ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΕΦΑΡΜΟΣΜΕΝΩΝ ΜΑΘΗΜΑΤΙΚΩΝ & ΦΥΣΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΟΜΕΑΣ ΦΥΣΙΚΗΣ

ΦΥΣΙΚΗ ΥΨΗΛΩΝ ΕΝΕΡΓΕΙΩΝ ΣΕ ΠΟΛΥΜΕΣΙΚΟ ΠΕΡΙΒΑΛΛΟΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΧΡΗΣΤΟΣ ΓΡΗΓΟΡΑΣ

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ: ΝΙΚΟΛΑΟΣ ΤΡΑΚΑΣ

ΑΘΗΝΑ, ΙΟΥΛΙΟΣ 2012

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΕΦΑΡΜΟΣΜΕΝΩΝ ΜΑΘΗΜΑΤΙΚΩΝ & ΦΥΣΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΟΜΕΑΣ ΦΥΣΙΚΗΣ

ΦΥΣΙΚΗ ΥΨΗΛΩΝ ΕΝΕΡΓΕΙΩΝ ΣΕ ΠΟΛΥΜΕΣΙΚΟ ΠΕΡΙΒΑΛΛΟΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΧΡΗΣΤΟΣ ΓΡΗΓΟΡΑΣ

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ: ΝΙΚΟΛΑΟΣ ΤΡΑΚΑΣ

ΕΓΚΡΙΘΗΚΕ ΑΠΟ ΤΗΝ ΤΡΙΜΕΛΗ ΕΠΙΤΡΟΠΗ ΣΤΙΣ 19 ΙΟΥΛΙΟΥ 2012

Νικόλαος Τράκας	Κωνσταντίνος Φαράκος	Κων/νος Αναγνωστόπουλος
Καθηγητής Ε.Μ.Π.	Αν. Καθηγητής Ε.Μ.Π.	Επ. Καθηγητής Ε.Μ.Π.

ΑΘΗΝΑ, ΙΟΥΛΙΟΣ 2012

Περιεχόμενα

ΠΕΡΙΛΗΨΗ

Σκοπός της εργασίας είναι η κατασκευή μιας διαδικτυακής εφαρμογής, για την καλύτερη κατανόηση και απλούστευση της βασικής λειτουργίας ενός ανιχνευτή σωματιδίων, με τη βοήθεια κινούμενων γραφικών, με χρήση HTML5, Javascript, CSS. Επιπρόσθετα, θα αναλυθεί ο τρόπος δημιουργίας κινούμενων γραφικών, καθώς και ορισμένες χρήσιμες λειτουργίες των παραπάνω γλωσσών προγραμματισμού, ώστε να αποτελέσουν βάση για την ανάπτυξη δυναμικών ιστοσελίδων.

Στο Κεφάλαιο 1 παρουσιάζεται ο Ανιχνευτής Σωματιδίων, τα κύρια μέρη του, και περιγράφεται η κύρια λειτουργία της εφαρμογής.

Στο Κεφάλαιο 2 παρέχονται πληροφορίες για το νέο πρότυπο HTML5, για τη δημιουργία κινούμενων γραφικών με χρήση JavaScript, και για τα CSS.

Στο Κεφάλαιο 3 αναλύεται ο μηχανισμός και ο κώδικας της εφαρμογής, μέσω των απαιτούμενων αρχείων.

Τέλος, στο Κεφάλαιο 4 αναπτύσσεται η σημασία της εκλαΐκευσης της Φυσικής.

Λέξεις Κλειδιά: HTML5, JavaScript, CSS, Ανιχνευτής Σωματιδίων, Κινούμενα Γραφικά, Εκλαΐκευση Φυσικής

ABSTRACT

The objective of the present study is the creation of online animated graphics, in order to simplify the understanding of the functionality and procedures taking place in a Particle Detector, using HTML5, JavaScript & CSS. In addition, the basic way of creating animated graphics will be analyzed, and some useful functions will be presented, in order to assist in the further development of dynamic web applications.

