Τ.Ε.Ι. ΗΠΕΙΡΟΥ - ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

με θέμα

Προσομοίωση φυσικών συστημάτων και απεικόνισης δεδομένων σε 3D Web περιβάλλον με τη χρήση της τεχνολογίας WebGL & HTML5

Ειρήνη Δ. Τσιάκαλου
Α.Μ.: 9388

Εξεταστική επιτροπή
Επιβλέπων καθηγητής: Χρήστος Κολιοπάνος
Μέλη: Κωνσταντίνος Αγγέλης, Λάμπρος Σακκάς

ΆΡΤΑ 2014

Περιεχόμενα

Ευχαριστίες

Θα ήθελα να ευχαριστήσω θερμά τον καθηγητή μου κ. Χρήστο Κολιοπάνο για την ανάθεση της παραπάνω πτυχιακής καθώς επίσης και για την καθοδήγηση που μου πρόσφερε κατά τη διάρκεια υλοποίησής της. Επίσης θερμές ευχαριστίες απευθύνω σε όλους τους καθηγητές μου που είχα όλα τα χρόνια της ακαδημαϊκής ζωής.

Τέλος ένα μεγάλο ευχαριστώ στην οικογένεια μου για την στήριξη που μου πρόσφερε.

Περίληψη

Στην εργασία αυτή εξετάζεται η προσομοίωση φυσικών συστημάτων και η απεικόνιση δεδομένων σε 3D Web περιβάλλον με την χρήση της τεχνολογίας WebGL και της HTML5. Με τον όρο προσομοίωση ορίζουμε τη μέθοδο εκείνη της μελέτης ενός συστήματος (πχ. ενός αντικειμένου, μιας διαδικασίας κα.) με την βοήθεια ενός άλλου συστήματος (υπολογιστική μηχανή). Η προσομοίωση δηλαδή είναι μια αναπαράσταση ή ένα μοντέλο που έχει κατασκευαστεί για να αναπαραστήσει και να επιτρέψει την κατανόηση της λειτουργίας ενός άλλου συστήματος. Μας επιτρέπει τον πειραματισμό πάνω σε ένα σύστημα χωρίς να έχουμε άμεση επαφή μαζί του. Η απεικόνιση των δεδομένων σε 3D περιβάλλον από την άλλη απαιτεί την χρήση μιας νέας σχετικά τεχνολογίας που καλείται WebGL. Η WebGL χρησιμοποιεί JavaScript σενάρια τα οποία αναμιγνύονται με ετικέτες (tags) της HTML5 και σε συνδυασμό με τις υπηρεσίες του Διαδικτύου (internet) επιτυγχάνεται η απεικόνιση του αντικειμένου σε 3D Web περιβάλλον.

Αυτό που εμείς θα επιχειρήσουμε είναι η δημιουργία μιας διαδραστικής εφαρμογής με χρήση 3D γραφικών. Πιο αναλυτικά με τον όρο διαδραστικά εννοούμε ότι το πληροφοριακό μας σύστημα περιέχει προγράμματα που του επιτρέπουν να ερμηνεύσει κάποιες πληροφορίες απεικονίζοντάς τις άμεσα στην οθόνη του υπολογιστή δίνοντας παράλληλα τη δυνατότητα στο χρήστη να ενεργήσει πάνω στα αντικείμενα. Με τον όρο γραφικά περιγράφουμε τη δημιουργία, την αποθήκευση και το χειρισμό μοντέλων και εικόνων. Σύμφωνα με τον William A. Fetter (1966) γραφικά είναι «μια συνειδητή διαχείριση και τεκμηριωμένη διαδικασία με σκοπό την κοινωνική ενημέρωση περιγραφικά και με ακρίβεια».

Η εργασία είναι χωρισμένη σε τρία μέρη που στο σύνολό τους δίνουν μια σαφή εικόνα για το πώς μια γεωμετρική εικόνα θα μεταβεί από το στάδιο του αρχικού ορισμού της στο τελικό στάδιο που είναι η εμφάνισή της σε 3D Web περιβάλλον .

Στο πρώτο κεφάλαιο περιγράφεται η τεχνολογία WebGL και πως αυτή μας επιτρέπει την δημιουργία διαδραστικών 3D γραφικών μέσα σε ένα πρόγραμμα περιήγησης web (Firefox) προσφέροντάς μας τη δυνατότητα της ρεαλιστικής απεικόνισης των δεδομένων στην οθόνη του υπολογιστή. Επίσης θα πειραματιστούμε στην δημιουργία 3D απεικονίσεων με τη χρήση Json.

Στο δεύτερο κεφάλαιο, όπου και αποτελεί το πρακτικό μέρος αυτής της πτυχιακής εργασίας, παρουσιάζονται όλα τα βήματα που

απαιτούνται για την δημιουργία εφαρμογής με σκοπό την απεικόνιση γεωμετρικών δεδομένων σε 3D Web περιβάλλον με τη χρήση της WebGL και της HTML5 καθώς επίσης και με τη χρήση ενός προγράμματος δημιουργίας 3D απεικονίσεων (Blender) μέσω της βιβλιοθήκης Three.js σε τοπικό περιβάλλον Apache. Οι δυνατότητες της εφαρμογής αυτής είναι η μετακίνηση του αντικειμένου γύρω από την τροχιά της κάμερας με κλικ και σύρσιμο του ποντικιού καθώς και η αλλαγή της εστιακής απόστασης του αντικειμένου με τη χρήση τροχού ποντικιού.

Τέλος το τρίτο μέρος αποτελεί το ΠΑΡΑΡΤΗΜΑ στο οποίο παραθέτονται γενικές πληροφορίες για το δίκτυο, το διαδίκτυο, το Διαδίκτυο, παγκόσμιο ιστό, το πρόγραμμα περιήγησης ιστού, τον εξυπηρετητή, τον εξυπηρετητή Apache, την HTML γλώσσα, την HTML5, την JavaScript γλώσσα και το CSS στυλ μορφοποίησης. Οι πληροφορίες αυτές είναι χρήσιμες για την καλύτερη κατανόηση της πτυχιακής εργασίας.

Σε όλες τις εικόνες της εργασίας παραθέτονται σχόλια για το τι περιγράφουν.

ΚΕΦΑΛΑΙΟ 1ο

Εισαγωγή

Η WebGL (Web Graphics Library) είναι μια νέα τεχνολογία η οποία μας επιτρέπει την δημιουργία διαδραστικών 3D γραφικών μέσα σε ένα πρόγραμμα περιήγησης web (Firefox) και μας προσφέρει τη δυνατότητα της ρεαλιστικής απεικόνισης των δεδομένων στην οθόνη του υπολογιστή μας. Αυτό επιτυγχάνεται χρησιμοποιώντας ένα JavaScript API το οποίο αλληλεπιδρά με την GPU (Graphics Processing Unit). Η WebGL χρησιμοποιεί JavaScript σενάρια τα οποία μπορούν να αναμειχθούν εύκολα με ετικέτες (tags) της HTML5. Πάνω σε αυτό θα βασιστούμε και θα αναλύσουμε την WebGL και των μεθόδων που χρησιμοποιεί για την κατασκευή 3D απεικονίσεων (3D scene).

1.1 Πως δουλεύει η WebGL ;

Όπως αναφέραμε πιο πάνω η WebGL είναι ένα JavaScript API μεταξύ της CPU (Central Processing Unit) και της GPU (Graphic Processing Unit) ενός υπολογιστή. Το API context (API-πλαίσιο) που πρέπει να αποκτήσουμε ενσωματώνεται στην HTML5 διαμέσου της ετικέτας <canvas>. Αυτό σημαίνει ότι δεν απαιτείται plug-in (πρόσθετο πρόγραμμα) στο πρόγραμμα περιήγησης. Απαραίτητη είναι η δημιουργία του shader-προγράμματος το οποίο χρησιμοποιεί GL Shading Language (GLSL, παρόμοια με τη C++) γλώσσα προγραμματισμού και σκοπό έχει να κάνει compile κατά το χρόνο εκτέλεσης (runtime). Στο shader πρόγραμμα μεταβαίνουν όλα τα attributes του αντικειμένου.

Η WebGL σχεδιάστηκε και συντηρείται από τη μη κερδοσκοπική εταιρία Khronos και βασίστηκε στο μοντέλο OpenGL ES 2.0. Η OpenGL ES είναι μια διεπαφή προγραμματισμού εφαρμογών (API) για την επικοινωνία με την κάρτα γραφικών. Στο σημείο αυτό επισημαίνουμε πως έχουμε τη δυνατότητα να δημιουργούμε WebGL "3D απεικονίσεις" (scenes) χωρίς προγραμματισμό χρησιμοποιώντας προγράμματα δημιουργίας τέτοιων απεικονίσεων όπως το Blender, Autodesk κα.

Ορισμένες από τις χρήσεις της είναι η οπτικοποίηση επιστημονικών δεδομένων, η αναπαράσταση και ο χειρισμός μοντέλων και σχεδίων, εικονικές περιηγήσεις, εφαρμογές ψυχαγωγίας κα.

1.2 Υποστηρίζει ο web browser την WebGL ;

Η υποστήριξη του προγράμματος περιήγησης γίνεται με την αναβάθμιση σε μια νεότερη έκδοση του web browser. Συμβατές εκδόσεις είναι η Firefox 4+, Chrome 9+, Safari 5.1 . Παράλληλα θα πρέπει να διαθέτουμε ένα κατάλληλο λειτουργικό σύστημα (OS Win7) καθώς και μια ισχυρή κάρτα γραφικών (INVDIA). Υπενθυμίζουμε ότι θα πρέπει να ενημερώνουμε συχνά (update) τους drivers (προγράμματα οδήγησης) γραφικών στην τελευταία τους έκδοση. Επίσης μπορούμε να ενισχύσουμε τους drivers γραφικών χρησιμοποιώντας το Microsoft Direct X.

1.3 JavaScript σενάρια αναμιγνύονται με tags της HTML5

Πριν εξηγήσουμε πως τα JavaScript σενάρια της WebGL αναμιγνύονται με στοιχεία της HTML5 θα αναφέρουμε κάποιες βασικές έννοιες. Αυτές είναι:

HTML Hypertext Markup Language: γλώσσα σήμανσης για την κατασκευή ιστοσελίδων. Χρησιμοποιεί **tags:** δομικά στοιχεία που εκτελούν εργασίες όπως η μορφοποίηση κειμένου, η ενσωμάτωση πολυμέσων, διαδραστικές εφαρμογές κα. (<title>Hello HTML</title>). Για την γραφή της χρησιμοποιούμε HTML editors: Notepad++, Geany κα. για Windows, Linux κα.

HTML5: εξελιγμένη εκδοχή της HTML παρέχει δυνατότητες που στηρίζονται σε HTML, CSS, DOM και JavaScript. Μειώνει τις ανάγκες για εξωτερικά plug-in πχ. το Flash, παρέχει εύκολο χειρισμό λαθών κα. Στοιχεία της HTML5: <canvas>, <video>, <audio>, <article>, <footer>, <header>, <nav>, <section> καθώς και νέα στοιχεία φόρμας όπως το ημερολόγιο, η ημερομηνία, η ώρα, το e-mail και url.

Για αναλυτικότερες πληροφορίες για την HTML και HTML5 στο ΠΑΡΑΡΤΗΜΑ.

1.3.1 JavaScript

Η **JavaScript** είναι μια γλώσσα προγραμματισμού των ηλεκτρονικών υπολογιστών για τον Παγκόσμιο ιστό (World Wide Web). Πρόκειται για μια γλώσσα προγραμματισμού σεναρίων, που

χρησιμοποιεί συναρτήσεις κατά πολύ επηρεασμένη από την C, ενσωματώνοντας χαρακτηριστικά όμως και από νεότερες γλώσσες όπως η Java.

Αρχικά χρησιμοποιήθηκε για προγραμματισμό από την πλευρά του πελάτη (client), που ήταν το πρόγραμμα περιήγησης του χρήστη και χαρακτηρίστηκε ως client-side γλώσσα προγραμματισμού. Αυτό σημαίνει ότι η επεξεργασία του κώδικα JavaScript και η παραγωγή του τελικού περιεχομένου της HTML δεν πραγματοποιείται στον εξυπηρετητή (server), αλλά στο πρόγραμμα περιήγησης των επισκεπτών (Firefox://localhost).

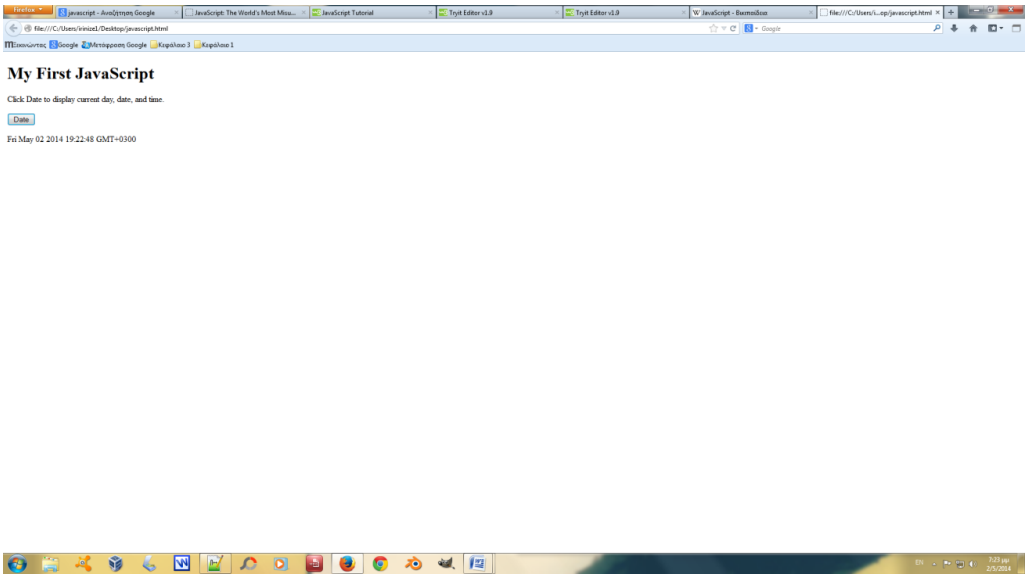
Χρησιμοποιείται για την κατασκευή ιστοσελίδων, για κατασκευή εγγράφων PDF κ.α. Τα τελευταία χρόνια βρίσκει πεδίο εφαρμογής στην ανάπτυξη εφαρμογών Ιστού από την πλευρά του εξυπηρετητή (server). Η JavaScript αποτελεί μια από τις δημοφιλέστερες γλώσσες προγραμματισμού ηλεκτρονικών υπολογιστών. (Για περισσότερη ενημέρωση επισκεφτείτε την διεύθυνση http://javascript.crockford.com/javascript.html

Να αναφέρουμε εδώ ότι πρέπει να ελέγξουμε εάν είναι ενεργοποιημένη η JavaScript στο πρόγραμμα περιήγησης Firefox.

Ακολουθεί ο index.html κώδικας για την ενσωμάτωση JavaScript σεναρίου.

```
<!DOCTYPE html>
<html>

<body>
        <h1>My First JavaScript</h1>
        <p>Click Date to display current day, date, and time.</p>

<button type="button" onclick="myFunction()">Date</button>

<p id="demo"></p>

<script>

function myFunction() {
   document.getElementById("demo").innerHTML = Date();
}
</script>

</body>
```

</html>



Εικόνα 1: απεικόνιση της ενσωμάτωσης JavaScript σεναρίου στο σώμα ενός .html εγγράφου (screenshot)

1.3.2 Το στοιχείο <canvas>

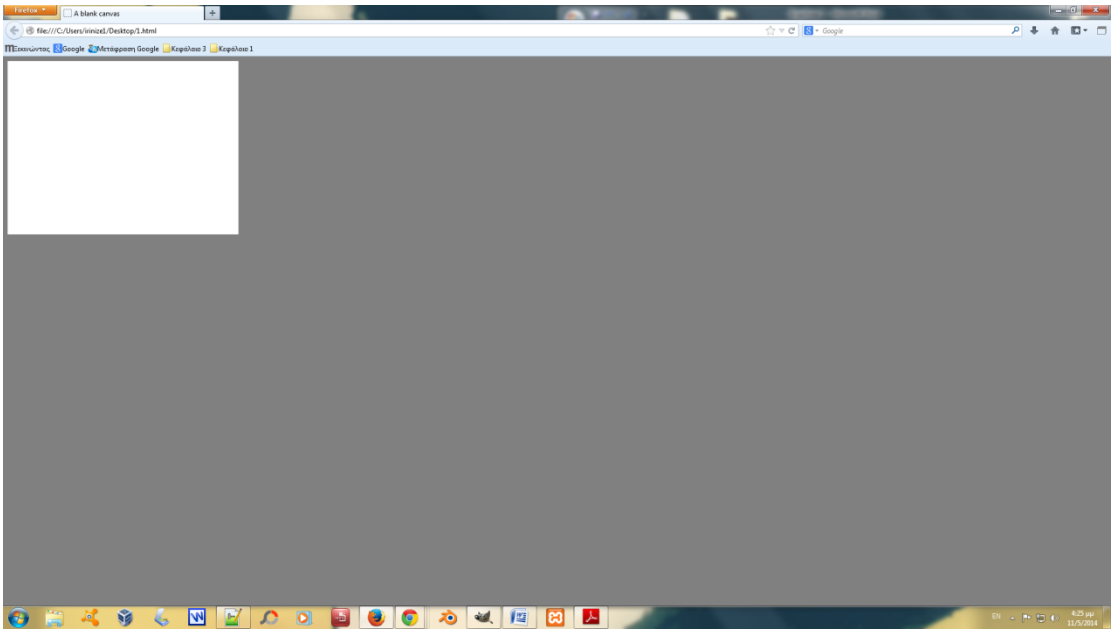
Η ανάμιξη των JavaScript σεναρίων της WebGL με την HTML5 γίνεται μέσω της ετικέτας canvas. Με τον όρο <canvas> εννοούμε το στοιχείο εκείνο της HTML5 που μας δίνει την δυνατότητα με τη χρήση JavaScript σεναρίων την εμφάνιση 3D γραφικών σε μια ιστοσελίδα. Ουσιαστικά πρόκειται για ένα "δοχείο" γραφικών όπου θα φιλοξενήσει το αντικείμενό μας ή διαφορετικά μια ορθογώνια περιοχή σε μια σελίδα HTML με όρια και περιεχόμενο.

1.3.3 Δημιουργία canvas σε HTML5 έγγραφο

Αναλυτικά ο κώδικας που απεικονίζει τη δημιουργία canvas σε ένα HTML έγγραφο.

```
<!doctype html>
<html>
	<head>
		<title>A blank canvas</title>
			<style>body{ background-color: grey; }
```

```
                canvas{ background-color: white; }
            </style>
    </head>

<body>

<canvas id="my-canvas" width="400" height="300">
Your browser does not support the HTML5 element.</canvas>

</body>

</html>
```



Εικόνα 1: απεικόνιση του στοιχείου canvas σε .html έγγραφο (screenshot)

Όπως βλέπουμε και στην εικόνα το περιεχόμενο του body είναι ένα μόνο στοιχείο canvas. Αν προβάλλουμε το έγγραφο με έναν παλιότερο browser που δεν υποστηρίζει το στοιχείο canvas θα εμφανιστεί το μήνυμα «Your browser does not support the HTML5 element».

1.4 Αποκτώντας το context

Όταν πρόκειται να σχεδιάσουμε (draw) σε ένα καμβά έχουμε

πολλές επιλογές για το πως θα παράγουμε την εικόνα μας. Αυτές οι επιλογές αντιστοιχούν σε διαφορετικά API (Application Programming Interface). Δηλαδή σε εφαρμογές, οι οποίες καθορίζουν πρωτόκολλα που προορίζονται να χρησιμοποιηθούν ως διεπαφές για την επικοινωνία μεταξύ των software με διαφορετική λειτουργικότητα και διαδικασίες εκτέλεσης. Η διαδικασία αυτή είναι γνωστή ως context of the canvas. Προς το παρόν έχουμε δύο ειδών canvas context που ονομάζονται "2D" και "webgl" και πρέπει να δηλωθεί ρητά ένα από τα δύο είδη.

Σε επίπεδο κώδικα για να αποκτήσουμε ένα context θα πρέπει να καλέσουμε τη μέθοδο (method) getContext η οποία δέχεται ως όνομα context το "2D" ή το "webgl". Το WebGL context όνομα που θα χρησιμοποιήσουμε είναι το " webgl", αλλά προς το παρόν το "experimental-webgl" είναι το όνομα που δέχονται οι περισσότεροι browsers. Η δήλωση API call για την απόκτηση του context είναι της μορφής gl = canvas.getContext("experimental-webgl") και τοποθετείται στην function setupWebGL().

Σε κάθε context εμπεριέχεται προαιρετικά και μια λίστα χαρακτηριστικών (attributes) του context που λέγεται WebGLContextAttributes. Οι ιδιότητες των χαρακτηριστικών αυτών μπορούν να μεταβιβαστούν στο αντικείμενο μας. Μερικές από τις ιδιότητες αυτές είναι:

```
dictionary WebGLContextAttributes {
boolean alpha = true;
boolean depth = true;
boolean stencil = false;
boolean antialias = true;
boolean premultipliedAlpha = true;
boolean preserveDrawingBuffer = false;
};
```

1.4.1 Index.html κώδικας για την απόκτηση του WebGL context

Αναλυτικά ο index.html κώδικας που εμπεριέχει το WebGL context σε ένα HTML έγγραφο.

```
<!doctype html>
<html>
    <head>
```

```
<title>WebGL Context</title>
<style>body{ background-color: red; }
canvas{ background-color: white; }
</style>

<script>
window.onload = setupWebGL;// καλείται η setupWebGL() και οι διεργασίες οι οποίες περιέχει με αποτέλεσμα την εμφάνισή του object στην οθόνη του υπολογιστή μας.

var gl = null;// η gl μεταβλητή περιέχει το context.
function setupWebGL()
{
var canvas = document.getElementById("my-canvas");
try{
gl = canvas.getContext("experimental-webgl");
}catch(e){
}
if(gl)
{//set the clear color to red
gl.clearColor(1.0, 0.0, 0.0, 1.0);
gl.clear(gl.COLOR_BUFFER_BIT);
}else{
alert( "Error: Your browser does not appear to support
WebGL.");
}
}// υλοποιώ τη συνάρτηση setupWebGL() για να αποκτήσω το context και ορίζω το χρώμα του canvas.
</script>

</head>

<body>
<canvas id="my-canvas" width="400" height="300">Your browser does not support the HTML5 canvas element.</canvas>

</body>

</html>
```

Αυτό που ουσιαστικά κάναμε εδώ είναι η ενσωμάτωση πρωτοκόλλων επικοινωνίας μεταξύ software με την επιλογή ενός

API. Να αναφέρουμε ότι δεν υπάρχουν αλλαγές στην οπτική απεικόνιση της σελίδας.

1.5 WebGL components

Στο σημείο αυτό θα κάνουμε μια περιγραφή των δομικών στοιχείων της WebGL ή αλλιώς WebGL components. Τα στοιχεία αυτά είναι:

- οι drawing buffers
- οι primitive types
- οι vertex storage mechanisms

1.5.1 Drawing buffers

Με τον όρο buffer εννοούμε το μπλοκ εκείνο της μνήμης το οποίο μπορεί να γραφτεί, να διαβαστεί και να αποθηκεύσει προσωρινά τα δεδομένα. Η WebGL διαθέτει τρείς τύπους buffer οι οποίοι είναι ο color buffer, ο depth buffer και ο stencil buffer.

