windows XP ή κάποιο νεότερο. Τουλάχιστον απαιτούνται οι ακόλουθες εκδόσεις των drivers :

- NVIDIA >= 257.21
- ATI/AMD >= 10.6
- Εκδόσεις των drivers της Intel από το Σεπτέμβριο του 2010 ή νεότερες

Firefox on OS X (πρώην MAC OS X)

Για την WebGL απαιτείται η έκδοση Mac OS 10.6 ή κάποια νεότερη.

Firefox on X Windows System ή X11

Στον Firefox 4 και 5 για το X11, μόνο ο driver της NVIDIA είναι στη λίστα των επιτρεπόμενων.

Στο Firefox 6 και στις νεότερες εκδόσεις για το X11, απαιτούνται οι ακόλουθες ελάχιστες εκδόσεις:

- Mesa> = 7.10.3 (ή Mesa> = 8.0 για τον οδηγό Nouveau)
- NVIDIA> = 257.21
- Οποιαδήποτε έκδοση του fglrx (driver για ATI graphic chips) εφαρμογής OpenGL 3.0 (αλλά ο fglrx είναι στη μαύρη λίστα, όταν η έκδοση του πυρήνα του Linux είναι 2.6.32)

Firefox Mobile

Ο Firefox για τα κινητά δεν έχει μέχρι στιγμής "μαύρη λίστα" για την WebGL.

Chrome on All Platforms

Σε όλα τα λειτουργικά συστήματα η WebGL είναι απενεργοποιημένη για τους drivers:

- Intel Mobile 945 Express family of chipsets
- NVIDIA GeForce FX Go5200

Chrome on Windows

- Σε όλες τις εκδόσεις των Windows, η WebGL είναι απενεργοποιημένη για όλους τους drivers γραφικών που είναι παλαιότεροι από την 1 Ιανουάριου του 2009.

Ειδικότερα, Στα Windows XP, η WebGL είναι απενεργοποιημένη στους :

- ATI/AMD παλαιότερους από την έκδοση 10.6
- NVIDIA drivers παλαιότερους από την έκδοση 257.21
- Intel drivers παλαιότερους από την έκδοση 14.42.7.5294

Επιπλέον η μαύρη λίστα περιλαμβάνει:

- Τους drivers FireNV 2400.
- Τους drivers Parallels παλαιότερους από την έκδοση 7.
- Τις κάρτες Video S3 Trio cards.

Chrome on OS X (πρώην MAC OS X)

Στο OS X, η WebGL είναι απενεργοποιημένη στην :

- ATI Radeon X1900

- NVIDIA GeForce 7300 GT

Επιπλέον, η WebGL δυνατότητα εξομάλυνσης των γραφικών, δηλαδή το antialiasing, όπως αναφέρεται, είναι μια μέθοδος που εφαρμόζεται σε εκτυπωτές και οθόνες για την εξομάλυνση των άκρων των πλάγιων γραμμών των γραφικών (Τα σφάλματα ανάγνωσης γραφικών αναφέρονται γενικά ως «aliasing»), είναι απενεργοποιημένη για όλες τις κάρτες στον OS X , παρόλο που η WebGL υποστηρίζετε σε αυτές τις κάρτες.

Chrome on Linux

Η WebGL είναι ενεργοποιημένη στην :

- ATI/AMD GPUs με τους AMD drivers 8.98 ή νεότερους.
- Intel GPUs με πρόσφατα προγράμματα οδήγησης Mesa
- Η τεχνική **Multisampling ή Multisample anti-aliasing (MSAA)** που χρησιμοποιείτε για την βελτίωση της ποιότητας των γραφικών είναι απενεργοποιημένη στις Intel Ivy Bridge cards.
- NVIDIA GPUs με NVIDIA drivers νεότερους από 295

Η WebGL είναι απενεργοποιημένη στην :

- NVIDIA Quadro FX 1500.
- NVIDIA +Intel GPUs.

Chrome on Android

Η WebGL είναι απενεργοποιημένη σε συσκευές που δεν υποστηρίζουν ARB_robustness ή EXT_robustness (δηλαδή συστήματα που δεν έχουν την ικανότητα να αντιμετωπίζουν τα λάθη ή να μπορούν να συνεχίσουν παρά τις εκάστοτε ανωμαλίες του κώδικα).

ΚΕΦΑΛΑΙΟ 4

ΕΙΣΑΓΩΓΗ ΣΤΗΝ HTML5

Εικόνα 22 - HTML 5

ΕΙΣΑΓΩΓΗ

Η HTML5 είναι η νέα έκδοση της γνωστής γλώσσας προγραμματισμού HTLM. Η προηγούμενη έκδοση, η οποία χρησιμοποιούμε ακόμα σήμερα, βγήκε το 1999. Από τότε όμως έχουν αλλάξει πολλά στον παγκόσμιο ιστό, και έτσι η ανάγκη για standards και πλήρη συμβατότητα οδήγησε τις ομάδες εργασίας World Wide Web Consortium (W3C) και Web Hypertext Application Technology Working Group (WHATWG) να συνεργαστούν και να δημιουργήσουν την νέα αυτή γλώσσα. Η HTML5 είναι ακόμα υπό ανάπτυξη, αλλά μερικοί browsers υποστηρίζουν από τώρα κάποια χαρακτηριστικά της και κάποιες λειτουργίες της.

Παρότι η σύνταξη μοιάζει αρκετά με το SGML (στην οποία βασίζεται η HTML), η HTML5 δεν προσπαθεί πλέον να αποτελεί

εφαρμογή του SGML, και ορίζεται ως αυτόνομη, μαζί με την XHTML5 η οποία βασίζεται στην XML. Οι συντάκτες της HTML5 είναι ο Ιαν Χίκσον της εταιρίας Google και ο Ντέιβ Χίατ της εταιρίας Apple .

Η HTML5 προορίζεται για αντικατάσταση της HTML 4.01, της XHTML 1.0, και της DOM Level 2 HTML. Ο σκοπός είναι η μείωση της ανάγκης για ιδιόκτητα plugin και πλούσιες διαδικτυακές εφαρμογές (RIA) όπως το Adobe Flash, το Microsoft Silverlight, το Apache Pivot, και η Sun JavaFX.