In Chapter 1 the Particle Detector and its layers are introduced, as well as the functionality of our web application.

In Chapter 2 information are provided about the new Standard of HTML5, and the basic pattern of building graphic animations, using JavaScript & CSS.

The code of the application is analyzed in Chapter 3, along with the connections between the application's files.

The importance of the outreach of Physics is the subject of Chapter 4.

Keywords: HTML5, JavaScript, CSS, Particle Detector, Animated Graphics, Outreach of Physics

ΠΡΟΛΟΓΟΣ

Με την εκπόνηση της παρούσας διπλωματικής εργασίας στον τομέα Φυσικής της σχολής Εφαρμοσμένων Μαθηματικών και Φυσικών Επιστημών, του Εθνικού Μετσόβιου Πολυτεχνείου, ολοκληρώνονται οι σπουδές μου ώς προπτυχιακός φοιτητής.

Αρχικά θα ήθελα να ευχαριστήσω τον Καθηγητή Ε.Μ.Π. κύριο Νικόλαο Τράκα, για την ανάθεση της συγκεκριμένης διπλωματικής εργασίας και τη συνεχή καθοδήγησή του, καθ'όλη τη διάρκεια της εκπόνησής της. Επίσης, θα ήθελα να ευχαριστήσω τον κύριο Μιχάλη Ψήτο για την φιλοξενία της ιστοσελίδας στον εξυπηρετητή positeam.net, και για τις πολύτιμες συμβουλές του.

Σε προσωπικό επίπεδο, θα ήθελα να ευχαριστήσω την οικογένεια μου για την συνολική συμπαράσταση και ενθάρρυνση κατά τη διάρκεια των σπουδών μου.

Κεφάλαιο 1:
Ο Ανιχνευτής Σωματιδίων

Ο ανιχνευτής σωματιδίων έχει ως σκοπό να καταγράφει και να οπτικοποιεί τις εκρήξεις σωματιδίων που προκύπτουν από τις συγκρούσεις μέσα σε επιταχυντές σωματιδίων. Οι πληροφορίες που θα συλλεχθούν, σχετικά με την ταχύτητα, τη μάζα, και το ηλεκτρικό φορτίο, βοηθούν τους επιστήμονες να ανακαλύψουν την ταυτότητα του σωματιδίου.

Οι σύγχρονες συσκευές ανίχνευσης σωματιδίων αποτελούνται από στρώματα υποανιχνευτών, κάθε ένα από το οποίο ανιχνεύει συγκεκριμένο είδος ή ιδιότητα σωματιδίου. Υπάρχουν 3 κύριοι τύποι υποανιχνευτών:

- Ανιχνευτές Τροχιών
- Καλορίμετρα
- Ανιχνευτές Μιονίων

1. Ανιχνευτές Τροχιών

Οι ανιχνευτές τροχιών αποκαλύπτουν τις διαδρομές ηλεκτρικά φορτισμένων σωματιδίων μέσα από τα ίχνη που αυτά αφήνουν πίσω τους, όπως για παράδειγμα, τα ίχνη που αφήνουν στην ατμόσφαιρα τα αεροσκάφη που δεν είναι ορατά εξαιτίας του μεγάλου ύψους πτήσης. Παρόμοια, όταν σωματίδια διαπερνούν ιονισμένα αέρια, με ταχύτητες που

πλησιάζουν την ταχύτητα φωτός, τότε μπορεί να αποκαλυφθεί η αλληλεπίδραση του σωματιδίου με τα άτομα του αερίου.

Οι περισσότερες σύγχρονες συσκευές ανίχνευσης τροχιών δεν αποτυπώνουν άμεσα τις τροχιές των σωματιδίων. Αντίθετα, παράγουν ηλεκτρικά σήματα που καταγράφονται σαν δεδομένα, και στη συνέχεια ένα πρόγραμμα ανακατασκευάζει τις τροχιές και τις εμφανίζει στην οθόνη.

Η εφαρμογή της εργασίας αναπαριστά γραφικά τη λειτουργία ενός ανιχνευτή τροχιών.