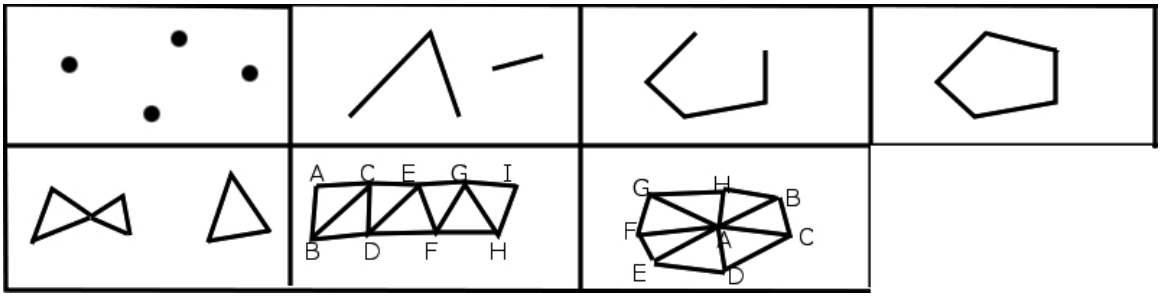
Πιο αναλυτικά ο color buffer αποθηκεύει πληροφορίες για το χρωματισμό σε RGB color model και προαιρετικά για την alpha-value δηλ. για το ποσοστό διαφάνειας/αδιαφάνειας (transparency /opacity). Ο depth buffer αποθηκεύει πληροφορίες για τον αριθμό των bits που χρησιμοποιούνται για να δηλώσουν το χρώμα ενός pixel σε μια εικόνα, ένα βίντεο ή κα. καθώς και πληροφορίες για τη z-value χρήσιμη για την αναπαράσταση 3D γραφικών (βάθος χρώματος). Ο stencil buffer αποθηκεύει πληροφορίες για να περιγράψει περιοχές στις οποίες θα κάνουμε ή όχι render (δημιουργία σκιών). Μόλις μια περιοχή της εικόνας σημανθεί ώστε να μη γίνει render η περιοχή αυτή καλείται masking area. Στον stencil buffer είναι αποθηκευμένη ολόκληρη η εικόνα καθώς και τα masking τμήματα. Μπορεί επίσης να χρησιμοποιηθεί σε συνδυασμό με το depth buffer.

1.5.2 Primitive κατηγορίες

Primitives στη WebGL καλούνται τα γραφικά δομικά στοιχεία (buildings blocks) με τα οποία κατασκευάζονται τα τρισδιάστατα

μοντέλα. Υπάρχουν τρία βασικά building blocks και επτά τρόποι χειρισμού τους. Τα building blocks είναι τα Points (σημεία) πρόκειται για κορυφές (vertices), οι Lines (γραμμές) που σχηματίζονται κατά ζεύγη (x,y) κορυφών και τα Triangles (τρίγωνα) που σχηματίζονται κατά τρίο (x,y,z) κορυφών και 7 τρόποι χειρισμού οι οποίοι είναι:

- points: vertices-κορυφές
- lines: δύο γραμμές μπορεί να έχουν κοινή κορυφή
- line_strip: είναι μια συλλογή κορυφών στην οποία, εκτός της πρώτης γραμμής, το αρχικό σημείο κάθε γραμμής είναι το τελικό σημείο της προηγούμενης (δεν ενώνονται ποτέ)
- line_loop: είναι μια συλλογή κορυφών στην οποία το αρχικό σημείο κάθε γραμμής είναι το τελικό σημείο της προηγούμενης (ενώνονται)
- triangles: τρίγωνα
- triangles_strip: χρησιμοποιεί τις δύο τελευταίες κορυφές του προηγούμενου τριγώνου για να σχηματίσει ένα νέο τρίγωνο σε συνδυασμό με την επόμενη κατά σειρά κορυφή. Πχ. ABC, (BC)D, (CD)E, (DE)F κτλ.
- triangles_fun: χρησιμοποιεί μια αρχική κορυφή η οποία αποτελεί μέρος κάθε σχηματιζόμενου τριγώνου



Εικόνα 2: 7 τρόποι χειρισμού των γραφικών δομικών στοιχείων (buildings blocks) της WebGL

1.5.3 Vertex Data - vertex storage mechanisms

Δυστυχώς σε κάποιες εκδόσεις της OpenGL ή του context 2D δεν μπορούμε να θέσουμε το color ή την position μιας κορυφής (vertex) απευθείας σε μια WebGL scene. Αυτό γιατί η WebGL δεν διαθέτει αυτή την λειτουργικότητα. Με την χρήση όμως των shaders το πρόβλημα αυτό λύνεται. Όλα τα δεδομένα που συσχετίζονται με μια κορυφή πρέπει να μεταδοθούν μέσω ενός JavaScript API στην

GPU (Graphics Processing Unit).

Για να γίνει εφικτό κάτι τέτοιο θα πρέπει να δημιουργήσουμε έναν vertex buffer object (VBO's) στον οποίο θα βρίσκονται αποθηκευμένα τα attributes και uniforms της κορυφής όπως η position, το color, τα normals και οι texture coordinates. Η υλοποίηση του γίνεται με τη function WebGLBuffer createBuffer().

Τα περιεχόμενα των VBO's εν συνεχεία στέλνονται στο shader και έτσι μπορούμε να χρησιμοποιήσουμε και να διαμορφώσουμε τα δεδομένα αυτά όπως εμείς επιθυμούμε.

1.5.3.1 VBO's - Vertex Buffer Objects

Buffers στους οποίους αποθηκεύουμε πληροφορίες για κάποιο συγκεκριμένο attribute των κορυφών μας. Αυτό το attribute μπορεί να είναι το position, το color, texture coordinates ή οτιδήποτε άλλο όπως προαναφέραμε. Για να δημιουργήσω έναν VBO σε επίπεδο κώδικα πρέπει να δημιουργήσω την συνάρτηση WebGLBuffer() η οποία θα αποθηκεύει τα δεδομένα και θα επιστρέφει ένα αποτέλεσμα. Στο σώμα της συνάρτησης δηλώνονται API calls.

var myBuffer = gl.createBuffer(); / δημιουργία buffer
gl.bindBuffer(gl. ARRAY_BUFFER, myBuffer); / δέσμευση buffer
Η παράμετρος gl.ARRAY_BUFFER ορίζει τον τύπο του buffer ο οποίος μπορεί να είναι είτε ARRAY_BUFFER που χρησιμοποιείται για attribute κορυφών όπως το color και η position ή ELEMENT_ARRAY_BUFFER που χρησιμοποιείται όταν οι κορυφές περιέχουν δείκτες.
var data = [1.0, 0.0, 0.0,
0.0, 1.0, 0.0,
0.0, 1.0, 1.0];/ δήλωση μεταβλητής που περιέχει
τιμές που αφορούν το position ή το color του αντικειμένου.
gl.bufferData(gl.ARRAY_BUFFER, data, STATIC_DRAW);/ τοποθετώ δεδομένα στον buffer. Η παράμετρος STATIC_DRAW θα θέσει τα δεδομένα μια φορά και τα οποία δεδομένα δεν αλλάζουν καθ' όλη τη χρήση της εφαρμογής απ' αυτό. Υπάρχουν επίσης και οι παράμετροι DYNAMIC_DRAW και STREAN_DRAW.

Ολοκληρωμένα η διαδικασία δημιουργίας και η δέσμευσης του buffer καθώς και η αποθήκευση δεδομένων έχει ως εξής:
var data = [1.0, 0.0, 0.0,

```
0.0, 1.0, 0.0,
0.0, 1.0, 1.0];
var myBuffer = gl.createBuffer();
gl.bindBuffer(gl.ARRAY_BUFFER, myBuffer);
gl.bufferData(gl.ARRAY_BUFFER, data, STATIC_DRAW);
```

1.5.3.2 Attributes and Uniforms

Εκτός από τα attributes (χαρακτηριστικά), έχουμε την δυνατότητα να μεταβιβάσουμε και uniforms (φόρμες) για κάθε κορυφή στους shaders. Αφού το shader πρόγραμμα είναι ένα μεταγλωττισμένο εξωτερικό πρόγραμμα πρέπει να είμαστε σε θέση να αναφέρουμε την θέση όλων των μεταβλητών μέσα στο πρόγραμμα. Μόλις αποκτήσουμε τη θέση μιας μεταβλητής, μπορούμε να στείλουμε τα δεδομένα στον shader από την web εφαρμογή.

Για να αποκτήσουμε τη θέση ενός attribute ή μιας uniform μέσω της WebGL θα πρέπει να χρησιμοποιήσουμε κάποιες API calls οι οποίες είναι οι Glint για τα attributes και η WebGLUniformLocation για τις uniforms. Και οι δύο επιστρέφουν τιμές οι οποίες αναφέρονται στη θέση του attribute ή της uniform μέσα στο shader. Και οι δύο κλήσεις περιέχουν τις δύο ίδιες παραμέτρους, η πρώτη είναι το WebGL πρόγραμμα και η δεύτερη το όνομα του attribute όπως δηλώθηκε στον vertex ή fragment shader. Οι API calls καλούνται μέσα στην συνάρτηση drawScene(). Το αποτέλεσμα ανατίθεται σε μια μεταβλητή.

```
vertexPositionAttribute=gl.getAttribLocation(glProgram,"aVertexPosi
tion");/ API call που επιστρέφει το attribute position
gl.enableVertexAttribArray(vertexPositionAttribute);
```

Αν στέλνουμε έναν πίνακα δεδομένων (array) σε ένα attribute τότε πρέπει να ενεργοποιήσουμε τον πίνακα δεδομένων. Εδώ γίνεται χρήση δείκτη (index) ό οποίος περιέχει τη θέση του attribute του προηγούμενου βήματος. Εδώ δεν επιστρέφεται κάποια τιμή.

Ως εδώ γνωρίζουμε τη θέση του attribute και έχουμε πει στο shader ότι θα χρησιμοποιήσουμε ένα πίνακα τιμών.

```
gl.bindBuffer(gl.ARRAY_BUFFER, myBuffer);/ δέσμευση buffer
gl.vertexAttribPointer(vertexPositionAttribute, 3, gl.FLOAT, false, 0,
0);
```

Τελικό στάδιο είναι η ερμηνεία των δεδομένων από το shader. Η παράμετρος 3(size) μας δείχνει τον αριθμό των components. Για color RGB-3 και RGBA-4 (alpha value), για position (x,y,z)-3 και (x,y,z,w)-4, για texture (s,t)-2 και η παράμετρος gl.FLOAT τον τύπο

δεδομένων.

Η διαδικασία για την εκχώρηση των τιμών σε έναν shader ενός attribute έχει ως εξής:

```
vertexPositionAttribute=gl.getAttribLocation(glProgram,"aVertexPosition");
gl.enableVertexAttribArray(vertexPositionAttribute);
gl.bindBuffer(gl.ARRAY_BUFFER, myBuffer);
gl.vertexAttribPointer(vertexPositionAttribute, 3, gl.FLOAT, false, 0, 0);
```

1.6 GL Shading Language/GLSL – Shaders

Η OpenGL Shading Language είναι μια γλώσσα προγραμματισμού υψηλού επιπέδου βασισμένη συντακτικά στη γλώσσα προγραμματισμού C. Δημιουργήθηκε για να δώσει στους προγραμματιστές άμεσο έλεγχο της διαδικασίας graphic pipeline χωρίς την χρήση της assembly ή κάποιας άλλης hardware γλώσσας.

Με τον όρο graphic pipeline εννοούμε τα βήματα που χρειάζονται έτσι ώστε μια εικόνα να μεταβεί από το στάδιο του αρχικού ορισμού της στο τελικό στάδιο που είναι η εμφάνιση στην οθόνη. Η διαδικασία pipeline είναι συγκεκριμένη και αποτελείται από διάφορα βήματα προκαθορισμένης σειράς. Μια pipeline διαδικασία μπορεί να διαθέτει fixed functionality (προκαθορισμένη λειτουργικότητα) ή programmable (προγραμματιζόμενη).

Όταν μιλάμε για fixed functionality μιλάμε για fixed εφαρμογή με συγκεκριμένες λειτουργίες οι οποίες είναι με καθορισμένη σειρά και που σε καμιά περίπτωση δεν μπορούμε να τις τροποποιήσουμε, απλώς μόνο να τις ενεργοποιήσουμε ή να τις απενεργοποιήσουμε πχ. φωτισμός, υφή. Μας επιτρέπει την εύκολη παραγωγή εικόνων διότι υπάρχουν ήδη στο σύστημα οι φόρμες φωτισμού και το shading πρόγραμμα. Μειονέκτημα αποτελεί ο περιορισμός αυτών που μπορούμε να δημιουργήσουμε αφού σε καμία περίπτωση δεν έχουμε την δυνατότητα να τροποποιήσουμε τις προκαθορισμένες ρυθμίσεις.

Από την άλλη μια programmable pipeline μας δίνει την δυνατότητα χρήσης μιας ευρύτερης γκάμας εφέ. Αυτό γιατί εμείς καθορίζουμε κομμάτια της διαδικασίας pipeline, συν του ότι έχουμε την δυνατότητα επανεγγραφής των δεδομένων που αφορούν τη θέση, το χρώμα, την υφή, μοντέλα φωτισμού κα. Αυτό γίνεται μέσω των shader προγραμμάτων. Shader προγράμματα ή αλλιώς shaders είναι προγράμματα υπολογιστή που χρησιμοποιούνται για τη

δημιουργία σκιών δηλ. την παραγωγή των κατάλληλων επιπέδων χρώματος σε μια εικόνα ή πιο ρεαλιστικά τη δημιουργία ειδικών εφέ. Τα κύρια programmable pipeline components είναι το vertex πρόγραμμα (vertex shader) και το fragment πρόγραμμα (fragment shader). Τα προγράμματα αυτά "τρέχουν" (run) στην κάρτα γραφικών (GPU).

Προγραμματιστικά ο vertex shader είναι μια συνάρτηση επεξεργασίας γραφικών που χρησιμοποιείται για να προσθέσει ειδικά εφέ σε ένα object σε ένα 3D περιβάλλον εκτελώντας μαθηματικές πράξεις στα δεδομένα των κορυφών του object (vertex data object). Κάθε κορυφή ορίζεται από πολλές μεταβλητές και αυτό γιατί η απεικόνιση σε 3D περιβάλλον απαιτεί την χρήση x, y και z συντεταγμένων. Κάθε κορυφή μπορεί επίσης να οριστεί από τις συντεταγμένες χρώματος, υφής και φωτισμού. Οι vertex shader αλλάζουν τις τιμές των δεδομένων έτσι ώστε κάθε κορυφή να εμφανίζεται με διαφορετικό χρώμα, υφή ή και διαφορετική θέση στο χώρο.

Ο fragment shader (FS) είναι ένα πρόγραμμα που όταν εκτελείται επεξεργάζεται τα δεδομένα από το στάδιο rasterized και το στάδιο της γραμμικής παρεμβολής (lineral interpolation) μας δίνει ως έξοδο την τιμή για το βάθος χρώματος (z-value).

Ο συνδυασμός και των δύο προγραμμάτων ενημερώνει το drawing, δηλ. το update του frame. Η WebGL χρησιμοποιεί την γλώσσα JavaScript για να συνδέσει τους shaders με το GLSL API. Η χρήση των shader γενικότερα μας δίνει τη δυνατότητα δημιουργίας εφέ με αρκετά αυξημένο βαθμό ρεαλισμού.

1.6.1 WebGL Graphics pipeline

Η διαδικασία ξεκινάει με την τοποθέτηση του πίνακα δεδομένων των κορυφών στους VBO's (buffers). Συνεχίζει με την ροή των δεδομένων από τους VBO's στο vertex shader πρόγραμμα (VS). Ο VS στέλνει δείκτες-πληροφοριών χρησιμοποιώντας την κλήση drawArrays ή drawElements. Το πρόγραμμα του vertex shader εκτελείται και καθορίζει την θέση κάθε κορυφής στην οθόνη. Προαιρετικά μπορεί να προσθέσει υπολογισμούς που μεταφέρονται στον fragment shader. Η έξοδος του VS (varying variables) μεταφέρεται στο επόμενο στάδιο της pipeline διαδικασίας κατά το οποίο η GPU παράγει primitive τύπους χρησιμοποιώντας τα δεδομένα των κορυφών και τους δείκτες.

Ακολουθεί η rasterized της primitive εικόνας (η εικόνα που απεικονίζεται με primitive τύπους – points, lines, triangle -). Για κάθε primitive τύπο δημιουργείται ένα “θραύσμα”/fragment. Με τον όρο rasterized δηλαδή εννοούμε τη διαδικασία μετατροπής μιας διανυσματικής εικόνας (γεωμετρικό σχήμα) σε μια εικόνα που αποτελείται από pixel ή θραύσματα (fragment) για την τελική έξοδό της στην οθόνη. Από το στάδιο αυτό απορρίπτεται κάθε primitive τύπος που βρίσκεται εκτός του viewport. Έπεται η γραμμική παρεμβολή (lineral interpolation) για κάθε θραύσμα (fragment). Με τον όρο αυτό εννοούμε την μετατροπή των σημείων των θραυσμάτων σε τιμές μιας χωρικής διάστασης (πολυώνυμο, x-y-z values).

Εν συνεχεία τα θραύσματα (fragment) μαζί με τις interpolation τιμές περνάνε στο fragment shader. Ο FS θέτει το χρώμα, την υφή ή ακόμη και λειτουργίες φωτισμού. Τα θραύσματα έχουν ως τελικό προορισμό την μετακύλησή τους στον frame buffer, ο οποίος αποθηκεύει την 2D εικόνα και προαιρετικά χρησιμοποιεί του depth και stencil buffers. Μέσω της συνάρτησης drawScene επιτυγχάνεται ο σχεδιασμός σε ένα canvas με σκοπό την απεικόνιση δεδομένων στην οθόνη. (drawing)

1.6.2 Παράδειγμα WebGL με την χρήση των shader προγραμμάτων - Ανάλυση

Ο κώδικας που παρουσιάζεται αναλυτικά στο 1.6.3 εμφανίζει δύο τρίγωνα που έχουν μια κοινή κορυφή, είναι χρωματισμένα και προβάλλονται στο κέντρο ενός canvas χρώματος πράσινου στο σώμα ενός .html εγγράφου. Ο κώδικας περιλαμβάνει JavaScript σενάρια για τη δημιουργία των shaders και JavaScript σενάρια που περιλαμβάνουν την υλοποίηση συναρτήσεων με σκοπό την απεικόνιση του object με χρώμα στην οθόνη του υπολογιστή. Συνοπτικά αυτά είναι:

<script id="shader-vs" type="x-shader/x-vertex">...</script>// δημιουργία vertex shader.

<script id="shader-fs" type="x-shader/x-fragment">...</script>// δημιουργία fragment shader.

<script>

function initWebGL()// αρχικοποίηση και απόκτηση context.

function setupWebGL()// καθορίζει το χρώμα του canvas και την προβολή του object.

function initShaders()// αρχικοποίηση των shader. Περιλαμβάνει την πηγή του κώδικα (shader source), τον μεταγλωττιστή των shaders (compile shaders), τη δημιουργία προγράμματος (create program), τη σύνδεση των shaders με το πρόγραμμα (attach and link shaders to the program) και τέλος τη χρήση του προγράμματος (use program).
function makeShader(src, type)// μεταγλώττιση του vertex shader (compile the vertex shader).
function setupBuffers()// δημιουργία buffer και ορισμός τιμών που αντικατοπτρίζουν ένα object με τη μορφή χαρακτηριστικών (attributes και uniforms variables).
function drawScene()// άντληση των δεδομένων για το σχεδιασμό που αφορούν το object.
</script>

1.6.3 Index.html κώδικας WebGL παραδείγματος με την χρήση των shader προγραμμάτων

Ακολουθεί ο index.html κώδικας που εμφανίζει δύο τρίγωνα που έχουν μια κοινή κορυφή, είναι χρωματισμένα και προβάλλονται στο κέντρο ενός canvas χρώματος πράσινου στο σώμα ενός .html εγγράφου.

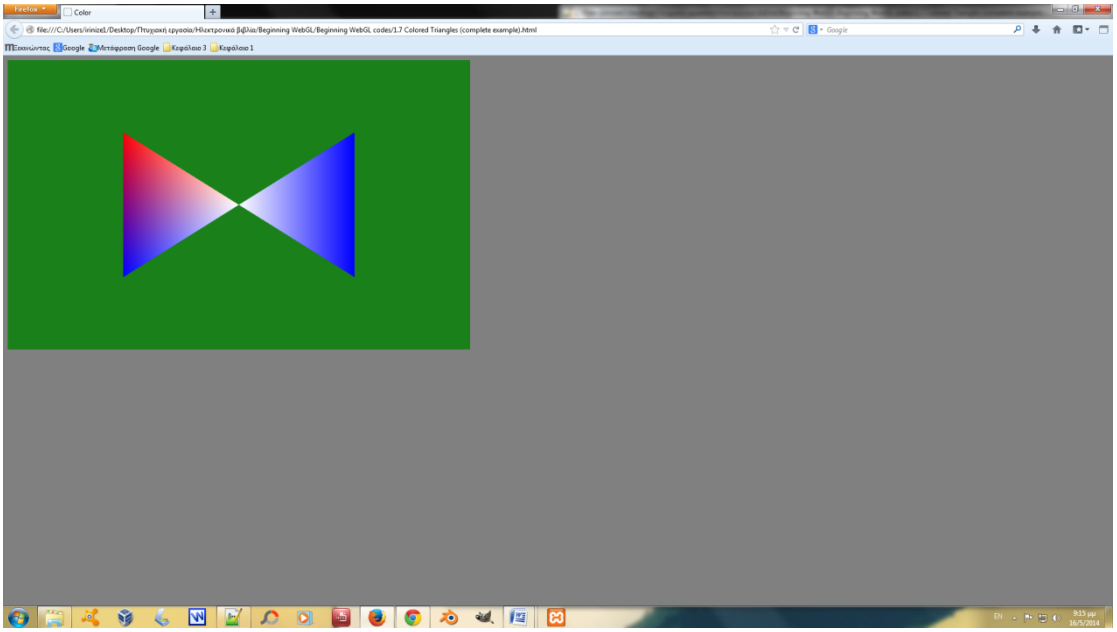
```
<!doctype html>
<html>
	<head>
		<title>Color</title>
		<style>
			body{ background-color: grey; }
			canvas{ background-color: white; }
		</style>

<script id="shader-vs" type="x-shader/x-vertex">
	attribute vec3 aVertexPosition;
	attribute vec3 aVertexColor;

	varying highp vec4 vColor;

	void main(void) {
gl_Position = vec4(aVertexPosition, 1.0);
```

```
                    vColor = vec4(aVertexColor, 1.0);
}
</script>

<script id="shader-fs" type="x-shader/x-fragment">
        varying highp vec4 vColor;

        void main(void) {
gl_FragColor = vColor;
                    }
</script>

<script>
        var gl = null,
        canvas = null,
        glProgram = null,
        fragmentShader = null,
        vertexShader = null;

        var vertexPositionAttribute = null,
        trianglesVerticeBuffer = null,
        vertexColorAttribute = null,
        trianglesColorBuffer = null;

                function initWebGL()
                        {
canvas = document.getElementById("my-canvas");
                        try
{
                        gl          =          canvas.getContext("webgl")          ||
canvas.getContext("experimental-webgl");
                                }catch(e){
                                }
                                if(gl)
                                {
                                        setupWebGL();
                                        initShaders();
                                        setupBuffers();
                                        drawScene();
                                }else{
                                        alert(   "Error: Your browser does not
appear to support WebGL.");
```

```
                    }
                }

            function setupWebGL()
                {
                //set the clear color to a shade of green
                gl.clearColor(0.1, 0.5, 0.1, 1.0);
                gl.clear(gl.COLOR_BUFFER_BIT);

                gl.viewport(0, 0, canvas.width, canvas.height);
                }

            function initShaders()
                {
//get shader source
var fs_source = document.getElementById('shader-fs').innerHTML,
vs_source = document.getElementById('shader-vs').innerHTML;

//compile shaders
vertexShader = makeShader(vs_source, gl.VERTEX_SHADER);
fragmentShader = makeShader(fs_source, gl.FRAGMENT_SHADER);

//create program
glProgram = gl.createProgram();

//attach and link shaders to the program
gl.attachShader(glProgram, vertexShader);
gl.attachShader(glProgram, fragmentShader);
gl.linkProgram(glProgram);

if (!gl.getProgramParameter(glProgram, gl.LINK_STATUS)) {
        alert("Unable to initialize the shader program.");
    }

//use program
gl.useProgram(glProgram);
                }

            function makeShader(src, type)
                {

//compile the vertex shader
```

```
var shader = gl.createShader(type);
gl.shaderSource(shader, src);
gl.compileShader(shader);

if (!gl.getShaderParameter(shader, gl.COMPILE_STATUS))
{
alert("Error compiling shader: " + gl.getShaderInfoLog(shader));
}
return shader;
}
function setupBuffers()
                        {
                        var triangleVertices = [
                                //left triangle
                                -0.5, 0.5, 1.0,
                                0.0, 0.0, 0.0,
                                -0.5, -0.5, 0.0,

                                //right triangle
                                0.5, 0.5, 0.0,
                                0.0, 0.0, 0.0,
                                0.5, -0.5, 0.0,
                                ];

                        trianglesVerticeBuffer = gl.createBuffer();
gl.bindBuffer(gl.ARRAY_BUFFER, trianglesVerticeBuffer);
gl.bufferData(gl.ARRAY_BUFFER, new Float32Array(triangleVertices),
gl.STATIC_DRAW);

                        var triangleVerticeColors = [
                                //left triangle
                                1.0, 0.0, 0.0,
                                1.0, 1.0, 1.0,
                                0.0, 0.0, 1.0,

                                //right triangle
                                0.0, 0.0, 1.0,
                                1.0, 1.0, 1.0,
                                0.0, 0.0, 1.0,
                                ];

trianglesColorBuffer = gl.createBuffer();
```

```
gl.bindBuffer(gl.ARRAY_BUFFER, trianglesColorBuffer);
gl.bufferData(gl.ARRAY_BUFFER, new Float32Array(triangleVerticeColors), gl.STATIC_DRAW);
                }

                function drawScene()
                {
vertexPositionAttribute = gl.getAttribLocation(glProgram, "aVertexPosition");
gl.enableVertexAttribArray(vertexPositionAttribute);
gl.bindBuffer(gl.ARRAY_BUFFER, trianglesVerticeBuffer);
gl.vertexAttribPointer(vertexPositionAttribute, 3, gl.FLOAT, false, 0, 0);

vertexColorAttribute = gl.getAttribLocation(glProgram, "aVertexColor");
gl.enableVertexAttribArray(vertexColorAttribute);
gl.bindBuffer(gl.ARRAY_BUFFER, trianglesColorBuffer);
gl.vertexAttribPointer(vertexColorAttribute, 3, gl.FLOAT, false, 0, 0);

gl.drawArrays(gl.TRIANGLES, 0, 6);
                }
</script>
        </head>
        <body onload="initWebGL()">
                <canvas id="my-canvas" width="800" height="500">
                Your browser does not support the HTML5 canvas element.
                </canvas>
        </body>
</html>
```