Ίσως κάποιοι πιο έμπειροι πουν πως η HTML5, από μόνη της είναι υπερεκτιμημένη, όμως ο συνδυασμός των τεχνολογιών **HTML5** και **WebGL**, έρχεται για να φέρει μια μεγάλη ανατροπή. Μια από τις πιο σημαντικές είναι η απεξάρτηση από τον Flash της Adobe και η ουσία για εμάς είναι πως θα μπορούμε να απολαμβάνουμε δικτυακά πράγματα που παλιότερα είχαμε προβλήματα, θέλαμε ισχυρό hardware και είχαμε άπειρους περιορισμούς.

Για αρκετούς, ένα από τα πιο ευχάριστα αυτών των τεχνολογιών είναι πλέον τα παιχνίδια στο cloud, τα οποία εξελίσσονται συνέχεια.

4.1 Χαρακτηριστικά της HTML5

Κάποιοι βασικοί κανόνες που έχουν οριστεί για την HTML5 είναι:

- Βάση για τα νέα χαρακτηριστικά να είναι οι HTML, CSS, DOM, και η JavaScript
- Ελαχιστοποίηση των plugins (όπως το Flash)
- Καλύτερη λειτουργία εντοπισμού λαθών
- Περισσότερο markup για να αντικατασταθεί το scripting
- Πλήρη συμβατότητα ανεξαρτήτως συσκευής

Κάποια από τα νέα χαρακτηριστικά της HTML5 είναι:

- Το στοιχείο canvas για το drawing.
- Την inline SVG.
- Τα στοιχεία video και audio για αναπαραγωγή πολυμέσων.
- 2D/3D γραφικά.
- Απόλυτη υποστήριξη του CSS3 2D/3D.
- Νέα στοιχεία περιεχομένου όπως τα footer, header, nav και section
- Νέα στοιχεία δημιουργίας φόρμας όπως τα calendar, date, time, email, url και search

Πλεονεκτήματα που προσφέρουν οι νέες APIs:

- Geological support: Υποστήριξη γεω-εντοπισμού, δηλαδή ο χρήστης μπορεί να δηλώσει την τοποθεσία του, κάτι το οποίο είναι πολύ χρήσιμο σε εφαρμογές πληροφόρησης και πλοήγησης.
- Client Side Database: Με την τοπική αποθήκευση (local storage), μπορούν οι διαδικτυακές εφαρμογές να αποθηκεύσουν μεγάλο όγκο δεδομένων τοπικά. Παλαιότερα, τα δεδομένα έπρεπε να αποθηκεύονται σε cookies, που περιλαμβάνονταν σε κάθε αίτημα του εξυπηρετητή, ενώ μέσω αυτής της API , παρέχεται μεγαλύτερη ασφάλεια και ταχύτητα χωρίς να επηρεάζεται η απόδοση.
- Offline Application Cache : Η εφαρμογή αποθηκεύεται προσωρινά στην κρυφή μνήμη και έτσι είναι προσβάσιμη και χωρίς σύνδεση στο διαδίκτυο.

- SSE(Server Side Events) : Η ιστοσελίδα κάνει αυτόματα ενημερώσεις απ ’τον server, ενώ παλαιότερα έπρεπε να ‘ρωτάει’ για τυχόν διαθέσιμες ενημερώσεις.

Παρακάτω θα αναλύσουμε τα πιο βασικά νέα στοιχεία της HTML5 τα οποία «κάνουν τη διαφορά» και αποδεικνύεται ότι η HTML5 σε συνδυασμό με την WebGL είναι πραγματικός δυναμίτης.

4.1.1 Video στην HTML5

Μέχρι σήμερα, για να παίξουμε ένα video σε έναν browser χρειαζόμασταν κάποιο plugin (πρόσθετο) όπως το Flash ή το QuickTime. Η HTML5 μας παρουσιάζει έναν νέο τρόπο, και αυτός είναι με την χρήση του στοιχείου video. Το μόνο που χρειάζεται να γράψουμε για να εμφανίσουμε το video μας είναι:

```
<video src="movie.ogg" controls="controls">
</video>
```

Εικόνα 23 - HTML 5 Βίντεο

Μια πιο εξειδικευμένη χρήση του στοιχείου video είναι η εξής:

```
<video width="320" height="240" controls="controls">
        <source src="movie.ogg" type="video/ogg">

        <source src="movie.mp4" type="video/mp4">

        Your browser does not support the video tag.
</video>
```

Ιδιότητες

- autoplay: Το video θα ξεκινήσει αυτόματα μόλις φορτώσει.
- controls :Θα εμφανίζονται ή όχι τα κουμπιά όπως το play, pause and volume.
- height : Το ύψος σε pixels
- loop : Αν θα επαναλαμβάνεται το video μετά το τέλος του.
- preload : Το video θα φορτώνεται μαζί με την σελίδα.
- src : Το URL του video
- width : Το πλάτος σε pixels

4.1.2 Audio στην HTML5

Ότι ισχύει σε σχέση με τα videos και τα plugins, ισχύει και για τα audio clips. Έτσι η HTML5 ενσωματώνει το στοιχείο audio για την αναπαραγωγή ήχων. Το μόνο που χρειάζεται να γράψουμε για να αναπαράγουμε τον ήχο μας είναι:

<audio src="song.ogg" controls="controls">

Your browser does not support the audio element of HTML5.

</audio>

Εικόνα 24 - HTML Ήχος

Ιδιότητες

- autoplay: Ο ήχος θα ξεκινήσει αυτόματα μόλις φορτώσει.
- controls: Θα εμφανίζονται ή όχι τα κουμπιά όπως το play.
- loop: Αν θα επαναλαμβάνεται ο ήχος μετά το τέλος του.
- preload: Ο ήχος θα φορτώνει μαζί με την σελίδα.
- src: Το URL του ηχητικού κομματιού μας.

4.1.3 Το στοιχείο CANVAS της HTML5

Η γνώση του HTML5 canvas είναι απαραίτητη τόσο στην προώθηση ιστοσελίδων όσο και στην κατασκευή ιστοσελίδων.