2. Θερμιδόμετρα(Καλορίμετρα)

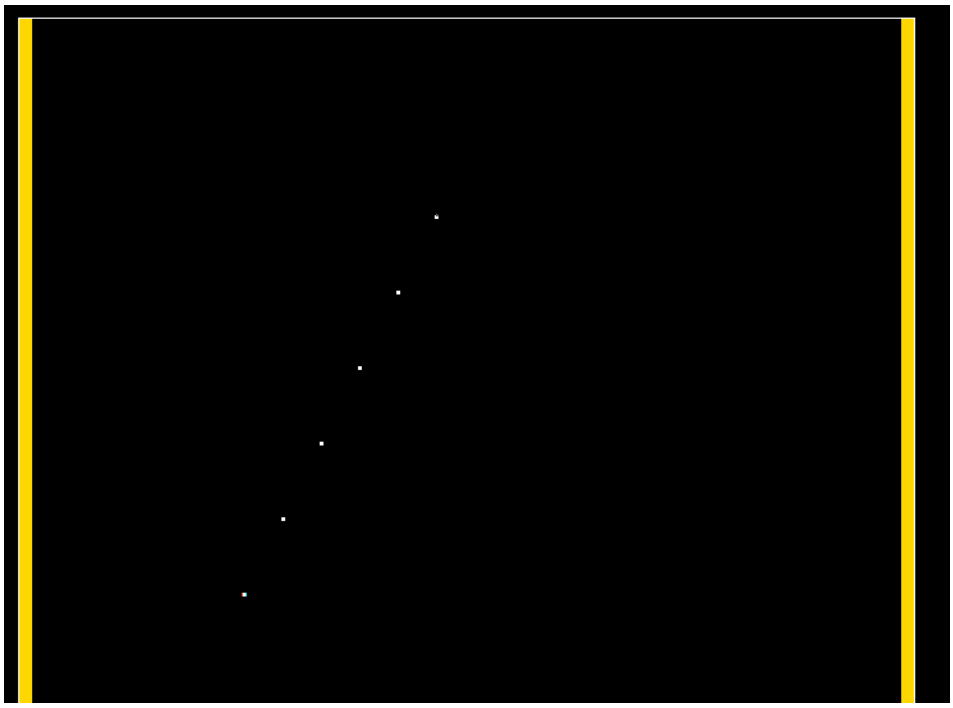
Υπολογίζουν την ενέργεια των ουδέτερων και φορτισμένων σωματιδίων. Συχνά τα θερμιδόμετρα σταματούν τα σωματίδια (απορροφούν την ενέργειά τους). Διαφορετικά είδη θερμιδομέτρων χρησιμοποιούνται για ηλεκτρόνια και φωτόνια και διαφορετικά για πρωτόνια, νετρόνια και πιόνια ή άλλα σωματίδια φτιαγμένα από quark (αδρόνια).