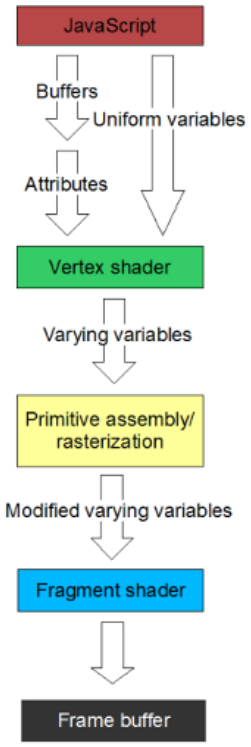


Εικόνα 3: απεικόνιση δύο τρίγωνων που έχουν μια κοινή κορυφή, είναι χρωματισμένα και προβάλλονται στο κέντρο ενός canvas χρώματος πράσινου στο σώμα ενός .html εγγράφου (screenshot)

1.7 Λειτουργία

Με βάση το παρακάτω διάγραμμα μπορούμε εν συντομία να αντιληφθούμε πως τα δεδομένα που βασίζονται σε JavaScript σενάρια, μέσω της συνάρτησης drawScene() μετατρέπονται σε pixel και εμφανίζονται σε ένα WebGL canvas στο σώμα της σελίδας μας.

Σχηματική περιγραφή



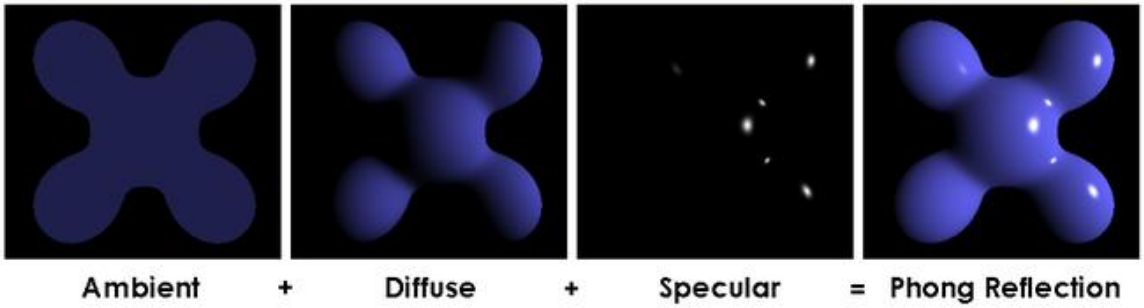
Εικόνα 4: σχήμα που απεικονίζει την διαδικασία μεταφοράς JavaScript δεδομένων και την μετατροπή τους σε pixel μέσω της συνάρτησης drawScene()

Κάθε φορά που καλούμε την συνάρτηση drawScene() οι διεργασίες της WebGL αντλούν τα δεδομένα που έχουμε ήδη καταχωρίσει με τη μορφή χαρακτηριστικών στους VBO's buffers (attributes και uniforms variables) και τα μεταφέρουν στον vertex shader. Ο vertex shader τοποθετεί τα δεδομένα που έχουν αντληθεί σε διάφορες μεταβλητές. Υποχρεωτική είναι η μεταβλητή gl_Position η οποία περιέχει τις συντεταγμένες των κορυφών του αντικειμένου. Αφού δημιουργήσουμε τον vertex shader η WebGL μετατρέπει την 3D εικόνα σε 2D με την χρήση των μεταβλητών αυτών.

Ακολουθεί η rasterized διαδικασία και η interpolation διαδικασία. Εν συνεχεία καλείται ο fragment shader για κάθε σημείο (pixel) της εικόνας. Σκοπός του fragment shader είναι να επιστρέψει το χρώμα αυτών των σημείων μέσω της μεταβλητής gl_FragColor. Όλα τα δεδομένα αποθηκεύονται στο frame buffer, δηλαδή η ολοκληρωμένη μας εικόνα είναι αποθηκευμένη σε μια προσωρινή μνήμη πλαισίου.

Στον fragment shader έχουμε τη δυνατότητα να προσθέσουμε

και ρυθμίσεις για το Phong Reflection model. Πρόκειται για ρυθμίσεις τοπικού φωτισμού σημείων πάνω σε μια επιφάνεια περιγράφοντας έτσι τον τρόπο με τον οποίο αντανακλάται το φως. Διάχυτη αντανάκλαση για τραχιές επιφάνειες και κατοπτρική αντανάκλαση για γυαλιστερές επιφάνειες. Λαμβάνεται πάντα υπόψη το φως περιβάλλοντος το οποίο υπάρχει πάντα στο στοιχείο <canvas>. Θα το δούμε πιο αναλυτικά στο επόμενο παράδειγμα. Οι Phong Reflection ρυθμίσεις προσδίδουν στοιχεία ρεαλισμού στο αντικείμενο.



Εικόνα 5: Phong Reflection model (ρυθμίσεις τοπικού φωτισμού σημείων)

Καθορίζω τους fragment shader και vertex shader χρησιμοποιώντας την tag <script>. Οι uniform μεταβλητές χρησιμοποιούνται για να αποθηκεύουν τα σημεία του φωτός (Phong Reflection model).

1.8 Βιβλιοθήκες λογισμικού & JavaScript βιβλιοθήκες

Με τον όρο βιβλιοθήκη λογισμικού εννοούμε την σουίτα εκείνη των δεδομένων και τον κώδικα προγραμματισμού, τα οποία μπορούμε να τα χρησιμοποιήσουμε για την ανάπτυξη προγραμμάτων λογισμικού και εφαρμογών. Ο σχεδιασμός τους είναι τέτοιος έτσι ώστε να είναι κατανοητή και από την πλευρά του προγραμματιστή και την πλευρά του compiler. Αποτελείται από κώδικα, τάξεις, διαδικασίες, σενάρια, δεδομένα διαμόρφωσης, πληροφορίες άδειας για το λογισμικό κα. Σκοπός των βιβλιοθηκών λογισμικού είναι η επίτευξη μεγαλύτερης λειτουργικότητας ή αυτοματοποίησης μιας διαδικασίας.

Με την χρήση της WebGL αυξάνονται οι απαιτήσεις του

συστήματος. Για ικανοποιήσουμε τις απαιτήσεις αυτές πρέπει να συνδέσουμε JavaScript βιβλιοθήκες ώστε να μπορούμε να χειριστούμε εξειδικευμένες λειτουργίες. Πρόκειται ουσιαστικά για ένα αποθετήριο λογισμικού που βρίσκεται σε μια ασφαλή "τοποθεσία" σε ένα εξυπηρετητή web και η απόκτησή της μας δίνει την δυνατότητα δημιουργίας εφαρμογών.

Στο παράδειγμα που θα ακολουθήσει οι βιβλιοθήκες που θα χρησιμοποιηθούν είναι οι glMatrix-0.9.5.min.js και webgl-utils.js. Η webgl-utils.js βιβλιοθήκη δίνει στον browser τη δυνατότητα να λειτουργεί με έναν πιο ανεξάρτητο τρόπο κάθε φορά που θα ενημερώνει (update) την απεικόνιση του αντικειμένου. Η βιβλιοθήκη αυτή είναι χρήσιμη για την function tick(). Η βιβλιοθήκη glMatrix-0.9.5.min.js είναι χρήσιμη για τις function που υποστηρίζουν την λειτουργία animate.

Σε επίπεδο κώδικα η σύνδεση βιβλιοθηκών γίνεται ως εξής:
<script type="text/javascript" src="glMatrix-0.9.5.min.js"></script>
<script type="text/javascript" src="webgl-utils.js"></script>

1.8.1 Three.js βιβλιοθήκη

Η Three.js είναι μια JavaScript/API βιβλιοθήκη με την οποία μπορούμε να δημιουργήσουμε και να εμφανίσουμε κινούμενα 3D γραφικά σε έναν Web browser (Firefox, Chrome etc.). Δημιουργήθηκε από τον Ricardo Cabello και βρίσκεται αναρτημένη στο αποθετήριο GitHub από το 2010. Η three.js βιβλιοθήκη μας παρέχει διάφορους τρόπους εμφάνισης του αντικειμένου και πρόκειται για μια καλά σχεδιασμένη βιβλιοθήκη, αρκετά διαισθητική στη χρήση της. Έχει την δυνατότητα αλλαγής των προκαθορισμένων ρυθμίσεων έτσι ώστε να μπορούν να ξαναγραφτούν ως νέες παράμετροι στο αντικείμενο. Η χρήση της προϋποθέτει την συμπερίληψη κάποιων scripts στον index κώδικά μας.

Παράδειγμα

Όταν το script βρίσκεται σε διαφορετικό φάκελο από το index
<script src="φάκελος/όνομα αρχείου.js" </script>
<script src="js/three.js" </script> index

Όταν το script βρίσκεται στον ίδιο φάκελο με το index

<script src=”όνομα αρχείου.js” </script>

1.9 Json

Json (Java Script Object Notation) είναι μια μορφή μορφοποίησης για την ανταλλαγή δεδομένων. Προήλθε από τη γλώσσα JavaScript και αποτελεί υποσύνολό της. Χρησιμοποιείται κατά κύριο λόγο για την μετάδοση των δεδομένων μεταξύ ενός server και μιας web εφαρμογής και χρησιμοποιεί τη βιβλιοθήκη JSONparser για την ανάλυση των δεδομένων αυτών. Το Json είναι ικανό να αναπαραστήσει strings, numbers, booleans και null καθώς objects.

Ένα object σε μορφή Json αποτελείται από δύο μέρη που λειτουργούν όμως ως ζεύγος και αυτά είναι το χαρακτηριστικό (attribute) και οι τιμές (values) και είναι της μορφής {“χαρακτηριστικό”: [τιμή]}. Για παράδειγμα

```
{
    “vertexPositions”: [2.745, -2.39, 2.785, … ],
}
```

Πρόκειται για ένα πρότυπο ικανό να αναγνωστεί από τον άνθρωπο. Η μορφή JSON καθορίστηκε αρχικά από τον Douglas Crockford. Προέρχεται από το ECMAScript Programming Language Standard.

1.9.1 Index.html κώδικας παραδείγματος WebGL με τη χρήση Json

Το παρακάτω παράδειγμα περιλαμβάνει τον κώδικα που εμφανίζει μια περιστρεφόμενη τσαγιέρα με την δυνατότητα αλλαγής των ρυθμίσεων του φωτισμού καθώς και την αλλαγή υφής. Η εμφάνιση του αντικειμένου γίνεται με τη χρήση Json τιμών.

Ακολουθεί ο index.html κώδικας.

```
<html>

<head>
<title>Learning WebGL</title>
<meta http-equiv="content-type" content="text/html; charset=ISO-8859-1">
```

```
<script type="text/javascript" src="glMatrix-0.9.5.min.js"></script>
<script type="text/javascript" src="webgl-utils.js"></script>

<script id="per-fragment-lighting-fs" type="x-shader/x-fragment">
  precision mediump float;

  varying vec2 vTextureCoord;
  varying vec3 vTransformedNormal;
  varying vec4 vPosition;

  uniform float uMaterialShininess;

  uniform bool uShowSpecularHighlights;
  uniform bool uUseLighting;
  uniform bool uUseTextures;
  uniform vec3 uAmbientColor;

  uniform vec3 uPointLightingLocation;
  uniform vec3 uPointLightingSpecularColor;
  uniform vec3 uPointLightingDiffuseColor;

  uniform sampler2D uSampler;

  void main(void) {
    vec3 lightWeighting;
    if (!uUseLighting) {
      lightWeighting = vec3(1.0, 1.0, 1.0);
    } else {
      vec3 lightDirection = normalize(uPointLightingLocation -
vPosition.xyz);
      vec3 normal = normalize(vTransformedNormal);

      float specularLightWeighting = 0.0;
      if (uShowSpecularHighlights) {
        vec3 eyeDirection = normalize(-vPosition.xyz);
        vec3 reflectionDirection = reflect(-lightDirection, normal);

        specularLightWeighting = pow(max(dot(reflectionDirection,
eyeDirection), 0.0), uMaterialShininess);
      }
```

```
        float diffuseLightWeighting = max(dot(normal, lightDirection),
0.0);
        lightWeighting = uAmbientColor
          + uPointLightingSpecularColor * specularLightWeighting
          + uPointLightingDiffuseColor * diffuseLightWeighting;
      }

      vec4 fragmentColor;
      if (uUseTextures) {
        fragmentColor = texture2D(uSampler, vec2(vTextureCoord.s,
vTextureCoord.t));
      } else {
        fragmentColor = vec4(1.0, 1.0, 1.0, 1.0);
      }
      gl_FragColor = vec4(fragmentColor.rgb * lightWeighting,
fragmentColor.a);
    }
</script>

<script id="per-fragment-lighting-vs" type="x-shader/x-vertex">
    attribute vec3 aVertexPosition;
    attribute vec3 aVertexNormal;
    attribute vec2 aTextureCoord;

    uniform mat4 uMVMatrix;
    uniform mat4 uPMatrix;
    uniform mat3 uNMatrix;
    varying vec2 vTextureCoord;
    varying vec3 vTransformedNormal;
    varying vec4 vPosition;
    void main(void) {
      vPosition = uMVMatrix * vec4(aVertexPosition, 1.0);
      gl_Position = uPMatrix * vPosition;
      vTextureCoord = aTextureCoord;
      vTransformedNormal = uNMatrix * aVertexNormal;
    }
</script>

<script type="text/javascript">

    var gl;
    function initGL(canvas) {
```

```
    try {
        gl = canvas.getContext("experimental-webgl");
        gl.viewportWidth = canvas.width;
        gl.viewportHeight = canvas.height;
    } catch (e) {
    }
    if (!gl) {
        alert("Could not initialise WebGL, sorry :-(");
    }
}

function getShader(gl, id) {
    var shaderScript = document.getElementById(id);
    if (!shaderScript) {
        return null;
    }

    var str = "";
    var k = shaderScript.firstChild;
    while (k) {
        if (k.nodeType == 3) {
            str += k.textContent;
        }
        k = k.nextSibling;
    }

    var shader;
    if (shaderScript.type == "x-shader/x-fragment") {
        shader = gl.createShader(gl.FRAGMENT_SHADER);
    } else if (shaderScript.type == "x-shader/x-vertex") {
        shader = gl.createShader(gl.VERTEX_SHADER);
    } else {
        return null;
    }

    gl.shaderSource(shader, str);
    gl.compileShader(shader);

    if (!gl.getShaderParameter(shader, gl.COMPILE_STATUS)) {
        alert(gl.getShaderInfoLog(shader));
        return null;
    }
```

```
        return shader;
    }
var shaderProgram;
    function initShaders() {
        var fragmentShader = getShader(gl, "per-fragment-lighting-fs");
        var vertexShader = getShader(gl, "per-fragment-lighting-vs");

        shaderProgram = gl.createProgram();
        gl.attachShader(shaderProgram, vertexShader);
        gl.attachShader(shaderProgram, fragmentShader);
        gl.linkProgram(shaderProgram);
        if (!gl.getProgramParameter(shaderProgram, gl.LINK_STATUS)) {
            alert("Could not initialise shaders");
        }

        gl.useProgram(shaderProgram);

        shaderProgram.vertexPositionAttribute =
gl.getAttribLocation(shaderProgram, "aVertexPosition");

gl.enableVertexAttribArray(shaderProgram.vertexPositionAttribute);

        shaderProgram.vertexNormalAttribute                    =
gl.getAttribLocation(shaderProgram, "aVertexNormal");
gl.enableVertexAttribArray(shaderProgram.vertexNormalAttribute);

        shaderProgram.textureCoordAttribute                    =
gl.getAttribLocation(shaderProgram,          "aTextureCoord");
gl.enableVertexAttribArray(shaderProgram.textureCoordAttribute);

        shaderProgram.pMatrixUniform                           =
gl.getUniformLocation(shaderProgram, "uPMatrix");
        shaderProgram.mvMatrixUniform                          =
gl.getUniformLocation(shaderProgram, "uMVMatrix");
        shaderProgram.nMatrixUniform                           =
gl.getUniformLocation(shaderProgram, "uNMatrix");
        shaderProgram.samplerUniform                           =
gl.getUniformLocation(shaderProgram, "uSampler");
        shaderProgram.materialShininessUniform                 =
gl.getUniformLocation(shaderProgram, "uMaterialShininess");
        shaderProgram.showSpecularHighlightsUniform            =
```

```
gl.getUniformLocation(shaderProgram, "uShowSpecularHighlights");
    shaderProgram.useTexturesUniform =
gl.getUniformLocation(shaderProgram, "uUseTextures");
    shaderProgram.useLightingUniform =
gl.getUniformLocation(shaderProgram, "uUseLighting");
    shaderProgram.ambientColorUniform =
gl.getUniformLocation(shaderProgram, "uAmbientColor");
    shaderProgram.pointLightingLocationUniform =
gl.getUniformLocation(shaderProgram, "uPointLightingLocation");
    shaderProgram.pointLightingSpecularColorUniform =
gl.getUniformLocation(shaderProgram,
"uPointLightingSpecularColor");
    shaderProgram.pointLightingDiffuseColorUniform =
gl.getUniformLocation(shaderProgram,
"uPointLightingDiffuseColor");
  }
  function handleLoadedTexture(texture) {
    gl.pixelStorei(gl.UNPACK_FLIP_Y_WEBGL, true);
    gl.bindTexture(gl.TEXTURE_2D, texture);
    gl.texImage2D(gl.TEXTURE_2D, 0, gl.RGBA, gl.RGBA,
gl.UNSIGNED_BYTE, texture.image);
    gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_MAG_FILTER,
gl.LINEAR);
    gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_MIN_FILTER,
gl.LINEAR_MIPMAP_NEAREST);
    gl.generateMipmap(gl.TEXTURE_2D);
    gl.bindTexture(gl.TEXTURE_2D, null);
  }
  var earthTexture;
  var galvanizedTexture;
  function initTextures() {
    earthTexture = gl.createTexture();
    earthTexture.image = new Image();
    earthTexture.image.onload = function () {
      handleLoadedTexture(earthTexture)
    }
    earthTexture.image.src = "earth.jpg";

    galvanizedTexture = gl.createTexture();
    galvanizedTexture.image = new Image();
    galvanizedTexture.image.onload = function () {
      handleLoadedTexture(galvanizedTexture)
```

```
    }
    galvanizedTexture.image.src                                    =
"arroway.de_metal+structure+06_d100_flat.jpg";
  }

  var mvMatrix = mat4.create();
  var mvMatrixStack = [];
  var pMatrix = mat4.create();
  function mvPushMatrix() {
    var copy = mat4.create();
    mat4.set(mvMatrix, copy);
    mvMatrixStack.push(copy);
  }

  function mvPopMatrix() {
    if (mvMatrixStack.length == 0) {
      throw "Invalid popMatrix!";
    }
    mvMatrix = mvMatrixStack.pop();
  }

  function setMatrixUniforms() {
    gl.uniformMatrix4fv(shaderProgram.pMatrixUniform,          false,
pMatrix);
    gl.uniformMatrix4fv(shaderProgram.mvMatrixUniform,         false,
mvMatrix);
    var normalMatrix = mat3.create();
    mat4.toInverseMat3(mvMatrix, normalMatrix);
    mat3.transpose(normalMatrix);
    gl.uniformMatrix3fv(shaderProgram.nMatrixUniform,          false,
normalMatrix);
  }

  function degToRad(degrees) {
    return degrees * Math.PI / 180;
  }

  var teapotVertexPositionBuffer;
  var teapotVertexNormalBuffer;
  var teapotVertexTextureCoordBuffer;
  var teapotVertexIndexBuffer;
 function handleLoadedTeapot(teapotData) {
```

```
      teapotVertexNormalBuffer = gl.createBuffer();
      gl.bindBuffer(gl.ARRAY_BUFFER, teapotVertexNormalBuffer);
      gl.bufferData(gl.ARRAY_BUFFER, new
Float32Array(teapotData.vertexNormals), gl.STATIC_DRAW);
      teapotVertexNormalBuffer.itemSize = 3;
      teapotVertexNormalBuffer.numItems =
teapotData.vertexNormals.length / 3;
      teapotVertexTextureCoordBuffer = gl.createBuffer();
      gl.bindBuffer(gl.ARRAY_BUFFER,
teapotVertexTextureCoordBuffer);
      gl.bufferData(gl.ARRAY_BUFFER, new
Float32Array(teapotData.vertexTextureCoords), gl.STATIC_DRAW);
      teapotVertexTextureCoordBuffer.itemSize = 2;
      teapotVertexTextureCoordBuffer.numItems =
teapotData.vertexTextureCoords.length / 2;

      teapotVertexPositionBuffer = gl.createBuffer();
      gl.bindBuffer(gl.ARRAY_BUFFER, teapotVertexPositionBuffer);
      gl.bufferData(gl.ARRAY_BUFFER, new
Float32Array(teapotData.vertexPositions), gl.STATIC_DRAW);
      teapotVertexPositionBuffer.itemSize = 3;
      teapotVertexPositionBuffer.numItems =
teapotData.vertexPositions.length / 3;

      teapotVertexIndexBuffer = gl.createBuffer();
      gl.bindBuffer(gl.ELEMENT_ARRAY_BUFFER,
teapotVertexIndexBuffer);
      gl.bufferData(gl.ELEMENT_ARRAY_BUFFER, new
Uint16Array(teapotData.indices), gl.STATIC_DRAW);
      teapotVertexIndexBuffer.itemSize = 1;
      teapotVertexIndexBuffer.numItems = teapotData.indices.length;

      document.getElementById("loadingtext").textContent = "";
    }

    function loadTeapot() {
      var request = new XMLHttpRequest();
      request.open("GET", "Teapot.json");
      request.onreadystatechange = function () {
        if (request.readyState == 4) {
          handleLoadedTeapot(JSON.parse(request.responseText));
        }
```