Με το HTML5 στοιχείο canvas αποκτάμε έναν εύκολο και ευέλικτο τρόπο για να ζωγραφίζουμε γραφικά χρησιμοποιώντας τη JavaScript . Μπορεί να χρησιμοποιηθεί για να ζωγραφίσουμε γραφικά, να πραγματοποιήσουμε ενώσεις φωτογραφιών ή να κάνουμε απλά animations.

Η Apple είχε πρωτοεμφανίσει το στοιχείο canvas για χρήση στο OSX Dashboard αρκετά χρόνια πριν, και σιγά σιγά οι browsers άρχισαν να το υποστηρίζουν σαν χαρακτηριστικό, όταν το ενσωμάτωσε στον Safari. Πλέον το στοιχείο canvas υποστηρίζεται πλήρως από τους περισσότερους browsers (για την ακρίβεια από όλους τους νέους browsers) και είναι πλέον στα επίσημα χαρακτηριστικά της HTML5.

Μια περιοχή canvas μπορεί να εμφανίσει διάφορα γραφικά σε μια σελίδα, όπως απλά διαγράμματα, εντυπωσιακά interfaces, κινούμενα γραφικά, γραφικές παραστάσεις και εξωτερικές εικόνες.

4.1.3.1 Δημιουργία περιοχής canvas στην σελίδα

Ξεκινάμε με ένα απλό στοιχείο canvas το οποίο έχει δυο συγκεκριμένα χαρακτηριστικά, το πλάτος και το ύψος, καθώς και τα βασικά HTML5 χαρακτηριστικά όπως είναι το id, name, class κλπ.

<canvas id="mycanvas" width="300" height="80"></canvas>

Με την ετικέτα <canvas> ορίζουμε μια περιοχή στην σελίδα μας, ο παραπάνω κώδικας προσθέτει μια περιοχή με συγκεκριμένο id και διαστάσεις 300x80.

Μπορούμε να έχουμε πρόσβαση στο στοιχείο canvas στο DOM χρησιμοποιώντας τη μέθοδο getElementById() ως εξής:

var canvas = document.getElementById("mycanvas");

Το παρακάτω κομμάτι κώδικα μας δείχνει τη χρήση του <canvas> σε ένα απλό HTML5 αρχείο.

```
<!DOCTYPE HTML>
<html>
    <head>
        <style>
        #mycanvas
            {
            border:1px solid red;
            }
        </style>
    </head>
    <body>
        <canvas id="mycanvas" width="100" height="100">
        </canvas>
    </body>
```

```
</html>
```

4.1.3.2 Σχεδίαση γραφικών στην περιοχή canvas

Η ετικέτα <canvas> δεν έχει από μόνη της σχεδιαστικές δυνατότητες. Η σχεδίαση γραφικών μέσα στην περιοχή γίνεται με την βοήθεια της γλώσσας JavaScript. Ο κώδικας JavaScript τοποθετείται μέσα σε μια function( ) η οποία είναι γραμμένη, έτσι ώστε να "τρέχει" κάθε φορά που φορτώνει η σελίδα:

Τρόπος σύνταξης του κώδικα JavaScript ο οποίος σχεδιάζει γραφικά στην περιοχή canvas:

```
<script type="text/javascript">
    window.onload = function()
        {
            var drawingCanvas = document.getElementById("mycanvas");
            var context = drawingCanvas.getContext("2d");
            context.fillStyle="#FF0000";
            context.fillRect(0,0,150,75);
        }
</script>
var drawingCanvas = document.getElementById("mycanva");
```

Στην πρώτη γραμμή του κώδικα της function καθορίζεται η περιοχή canvas στην οποία θα σχεδιαστούν γραφικά.

```
var context = drawingCanvas.getContext("2d");
```

Έπειτα, δημιουργείται ένα αντικείμενο getContext("2d") το οποίο είναι μέρος της HTML5 και το οποίο έχει πολλούς μεθόδους για τη σχεδίαση γραφικών.

context.fillStyle="#FF0000";

context.fillRect(0,0,150,75);

Οι δύο επόμενες γραμμές ορίζουν ένα κόκκινο (#FF0000) ορθογώνιο με διαστάσεις 150x75. Η μέθοδος *fillStyle* ορίζει χρώμα στο γραφικό, ενώ η μέθοδος *fillRect* ορίζει τις συντεταγμένες ενός ορθογωνίου επάνω στην περιοχή canvas. Στην προκειμένη περίπτωση δημιουργεί ένα ορθογώνιο το οποίο η επάνω αριστερή του γωνία θα είναι στο σημείο 0,0 της περιοχής canvas και θα έχει μήκος 150 pixels και ύψος 75 pixels.

4.1.4 Το στοιχείο SVG της HTML5

Το SVG, είναι το ακρωνύμιο του «κλιμακώσιμα διανυσματικά γραφικά», που είναι μια αυξανόμενα δημοφιλής μορφή αρχείων για διανυσματικά γραφικά, όπου τα γραφικά στοιχεία παριστάνονται σε μορφή ανεξάρτητη από την ανάλυση, σε αντίθεση με τα εικονογραφικά, όπου τα γραφικά στοιχεία παριστάνονται ως διανύσματα εικονοστοιχείων. Είναι μια γλώσσα inline, δηλαδή ενσωματώνεται στην HTML5.

Η μορφή SVG χειρίζεται πολλά γραφικά στοιχεία μεταξύ αυτών, σχήματα όπως τετράγωνα, ορθογώνια, κύκλους, ελλείψεις, κανονικά πολύγωνα κ.α.