3. Ανιχνευτές Μιονίων

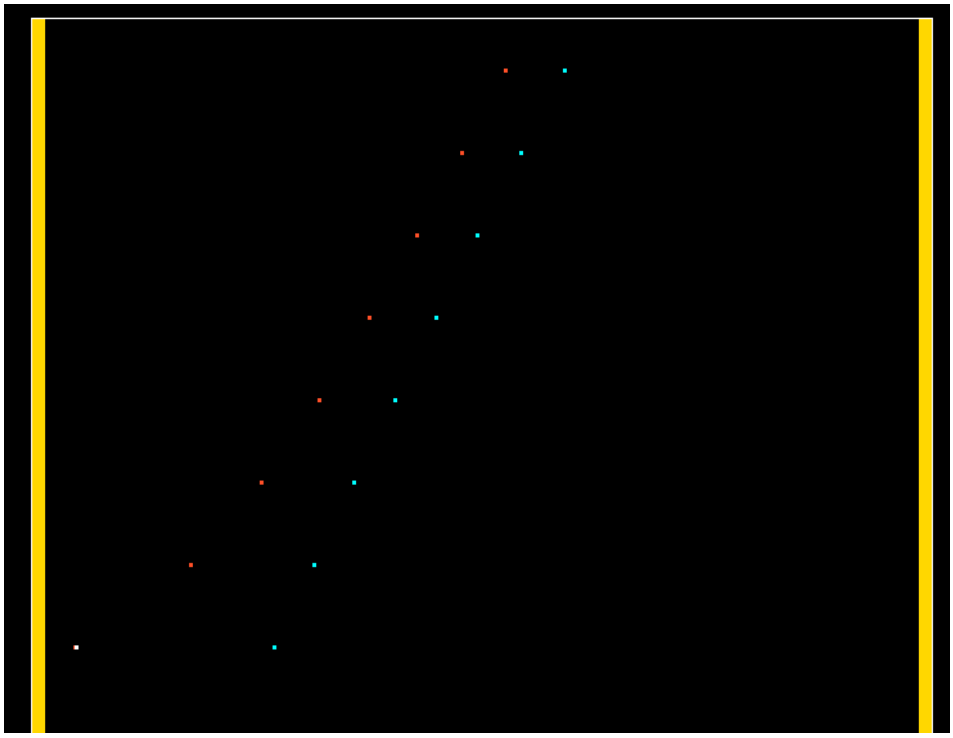
Οι θάλαμοι μιονίων είναι ανιχνευτές τροχιών εξειδικευμένοι να ανιχνεύουν μιόνια. Αυτά τα σωματίδια αλληλεπιδρούν ελάχιστα με την ύλη, και μπορούν να ταξιδεύουν μεγάλες αποστάσεις, διαπερνώντας αρκετά μέτρα πυκνής ύλης. Τα μιόνια διαπερνούν τα διαδοχικά στρώματα των ανιχνευτών σωματιδίων. Οι θάλαμοι μιονίων συνήθως αποτελούν το τελευταίο από τα στρώματα.

4. Ο ανιχνευτής τροχιών σε πολυμεσικό περιβάλλον

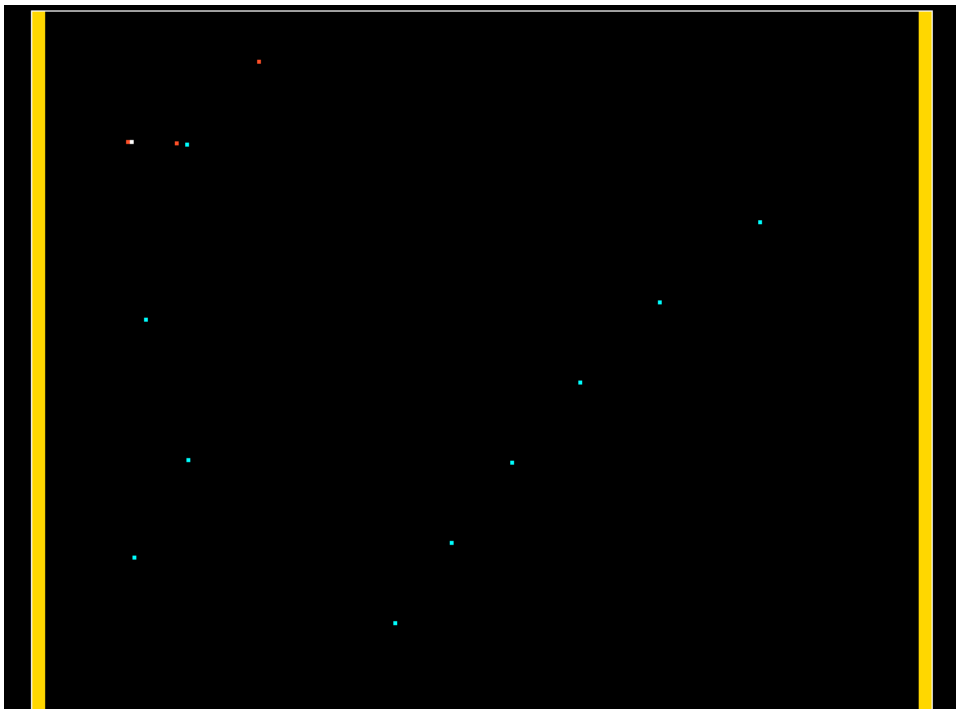
Η εφαρμογή που θα αναπτυχθεί έχει σαν σκοπό να οπτικοποιήσει το πείραμα που διεξάγεται σε έναν ανιχνευτή τροχιών. Ένα σωματίδιο διαπερνά, με ταχύτητα που προσεγγίζει αυτήν του φωτός, ένα θάλαμο με εύκολα ιονίσιμο αέριο, προκαλώντας τη διάσπαση ορισμένων ατόμων σε ηλεκτρόνια και θετικά ιόντα(πορτοκαλί και μπλε αντίστοιχα), όπως φαίνεται στο σχήμα 1.1. Στη συνέχεια, ηλεκτρικά φορτισμένα σύρματα (οι κίτρινες κάθετες γραμμές) έλκουν τα ηλεκτρόνια και τα θετικά ιόντα. Τα ηλεκτρόνια, καθώς κινούνται, προκαλούν με τη σειρά τους επιπλέον διασπάσεις των ατόμων του αερίου (Φαινόμενο Avalanche) όπως απεικονίζονται στο σχήμα 1.3. Οι μετρήσεις του ρεύματος που καταγράφονται, δίνουν τη δυνατότητα να οπτικοποιηθεί στη συνέχεια η τροχιά του σωματιδίου, και να εξαχθούν περαιτέρω συμπεράσματα(Σχήμα 1.4). Για λόγους οικονομίας υπολογιστικών πόρων, οι ηλεκτρικές δυνάμεις που αναπτύσσονται στην εφαρμογή είναι μόνο ελκτικές, και μόνο μεταξύ ηλεκτρικά φορτισμένων γραμμών και αντίστοιχων ιόντων.