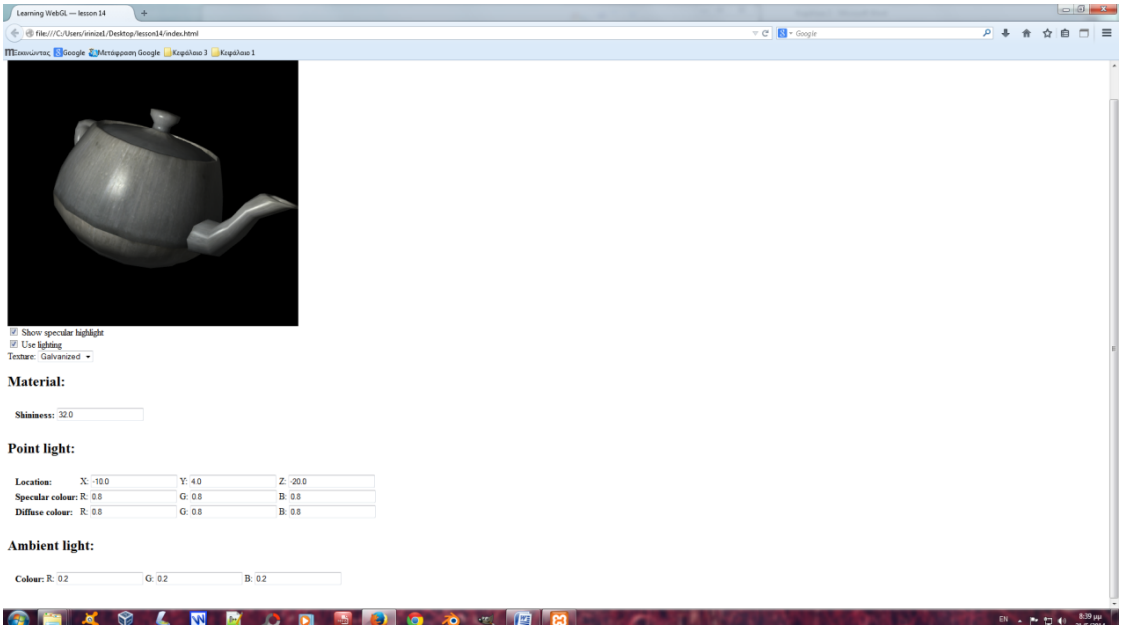
```
    }
    request.send();
  }
  var teapotAngle = 180;
 function drawScene() {
    gl.viewport(0, 0, gl.viewportWidth, gl.viewportHeight);
    gl.clear(gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);

    if (teapotVertexPositionBuffer == null || teapotVertexNormalBuffer == null || teapotVertexTextureCoordBuffer == null || teapotVertexIndexBuffer == null) {
      return;
    }

    mat4.perspective(45, gl.viewportWidth / gl.viewportHeight, 0.1, 100.0, pMatrix);

    var specularHighlights = document.getElementById("specular").checked;
    gl.uniform1i(shaderProgram.showSpecularHighlightsUniform, specularHighlights);

    var lighting = document.getElementById("lighting").checked;
    gl.uniform1i(shaderProgram.useLightingUniform, lighting);
    if (lighting) {
      gl.uniform3f(
        shaderProgram.ambientColorUniform,
        parseFloat(document.getElementById("ambientR").value),
        parseFloat(document.getElementById("ambientG").value),
        parseFloat(document.getElementById("ambientB").value)
      );

      gl.uniform3f(
        shaderProgram.pointLightingLocationUniform,

parseFloat(document.getElementById("lightPositionX").value),

parseFloat(document.getElementById("lightPositionY").value),

parseFloat(document.getElementById("lightPositionZ").value)
      );
```

```
        gl.uniform3f(
          shaderProgram.pointLightingSpecularColorUniform,
          parseFloat(document.getElementById("specularR").value),
          parseFloat(document.getElementById("specularG").value),
          parseFloat(document.getElementById("specularB").value)
        );

        gl.uniform3f(
          shaderProgram.pointLightingDiffuseColorUniform,
          parseFloat(document.getElementById("diffuseR").value),
          parseFloat(document.getElementById("diffuseG").value),
          parseFloat(document.getElementById("diffuseB").value)
        );
      }

      var texture = document.getElementById("texture").value;
      gl.uniform1i(shaderProgram.useTexturesUniform, texture !=
"none");

      mat4.identity(mvMatrix);

      mat4.translate(mvMatrix, [0, 0, -40]);
      mat4.rotate(mvMatrix, degToRad(23.4), [1, 0, -1]);
      mat4.rotate(mvMatrix, degToRad(teapotAngle), [0, 1, 0]);
      gl.activeTexture(gl.TEXTURE0);
      if (texture == "earth") {
        gl.bindTexture(gl.TEXTURE_2D, earthTexture);
      } else if (texture == "galvanized") {
        gl.bindTexture(gl.TEXTURE_2D, galvanizedTexture);
      }
      gl.uniform1i(shaderProgram.samplerUniform, 0);

      gl.uniform1f(shaderProgram.materialShininessUniform,
parseFloat(document.getElementById("shininess").value));

      gl.bindBuffer(gl.ARRAY_BUFFER, teapotVertexPositionBuffer);
      gl.vertexAttribPointer(shaderProgram.vertexPositionAttribute,
teapotVertexPositionBuffer.itemSize, gl.FLOAT, false, 0, 0);

      gl.bindBuffer(gl.ARRAY_BUFFER,
teapotVertexTextureCoordBuffer);
      gl.vertexAttribPointer(shaderProgram.textureCoordAttribute,
```

```
teapotVertexTextureCoordBuffer.itemSize, gl.FLOAT, false, 0, 0);

      gl.bindBuffer(gl.ARRAY_BUFFER, teapotVertexNormalBuffer);
      gl.vertexAttribPointer(shaderProgram.vertexNormalAttribute,
teapotVertexNormalBuffer.itemSize, gl.FLOAT, false, 0, 0);
      gl.bindBuffer(gl.ELEMENT_ARRAY_BUFFER,
teapotVertexIndexBuffer);
      setMatrixUniforms();
      gl.drawElements(gl.TRIANGLES,
teapotVertexIndexBuffer.numItems, gl.UNSIGNED_SHORT, 0);
   }

   var lastTime = 0;
   function animate() {
      var timeNow = new Date().getTime();
      if (lastTime != 0) {
         var elapsed = timeNow - lastTime;
         teapotAngle += 0.05 * elapsed;
      }
      lastTime = timeNow;
   }
   function tick() {
      requestAnimFrame(tick);
      drawScene();
      animate();
   }
 function webGLStart() {
      var canvas = document.getElementById("lesson14-canvas");
      initGL(canvas);
      initShaders();
      initTextures();
      loadTeapot();

      gl.clearColor(0.0, 0.0, 0.0, 1.0);
      gl.enable(gl.DEPTH_TEST);

      tick();
   }

</script>
```

```
<style type="text/css">
  #loadingtext {
    position:absolute;
    top:250px;
    left:150px;
    font-size:2em;
    color: white;
  }
</style>

</head>

<body onload="webGLStart();">
  <canvas id="lesson14-canvas" style="border: none;" width="500" height="500"></canvas>

  <div id="loadingtext">Loading world...</div>
  <br/>

  <input type="checkbox" id="specular" checked /> Show specular highlight<br/>
  <input type="checkbox" id="lighting" checked /> Use lighting<br/>

  Texture:
  <select id="texture">
    <option value="none">None</option>
    <option selected value="galvanized">Galvanized</option>
    <option value="earth">Earth</option>
  </select>
  <br/>

  <h2>Material:</h2>

  <table style="border: 0; padding: 10px;">
    <tr>
      <td><b>Shininess:</b>
      <td><input type="text" id="shininess" value="32.0" />
    </tr>
  </table>

  <h2>Point light:</h2>
```

```
  <table style="border: 0; padding: 10px;">
    <tr>
      <td><b>Location:</b>
      <td>X: <input type="text" id="lightPositionX" value="-10.0" />
      <td>Y: <input type="text" id="lightPositionY" value="4.0" />
      <td>Z: <input type="text" id="lightPositionZ" value="-20.0" />
    </tr>
    <tr>
      <td><b>Specular colour:</b>
      <td>R: <input type="text" id="specularR" value="0.8" />
      <td>G: <input type="text" id="specularG" value="0.8" />
      <td>B: <input type="text" id="specularB" value="0.8" />
    </tr>
    <tr>
      <td><b>Diffuse colour:</b>
      <td>R: <input type="text" id="diffuseR" value="0.8" />
      <td>G: <input type="text" id="diffuseG" value="0.8" />
      <td>B: <input type="text" id="diffuseB" value="0.8" />
    </tr>
  </table>

  <h2>Ambient light:</h2>

  <table style="border: 0; padding: 10px;">
    <tr>
      <td><b>Colour:</b>
      <td>R: <input type="text" id="ambientR" value="0.2" />
      <td>G: <input type="text" id="ambientG" value="0.2" />
      <td>B: <input type="text" id="ambientB" value="0.2" />
    </tr>
  </table>
  <br/>

</body>

</html>
```



Εικόνα 6: απεικόνιση περιστρεφόμενης τσαγιέρας με την δυνατότητα αλλαγής των ρυθμίσεων του φωτισμού καθώς και την αλλαγή υφής στο σώμα ενός .html εγγράφου (screenshot)

1.9.2 Index.html κώδικας παραδείγματος WebGL με τη χρήση Json - Ανάλυση

Στο σημείο αυτό θα εξηγήσουμε τις λειτουργίες κάποιων βασικών συναρτήσεων που εμπεριέχονται στον κώδικά μας.

JavaScript βιβλιοθήκες

<script type="text/javascript" src="glMatrix-0.9.5.min.js"></script>// σύνδεση βιβλιοθήκης χρήσιμη για το update του frame [function tick()].
<script type="text/javascript" src="webgl-utils.js"></script>// σύνδεση βιβλιοθήκης χρήσιμης για τις function που υποστηρίζουν την λειτουργία animate. Οι function αυτές είναι οι mvPushMatrix(),mvPopMatrix() και setMatrixUniforms().

JavaScript σενάρια VS και FS

<script id="per-fragment-lighting-fs" type="x-shader/x-fragment">
</script>// δημιουργία fragment-shader προγράμματος. Οι μεταβλητές και οι φόρμες που υλοποιούνται εδώ αφορούν την κατεύθυνση και το χρώμα του φωτός (στοιχεία διάχυσης φωτός: shininess, specular, diffuse)

<script id="per-fragment-lighting-vs" type="x-shader/x-vertex"></script>// δημιουργία vertex-shader προγράμματος. Είναι υπεύθυνο για την οριστικοποίηση της τελικής θέσης του αντικειμένου. Προαιρετικά για τα normals, texture, lighting, και color.

JavaScript σενάρια με την ενσωμάτωση συναρτήσεων

<script type="text/javascript">

function initGL(canvas)// συνάρτηση με την οποία αρχικοποιούμε και αποκτούμε το API context.

function getShader(gl, id)// υλοποίηση VS και FS shader σε γλώσσα GLSL.

function initShaders()// αρχικοποιεί και συνδέει τα shader προγράμματα και εν συνεχεία τα αποδίδει σε WebGL μέσω της μεταβλητής shaderProgram. Επίσης παίρνει αναφορά για ένα attribute και το αποθηκεύει σε μια νέα μεταβλητή.

function handleLoadedTexture(texture)// μετατρέπει την εικόνα σε 2D μέσω της texImage2D και την προωθεί στην κάρτα γραφικών.

function initTextures()// συνάρτηση στην οποία βρίσκονται αποθηκευμένες οι εικόνες που πρόκειται να χρησιμοποιήσουμε ως υφή.

function mvPushMatrix()// συνάρτηση που υποστηρίζει τη λειτουργία animate.

function mvPopMatrix()// συνάρτηση που υποστηρίζει τη λειτουργία animate.

function setMatrixUniforms()// συνάρτηση με την οποία μετακινείται το model-view και το projection του animate στη WebGL.

function degToRad(degrees)// συνάρτηση που καθορίζει τις μοίρες περιστροφής του αντικειμένου.

function handleLoadedTeapot(teapotData)// παίρνει τις τιμές από τις λίστες του .json αρχείου και τις τοποθετεί σε WebGL πίνακες. Οι πίνακες με την σειρά τους περνάνε στην κάρτα γραφικών μέσω των buffer.Στην συνάρτηση αυτή δηλώνονται επίσης τέσσερις (4) μεταβλητές, οι var teapotVertexPositionBuffer; teapotVertexNormalBuffer; teapotVertexTextureCoordBuffer; teapotVertexIndexBuffer; στις οποίες βρίσκονται αποθηκευμένα τα attribute των κορυφών (VBO's). Εδώ βρίσκονται αποθηκευμένα τα attribute position, color, normals και texture coordinates κάθε κορυφής.

function loadTeapot()// δημιουργούμε ένα XMLHttpRequest το οποίο χρησιμοποιούμε για να φορτώσουμε το αρχείο JSON. Στο

σημείο αυτό του κώδικα εισάγουμε μια συνάρτηση επανάκλησης η οποία καλείται μέχρις ότου η διαδικασία της φόρτωσης του αντικειμένου φθάσει στο επιθυμητό επίπεδο (readyState of 4) πράγμα που σημαίνει ότι το αρχείο μας έχει φορτωθεί πλήρως.
function drawScene()// συνάρτηση στην οποία υλοποιείται η άντληση των δεδομένων που αφορούν το αντικείμενο.

function animate()// συνάρτηση η οποία καθορίζει το ποσοστό της περιστροφής του αντικειμένου.

function tick()// συνάρτηση που πρέπει να καλείται τακτικά. Σκοπός της είναι να κάνει update στην απεικόνιση του αντικειμένου σε τακτά χρονικά διαστήματα.

function webGLStart()// συνάρτηση η οποία καλεί άλλες συναρτήσεις με αποτέλεσμα την δημιουργία του αντικειμένου.
</script>

</head>

<body onload="webGLStart();">// με τη δήλωση αυτή εμφανίζεται στο σώμα της σελίδας το αντικείμενο.

<canvas id="lesson14-canvas" style="border: none;" width="500" height="500"></canvas>

</body>

</html>

1.10 Αρχεία .obj – Γεωμετρικά αρχεία

Τα αρχεία .obj είναι μια μορφή αρχείου που χρησιμοποιείται για την αποθήκευση γεωμετρικών αντικειμένων σε ASCII format και τα οποία αποτελούνται από γραμμές (lines), πολύγωνα (polygons), καμπύλες (curves) και επιφάνειες (surfaces) και αναπτύχθηκε από την Wavefront Technologies.

Ειδικότερα σε ένα .obj αρχείο ορίζεται η vertex position (θέση κάθε κορυφής), η texture coordinates vertex (θέση UV κάθε κορυφής), τα vertex normals και τα faces καταφέρνοντας έτσι οποιοδήποτε γεωμετρικό αντικείμενο (πολύγωνο) να οριστεί σαν μια λίστα από vertices (κορυφές) και texture vertices (κορυφές υφής).

Ένα αρχείο .obj αποτελείται από τις v λίστες (συντεταγμένες κορυφής) και f λίστες (συντεταγμένες υφής) και είναι της μορφής.

```
v 1 1 1
v 1 1 -1
v 1 -1 1
v 1 -1 -1
v -1 1 1
f 1 3 4 2
f 5 7 8 6
f 1 5 6 2
f 3 7 8 4
f 1 5 7 3
f 2 6 8 4
```

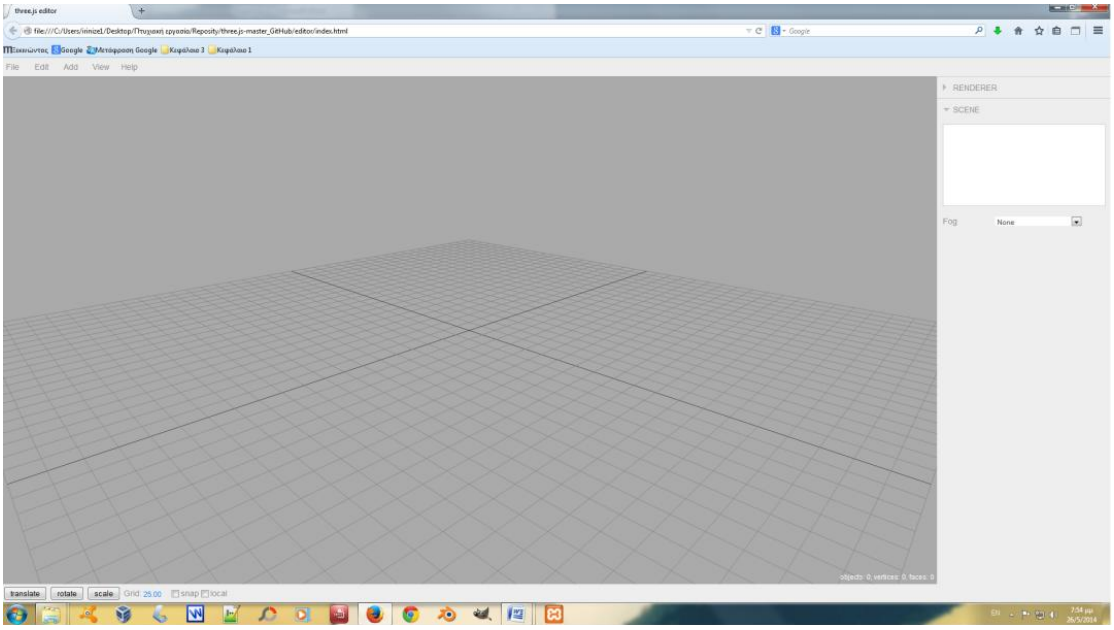
1.11 Άσκηση

Στο σημείο αυτό στηριζόμενοι στο παραπάνω παράδειγμα επιχειρούμε την αλλαγή των τιμών που αντιπροσωπεύουν το αντικείμενο Teapot με τις τιμές που εμείς μετατρέψαμε σε .json ενός αντικειμένου .obj που κάναμε download από το Διαδίκτυο.

Τα βήματα που θα ακολουθήσουμε είναι τα εξής:

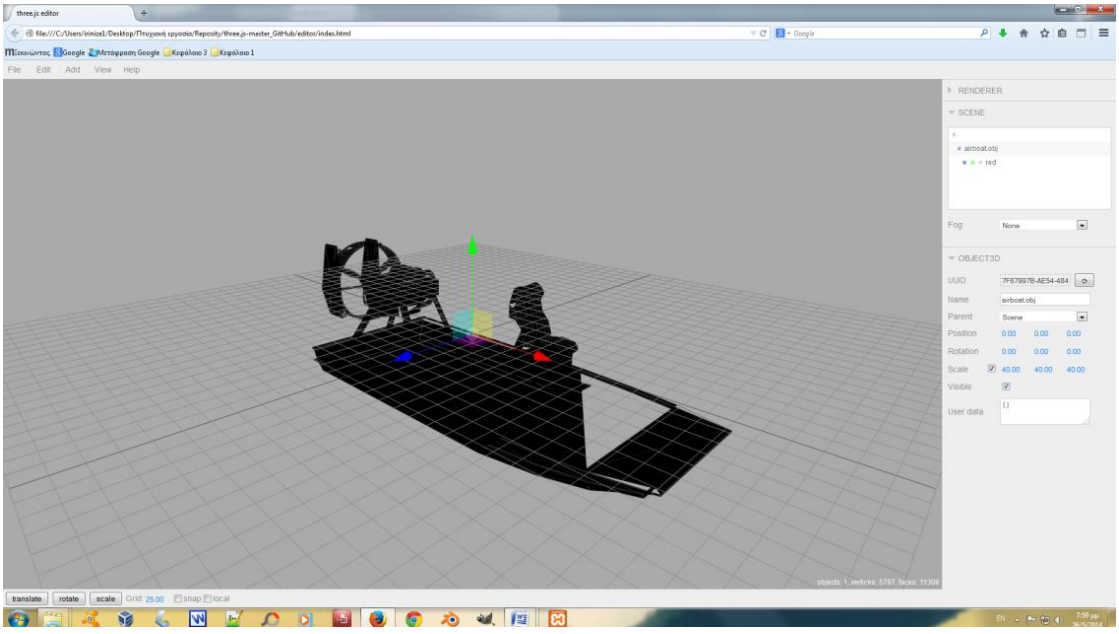
"Κατεβάζουμε" (download) ένα αντικείμενο σε .obj format από την διεύθυνση http://people.sc.fsu.edu/~jburkardt/data/obj/obj.html Το αρχείο που έχουμε επιλέξει είναι το airboat.obj και airboat.png. Κάνοντας κλικ πάνω στο αρχείο airboat.obj αυτό που θα δούμε είναι μια λίστα παραμέτρων τις οποίες τις αντιγράφουμε και τις αποθηκεύουμε μέσω του Notepad++ στον υπολογιστή μας. Αποθηκεύουμε ως airboat.obj.

Επισκεπτόμαστε τη σελίδα https://github.com/mrdoob/three.js/ ή τη http://threejs.org/ και κάνουμε download και αποθήκευση της βιβλιοθήκης Three.js στον υπολογιστή μας. Αυτό που εμείς θα αναζητήσουμε είναι ο φάκελος editor και πιο συγκεκριμένα το index.html έγγραφο. Κάνοντας δεξί κλικ πάνω στο index.html επιλέγουμε Άνοιγμα με_Firefox και εμφανίζεται στην οθόνη του υπολογιστή μας ένας three.js editor.



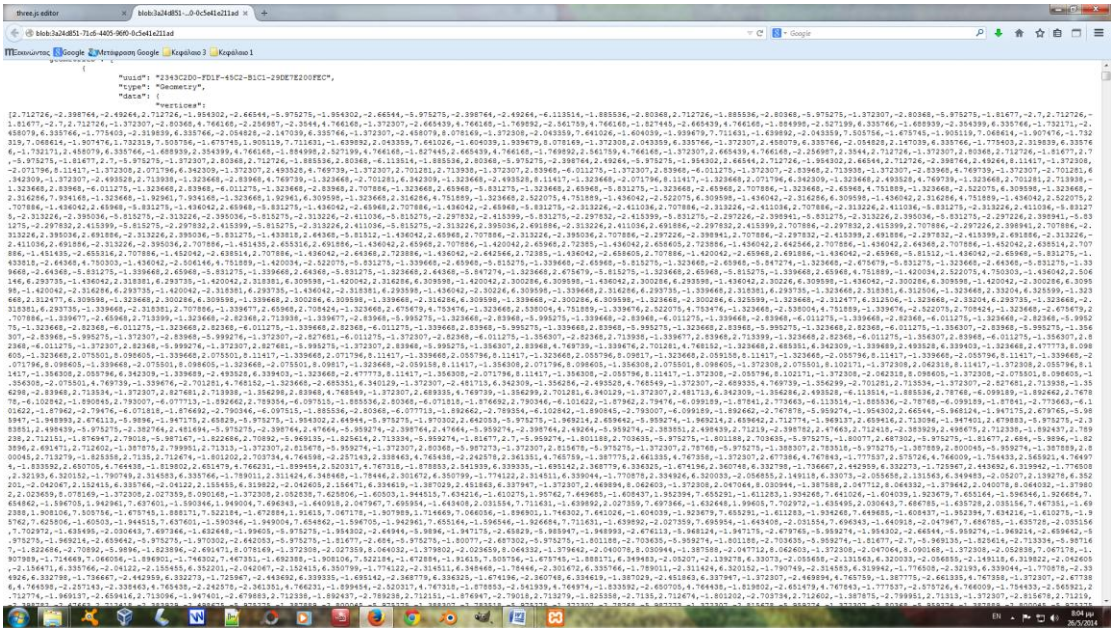
Εικόνα 7: απεικόνιση three.js editor (screenshot)

Εν συνεχεία κάνουμε File_Import και μας ανοίγει το πλαίσιο διαλόγου από το οποίο θα επιλέξουμε το .obj αντικείμενο που αποθηκεύσαμε στον υπολογιστή μας στο προηγούμενο βήμα δηλ. το airboat.obj. Αυτό που θα δούμε στην οθόνη του υπολογιστή μας είναι



Εικόνα 8: απεικόνιση εισαγωγής .obj αρχείου στον three.js editor (screenshot)

Επόμενό μας βήμα είναι η εξαγωγή των τιμών που απεικονίζουν το αντικείμενο. Αυτό γίνεται κάνοντας File_Export Scene.