Χαρακτηριστικά και πλεονεκτήματα του SVG:

- Το SVG ορίζει τα γραφικά σε μορφή XML
- Τα Γραφικά SVG δεν χάνουν την ποιότητα αν αλλάξουν μέγεθος

- Κάθε στοιχείο και κάθε χαρακτηριστικό σε αρχεία SVG μπορεί να είναι κινούμενο

- Το SVG είναι πρότυπο του W3C

- Οι εικόνες SVG μπορούν να εκτυπωθούν με υψηλή ποιότητα σε οποιαδήποτε ανάλυση

- Μπορούν να δημιουργηθούν και να επεξεργαστούν με οποιοδήποτε πρόγραμμα επεξεργασίας κειμένου

4.1.4.1 Διαφορές του στοιχείου CANVAS και SVG

Canvas

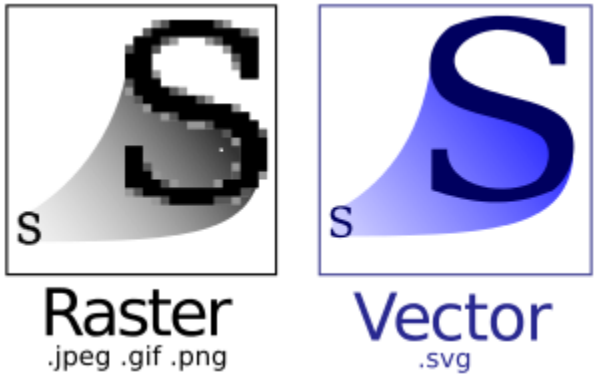
1. Εξαρτάται από την ανάλυση.
2. Βασίζεται στη δημιουργία δυσδιάστατων γραφικών με JavaScript.
3. Δεν υποστηρίζει event-handlers (χειρισμό συμβάντων).
4. Χαμηλή ποιότητα απόδοσης δυνατοτήτων κειμένου.
5. Κατάλληλο για υψηλής ποιότητας παιχνιδιών.
6. Η απόδοση γίνεται pixel προς pixel. Αυτό σημαίνει ότι τη στιγμή που το γραφικό δημιουργηθεί , έχει "ξεχαστεί" από το πρόγραμμα περιήγησης. Εάν η θέση του πρέπει να αλλάξει, ολόκληρη η σκηνή πρέπει να δημιουργηθεί εκ νέου.
7. Η εικόνα που προκύπτει μπορεί να αποθηκευτεί ως .png ή .jpg .

SVG

1. Δεν εξαρτάται από την ανάλυση.
2. Περιγράφει δυσδιάστατα γραφικά σε XML, που σημαίνει ότι κάθε στοιχείο είναι διαθέσιμο μέσω του svg DOM.
3. Υποστηρίζει event-handlers (χειρισμό συμβάντων).
4. Κατάλληλο για εφαρμογές σε μεγάλες περιοχές απόδοσης (Google Maps).
5. Ακατάλληλο για παιχνίδια, εξαιτίας αργής απόδοσης στην πολυπλοκότητα.

6. Κάθε σχήμα είναι ένα αντικείμενο. Εάν τα χαρακτηριστικά ενός αντικειμένου SVG αλλάξουν, το πρόγραμμα περιήγησης μπορεί να επανασυνδεθεί αυτόματα με το σχήμα.
7. Η εικόνα που προκύπτει μπορεί να αποθηκευτεί ως .svg.

Παρακάτω βλέπουμε μια εικόνα όπου δείχνει τη διαφορά μεταξύ διανυσματικών (vector) γραφικών και γραφικών ψηφίδων (bitmap).



Εικόνα 25 – Διαφορά μεταξύ vector και bitmap

ΚΕΦΑΛΑΙΟ 5

ΔΙΑΦΟΡΕΣ ΤΕΧΝΟΛΟΓΙΕΣ ΓΡΑΦΙΚΩΝ ΙΣΤΟΥ

5.1 Τεχνολογία VRML (Virtual Reality Modeling Language)

Εικόνα 26 – Τεχνολογία VRML

Η γλώσσα δημιουργίας εικονικής πραγματικότητας (VRML) είναι μία γλώσσα για την περιγραφή εικονικών κόσμων που είναι συνδεμένοι με το Internet και διασυνδεόμενοι με το παγκόσμιο Διαδίκτυο.

Η VRML, αποτελεί την πρώτη προσπάθεια για τη δημιουργία τρισδιάστατου προτύπου για το διαδίκτυο και βασίζεται στην OpenGL. Η όλη ιδέα ξεκίνησε στο πρώτο συνέδριο για το διαδίκτυο που πραγματοποιήθηκε στο κέντρο CERN της Γενεύης, όπου εισήχθη η ιδέα της VRML ως προτύπου για τρισδιάστατο διαδίκτυο. Το κίνητρο είχε δοθεί για τη δημιουργία μιας ομάδας εργασίας, η οποία παρουσίασε τελικά την VRML.

Αρχικά παρουσιάζεται η VRML1, η οποία έχει περιορισμένες δυνατότητες αλληλεπίδρασης και κίνησης, επιτρέποντας τη δημιουργία στατικών εικονικών κόσμων. Εξαιτίας σημαντικών ελλείψεων , γίνεται αναθεώρηση των προδιαγραφών και δημιουργείται η δεύτερη έκδοση, η οποία αποτελεί πρότυπο και καλείται VRML97(γνωστή και ως VRML 2.0).

Με τη VRML97, παρέχεται η δυνατότητα τρισδιάστατων δυναμικών σκηνών, όπου ο χρήστης μπορεί να πλοηγηθεί μέσω VRML browsers, που συνήθως αποτελούν plug-in για web browsers.

Τη βάση ανάπτυξης της VRML, αποτελεί η HTML , επομένως ακολουθεί τις ίδιες αρχές διαχείρισης των αντικειμένων.

Ειδικότερα η αρχές σχεδιασμού της αφορούν:

- Επεκτασιμότητα, δηλαδή δυνατότητα προσθήκης καινούριων τύπων.
- Αντικείμενα που δε προδιαγράφονται από τη VRML.
- Δυνατότητα ανάπτυξης σε διαφορετικά συστήματα.
- Τη δημιουργία πολύ μεγάλων τρισδιάστατων κόσμων.

Για να μπορέσουμε να δούμε τα αρχεία VRML, απαιτείται η εγκατάσταση ενός VRML plugin στον φυλλομετρητή μας, όπως ο Cortona 3D Viewer (Windows), ή ο FreeWRL (Mac / Linux).

Να τονίσουμε ότι τα εργαλεία ανάπτυξης ενός αρχείου VRML μπορεί να είναι είτε ένας Text editor, είτε εφαρμογή που είναι ειδική για τη δημιουργία κόσμων(λιγότερη ευελιξία), είτε με ένα πρόγραμμα σχεδίασης τρισδιάστατων γραφικών που να περιέχει έναν μετατροπέα σε VRML. Παρόλο που ο τελευταίος παρέχει πολύ ισχυρά εργαλεία σχεδίασης, ο κώδικας που παράγεται είναι αναποτελεσματικός και επίσης δεν υποστηρίζει αρκετές δυνατότητες της γλώσσας.