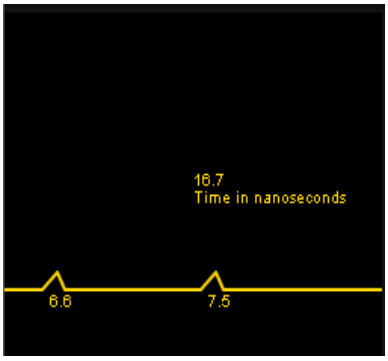
Σχήμα 1.1



Σχήμα 1.2



Σχήμα 1.3



Σχήμα 1.4

Κεφάλαιο 2:
HTML 5, JavaScript, CSS

Η εφαρμογή βασίζεται στο συνδυασμό της γλώσσας σήμανσης HTML5, με τον έλεγχο της εμφάνισης μέσω της αλληλουχίας φύλλων στυλ CSS, και της αντικειμενοστραφούς γλώσσας προγραμματισμού JavaScript.

1. HTML5

Η HTML5 (Hypertext Markup Language) είναι το νέο πρότυπο για την γλώσσα HTML. Προέκυψε από τη συνεργασία των World Wide Web Consortium (W3C) και Web Hypertext Application Technology Working Group (WHATWG), προκειμένου να δημιουργήσουν μια νέα έκδοση της γλώσσας HTML, λαμβάνοντας υπόψη τις σημαντικές αλλαγές και ανάγκες στη δημιουργία ιστοσελίδων και εφαρμογών, που δημιουργήθηκαν μετά από την τελευταία έκδοση της γλώσσας το 1999(HTML 4.01).