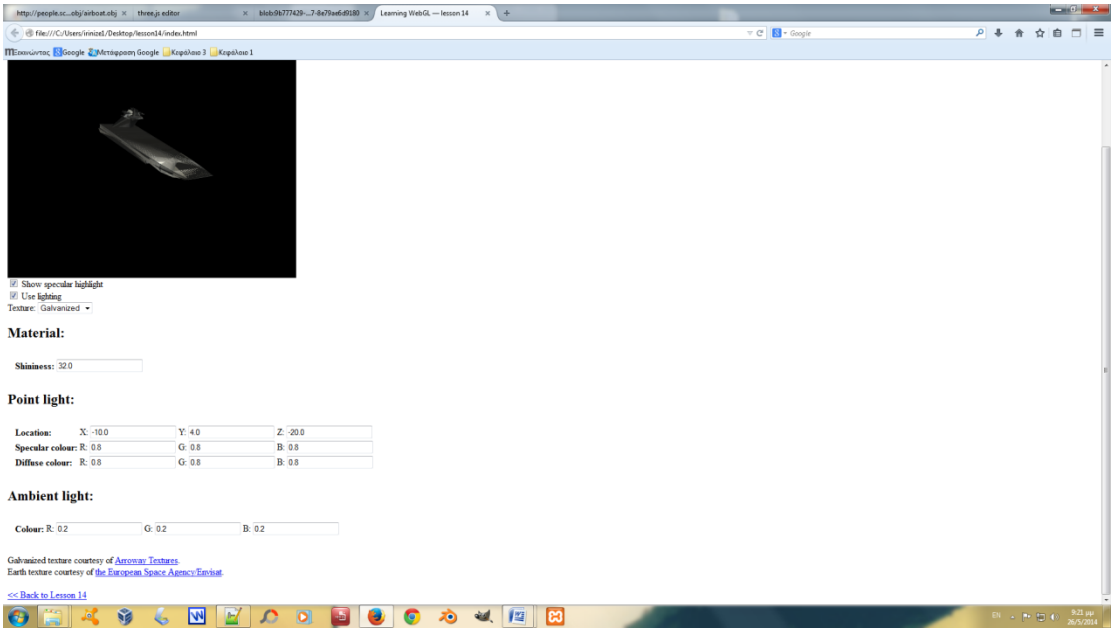


Εικόνα 9: απεικόνιση εξαγωγής τιμών από τον three.js editor (screenshot)

Κάνουμε copy-paste τις τιμές και τις αποθηκεύουμε ως airboat.json μέσω του Notepad++.
Οι τελικές τιμές μας είναι οι

```
{       "vertices": [2.712726,-2.398764,...,
        "normals": [0.0081740019916676 04,0,...,
        "faces": [16,5048,5018,5032,0,16,...
}
```

Ο πίνακας "vertices": [2.712726,-2.398764,..., περιέχει τιμές που αφορούν τη θέση του αντικειμένου ουσιαστικά δηλαδή το airboat αντικείμενο. Οι πίνακες "normals": [0.0081740019916676 04,0,..., και "faces": [16,5048,5018,5032,0,16,... περιέχουν τιμές που αφορούν τα στοιχεία ρεαλισμού που προσδίδονται στο αντικείμενο (φωτισμός). Βασιζόμενοι στο παράδειγμα Teapot αντικαθιστούμε τις τιμές που αφορούν τη θέση του αντικειμένου δηλ. το "vertices" του airboat με τις τιμές "vertexPositions" του Teapot. Αυτό που επιτύχαμε είναι



Εικόνα 10: απεικόνιση περιστρεφόμενου airboat με την δυνατότητα αλλαγής των ρυθμίσεων του φωτισμού καθώς και την αλλαγή υφής στο σώμα ενός .html εγγράφου με τη χρήση .json (screenshot)

Συμπέρασμα: it works!

ΚΕΦΑΛΑΙΟ 2ο

Εισαγωγή

Σκοπός του πρακτικού μέρους αυτής εδώ της εργασίας είναι η απεικόνιση γεωμετρικών δεδομένων σε 3D Web περιβάλλον με τη χρήση της WebGL και της HTML5 καθώς επίσης και με τη χρήση ενός προγράμματος δημιουργίας 3D απεικονίσεων (Blender) μέσω της βιβλιοθήκης Three.js σε τοπικό περιβάλλον Apache. Οι δυνατότητες της εφαρμογής αυτής είναι η μετακίνηση του αντικειμένου γύρω από την τροχιά της κάμερας με κλικ και σύρσιμο του ποντικιού καθώς και η αλλαγή της εστιακής απόστασης του αντικειμένου με τη χρήση τροχού ποντικιού.

Για την επίτευξη του πρακτικού μέρους θα πρέπει το υπολογιστικό μας σύστημα να διαθέτει το κατάλληλο hardware και software. Στην παράγραφο 2.1 και 2.2 περιγράφονται τα χαρακτηριστικά του υπολογιστικού μηχανήματος καθώς επίσης και οι απαιτήσεις της WebGL σε υλικό και σε λογισμικό εξοπλισμό.

2.1 Χαρακτηριστικά υπολογιστή

Το υπολογιστικό μηχάνημα που υλοποιήθηκε η εργασία διαθέτει λειτουργικό σύστημα OS Win7 με Έκδοση Windows: Windows 7 Prenium, Service Pack 1 και επεξεργαστή: Pentium(R) Dual Core CPU E55300 @ 2.60GHz. Διαθέτει επίσης μνήμη RAM: 3.00GB, τύπο συστήματος: Λειτουργικό σύστημα 32-bit και κάρτα γραφικών: NVIDIA GeForce 7300GS.

2.2 Απαιτήσεις Hardware και Software

Στην παράγραφο αυτή αναλύουμε τις απαιτήσεις του υπολογιστικού μας συστήματος για την χρήση της WebGL καθώς επίσης και τον τρόπο απόκτησής τους. Θα αναφερθούμε αρχικά στον υλικό εξοπλισμό και μετά στα λογισμικά προγράμματα.

2.2.1 Hardware

Ισχυρή κάρτα γραφικών (INVDIA). Η κάρτα γραφικών είναι τμήμα ενός υπολογιστή, το οποίο λαμβάνει δεδομένα από την Κεντρική Μονάδα Επεξεργασίας (CPU) για να τα μετατρέψει σε εικόνα, η οποία θα προβληθεί στην οθόνη. Εμείς έχουμε επιλέξει την NVIDIA GeForce 7300GS.

2.2.2 Software

Επιλογή web browser και αναβάθμιση σε έκδοση συμβατή με την WebGL. Διαθέσιμα προγράμματα περιήγησης μπορείτε να βρείτε στη σελίδα http://www.browserchoice.eu/BrowserChoice/browserchoice_el.htm Για να έχουμε τη δυνατότητα να δούμε τα WebGL παραδείγματα θα πρέπει να καθορίσουμε την συμβατότητα του προγράμματος περιήγησης (web browser). Συμβατές εκδόσεις αποτελούν οι Firefox 4+, Chrome 9+, Safari 5.1.

Update τους drivers (προγράμματα οδήγησης) γραφικών στην τελευταία τους έκδοση. Στη διεύθυνση http://www.nvidia.com/Download/index.aspx?lang=en-us μπορούμε να βρούμε διαθέσιμους drivers.

Ενίσχυση των drivers γραφικών χρησιμοποιώντας το Microsoft DirectX.

Πρόκειται για μια συλλογή διεπαφών (API'S) για την διεκπεραίωση καθηκόντων που σχετίζονται με τα πολυμέσα και ιδιαίτερα με προγραμματισμό παιχνιδιών και βίντεο. Το DirectX runtime είναι διαθέσιμο δωρεάν από την σελίδα της Microsoft στη διεύθυνση https://www.microsoft.com/en-us/download/search.aspx?q=directx

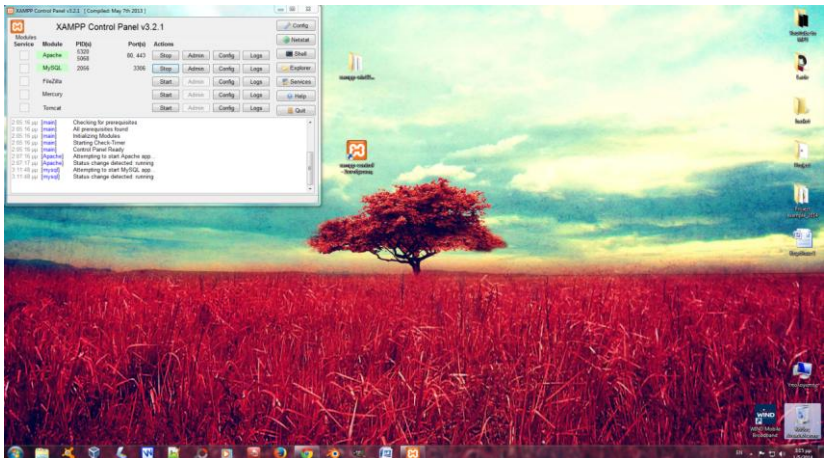
Έλεγχος ενεργοποίησης της γλώσσας JavaScript στο πρόγραμμα περιήγησης Firefox. Τα βήματα για την ενεργοποίηση της JavaScript είναι τα εξής: Στη γραμμή διευθύνσεων, πληκτρολογήστε about: config και πατήστε enter. Κάντε κλικ στο κουμπί "Θα πρέπει να είστε προσεκτικοί, το υπόσχομαι" εάν εμφανιστεί ένα μήνυμα προειδοποίησης. Στο πλαίσιο αναζήτησης πληκτρολογήστε javascript.enabled.Κάντε δεξί κλικ στο "javascript.enabled" και επιλέξτε "Αλλαγή" για να αλλάξετε την τιμή από "false" σε "true".

Εγκατάσταση Notepad++ (κειμενογράφος κώδικα). Πρόκειται για απλή εγκατάσταση προγράμματος.

Απόκτηση της βιβλιοθήκης Three.js. Μέλημά μας είναι να "κατεβάσουμε" (download) την βιβλιοθήκη από το αποθετήριο του GitHub που βρίσκεται στη διεύθυνση https://github.com/mrdoob/three.js/ ή από την ιστοσελίδα του Three.org που βρίσκεται στη διεύθυνση http://threejs.org/ και να την αποθηκεύσουμε στον υπολογιστή μας. Η χρήση της προϋποθέτει την συμπερίληψη κάποιων scripts στον index κώδικά μας.

Τα βήματα για την απόκτηση της βιβλιοθήκης είναι τα εξής: Κάνουμε download τη βιβλιοθήκη three.js ως εξής: http://threejs.org/ , αριστερό frame, download, extract. Βρίσκουμε το φάκελο build και αντιγράφουμε το script three.min.js στο φάκελο που αναπτύσσουμε το project μας. Βρίσκουμε επίσης το φάκελο examples-js-controls και αντιγράφουμε το script OrbitControls.js στο φάκελο που αναπτύσσουμε το project μας. Συμπεριλαμβάνουμε τα scripts στον index κώδικά μας.

Απόκτηση XAMPP (Apache). Πρόκειται για ένα πακέτο ελεύθερου και ανοιχτού κώδικα λογισμικού. Περιέχει εξυπηρετητή ιστοσελίδων (http Apache), βάση δεδομένων (MySQL) και έναν διερμηνέα κώδικα ο οποίος είναι γραμμένος σε γλώσσα προγραμματισμού PHP και Perl. Τα βήματα για την απόκτηση του XAMPP είναι τα εξής: Επισκεφθείτε τη διεύθυνση http://sourceforge.net/projects/xampp/ και "κατεβάστε" το κατάλληλο αρχείο για τα χαρακτηριστικά του υπολογιστή. Αντιγράψτε το αρχείο xampp-win32-1.8.3-3-VC11-installer στον τοπικό δίσκο C και εν συνεχεία βρείτε το Xampp control κάντε run. Εάν θέλουμε μπορούμε να δημιουργήσουμε μια συντόμευση του xampp-control κάνοντας δεξί κλικ στο εικονίδιο και επιλέγοντας το πεδίο Δημιουργία συντόμευσης. Ακολουθεί η ενεργοποίηση του Apache.

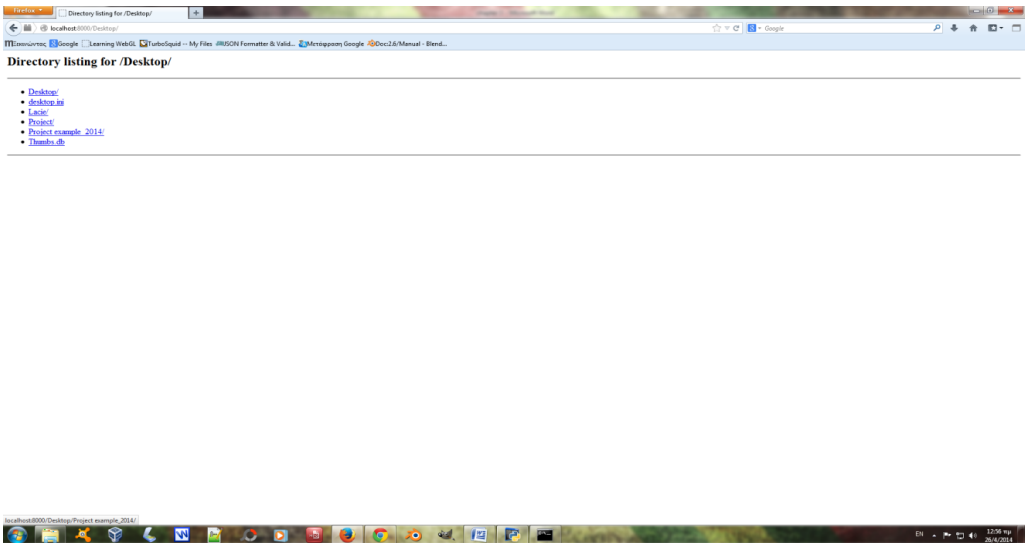


Εικόνα 11: απεικόνιση ενεργοποίησης Apache (screenshot)

Ακολουθεί το άνοιγμα μιας θύρας. Για να ανοίξτε μια θύρα στο router θα πρέπει να γνωρίζετε την ip του router του υπολογιστή. Επισκεφθείτε τη διεύθυνση http://portforward.com/english/routers/port_forwarding/routerindex.htm , επιλέξτε από τον κατάλογο που εμφανίζεται το όνομα του δικού σας router, συνεχίστε όπως κατευθύνεστε από τη σελίδα ακριβώς έτσι ώστε να επιλέξετε την δική σας έκδοση του router. Σημαντική ερώτηση αποτελεί το τι θύρα θέλουμε να ανοίξουμε πχ. για ιστοσελίδες, παιχνίδια, αρχεία κα. Επιλέγουμε HTTP.

Αυτό που μόλις κάναμε θα εξυπηρετήσει τα αρχεία από τον τρέχοντα κατάλογο σε localhost μέσω της θύρας 80. Να τονίσουμε εδώ ότι ο φάκελος με το project μας θα πρέπει να είναι τοποθετημένος σύμφωνα με το μονοπάτι C: - xampp – htdocs.

Στην πορεία και όταν πλέον η διαδικασία για την εμφάνιση 3D αντικειμένου θα έχει ολοκληρωθεί θα επιλέξουμε από την λίστα που θα μας εμφανιστεί το φάκελο Project alfa 147_2014. Κάνοντας κλικ πάνω του θα εμφανιστεί στην οθόνη του υπολογιστή μας το index αρχείο που δημιουργήσαμε. Γράφουμε λοιπόν στο πρόγραμμα περιήγησης Firefox http://localhost/, πατάμε enter και επιλέγουμε τον κατάλληλο φάκελο.



Εικόνα 12: απεικόνιση καταλόγου localhost (screenshot)

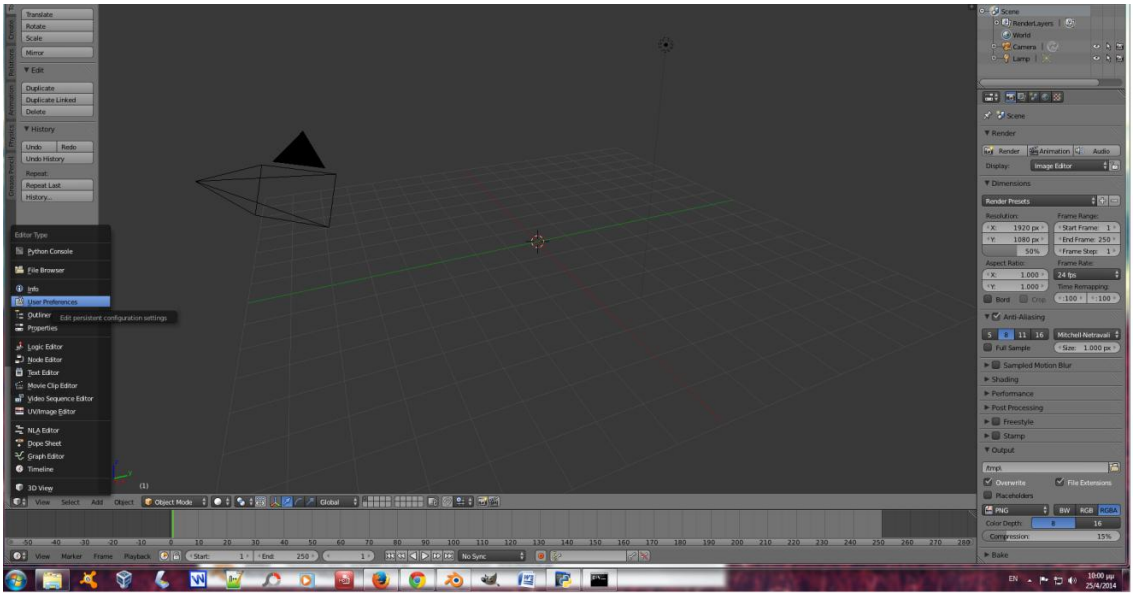
Εγκατάσταση Blender. Ένας εύκολος τρόπος δημιουργίας σύνθετων WebGL μοντέλων είναι να χρησιμοποιήσουμε ένα εξειδικευμένο πακέτο (λογισμικό) για 3D γραφικά και να τα ανεβάσουμε σε WebGL περιβάλλον. Το Blender είναι ένα δωρεάν ανοιχτού τύπου λογισμικό για 3D γραφικά που μπορεί να

χρησιμοποιηθεί για να δημιουργήσει μοντέλα, κινούμενα μοντέλα καθώς επίσης να εισάγει ή να εξάγει μοντέλα με διαφορετικές μορφοποιήσεις όπως .js, .obj κα. Το Blender μπορούμε να το "κατεβάσουμε" από την διεύθυνση http://www.blender.org/ Πρόκειται για απλή εγκατάσταση προγράμματος.

Εγκατάσταση add-ons three.js στο Blender. Επιπλέον θα πρέπει να εγκαταστήσουμε και να ενεργοποιήσουμε την three.js βιβλιοθήκη. Τα βήματα για την απόκτηση της ενεργοποίηση της βιβλιοθήκης three.js στο Blender είναι τα εξής:

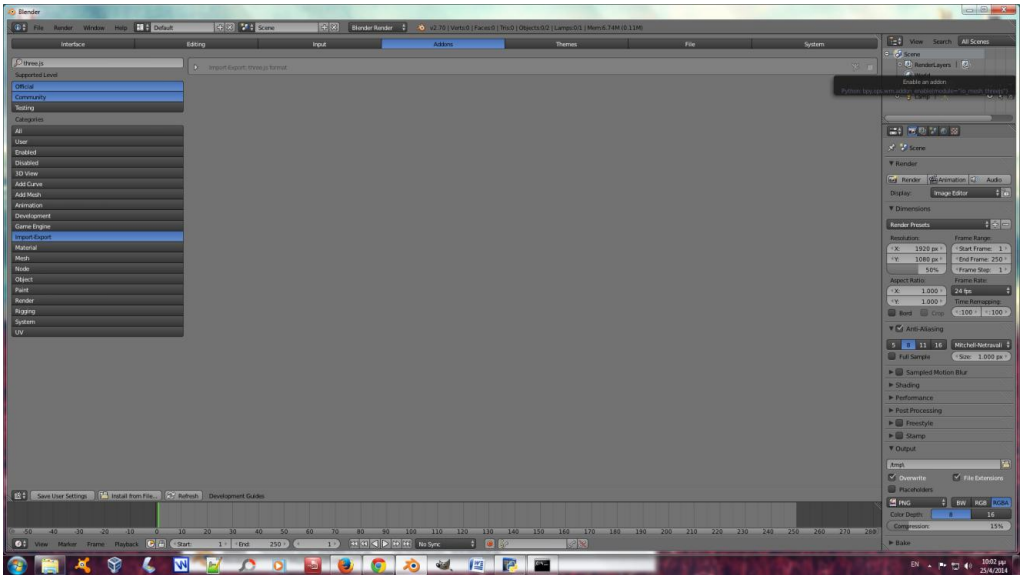
Για την εγκατάσταση πρόσθετων στο Blender και ειδικότερα της three.js βιβλιοθήκης θα πρέπει να επισκεφτούμε το αποθετήριο του GitHub στη διεύθυνση https://github.com/mrdoob/three.js/tree/master/utils/exporters/blender/2.65/scripts/addons Επιλέγουμε το φάκελο 2.65_scripts_addons_io_mesh_threejs. Το io_mesh_threejs είναι το αρχείο που θα αντιγράψουμε στο φάκελο addons που βρίσκεται στο μονοπάτι C:\Χρήστες\irinize1\AppData\Roamming\Blender Foundation\Blender\2.67\scripts\addons. Αν δεν υπάρχει ο φάκελος io_mesh_threejs τον δημιουργούμε.

Για την ενεργοποίηση της three.js βιβλιοθήκης ανοίγουμε το Blender και πηγαίνουμε στο Users Preferences



Εικόνα 13: (screenshot)

κάνοντας κλικ εμφανίζεται στην οθόνη μας η εικόνα



Εικόνα 14: (screenshot)

Πληκτρολογούμε στην αναζήτηση three.js και πατώντας enter οδηγούμαστε κατευθείαν στο εικονίδιο –δεξιά του frame- που ενεργοποιεί την Three.js βιβλιοθήκη. Αποθηκεύω κάνοντας Save Users Settings.