Λόγω των υψηλών απαιτήσεων της σχετικά με τη μνήμη και τη δύναμη, χρειάζεται ένα πολύ εξελιγμένο υπολογιστή. Επιπλέον, αυτές οι ελάχιστες απαιτήσεις συστήματος δεν παρέχουν την καλύτερη ποιότητα θέασης για το χρήστη.

Οι VRML κόσμοι αποθηκεύονται σήμερα και κατεβαίνουν ως αρχεία κειμένων, που λέγεται ότι συμβάλλει στον υπερβολικό χρόνο λήψης . Ωστόσο, η Apple και η IBM έχουν εργαστεί μαζί για να δημιουργήσουν ένα δυαδικό αρχείο προδιαγραφής που επιτρέπει αρχεία VRML να είναι πολύ μικρότερα. Πλέον η VRML έχει αντικατασταθεί από την X3D.

5.2 Τεχνολογία X3D

Εικόνα 27 - Τεχνολογία X3D

Το πρότυπο X3D που το ακρωνύμιο σημαίνει eXtensible 3D Graphics, αποτελεί διάδοχο της VRML. Είναι ένας τύπος αρχείου για τρισδιάστατα γραφικά, που ακολουθεί τις αρχές της XML, για πραγματικού χρόνου μετάδοση τρισδιάστατων δεδομένων, προς όλες τις εφαρμογές, τόσο τοπικές όσο και δικτυακές.

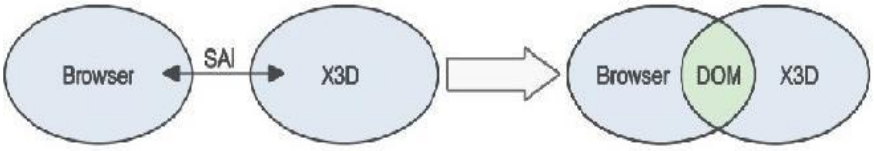
Μπορεί να χρησιμοποιηθεί στην αρχιτεκτονική, στην ιατρική, στην εκπαίδευση, καθώς και σε πληθώρα άλλων τομέων και λόγω των πλούσιων χαρακτηριστικών, των εξελιγμένων APIs και των πιο αυστηρών προδιαγραφών, αποτελεί πιο ώριμο πρότυπο από την VRML.

Το X3D είναι ένα επεκτάσιμο και ανοιχτό πρότυπο λογισμικού για τον ορισμό πραγματικού χρόνου (real-time), τρισδιάστατου περιεχομένου και μοντελοποίηση οπτικών εφέ, το οποίο μπορεί να χρησιμοποιηθεί σε μια μεγάλη γκάμα συσκευών και εφαρμογών.

Η X3DOM, αποτελεί προσπάθεια της Fraunhofer Institute , για την υποστήριξη απεικόνισης τρισδιάστατων σκηνών που περιγράφονται σε μορφή X3D , μέσα στο περιβάλλον του φυλλομετρητή χωρίς τη χρήση plugins, χρησιμοποιώντας μόνο τη WebGL και τη JavaScript .

Επιτρέπει την απεικόνιση 3D αντικειμένων, χωρίς να απαιτούνται κάποια πρόσθετα λογισμικά (όπως ο adobe flash player ή java platform), αλλά με εγγενής υποστήριξη μέσω της HTML5 και με έναν web browser που να την υποστηρίζει.

Η SAI (Scene Access Interface), που φαίνεται στην παρακάτω εικόνα είναι μια διεπαφή που έδινε τη δυνατότητα σε έναν προγραμματιστή να αλλάξει ή να χτίσει κόσμους X3D.



Εικόνα 28 - Scene Access Interface

Χαρακτηριστικά X3D:

1. XML, το κλειδί για την ενσωμάτωση υπηρεσιών web, κατανεμημένων δικτύων, cross-platform.
2. Επεκτασιμότητα, επιτρέπει την πρόσθεση συστατικών για την επέκταση της λειτουργικότητας σε εφαρμογές και υπηρεσίες.
3. Τυποποιημένα σύνολα επεκτάσεων καλύπτουν συγκεκριμένες ανάγκες εφαρμογών.
4. Εξελικτική ,εύκολο να ενημερώσει VRML97 κώδικα σε X3D.
5. Γραφικά υψηλής ποιότητας σε πραγματικό χρόνο.
6. Καλά-ορισμένη, δηλαδή είναι ευκολότερο να χτίσει συνεπή και χωρίς λάθος υλοποιήσεις.

Δυνατότητες X3D:

1. Τρισδιάστατα γραφικά .
2. Δισδιάστατα γραφικά.
3. Κίνηση(animation).
4. Χωρικό ήχο και video, δυνατότητα τοποθέτησης οπτικοακουστικών πη-γών πάνω στη σκηνή.
5. Αλληλεπίδραση με το χρήστη.
6. Πλοήγηση με χρήση κάμερας.
7. Scripting, επιτρέποντας δυναμικές αλλαγές στη σκηνή μέσω προγραμ-ματισμού και scripting.
8. Χρήση διαδικτύου(Networking), δυνατότητα σύνθεσης X3D σκηνής, αξι-οποιώντας ποικίλα στοιχεία του διαδικτύου.
9. Επικοινωνία σε πραγματικό χρόνο και φυσική εξομοίωση.

5.3 Τεχνολογία O3D

Εικόνα 29 - Τεχνολογία O3D

Η O3D, είναι μια JavaScript API ανοιχτού κώδικα (open-source), η οποία δημιουργήθηκε από τη Google για τη δημιουργία τρισδιάστατων διαδραστικών εφαρμογών που τρέχουν σε έναν web browser ή σε ένα XUL desktop application.

Στις 7 Μαΐου 2010, η Google ανακοίνωσε ότι η O3D θα άλλαζε από plugin web browser, σε μια βιβλιοθήκη JavaScript υλοποιημένη πάνω στην WebGL.