Ορισμένοι από τους κανόνες πάνω στους οποίους βασίζεται η πέμπτη αυτή σημαντική αναθεώρηση είναι:

- Τα νέα χαρακτηριστικά θα πρέπει να βασίζονται στις HTML, CSS, DOM, JavaScript
- Μείωση της ανάγκης για χρήση εξωτερικών πρόσθετων (π.χ. Flash)
- Καλύτερη αντιμετώπιση σφαλμάτων
- Περισσότερη σήμανση (markup) προς αντικατάσταση σύνθετων κειμένων κώδικα
- Δημοσίευση διαδικασίας ανάπτυξης

Ενδιαφέροντα Νέα Χαρακτηριστικά

- Το στοιχείο <canvas> - Σχεδιασμός 2D γραφικών (κυρίως με χρήση JavaScript)
- Τα στοιχεία <video> & <audio> - Αναπαραγωγή Πολυμέσων
- Υποστήριξη για τοπική αποθήκευση δεδομένων
- Στοιχεία ειδικού περιεχομένου - <article>, <footer>, <header>, <nav>, <section>

Συμβατότητα Περιηγητών

Η HTML5 δεν είναι ακόμα επίσημο πρότυπο, και κανένας περιηγητής δεν είναι πλήρως συμβατός, όμως συνεχώς αυξάνεται ο αριθμός των νέων χαρακτηριστικών που υποστηρίζονται από τις τελευταίες εκδόσεις των περιηγητών.

2. JavaScript

Η JavaScript αποτελεί μια ευρέως διαδεδομένη γλώσσα προγραμματισμού ιστοσελίδων και εφαρμογών, με το σημαντικό πλεονέκτημα ότι ο κώδικας εκτελείται στον υπολογιστή του επισκέπτη της ιστοσελίδας (client-side).

Συνήθως, η δημιουργία κινούμενων γραφικών στο στοιχείο <canvas> της HTML5 γίνεται με τη χρήση της αντικειμενοστραφούς JavaScript, και απαιτούνται τα εξής βήματα:

```
function animate(){

    update(); // Ανανέωση των δεδομένων( π.χ. Ενημέρωση για τη νέα θέση των γραφικών)

    clear(); // Καθαρισμός του στοιχείου <canvas> από προηγούμενο σχέδιο

    draw(); // Σχεδιασμός των γραφικών, βάση των νέων δεδομένων που προέκυψαν με τη μέθοδο update();

    request_new_frame(); // Αίτηση νέου πλαισίου(frame)

    animate(); // Ανάκληση της λειτουργίας , για το καινούριο frame

}
```

3. CSS

Με τη χρήση της CSS, ορίζονται οι ιδιότητες διάταξης και εμφάνισης των στοιχείων της σελίδας. Ο συνδυασμός με JavaScript μπορεί να αποφέρει εντυπωσιακά αποτελέσματα, ενώ ήδη υπάρχουν έτοιμες βιβλιοθήκες και διεπαφές προγραμματισμού εφαρμογών (API), που ενσωματώνονται εύκολα σε μια ιστοσελίδα. Στην εφαρμογή μας έχουν χρησιμοποιηθεί οι βιβλιοθήκες JQuery, και YUI. Το ειδικό εφέ που εμφανίζεται με κλικ στο “Toggle”, αποτελεί παράδειγμα χρήσης των παραπάνω βιβλιοθηκών.

Κεφάλαιο 3:
Ο Κώδικας της Εφαρμογής

Στη συνέχεια, η δομή της ιστοσελίδας μας θα αναλυθεί στα επιμέρους στοιχεία και λειτουργίες της. Κομμάτια κώδικα που βρίσκονται μέσα σε σχόλια, έχουν σαν σκοπό να αποτελέσουν βοήθημα ή παράδειγμα για ανάπτυξη παρόμοιων εφαρμογών, έστω και αν δεν χρησιμοποιούνται στην παρούσα εφαρμογή.