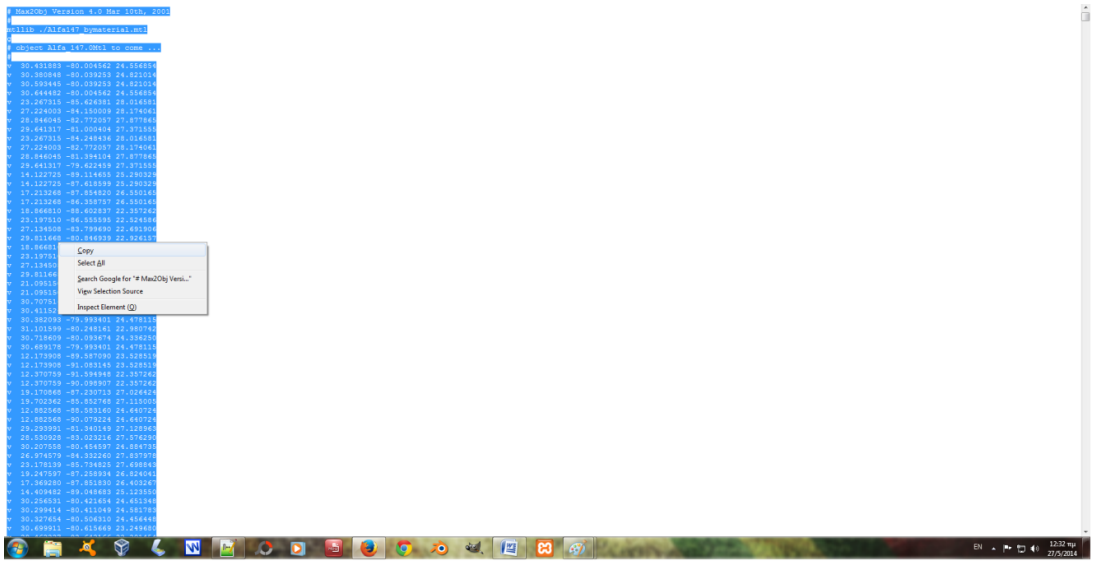
2.3 Πρακτικό μέρος (φάση πρώτη)

Θα ξεκινήσουμε το πρακτικό μέρος αναφέροντας εν συντομία τα βήματα που θα πρέπει να κάνουμε έτσι ώστε να δημιουργήσουμε μια 3D απεικόνιση. Τα βήματα αυτά είναι η απόκτηση του .obj αρχείου με τη χρήση internet, η αποθήκευσή του στον υπολογιστή μας, η εισαγωγή του στο Blender και η έξοδος του ως αρχείο three.js.

Απόκτηση .obj αρχείου. "Κατεβάζουμε" (download) ένα αντικείμενο σε .obj format από την διεύθυνση http://people.sc.fsu.edu/~jburkardt/data/obj/obj.html Το αρχείο που έχουμε επιλέξει είναι το alfa 147.obj. Κάνοντας κλικ πάνω στο αρχείο, αυτό που θα δούμε είναι μια λίστα παραμέτρων.

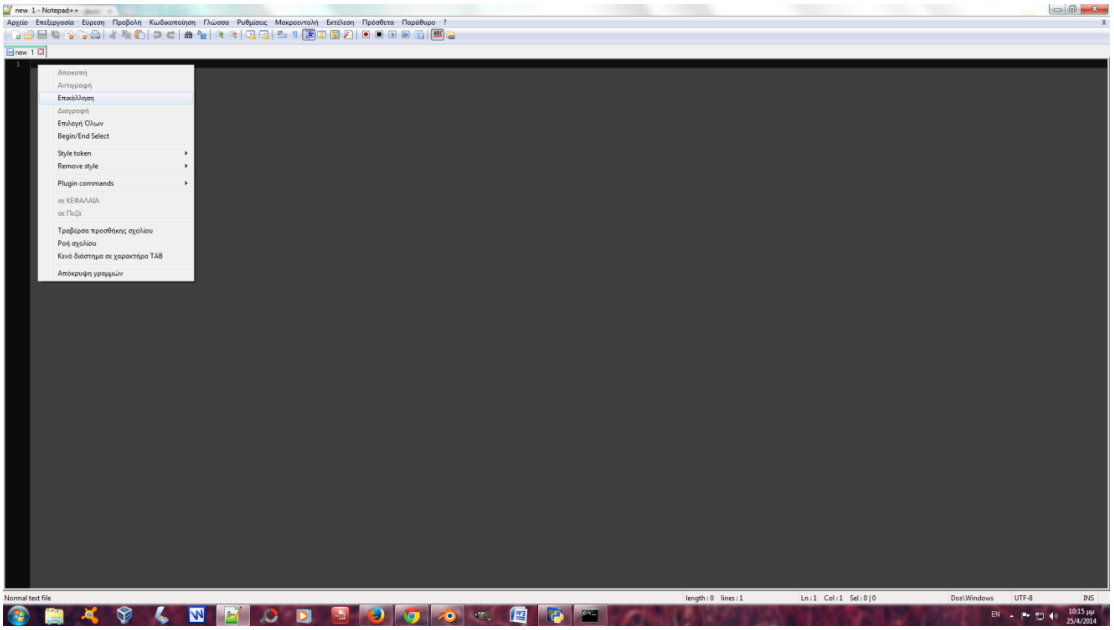
Τα βήματα για την απόκτηση της λίστας παραμέτρων του .obj αρχείου είναι τα εξής:

Αντιγραφή των τιμών. Επιλογή_ δεξί κλικ_ Αντιγραφή



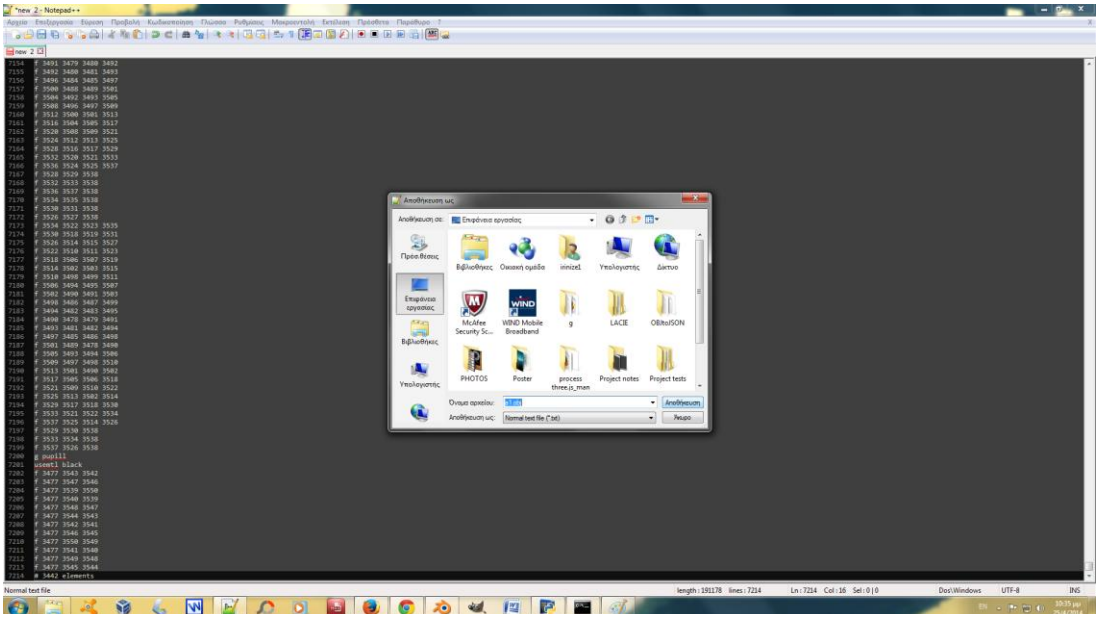
Εικόνα 15: απεικόνιση παραμέτρων .obj αρχείου (screenshot)

τις οποίες τις επικολλούμε στο Notepad++. File_New_Select_Paste



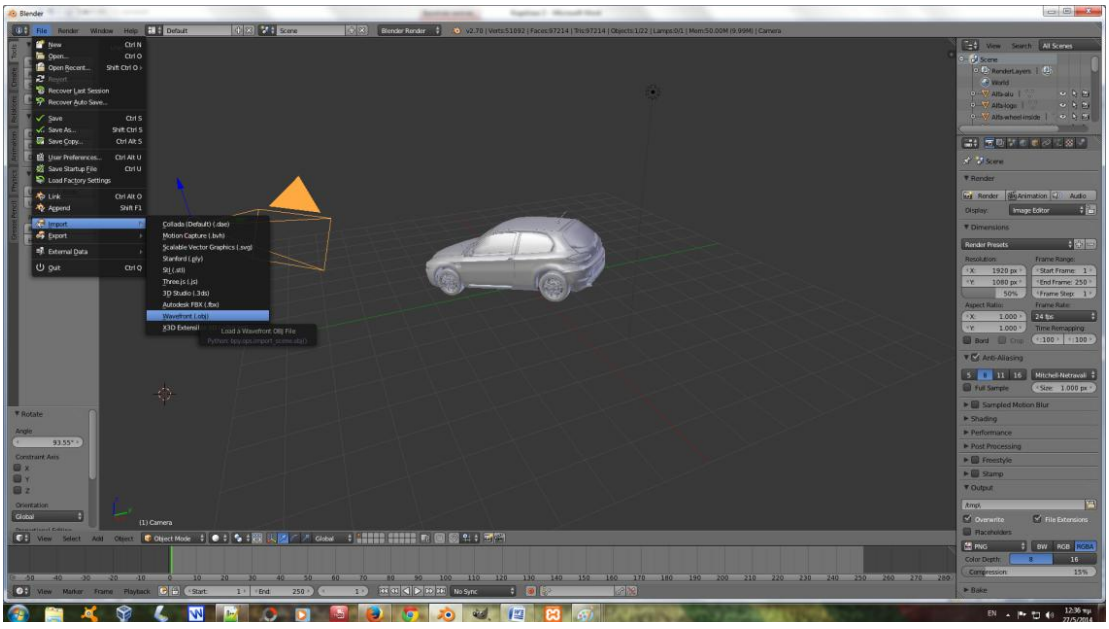
Εικόνα 16: απεικόνιση διαδικασίας αντιγραφής των παραμέτρων του .obj αρχείου 1 (screenshot)

και τις οποίες εν συνεχεία θα τις αποθηκεύσουμε ως .obj.



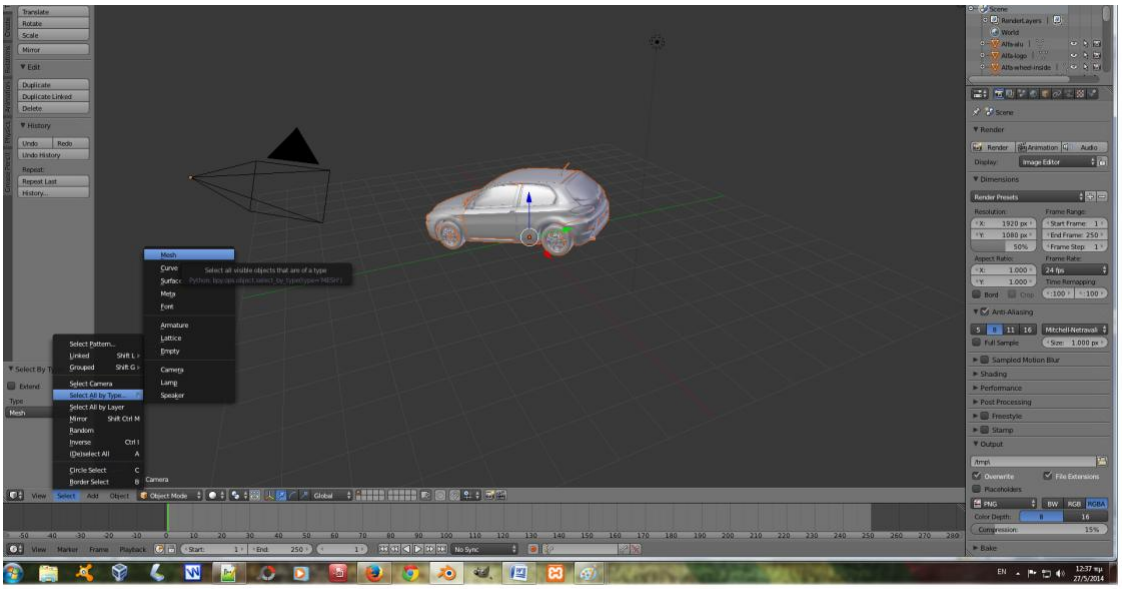
Εικόνα 17: απεικόνιση διαδικασίας αντιγραφής των παραμέτρων του .obj αρχείου 2 (screenshot)

Εισαγωγή – εξαγωγή 3D μοντέλων. Τα βήματα για την εισαγωγή και την επιλογή του .obj αρχείου καθώς και για την εξαγωγή του ως three.js αρχείο είναι τα εξής:
Εισαγωγή .obj μοντέλου: File_ Import_ Wavefront (.obj). Επιλέγουμε το αρχείο που αποθηκεύσαμε στο προηγούμενο βήμα δηλ. το alfa 147.obj.



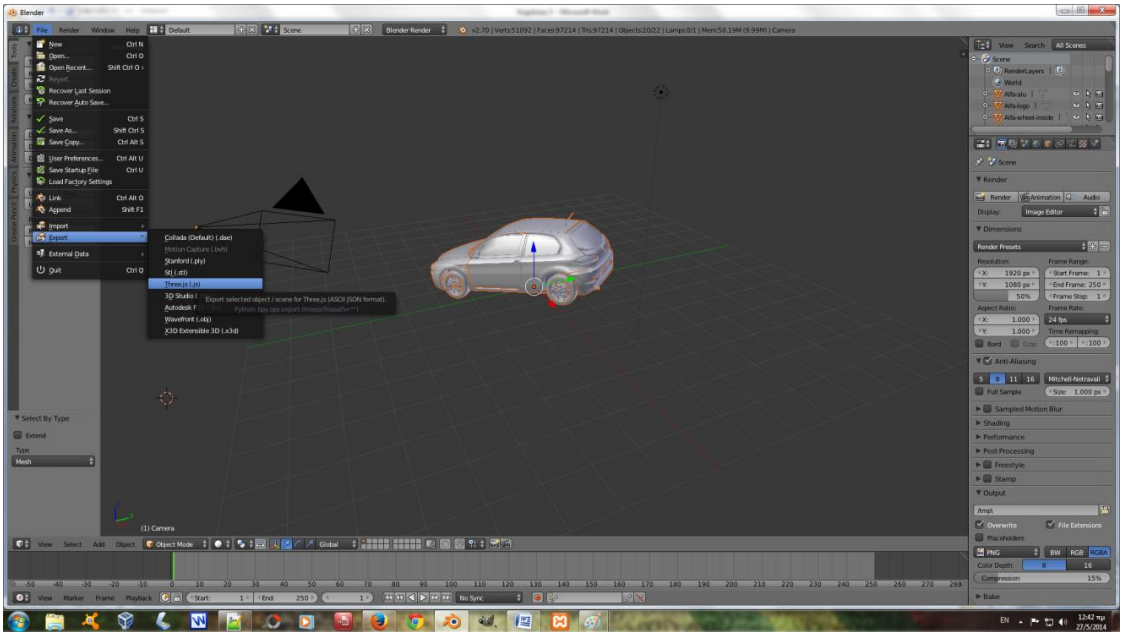
Εικόνα 18: απεικόνιση διαδικασίας εισαγωγής των παραμέτρων του .obj αρχείου (screenshot)

Επιλογή αντικεμένου: Select_ Select All by Type_ Mesh



Εικόνα 19: απεικόνιση διαδικασίας επιλογής του.obj αρχείου (screenshot)

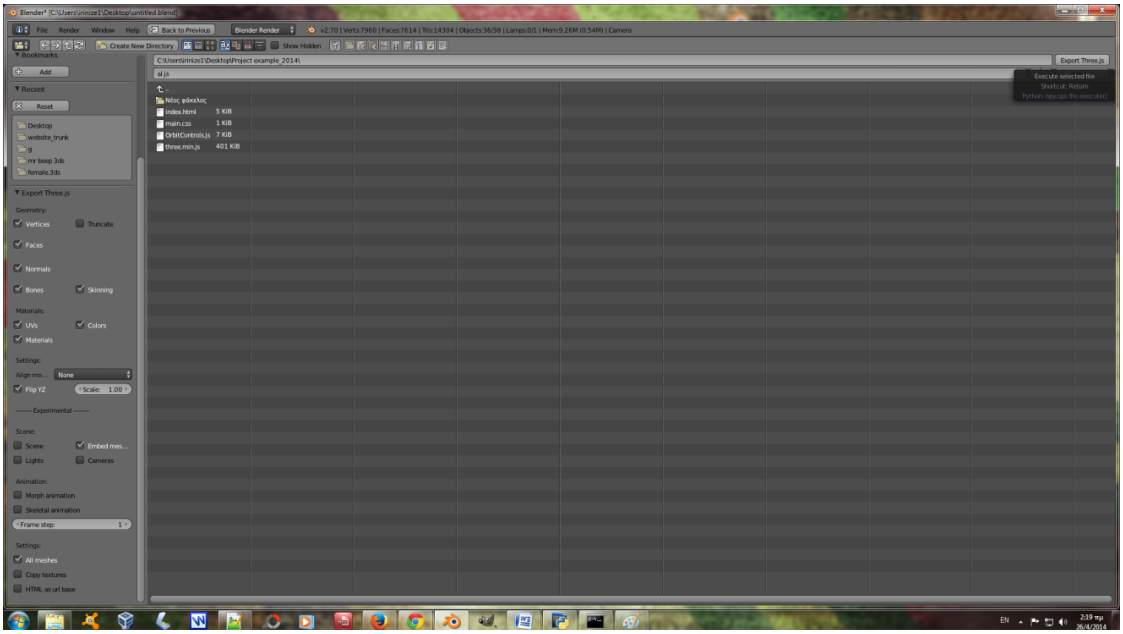
Εξαγωγή του αντικειμένου σε .js format: File_Export_Three.js (.js)



Εικόνα 20: απεικόνιση διαδικασίας εξαγωγής του .obj αρχείου σε three.js μορφοποίηση (screenshot)

Πριν την τελική εξαγωγή του αντικειμένου σε .js format ενεργοποιούμε -στο αριστερό frame και στο πεδίο Export three,js-

τις παραμέτρους Vertices, Faces, Normals, Bones, Skinning, UV's, Colors, Materials, Flip YZ, Embed mesh και All meshes. Η αποθήκευση γίνεται στον ίδιο φάκελο με το index αρχείο μας.



Εικόνα 21: απεικόνιση διαδικασίας ρυθμίσεων για την εξαγωγή του .obj αρχείου σε three.js μορφοποίηση (screenshot)

2.4 Πρακτικό μέρος – Index.html κώδικας (φάση δεύτερη)

Ακολουθεί ο index.html κώδικας που μας επιτρέπει την απεικόνιση γεωμετρικών δεδομένων σε 3D Web περιβάλλον με τη χρήση της WebGL και της HTML5 καθώς επίσης και με τη χρήση ενός προγράμματος δημιουργίας 3D απεικονίσεων (Blender) μέσω της βιβλιοθήκης Three.js σε τοπικό περιβάλλον Apache. Οι δυνατότητες της εφαρμογής αυτής είναι η μετακίνηση του αντικειμένου γύρω από την τροχιά της κάμερας με κλικ και σύρσιμο του ποντικιού καθώς και η αλλαγή της εστιακής απόστασης (ζουμ) του αντικειμένου με τη χρήση τροχού ποντικιού.

Αναλυτικά ο index.html κώδικας

```
<!doctype html>
<html lang="en">
<head>
```

```
                <title>3D OBJECTS FOR WEB</title>
                <meta charset="utf-8">

<script src="three.min.js"></script>
<script src="OrbitControls.js"></script>
                <link rel="stylesheet" href="main.css"
        </head>

<body style="margin: 0;">

                <header id="top_header">
                <h1>ΕΡΓΑΣΙΑ: Προσομοίωση φυσικών συστημάτων και απεικόνισης δεδομένων σε 3D Web περιβάλλον με τη χρήση της τεχνολογίας WEBGL και HTML5</br>
ΟΝΟΜΑΤΕΠΩΝΥΜΟ: Ειρήνη Τσιάκαλου, Α.Μ.: 9388</h1>
                </header>

<script>

   // Set up the scene, camera, and renderer as global variables.
   var scene, camera, renderer;

   init();
   animate();

   // Sets up the scene.
   function init() {

     // Create the scene and set the scene size.
     scene = new THREE.Scene();
     var WIDTH = "800",
        HEIGHT = "800";

     // Create a renderer and add it to the DOM.
     renderer = new THREE.WebGLRenderer ({antialias:true});
     renderer.setSize(WIDTH, HEIGHT);
     document.body.appendChild (renderer.domElement);

     // Create a camera, zoom it out from the model a bit, and add it to the scene.
     camera = new THREE.PerspectiveCamera(45, WIDTH / HEIGHT, 0.1, 20000);
```

```
    camera.position.set(0,10,0);
    scene.add(camera);

    // Create an event listener that resizes the renderer with the
browser window.
    window.addEventListener('resize', function() {
      var WIDTH = window.innerWidth,
        HEIGHT = window.innerHeight;
      renderer.setSize(WIDTH, HEIGHT);
      camera.aspect = WIDTH / HEIGHT;
      camera.updateProjectionMatrix();
    });

    // Set the background color of the scene.
    renderer.setClearColorHex(0x778899, 1)
    // Create a light, set its position, and add it to the scene.
    var light = new THREE.PointLight(0xffffff);
    light.position.set(-100,200,100);
    scene.add(light);

    // Load in the mesh and add it to the scene.
    var loader = new THREE.JSONLoader();
    loader.load( "alfa 147.js", function(geometry){
      var material = new THREE.MeshLambertMaterial({color:
0x87cefa});
      mesh = new THREE.Mesh(geometry, material);
      scene.add(mesh);
    });

    // Add OrbitControls so that we can pan around with the mouse.
    controls = new THREE.OrbitControls(camera,
renderer.domElement);

  }


  // Renders the scene and updates the render as needed.
  function animate() {
    requestAnimationFrame(animate);

    // Render the scene.
    renderer.render(scene, camera);
```

```
    controls.update();

  }

 </script>
<section id="main_section">
        <article>
                <header>
                        <hgroup>
                        <h1>Χαρακτηριστικά υπολογιστή - Features computer</h1>
                        </hgroup>
                </header>
<p id="tuna"><h2>1. Έκδοση Windows</h2></br> Windows 7 Home Premium, Service Pack 1</br>
                  <h2>2. Λειτουργικό Σύστημα</h2></br>
Επεξεργαστής: Pentium(R) Dual-Core CPU  E5300  @ 2.60GHz</br>
Μνήμη RAM: 3.00GB</br>
Τύπος συστήματος: Λειτουργικό σύστημα 32-bit</br>
Κάρτα γραφικών: NVIDIA GeForce 7300GS
        </article>

        <article>
                <header>
                        <hgroup>
                        <h1>Βήματα - Steps</h1>
                        </hgroup>
                </header>
<p id="tuna">1. Συμβατότητα προγράμματος περιήγησης - (<a href="https://www.microsoft.com/en-us/download/search.aspx?q=directx" target="_blank">DirectX</a>,
<a href="http://www.nvidia.com/Download/index.aspx?lang=en-us" target="_blank">drivers GPU</a>
&
<a href="https://support.mozilla.org/en-US/kb/upgrade-graphics-drivers-use-hardware-acceleration?redirectlocale=en-US&redirectslug=how-do-i-upgrade-my-graphics-drivers)" target="_blank">Firefox</a>)</br>

2. Αποκτώντας την three.js βιβλιοθήκη - (<a href="http://threejs.org/" target="_blank">Three.js library</a>)</br>
```

3. Καθορίζοντας το τοπικό περιβάλλον - (XAMPP)</br>

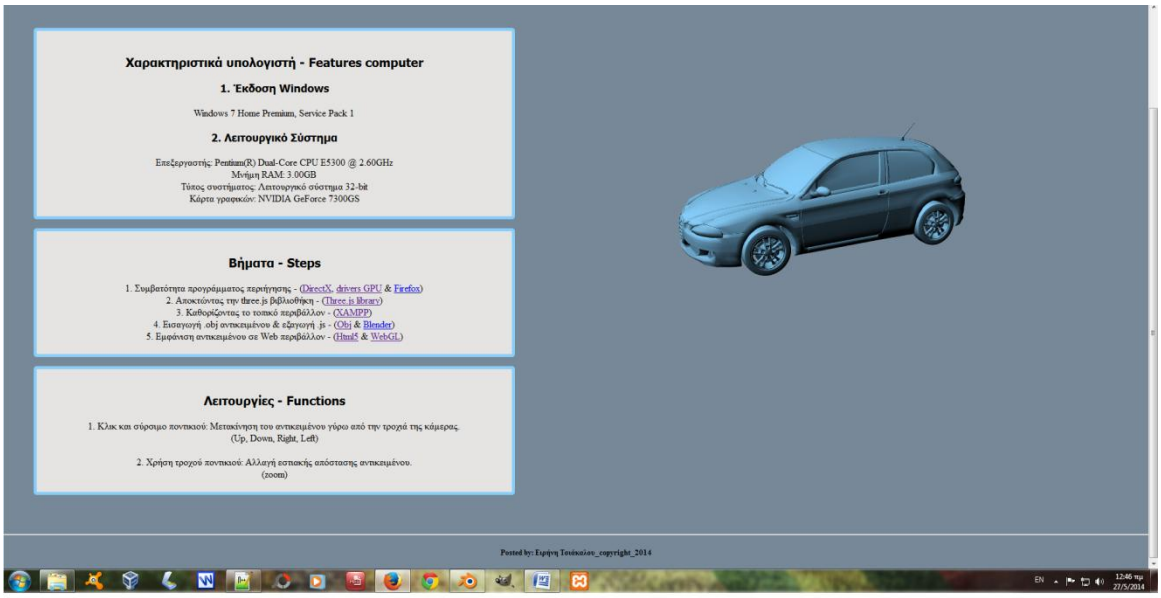
4. Εισαγωγή .obj αντικειμένου & εξαγωγή .js - (Obj & Blender)</br>

5. Εμφάνιση αντικειμένου σε Web περιβάλλον - (Html5 & WebGL)

```
        </article>
        <article>
            <header>
                <hgroup>
                <h1>Λειτουργίες - Functions</h1>
                </hgroup>
            </header>
<p id="tuna">1. Κλικ και σύρσιμο ποντικιού: Μετακίνηση του αντικειμένου γύρω από την τροχιά της κάμερας. </br>(Up, Down, Right, Left)</p></br>
<p id="tuna">2. Χρήση τροχού ποντικιού: Αλλαγή εστιακής απόστασης αντικειμένου. </br>(zoom)</p>
        </article>

</section>

<footer id="the_footer">
  <p><h5>Posted by: Ειρήνη Τσιάκαλου_copyright_2014</h5></p>
</footer>

</body>

</html>
```

2.4.1 Εμφάνιση μέσω localhost

Γράφουμε στο πρόγραμμα περιήγησης Firefox http://localhost/ και πατάμε enter. Επιλέγουμε από την λίστα που θα εμφανιστεί το φάκελο Project alfa 147_2014 και κάνοντας κλικ θα εμφανιστεί στο 3D αντικείμενό μας.