Αρχικά, η O3D χρησιμοποίησε μια αρχιτεκτονική plugin ,η οποία επέτρεπε στους προγραμματιστές να ενσωματώσουν προσαρμοσμένες λειτουργίες, όπως φωτορεαλιστικές κινήσεις. Είναι σημαντικό να σημειωθεί ότι το plugin γράφτηκε σε C , που επικοινωνεί απευθείας με το υλικό και έτσι η ταχύτητα της σκηνής απόδοσης εξαρτιόταν σε μεγάλο βαθμό από την κάρτα γραφικών του υπολογιστή. Τώρα, μεγάλο μέρος αυτής της ίδιας λειτουργικότητας είναι ενσωματωμένη στο WebGL.

Η O3D μπορεί να κατασκευάστηκε για διάφορες περιοχές εφαρμογών, ωστόσο, είναι προσανατολισμένη προς τα παιχνίδια, τις διαφημίσεις, τις επιδείξεις προϊόντων, προσομοιώσεις, μηχανολογικές εφαρμογές, συστήματα ελέγχου και παρακολούθησης και μαζικούς online εικονικούς κόσμους.

Η O3D είναι διαθέσιμη στον Internet Explorer, Firefox, Chrome και Safari και λειτουργεί σε οποιοδήποτε Linux, Mac και Windows που έχει

DirectX 9, PS 2.0 κάρτα γραφικών. Επίσης επί του παρόντος δεν είναι διαθέσιμη σε όλες τις πλατφόρμες κινητής τηλεφωνίας.

Το κύριο πλεονέκτημα O3D είναι ότι μπορεί να φορτώσει, να αποδώσει φωτορεαλιστικά και να μετατρέψει τα μοντέλα και τις αντίστοιχες συστάσεις τους δυναμικά, χρησιμοποιώντας AJAX ή και COMET σε πραγματικό χρόνο.

Η σύνταξη του πηγαίου κώδικα, των πόρων εφαρμογής και των αντικειμένων δεν είναι πλέον απαραίτητη, δεδομένου ότι όλες αυτές οι πτυχές έχουν φορτωθεί σε πραγματικό χρόνο.

Τα αρχεία που χρησιμοποιεί, περιορίζονται στο πρότυπο collada (μια μορφή ανταλλαγής αρχείων για διαδραστικές εφαρμογές 3D, που ορίζει ένα ανοιχτό πρότυπο XML για την ανταλλαγή ψηφιακών στοιχείων μεταξύ των διαφόρων γραφικών εφαρμογών λογισμικού) και Google Sketch up. Εν τέλει ανακοινώθηκε, ότι δε θα αποτελεί αυτόνομη τεχνολογία, αλλά θα εκτελείται χρησιμοποιώντας τη WebGL.

Στο παρακάτω σύνδεσμο, παρουσιάζεται ένα παιχνίδι / demo (προσομοιώνεται το γνωστό σε όλους παιχνίδι μπιλιάρδο) από την Google, που χρησιμοποιεί την O3D, βασισμένη στη JavaScript και τη WebGL.

http://o3d.googlecode.com/svn/trunk/samples_webgl/o3d-webgl-samples/pool.html

Εικόνα 30 - Παράδειγμα παιχνιδιού (Pool)

5.4 Τεχνολογία JAVA3D

Η java3D, προσφέρει μια συλλογή από υψηλού επιπέδου δομές για τη δημιουργία, την απόδοση και το χειρισμό ενός 3D γράφου σκηνής, ο οποίος αποτελείται από γεωμετρικά σχήματα, υλικά , φώτα, ήχους κ.α.

Java 3D

Developer(s)	Sun Microsystems & JogAmp Community
Stable release	**1.5.2** / 2008
Preview release	**1.6.0-pre8** / 30-Jul-2013
Operating system	Cross-platform
Type	3D computer graphics software (library/API)
License	GPL version 2+GPL linking exception
Website	JogAmp's Java3D Continuation forum

Εικόνα 31 - Java 3D

Υπάρχουν 2 παραλλαγές της Java3D, η μία στηρίζεται στην OpenGL και η άλλη στο DirectX Graphics.

Ο προγραμματιστής, βλέπει το σύστημα γραφικών μέσω των βιβλιοθηκών:

- OpenGL
- DirectX
- Java 3D

Το 1992 η εταιρία Silicon Graphics, παρουσίασε στην αγορά το πακέτο **OpenGL**(Open Graphics Library), είναι μια βιβλιοθήκη 2D/3D γραφικών ανεξάρτητης συσκευής(desktop, laptop, mobile phones) και ανεξαρτήτου λειτουργικού συστήματος (Windows , Linux/Unix, Mac OS X).

Είναι ένα low-level API, βασισμένο σε μια διασωλήνωση γραφικών (graphic pipeline) για τον χειρισμό των εικονοστοιχείων και των κορυφών.

Δύο χρόνια αργότερα, η Microsoft παρουσίασε το δικό της προϊόν, με τη μορφή του API που ονομάστηκε DirectX, που ενσωμάτωνε δυνατότητες παραγωγής multimedia για τη νέα γενιά συστημάτων που ξεκίνησε με τα Windows 95.

Το DirectX συνιστούσε τη μετεξέλιξη του WinG (που λειτουργούσε σε περιβάλλον Windows 3.x) και αποτέλεσε ικανό κίνητρο για τους κατασκευαστές παιχνιδιών να εγκαταλείψουν το MS-DOS ως βάση ανάπτυξης του λογισμικού τους και να στραφούν στα Windows, αφού τους έδινε την ίδια ελευθερία και ταχύτητα διαχείρισης του hardware.

Διάφορες εναλλακτικές προτάσεις εμφανίστηκαν στην αγορά, με τη μορφή των SDL, OpenMAX, OpenML, OpenCL, FMOD, αλλά η κυριαρχία των OpenGL και DirectX δεν αμφισβητήθηκε.

Το DirectX graphics, υποστηρίζει μια κλασική διασωλήνωση γραφικών, η οποία περιγράφει κάθε γεωμετρικό σχήμα μέσω εικονοστοιχείων και κορυφών. Αποτελεί τμήμα του DirectX, μιας συλλογής μονάδων λογισμικού για βιντεοπαιχνίδια για την πλατφόρμα των Windows. Τα υπόλοιπα DirectX APIs, υποστηρίζουν 3D ήχο, δικτύωση , πολυμέσα κ.α.