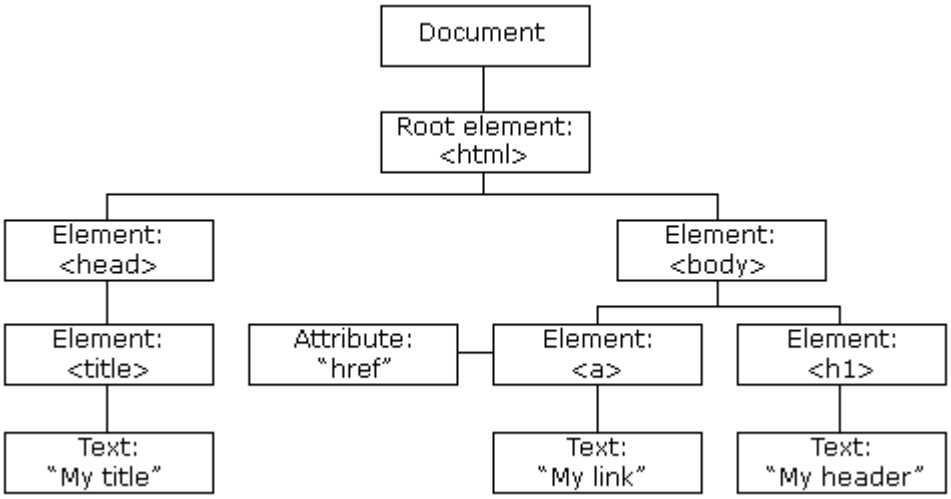
Εικόνα 22: απεικόνιση γεωμετρικών δεδομένων σε 3D Web περιβάλλον με τη χρήση της WebGL και της HTML5 σε τοπικό περιβάλλον Apache. Δυνατότητες εφαρμογής: η μετακίνηση του αντικειμένου γύρω από την τροχιά της κάμερας και η αλλαγή της εστιακής απόστασης (ζουμ)

2.4.2 Index.html κώδικας πρακτικού μέρους - Ανάλυση

```
<!doctype html>
<html lang="en">
<head>
                <title>...</title>
                <meta charset="utf-8">
<script src="three.min.js"></script>
<script src="OrbitControls.js"></script>

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

```
<body style="margin: 0;">

<script>

var scene, camera, renderer;// δημιουργία μεταβλητών για την scene, την μηχανή λήψης (φωτογραφική μηχανή - camera) και τη σκίαση (render) ως καθολικές μεταβλητές.
   init();
   animate();// με την κλήση των συναρτήσεων επιτυγχάνεται η απεικόνιση του αντικειμένου.
   function init()// υλοποίηση συνάρτησης η οποία ρυθμίζει τη scene (Sets up). Πιο αναλυτικά δημιουργεί τη scene και καθορίζει το μέγεθος της, δημιουργεί τη σκίαση την οποία μεταφέρει στο DOM, δημιουργεί την camera, καθορίζει τη θέση της και την προσθέτει στη scene. Επαναπροσδιορίζει το μέγεθος της σκίασης με βάση το παράθυρο του προγράμματος περιήγησης, ορίζει το χρώμα του background της scene και δημιουργεί φως στο οποίο ορίζει τη θέση του και το τοποθετεί στη scene. Εν συνεχεία τοποθετεί την γεωμετρική εικόνα (mesh-πλέγμα) μέσω της βιβλιοθήκης three.js στη scene τέλος προσθέτει το OrbitControls.js σενάριο για την μετακίνηση του ποντικιού (mouse).
   function animate()// ενημερώνει και αποδίδει την scene (refresh a web page)

 </script>

<section id="main_section">
        <article>
                <header>
                        <hgroup>…</hgroup>
                </header>
<p id="tuna">…</p>

        </article>
</section>

<footer id="the_footer">…</footer>

</body>

</html>
```

Να αναφέρουμε στο σημείο αυτό ότι DOM (Document Object Model) είναι το έγγραφο εκείνο με το οποίο η JavaScript μπορεί να έχει πρόσβαση και να τροποποιήσει τα στοιχεία ενός .html εγγράφου. Όταν μια ιστοσελίδα πρόκειται να "φορτωθεί" το πρόγραμμα περιήγησης δημιουργεί ένα DOM έγγραφο με τη μορφή δέντρου (tree) της ιστοσελίδας αυτής. Με το Object Model αυτό, η JavaScript έχει τη δυνατότητα να δημιουργήσει δυναμικές ιστοσελίδες με τη δυνατότητα αλλαγής των στοιχείων (element) της ιστοσελίδας και των χαρακτηριστικών, την τροποποίηση του CSS στυλ μορφοποίησης, την πρόσθεση ή την αφαίρεση στοιχείων κα. Πρόκειται δηλαδή για μια διεπαφή (interface) ανταλλαγής πληροφοριών μεταξύ διαφορετικών υπολογιστικών μηχανημάτων και λογισμικών προγραμμάτων.



Εικόνα 22: σχηματική απεικόνιση DOM εγγράφου με τη μορφή δέντρου

ΠΑΡΑΡΤΗΜΑ

Δίκτυο, διαδίκτυο και Διαδίκτυο

Δίκτυο υπολογιστών είναι ένα σύστημα επικοινωνίας που συνδέει δύο ή και περισσότερους αυτόνομους και ανεξάρτητους υπολογιστές καθώς και περιφερειακές συσκευές. Δύο υπολογιστές θεωρούνται διασυνδεδεμένοι όταν μπορούν να ανταλλάσσουν μεταξύ τους πληροφορίες. Σκοπός του δικτύου είναι η εξυπηρέτηση των αναγκών που προέκυψαν από την εξάπλωση της χρήσης των υπολογιστών.

Για να επικοινωνήσουν δύο υπολογιστικά συστήματα θα πρέπει να υπάρξει μεταξύ τους φυσική και λογική σύνδεση. Η διασύνδεση σε φυσικό επίπεδο επιτυγχάνεται χρησιμοποιώντας κάποιο φυσικό μέσο μετάδοσης πχ. ομοαξονικό καλώδιο, οπτικές ίνες κα. Αναγκαίος είναι ο καθορισμός του τρόπου με τον οποίο διασυνδέονται μεταξύ τους οι συσκευές του δικτύου, δηλ. η τοπολογία του δικτύου (πχ. τοπολογία δακτυλίου, αστέρα, διαύλου κτλ.). Η διασύνδεση σε λογικό επίπεδο αφορά την χρήση πρωτοκόλλων επικοινωνίας.

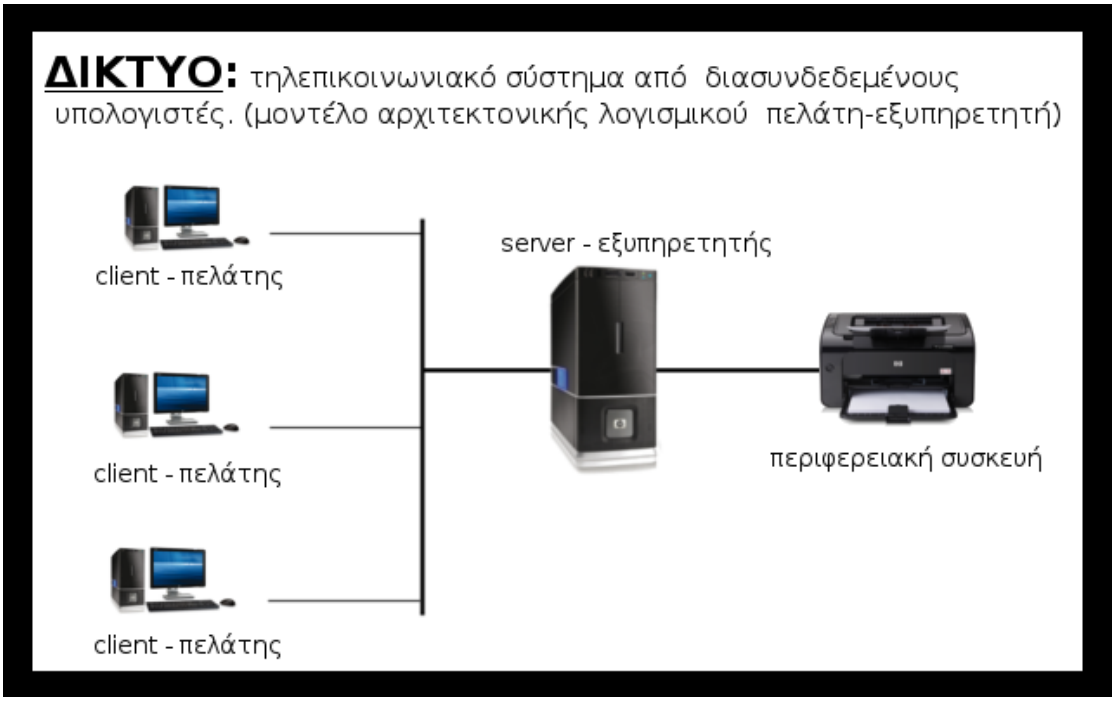
Για την υλοποίηση ενός δικτύου βασικά στοιχεία αποτελούν α) εξυπηρετητής αρχείων (file server) δηλ. ένας ισχυρός μικροϋπολογιστής που τρέχει το λειτουργικό σύστημα του δικτύου και διαχειρίζεται τη ροή των δεδομένων. Διαθέτει μεγάλη αποθηκευτική ικανότητα β) οι σταθμοί εργασίας, δηλ. προσωπικοί υπολογιστές με δικό τους λειτουργικό σύστημα οι οποίοι είναι συνδεδεμένοι μέσω καλωδίων και καρτών επικοινωνίας γ) οι κάρτες διασύνδεσης δικτύου (NIC – Network Interface Card) οι οποίες περιέχονται και στους file server και στις υπολογιστικές μηχανές. Μέσω της κάρτας αυτής γίνεται εφικτή η σύνδεση με τις υπόλοιπες συσκευές του δικτύου δ) περιφερειακές συσκευές και ε) το καλώδιο σύνδεσης.

Το δίκτυο στηρίζεται στο μοντέλο αρχιτεκτονικής λογισμικού πελάτη-εξυπηρετητή (client-server) σύμφωνα με το οποίο ο server οργανώνει και διαχειρίζεται το αρχείο δεδομένων, δέχεται ερωτήματα και απαντά στον client. Ο client με τη σειρά του θέτει ερωτήματα στο server και έχει τη δυνατότητα να αποκωδικοποιήσει τις απαντήσεις του server.

Με τον όρο Server (εξυπηρετητής) εννοούμε την υπολογιστική μηχανή εκείνη η οποία απαντάει στις αιτήσεις που γίνονται από τον

client. Οι δυνατότητες της μηχανής αυτής είναι να αποθηκεύει, να ανακτά και να προστατεύει πληροφορίες, να επιθεωρεί τις αιτήσεις των client και να διαχειρίζεται πληροφορίες.

Με τον όρο client-πελάτης εννοούμε επίσης μια υπολογιστική μηχανή (πχ. προσωπικός υπολογιστής) η οποία μπορεί να αιτηθεί υπηρεσίες από τον server και ξεκινά πάντα την επικοινωνία. Μια client μηχανή θα πρέπει να διαθέτει λογισμικό γραφικών διεπαφών χρηστών (GUIs), να δημιουργεί τις αιτήσεις για πληροφορίες και να τις στέλνει στον server και να αποθηκεύει τις πληροφορίες που επιστρέφονται .



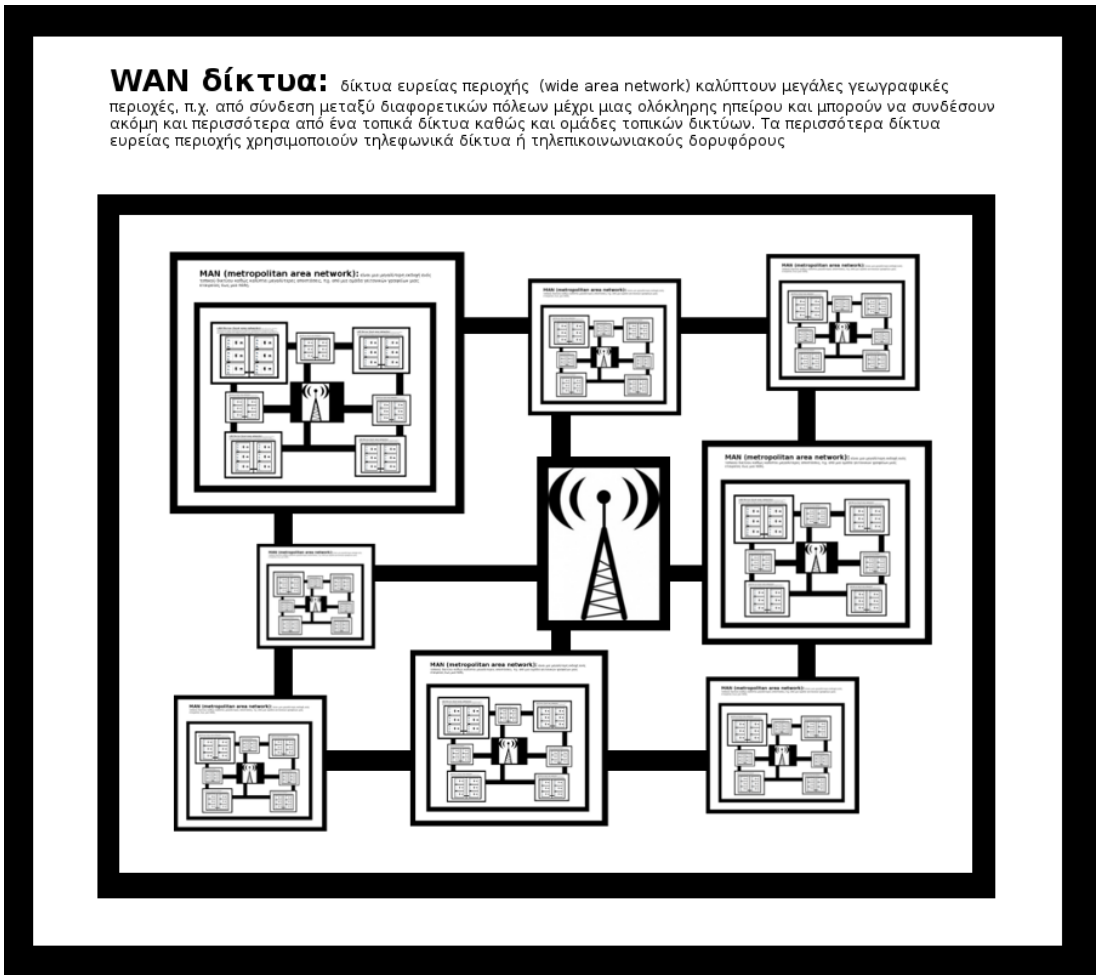
Εικόνα 24: δίκτυο υπολογιστών

Τα δίκτυα ανάλογα με την γεωγραφική τους κάλυψη κατηγοριοποιούνται ως εξής: τοπικά (LAN), μητροπολιτικά (MAN), και ευρείας κάλυψης (WAN). Σ΄ αυτό που εμείς θα εστιάσουμε περισσότερο είναι τα δίκτυα ευρείας κάλυψης WAN. Πρόκειται για δίκτυα που α) καλύπτουν μεγάλες γεωγραφικές περιοχές σε επίπεδο διαφορετικών πόλεων μέχρι μιας ηπείρου και β) συνδέουν περισσότερα από ένα τοπικά δίκτυα ή ομάδες τοπικών δικτύων. Τα WAN χρησιμοποιούν τηλεφωνικά δίκτυα ή τηλεφωνικούς δορυφόρους για τη μεταφορά των δεδομένων.

Κατηγορία του WAN αποτελεί το διαδίκτυο. Το διαδίκτυο είναι

δίκτυο ευρείας περιοχής με γεωγραφική κάλυψη μιας ή και περισσότερων ηπείρων διασυνδέοντας επιμέρους δίκτυα. Στα διαδίκτυα είναι δυνατή η συνύπαρξη διαφορετικών τεχνολογιών και λειτουργικών συστημάτων. Το Διαδίκτυο είναι το μεγαλύτερο δίκτυο τύπου διαδικτύου.

(Ο όρος διαδίκτυο γράφεται με μικρό το πρώτο γράμμα ενώ ο όρος Διαδίκτυο αναφέρεται μόνο στο συγκεκριμένο μεγαλύτερο τέτοιο είδος δικτύου.)



Εικόνα 25: δίκτυο ευρείας περιοχής WAN

Διαδίκτυο – Σύνδεση στο Διαδίκτυο μέσω Internet

Όπως αναφέραμε Διαδίκτυο είναι ένα παγκόσμιο σύστημα διασυνδεδεμένων δικτύων υπολογιστών. Για να είναι εφικτή η επικοινωνία μεταξύ υπολογιστών θα πρέπει να έχουν κοινά πρωτόκολλα επικοινωνίας, δηλ. κοινό τρόπο κωδικοποίησης των πληροφοριών έτσι ώστε οι πληροφορίες αυτές να γίνονται

κατανοητές από όλους τους υπολογιστές. Το πρωτόκολλο το οποίο χρησιμοποιεί το Διαδίκτυο είναι το TCP/IP και αποτελείται από δύο μέρη α) TCP (Transmission Control Program) το οποίο είναι υπεύθυνο για την μεταφορά των πακέτων από το ένα σημείο στο άλλο και β) το IP (Internet Protocol) το οποίο δίνει στα πακέτα συγκεκριμένες διευθύνσεις. Μια διεύθυνση IP είναι της μορφής 192.168.0.8. Οι δύο πρώτες τριάδες αριθμών (192.168) αντιπροσωπεύουν το δίκτυο ενώ οι δύο επόμενες (0.8) τον host. Κάθε μηχάνημα (υπολογιστής, κινητό, εκτυπωτής) που συνδέεται σε ένα δίκτυο ή στο Διαδίκτυο διαθέτει μια τέτοια διεύθυνση η οποία είναι μοναδική.

Το Διαδίκτυο στηρίζεται επίσης στο μοντέλο αρχιτεκτονικής λογισμικού πελάτη-εξυπηρετητή (client-server). Η λέξη client θα αντικατασταθεί με τη λέξη host και η λέξη server με τη λέξη web server.

Η πιο διαδεδομένη υπηρεσία του Διαδικτύου είναι ο Παγκόσμιος ιστός (World Wide Web). Άλλες υπηρεσίες είναι οι ακόλουθες:

- E-mail (ηλεκτρονικό ταχυδρομείο), για την ανταλλαγή ηλεκτρονικών μηνυμάτων.
- F.T.P. (File Transfer Protocol), για την ανίχνευση, τον εντοπισμό και την μεταφορά αρχείων στον υπολογιστή μας τα οποία βρίσκονται αποθηκευμένα σε εξυπηρετητές (servers) που ονομάζονται ftp servers.
- Telnet, για την πρόσβαση σε απομακρυσμένο υπολογιστή.
- IRC (Internet Relay Chat – Αναμετάδοση συζήτησης), για την πραγματοποίηση συζήτησης σε πραγματικό χρόνο με άλλους χρήστες του διαδικτύου. Όλοι είναι συνδεδεμένοι με ένα chat server.
- News Groups (Ομάδες ειδήσεων), για την επικοινωνία χρηστών μέσω πινάκων ανακοινώσεων.

Για να μπορεί κάποιος να συνδεθεί στο Διαδίκτυο θα πρέπει να απευθυνθεί αρχικά σε έναν παροχέα Internet. Οι παροχείς Internet προσφέρουν πρόσβαση σε μεμονωμένους υπολογιστές στο Διαδίκτυο μέσω τηλεφωνικής γραμμής. Για την επίτευξη της σύνδεσης πρέπει να διαθέτουμε modem ή router και το κατάλληλο λογισμικό.

World Wide Web (www) – Παγκόσμιος Ιστός

Το World Wide Web (www) - Παγκόσμιος ιστός, δημιουργήθηκε το 1989 στο CERN (EUROPEAN PARTCLE PHYSICS LABORATORY) και σχεδιάστηκε από τον Tim Berners-Lee. Πρόκειται για ένα διαδίκτυο το οποίο απαρτίζεται από δίκτυα συνδεδεμένων υπολογιστών (πχ. δίκτυα υπολογιστών επιχείρησης) σε τοπικό επίπεδο με κάποια τοπολογία και δίκτυα (πχ. εθνικά ή υπερεθνικά δίκτυα) σε παγκόσμιο επίπεδο.
Στο σημείο αυτό θα πρέπει να τονίσουμε ότι το Διαδίκτυο και το World Wide Web (η πιο διαδεδομένη υπηρεσία του Διαδικτύου) είναι ξεχωριστές αλλά και συσχετιζόμενες έννοιες. Με την χρήση του όρου Web εννοούμε τον τρόπο που έχουμε πρόσβαση στην πληροφορία μέσω του Διαδικτύου.

Το πληροφοριακό αυτό σύστημα αυτό κάνει χρήση πρωτοκόλλου επικοινωνίας το οποίο λέγεται HTTP (HYPERTEXT TRANSFER PROTOCOL – Πρωτόκολλο μεταφοράς Υπερκειμένου). Το HTTP πρωτόκολλο περιέχει κανόνες για την ανταλλαγή μηνυμάτων μεταξύ υπολογιστών, για το πώς αυτά θα πρέπει να μορφοποιηθούν και να διαβιβαστούν καθώς επίσης και τι ενέργειες θα πρέπει να γίνουν από την πλευρά του Web server (εξυπηρετητής ιστοσελίδων) και του web browser (πρόγραμμα περιήγησης Web πχ. Firefox) ώστε να έχουμε μια επιτυχημένη επικοινωνία.

Κύριο συστατικό, παράλληλα με το HTTP πρωτόκολλο, το οποίο ελέγχει το πώς το World Wide Web λειτουργεί είναι η HTML γλώσσα σήμανσης η οποία καλύπτει το φάσμα των ενεργειών για το πώς μορφοποιούνται και πως εμφανίζονται οι ιστοσελίδες στην οθόνη του υπολογιστή μας.

Καινοτομία στην τεχνολογία του Παγκόσμιου ιστού αποτελεί η δημιουργία “Υπερκειμένων” (hypertext) δηλ. η δημιουργία μιας διασύνδεσης πολλών μη ιεραρχημένων στοιχείων (links) καθώς επίσης και η δημιουργία άλλων μορφών απεικόνισης δεδομένων πέραν του κειμένου όπως εικόνα και ήχος που λέγονται “Υπερμέσα” (hypermedia).

Ο Παγκόσμιος ιστός στηρίζεται επίσης στο μοντέλο πελάτη-εξυπηρετητή (client-server). Το ρόλο του server αναλαμβάνουν λογισμικά προγράμματα εξυπηρετητή γνωστά ως Web servers (πχ. Apache) που σκοπό έχουν την οργάνωση και τη διαχείριση των πληροφοριών μέσω ιστοσελίδων. Ιστοσελίδα είναι μια εφαρμογή “Υπερμέσου”.

Πρόγραμμα περιήγησης ιστού - Web browser

Web browser είναι ένα λογισμικό που επιτρέπει στο χρήστη να προβάλλει και να αλληλεπιδρά με, κείμενα, εικόνες, βίντεο μουσική, παιχνίδια τα οποία βρίσκονται αναρτημένα σε μια ιστοσελίδα ενός ιστότοπου στον Παγκόσμιο ιστό ή σε ένα τοπικό δίκτυο. Τα στοιχεία αυτά μπορεί να περιέχουν υπερσυνδέσμους προς άλλες ιστοσελίδες του ίδιου ή διαφορετικού ιστότοπου. Σκοπός web browser του είναι να διαβάζει έγγραφα HTML και να τα συνθέτει σε ιστοσελίδες έτσι ώστε κάποιος χρήστης να μπορεί να τις διαβάσει, να ακούσει κάποιο ενσωματωμένο ήχο ή να παρακολουθήσει κάποιο βίντεο. Δεν εμφανίζει τις ετικέτες (tags), απλώς τις χρησιμοποιεί για να ερμηνεύσει το περιεχόμενο της σελίδας. Παρέχει μεθόδους δημιουργίας δομημένων εγγράφων, δηλ. εγγράφων που αποτελούνται από το περιεχόμενο που μεταφέρουν και από τον κώδικα μορφοποίησης του περιεχομένου, καθορίζοντας δομικά στοιχεία για το κείμενο όπως κεφαλίδες, παραγράφους, λίστες, συνδέσμους κ.α. Μπορούν επίσης να ενσωματώνουν σενάρια εντολών σε γλώσσες όπως η JavaScript, τα οποία επηρεάζουν την συμπεριφορά των ιστοσελίδων HTML. Πιο κάτω θα εξηγήσουμε τις έννοιες JavaScript και CSS.

Εξυπηρετητής - Web Server

Οι Web Servers είναι υπολογιστικές μηχανές που επιτρέπουν σε άλλες υπολογιστικές μηχανές να έχουν πρόσβαση στα αρχεία που ο web browser διαχειρίζεται με τη βοήθεια του HTTP πρωτοκόλλου. Ο διαχειριστής του web server έχει τον απόλυτο έλεγχο της υπολογιστικής αυτής μηχανής ενώ ο απλός χρήστης, περιορίζεται μόνο, στο να "ζητήσει" (request) από τον web server να εμφανίσει μια συγκεκριμένη ιστοσελίδα. Υπάρχουν δύο κατηγοριών web server, οι dedicated servers οι οποίοι ασχολούνται αποκλειστικά με τον έλεγχο των ιστοσελίδων (Google, Yahoo) και οι integrated servers οι οποίοι ασχολούνται εκτός από τον έλεγχο των ιστοσελίδων και με άλλους σκοπούς.

Τρόπος λειτουργίας

Ένας Web server θα πρέπει να είναι ικανός να ανταποκρίνεται σε εκατοντάδες αιτήματα κάθε στιγμή αν και αυτό εξαρτάται από

πολλούς παράγοντες όπως το υλικό-hardware του server, την επισκεψιμότητα (traffic) των σελίδων κα.
Παράλληλα με το υλικό (hardware), δηλ. το σύνολο των φυσικών εξαρτημάτων ενός υπολογιστή σημαντικό ρόλο παίζει και το λογισμικό (software) δηλ. το σύνολο των προγραμμάτων υπολογιστών, διαδικασιών και οδηγιών χρήσης που εκτελούν ορισμένες εργασίες σε ένα υπολογιστικό σύστημα. Ο συνδυασμός αυτών των δύο μας δίνει ως αποτέλεσμα τη δημιουργία ενός ισχυρού web server.

Ο μηχανισμός που χρησιμοποιείται προκειμένου να "ανέβει" (upload) μια ιστοσελίδα στην οθόνη του υπολογιστή μας είναι ο ακόλουθος

- στον υπολογιστή μας τρέχει ένας web browser (Mozilla Firefox)
- ο web browser συνδέεται με τον server και αιτείται πρόσβαση σε μια ιστοσελίδα (www.teiep.gr)
- ο server λειτουργεί ως web server
- ο web server στέλνει στον υπολογιστή την αιτούμενη σελίδα

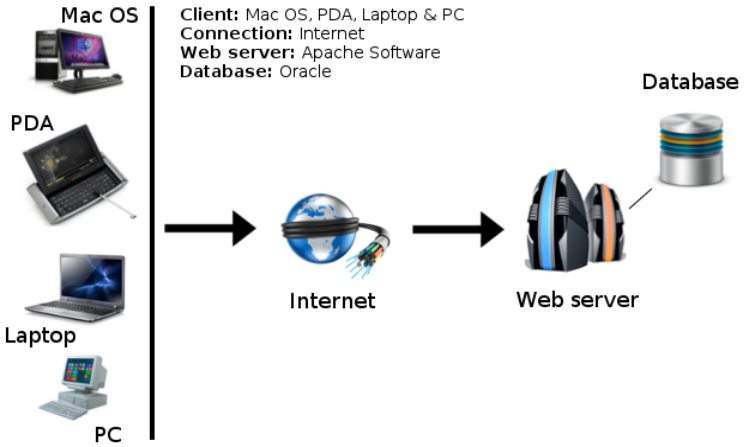
Εικόνα 26: μηχανισμός που χρησιμοποιείται προκειμένου να "ανέβει" (upload) μια ιστοσελίδα στην οθόνη του υπολογιστή

Web Server (Apache)

Ο Apache είναι ένας εξυπηρετητής του παγκόσμιου ιστού. Κάθε φορά που ένας χρήστης επισκέπτεται έναν ιστότοπο, το πρόγραμμα περιήγησης (Firefox) επικοινωνεί με έναν εξυπηρετητή (web server) μέσω του HTTP πρωτοκόλλου, ο οποίος παράγει τις ιστοσελίδες και τις αποστέλλει στον web browser.
Ο web server Apache είναι ένας ανοιχτού κώδικα εξυπηρετητής HTTP συμβατός με όλα τα λειτουργικά συστήματα (Win και Unix). Στόχος του είναι να παρέχει ένα ασφαλές, αποτελεσματικό και επεκτάσιμο web server που παρέχει υπηρεσίες HTTP. Ο Apache σχεδιάστηκε από την Apache Software Foundation (1996). Πρόκειται για έναν από τους δημοφιλέστερους web servers.

Για να γίνει εφικτή η δημιουργία ενός web server στον προσωπικό μας υπολογιστή θα πρέπει να ανοίξουμε μια θύρα (port) στο router του υπολογιστή μας για τη δημιουργία επικοινωνίας. Με τον όρο router (δρομολογητής) εννοούμε μια ηλεκτρονική συσκευή που σκοπός της είναι η αποστολή και η λήψη πακέτων δεδομένων μεταξύ ενός ή περισσοτέρων εξυπηρετητών (server), άλλων δρομολογητών και πελατών (clients).

Αυτό που τελικά προκύπτει είναι ότι η χρήση κάποιου προγράμματος περιήγησης θα μας δώσει τη δυνατότητα να μπούμε στο router και να ανοίξουμε μια θύρα επικοινωνίας. Μέσω του internet και μέσω του υπολογιστή μας όλοι οι χρήστες που θα μπαίνουν στην σελίδα μας (clients) θα μπορούν να μπαίνουν από οποιαδήποτε υπολογιστική μηχανή πχ. Laptop, PDA, Mac OS, κανονικοί υπολογιστές.



Εικόνα 27: Apache είναι ένας εξυπηρετητής του παγκόσμιου ιστού

HTML γλώσσα σήμανσης - Markup Language

Η HTML είναι ακρωνύμιο των όρων Hypertext Markup Language και είναι η κύρια γλώσσα σήμανσης για την κατασκευή ιστοσελίδων. Γράφεται υπό μορφή στοιχείων τα οποία αποτελούνται από ετικέτες-σημάνσεις (tags). Πρόκειται για δομικά στοιχεία που εκτελούν εργασίες όπως η μορφοποίηση κειμένου, η ενσωμάτωση πολυμέσων, διαδραστικές εφαρμογές κα. Οι ετικέτες περικλείονται μέσα σε αγκύλες (<ετικέτα>). Ορισμένες ετικέτες εργάζονται κατά ζεύγη, χρειάζονται δηλαδή αρχική σήμανση - <ετικέτα> - και τελική σήμανση - </ετικέτα> -. Παράδειγμα: <title>Hello HTML</title>.

Ο web browser είναι υπεύθυνος για τη διαβίβαση εγγράφων .html και η σύνθεσή τους σε ιστοσελίδες. Η γραφή του HTML κώδικα γίνεται μέσω λογισμικών προγραμμάτων επεξεργασίας HTML (HTML editors) εκ των οποίων τα πιο διαδεδομένα είναι το Notepad++, Geany κα. για Windows, Linux κα. Οι περισσότερο χρησιμοποιούμενοι Web browser είναι ο Mozilla Firefox, Google Chrome, Apple Safari κα.

Παραδείγματα ετικετών (tags)

Δηλώσεις

<!DOCTYPE html>: δήλωση html εγγράφου.
<html></html>: αντιπροσωπεύει το root (ρίζα) ενός html εγγράφου.
<head></head>: αντιπροσωπεύει μια συλλογή από metadata για το html έγγραφο.

Οι ακόλουθες ετικέτες μπορούν να συμπεριληφθούν ανάμεσα στο <head>.
<title>name the browser bar</title>: καθορίζει τον τίτλο του html εγγράφου.
<style type=''text/css''>body {background-color:yellow} p {color:blue}</style>: καθορίζει πληροφορίες για το στυλ του html εγγράφου.
<link rel=''stylesheet'' type=text/css'' href=mystyle.css''>: καθορίζει την σύνδεση του html με άλλες πηγές. Εδώ συνδέεται με τις μορφοποιήσεις του κειμένου με τη χρήση css.
<script></script>: επιτρέπει στους συγγραφείς να συμπεριλάβουν στον κώδικά τους δυναμικά σενάρια εντολών και δεδομένα.

Σώμα ιστοσελίδας (body)

Οι ακόλουθες ετικέτες μπορούν επίσης να συμπεριληφθούν ανάμεσα στο <body>.

<!-- comment -->: σχόλια.
<h1>heading 1</h1>: καθορίζει επικεφαλίδα (μεγάλο).
<h2>heading 2</h2>
<h3>heading 3</h3>
<h4>heading 4</h4>
<h5>heading 5</h5>: καθορίζει επικεφαλίδα (μικρό).
<hr>a line</hr>: εμφανίζει μια οριζόντια γραμμή στην HTML σελίδα μας.

line brakes</br>: αλλαγή γραμμής.
a paragraph : καθορίζει μια παράγραφο.
bold: έντονη γραφή.
italic: πλάγια γραφή.

a link: δημιουργία συνδέσμου.
a link in a new blank: δημιουργία συνδέσμου σε νέο παράθυρο.

<img scr=''photo.jpg'' width=''250'' height=''400''>: εισαγωγή εικόνας από αρχείο και διαστάσεις.
<img scr=''url'' alt=''name photo''>: εισαγωγή εικόνας από το διαδίκτυο.

Εισαγωγή φορμών

1. <map name="planetmap">
<area shape="rect" coords="0,0,82,126" href="sun.htm" alt="Sun">
<area shape="circle" coords="90,58,3" href="mercur.htm" alt="Mercury">
<area shape="circle" coords="124,58,8" href="venus.htm" alt="Venus">
</map>: δημιουργία χάρτη.

2. <table border="1">
<tr>
<td>row 1, cell 1</td>
<td>row 1, cell 2</td>

</tr>
<tr>
<td>row 2, cell 1</td>
<td>row 2, cell 2</td>
</tr>
</table>: δημιουργία πίνακα.

3. list: καθορίζει μια λίστα.

Coffee
Milk
: δημιουργία μη ταξινομημένης λίστας.

Coffee
Milk
: δημιουργία ταξινομημένης λίστας.

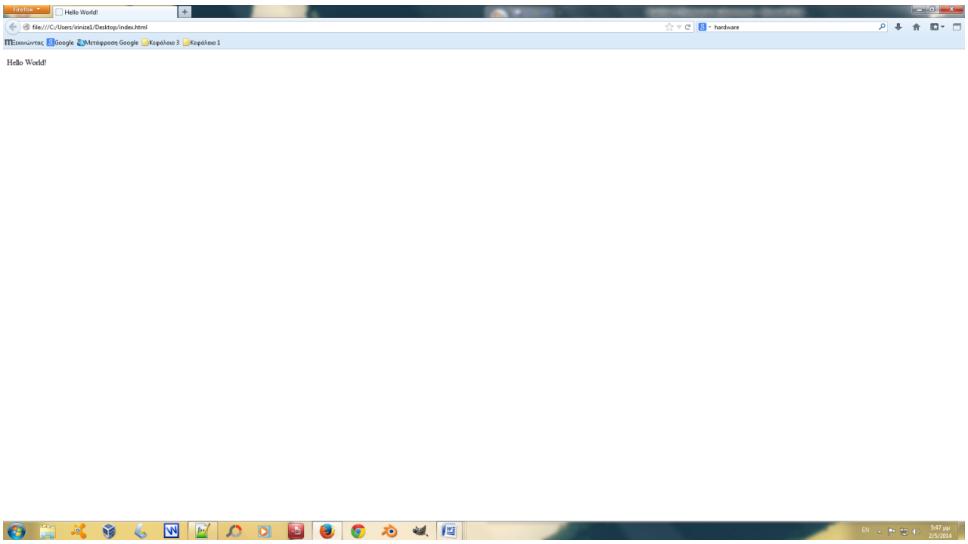
<dl>
<dt>Coffee</dt>
<dd>- black hot drink</dd>

<dt>Milk</dt>
<dd>- white cold drink</dd>
</dl>: εναλλάσσει την εσοχή των αντικειμένων της λίστας.

Παράδειγμα HTML

Το παρακάτω παράδειγμα είναι ένα κλασικό παράδειγμα Hello World! που χρησιμοποιεί εννέα (9) γραμμές κώδικα.