5.4.1 Direct X Graphics ή OpenGL?

Η απάντηση στην ερώτηση για το ποία έκδοση της Java3D είναι καλύτερη, απαιτεί διερεύνηση των σχετικών πλεονεκτημάτων της OpenGL έναντι του DirectX Graphics.Όσον αφορά τις τεχνικές λεπτομέρειες, τα δύο APIs , είναι σχεδόν ισοδύναμα, καθώς βασίζονται στην ίδια ιδέα, της αρχιτεκτονικής διασωλήνωσης των γραφικών.

Η πιο σημαντική διαφορά, εντοπίζεται στην φορητότητα τους (portability), με την OpenGL από τη μια να καλύπτει ένα μεγάλο εύρος από πλατφόρμες και λειτουργικά συστήματα (αφού αποτελεί μια συνεργασία πολλών εταιριών) και από την άλλη το DirectX να

περιορίζεται αποκλειστικά σε υπολογιστές με Windows (άρα ελέγχεται αποκλειστικά από αυτήν) και στην παιχνιδομηχανή Xbox.

5.4.2 Πλεονεκτήματα της Java 3D

Τα κύρια πλεονεκτήματα της είναι:

1. Χρησιμοποιεί υψηλού επίπεδου μοντέλο για την περιγραφή της σκηνής, ο οποίος απλουστεύει τον 3D προγραμματισμό και επιταχύνει τον κώδικά, που με τη σειρά του είναι ευανάγνωστος, επαναχρησιμοποιήσιμος και εύκολος να γραφτεί.
2. Η επίδοσή της.
3. Το σύνολο των μοναδικών της χαρακτηριστικών.
4. Το γεγονός ότι η java ως γλώσσα προγραμματισμού είναι πολύ διαδεδομένη και διαθέτει ένα τεράστιο αριθμό υποστηρικτικών πακέτων APIs.
5. Η εκτενής τεκμηρίωση και τα πολλά παραδείγματα.

5.4.3 Μειονεκτήματα της Java 3D

Τα κύρια μειονεκτήματα της είναι:

1. Εάν η προτεραιότητα του προγραμματιστή είναι η μεγαλύτερη απόδοση της εφαρμογής του, τότε αντί της java3D, είναι προτιμότερο να χρησιμοποιήσει OpenGL και C.
2. Συλλέκτης απορριμμάτων της Java (GC –Garbage Collector). Κατά το χρόνο εκτέλεσης της Java3D, δημιουργούνται αντικείμενα που στην πορεία γίνονται σκουπίδια. Αυτά στη συνέχεια καθώς συλλέγονται από τον GC, μπορεί να δημιουργηθεί σημαντική επιβράδυνση του συστήματος με αποτέλεσμα να χαθούν κάποια frames(μείωση ρεαλισμού).

5.5 Τεχνολογία CSS3

Εικόνα 32 - Τεχνολογία CSS3

Τα αρχικά της λέξης CSS αντιστοιχούν στο Cascading Style Sheets. Είναι μια απλή γλώσσα που μας βοηθάει να ορίσουμε με σαφήνεια και ιδιαίτερη ευελιξία, τον τρόπο με τον οποίο θα εμφανίζονται τα διάφορα στοιχεία στην ιστοσελίδα μας.

Το γενικό αίσθημα πίσω από το 3D στο DOM είναι ότι η HTML περιγράφει τη δομή ενώ η CSS 3D περιγράφει την οπτική παρουσίαση. Εφ 'όσον τηρείται αυτή η σχέση, υφίσταται ένα πολύ λογικό μονοπάτι διασύνδεσης μεταξύ τους. Πράγματα όπως το μέγεθος των γραμματοσειρών, το χρώμα που θα έχει το φόντο και τα περιθώρια καθορίζονται από την CSS. Οι δυνατότητες της γλώσσας βέβαια, δεν σταματούν μόνο σε αυτά.

Η CSS3 είναι η τρίτη έκδοση της γλώσσας αυτής CSS και είναι πλήρως συμβατή με τις προηγούμενες εκδόσεις, ενώ χωρίζεται σε ενότητες, διατηρώντας και τις παλιές προδιαγραφές του CSS.

Μερικές από τις πιο σημαντικές ενότητες του CSS3 είναι τα εφέ κειμένου(Text Effects), το φόντο και τα περιγράμματα, τα animations καθώς και οι 2D/3D μεταμορφώσεις(CSS3 3D Transforms). Μπορούμε

να μετακινήσουμε, να περιστρέψουμε, καθώς και να τεντώσουμε κάποια στοιχεία.

Ο μετασχηματισμός είναι ένα εφέ, που επιτρέπει σε ένα στοιχείο να αλλάξει το σχήμα, το μέγεθος αλλά και τη θέση του.

Μερικές από τις πιο σημαντικές ενότητες CSS3 είναι:

- Οι επιλογείς (Selectors)
- Το Φόντο και τα όρια (Backgrounds and Borders)
- Τα εφέ κειμένου (Text Effects)
- Οι 2D / 3D Μετασχηματισμοί (2D/3D Transformations)
- Τα Κινούμενα σχέδια (Animations)
- Περιβάλλον χρήστη (User Interface)