```
<!DOCTYPE html>
<html>
        <head>
            <title>Hello World!</title>
        </head>
    <body>
         <p>Hello World!</p>
    </body>
</html>
```



Εικόνα 27: απεικόνιση .html εγγράφου (screenshot)

HTML5

Η HTML 5 είναι η πέμπτη κατά σειρά αναθεώρηση του προτύπου HTML. Δημιουργήθηκε από την συνεργασία μεταξύ World Wide Web Consortium (W3C) και της Web Hypertext Application Technology Working Group (WHATWG). Η εξελιγμένη αυτή εκδοχή της HTML παρέχει δυνατότητες που στηρίζονται σε HTML, CSS, DOM και JavaScript. Μειώνει τις ανάγκες για εξωτερικά plug-in (πχ. Flash), παρέχει εύκολο χειρισμό λαθών κα.

Μερικά από τα νέα χαρακτηριστικά της HTML5 είναι το στοιχείο <canvas>, τα στοιχεία <video>, <audio> για την αναπαραγωγή πολυμέσων, νέο περιεχόμενο ετικετών όπως <article>, <footer>, <header>, <nav>, <section> καθώς και νέα στοιχεία φόρμας όπως το ημερολόγιο, η ημερομηνία, η ώρα, το e-mail και url.

CSS

Όπως προαναφέραμε οι Web Browsers μπορούν να αναφέρονται σε στυλ μορφοποίησης CSS για να ορίσουν την εμφάνιση και την διάταξη του κειμένου.

Ειδικότερα, το CSS είναι μια υπολογιστική γλώσσα που χρησιμοποιείται για τον έλεγχο της εμφάνισης ενός εγγράφου που έχει γραφτεί σε μια γλώσσα σήμανσης html. Αναπτύχθηκε προκειμένου να αναπτύσσει στιλιστικά μια ιστοσελίδα, δηλαδή να διαμορφώνει χαρακτηριστικά όπως το χρώμα, τη στοίχιση κ.α.

Επισκεφτείτε την διεύθυνση http://www.w3.org/Style/CSS/ για περισσότερες πληροφορίες.

Index.html κώδικας παραδείγματος HTML5 εγγράφου

Παρακάτω παρουσιάζεται ο index.html και ο main.css κώδικας για τη δημιουργία ιστοσελίδας με την χρήση της HTML5.

Index κώδικας

```
<!doctype html>
<html lang="en">
<head>
		<meta charset="utf-8" />
		<title>Website</title>
		<link rel="stylesheet" href="main.css"
</head>

<body>
	<div id="big_wrapper">
		<header id="top_header">
			<h1>Name</h1>
		</header>

<nav id="top_menu">
		<ul>
			<li>Home</li>
			<li>My work</li>
			<li>Contact</li>
		</ul>
</nav>
<section id="main_section">
	<article>
		<header>
			<hgroup>
```

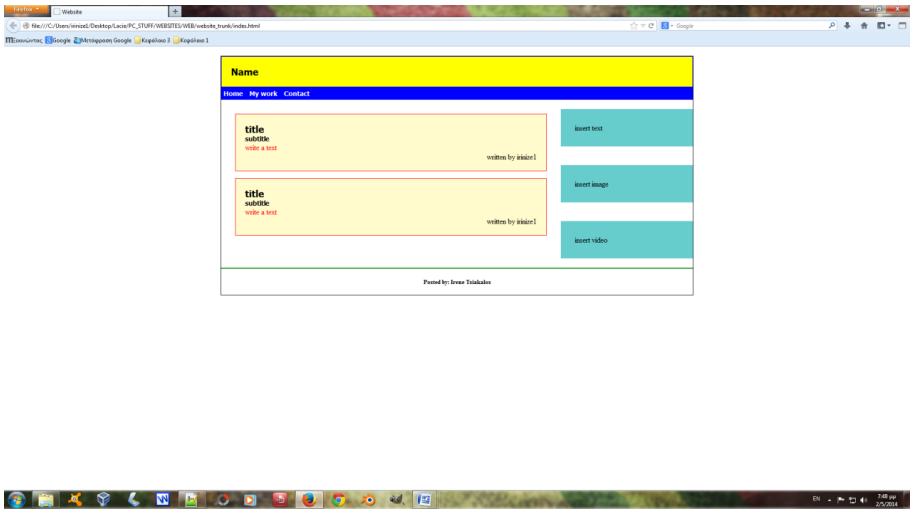
```
                    <h1>title</h1>
                    <h2>subtitle</h2>
                </hgroup>
            </header>
                <p id="tuna">write a text </p>
                <footer>
                    <p>written by irinize1</p>
                </footer>
        </article>
        <article>
            <header>
                <hgroup>
                    <h1>title</h1>
                    <h2>subtitle</h2>
                </hgroup>
            </header>
                <p id="tuna">write a text </p>
                <footer>
                    <p>written by irinize1</p>
                </footer>
        </article>
</section>

<aside id="side_news">insert text</aside>
<aside id="side_news">insert image</aside>
<aside id="side_news">insert video</aside>

<footer id="the_footer">
            <p><h5>Posted by: Irene Tsiakalos</h5></p>
</footer>
</div>

</body>

</html>
```

CSS κώδικας

```
#tuna{color:red;}

*{
```

```
        margin:       0px;
        padding: 0px;
}
h1 {
        font: bold 20px Tahoma;
}
h2 {
        font: bold 14px Tahoma;
}
header, section, footer, aside, nav, article, hgroup{
        display:block;
}
body{
        text-align:center;
}
#big_wrapper{
        border:       1px solid black;
        width:1000px;
        margin: 20px auto;
        text-align:left;
}
#top_header{
        background:yellow;
        border: 1px solid blue;
        padding: 20px;
}
#top_menu {
        background-color:blue;
        color:white;
}
#top_menu li{
        display:inline-block;
        list-style:none;
        padding: 5px;
        font: bold 14px Tahoma;
}
#main_section{
        float: left;
        width:660px;
        margin:30px; /*720px, 280 left*/
}
#side_news{
```

```
        float: left;
        width: 220px;
        margin: 20px 0px;
        padding: 30px;
        background: #66CCCC;
}
#the_footer{
        clear: both;
        text-align:center;
        padding:20px;
        border-top: 2px solid green;
}
article {
        background: #FFFBCC;
        border: 1px solid red;
        padding: 20px;
        margin-bottom: 15px;
}
article footer{
        text-align:right;
}
```



Εικόνα 28: απεικόνιση ιστοσελίδας με τη χρήση της HTML5 (screenshot)

ΒΙΒΛΙΟΓΡΑΦΙΑ

ΕΙΣΑΓΩΓΗ

ΙΣΤΟΣΕΛΙΔΑ
http://webcache.googleusercontent.com/search?q=cache:gZrtsymAM-gJ:aetos.it.teithe.gr/~praptis/CG/Graphics01%2520-%2520Introduction.pdf+&cd=3&hl=el&ct=clnk&gl=gr

ΚΕΦΑΛΑΙΟ 1ο

ΒΙΒΛΙΑ
Tay Vaughan (απόδοση Γιάννης Β. Σαμαράς), Πολυμέσα: Αναλυτικός οδηγός Έκδοση 7η , Εκδόσεις Μ. Γκιούρδας, 2008

ΗΛΕΚΤΡΟΝΙΚΑ ΒΙΒΛΙΑ
Beginning WebGL for HTML5, Brian Danchilla
http://learningwebgl.com/blog/

ΙΣΤΟΣΕΛΙΔΕΣ
http://html5.org/
http://www.khronos.org/
http://www.json.org/
http://www.ietf.org/rfc/rfc4627.txt
http://www.fileformat.info/format/wavefrontobj/egff.htm
http://notepad-plus-plus.org/
http://people.sc.fsu.edu/~jburkardt/data/obj/obj.html.
https://github.com/mrdoob/three.js
http://threejs.org/

ΚΕΦΑΛΑΙΟ 2ο

ΙΣΤΟΣΕΛΙΔΕΣ
https://www.microsoft.com/enus/download/search.aspx?q=directx
http://www.nvidia.com/Download/index.aspx?lang=en-us
http://www.browserchoice.eu/BrowserChoice/browserchoice_el.htm
http://threejs.org/
http://sourceforge.net/projects/xampp/
http://portforward.com/english/routers/port_forwarding/routerindex.htm
http://www.blender.org/
https://github.com/mrdoob/three.js/tree/master/utils/exporters/blender
http://people.sc.fsu.edu/~jburkardt/data/obj/obj.html

ΠΑΡΑΡΤΗΜΑ

ΒΙΒΛΙΑ

Σπυριδούλα Μαργαρίτη-Ελευθέριος Σταύρου. Τοπικά & αστικά δίκτυα Έκδοση 1η, Εκδόσεις Νέων Τεχνολογιών, 2007
Andrew S. Tanenbaum, Computer Networks Έκδοση 4η, Εκδόσεις Κλειδάριθμος, 2003
William Stallings, Data and Computer Communications Έκδοση 6η, Εκδόσεις Τζιόλα, 2008
Douglas E. Comer, Internetworking with TCP/IP Έκδοση 4η, Εκδόσεις Κλειδάριθμος, 2001

ΙΣΤΟΣΕΛΙΔΕΣ

http://www.apache.org/