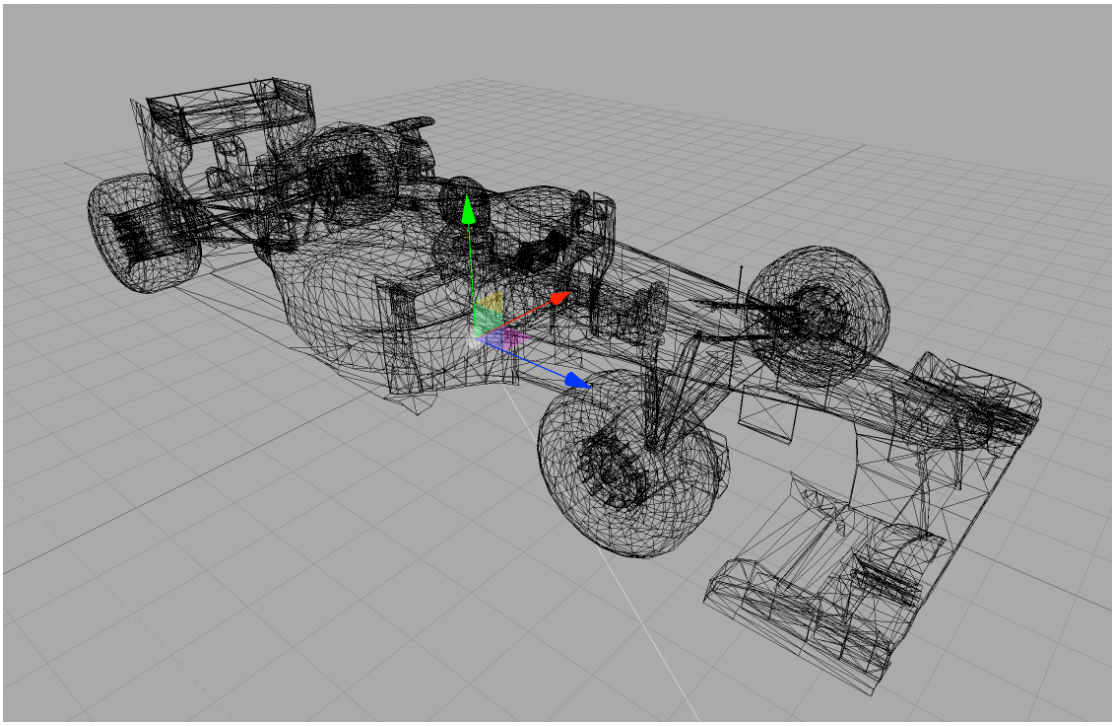
ΣΥΜΠΕΡΑΣΜΑΤΑ

Η δημιουργία της παρούσας πτυχιακής εργασίας , είχε ως σκοπό την ανάλυση των Web βιβλιοθηκών γραφικών καθώς και τον τρόπο δημιουργίας κώδικα σε JavaScript, βασιζόμενο στην Three.js, μια ελαφριά, cross-browser JavaScript βιβλιοθήκη που χρησιμοποιείται για να δημιουργήσει και να εμφανίσει κινούμενα 3D γραφικά υπολογιστών σε ένα πρόγραμμα περιήγησης. Ο πηγαίος κώδικας φιλοξενείται σε ένα αποθετήριο (χώρο αποθήκευσης) στο Cloud.

Έτσι με γνώσεις προγραμματισμού διαδικτυακών εφαρμογών και με τη χρήση της 3D JavaScript βιβλιοθήκης, διαμορφώσαμε παραδείγματα με σκηνές στις οποίες προσθέσαμε αντικείμενα, τα οποία αλληλεπιδρούσαν με το χρήστη. Επίσης, προσθέσαμε φωτισμό, χρώμα, σκιές και κίνηση χρησιμοποιώντας την Three.js.

Επιπλέον, εξετάσαμε την HTML5 και τα πλεονεκτήματα που έχει σχετικά με προηγούμενες εκδόσεις της HTML, καθώς και τις διαφορές ανάμεσα σε άλλες τεχνολογίες.

Τέλος, αναφέραμε την υποστήριξη και συμβατότητα που αυτή έχει σε σχέση με τα λειτουργικά περιβάλλοντα.



Εικόνα 33 - Three.js

ΒΙΒΛΙΟΓΡΑΦΙΑ

1) Πασχάλης Ράπτης Γραφικά Υπολογιστών
http://aetos.it.teithe.gr/~praptis/CG/CG-page.htm

2) Brian Danchilla, Beginning WebGL for HTML5.
www.it-ebooks.info

3) Nick Pettit (2013) The Beginner's Guide to three.js.
http://blog.teamtreehouse.com/the-beginners-guide-to-three-js

4) Mitch Williams, WebGL, Create interactive 3D content for web pages and mobile devices.
www.it-ebooks.info

5) Brian Danchilla, (2012) , Beginning WebGL for HTML5,
Xavier Boyry, WebGL Academy: Tutorial to learn WebGL.
http://www.webglacademy.com/

6) Dimitris Kamileris Heraklion (22/8/2012) Real-time shader-based Per-fragment lighting for dynamic objects in WebGL.
http://dkamileris.weebly.com/uploads/4/5/7/6/45765585/diploma_thesis.pdf

7) Jos Dirksen, Three.js Cookbook
www.it-ebooks.info

8) Jerome Etienne (2012) Casting Shadows.
http://learningthreejs.com/blog/2012/01/20/casting-shadows/

9) Sumeet Arora, WebGL Game Development.
www.it-ebooks.info

10)
HTML Canvas Deep Dive
http://files.joshondesign.com/books/canvasdeepdive/

11)

Script Tutorials
https://www.script-tutorials.com/

12)
Wikipedia Three.js
https://en.wikipedia.org/wiki/Three.js

13)
Three.js/doc
http://threejs.org/docs/#Reference/Extras.Geometries/BoxGeometry

14)
Jos Dirksen, Learning Three.js: The JavaScript 3D Library for WebGL.
www.it-ebooks.info

15)
Three.js examples
http://threejs.org/

16)
Βαμβασάκης Δημήτριος, (2005), Πτυχιακή στο Opencart CMS eShop με ενσωμάτωση X3D Objects σε HTML5, Greece.

17)
Μιχάλης Σαλαμπάσης, (2008), Εισαγωγή στον προγραμματισμό διαδικτυακών εφαρμογών, Ελλάδα.

18)
Wikipedia, List of web browsers.
https://en.wikipedia.org/wiki/List_of_web_browsers

19)
Nicholas C. Zakas, Professional JavaScript for Web Developers
www.it-ebooks.info

20)
HTML5, W3C.
http://www.w3.org/TR/html5/

21)
Jos Dirksen, Three.js Essentials, Create and animate beautiful 3D graphics with this fast-paced tutorial.
www.it-ebooks.info

22)
HTML5 Tutorial, The world's largest web developer site.
http://www.w3schools.com/html/html5_intro.asp