1. Το αρχείο **index.html**

Το βασικό αρχείο index.html, όπως υποδηλώνει το όνομά του, αποτελεί το ευρετήριο της ιστοσελίδας. Εδώ, περιλαμβάνονται οι τοποθεσίες των JavaScript & CSS αρχείων .js και .css αντίστοιχα, ενώ ορίζονται και τα html στοιχεία της ιστοσελίδας μας.

```
<!DOCTYPE html> <!--Doctype Declaration in HTML5-->
<html>

   <meta property="og:image" content="" />
   <meta property="og:title" content="High Energy Physics  -
Particle Detector Animation" />
   <meta property="og:type" content="Animation" />
   <meta property="og:url"
content="http://positeam.net/HEPME2/HEPME3/" />
   <meta property="og:description" content="" />
   <meta property="fb:admins" content="" />
      <title>High Energy Physics  - Particle Detector
Animation</title>
      <link rel="stylesheet" type="text/css"
href="http://yui.yahooapis.com/3.3.0/build/cssreset/reset-
min.css">
      <link rel="stylesheet" type="text/css"
href="http://ajax.googleapis.com/ajax/libs/jqueryui/1.8.13/themes/
ui-darkness/jquery-ui.css"/>
```

```
      <link rel="stylesheet" type="text/less"
href="particleController.css"/>
      <script
src="http://ajax.googleapis.com/ajax/libs/jquery/1.6.1/jquery.min.
js" type="text/javascript" charset="utf-8"></script>
      <script
src="https://ajax.googleapis.com/ajax/libs/jqueryui/1.8.13/jquery-
ui.min.js" type="text/javascript" charset="utf-8"></script>
      <script src="less-1.1.3.min.js"       type="text/javascript"
charset="utf-8"></script>
    <script src="Display.js"          type="text/javascript"
charset="utf-8"></script>
      <script src="DisplayCurrent.js"
type="text/javascript" charset="utf-8"></script>
    <script src="FieldLine.js"            type="text/javascript"
charset="utf-8"></script>
    <script src="NegativeFieldLine.js"
type="text/javascript" charset="utf-8"></script>
    <script src="ParticleEmitter.js"  type="text/javascript"
charset="utf-8"></script>
      <script src="Particle.js"         type="text/javascript"
charset="utf-8"></script>
    <script src="AvalancheParticle.js"
type="text/javascript" charset="utf-8"></script>
    <script src="Ion.js"         type="text/javascript"
charset="utf-8"></script>
      <script src="AvalancheEmitter.js"  type="text/javascript"
charset="utf-8"></script>
      <script src="ParticleSystem.js"   type="text/javascript"
charset="utf-8"></script>
    <script src="CurrentSystem.js"   type="text/javascript"
charset="utf-8"></script>
    <script src="IonCurrentSystem.js"   type="text/javascript"
charset="utf-8"></script>
      <script src="Field.js"            type="text/javascript"
charset="utf-8"></script>
    <script src="NegativeField.js"
type="text/javascript" charset="utf-8"></script>
    <script src="particleController.js"
type="text/javascript" charset="utf-8"></script>
      <script src="Util.js"             type="text/javascript"
charset="utf-8"></script>
    <script type="text/javascript"
src="https://apis.google.com/js/plusone.js"></script>

</head>

<body>
<div>
<div id="header"><h1 style="font-size:24px; text-
align:center;">High Energy Physics, using HTML5 - Javascript -
CSS</h1>
<p>The animation below represents the procedures taking place in a
particle detector, in order to detect, track, and/or identify
```

```
high-energy particles, such as those produced by nuclear decay,
cosmic radiation, or reactions in a particle accelerator.
</p>
</div>

      <div id="container">
            <span><canvas id="display" width="700"
height="576"></canvas></span>
      </div>

      <div id="controlContainer1" class="controlContainer">
      <div class="showHideControls">
         <p>Toggle</p>
      </div>
         <div class="controls" id="welcomeMessage">

      <p>The Current Meters below provide details about the
current created during the absorption of the ions and the
electrons by the electrically charged lines.!</p>
<p></p>
<p style="text-align:center">
</p>
<p>
   <!--Based on
http://jarrodoverson.com/static/demos/particleSystem.html-->
</p>

         </div>
       <canvas id="current" width="250" height="200"></canvas>
       <canvas id="ionCurrent" width="250" height="200"></canvas>
      </div>

      <div id="controlContainer2" class="controlContainer">
      <div class="showHideControls">
         <p>Toggle Controls</p>
      </div>
    <div class="controls" id="displayControls" style=""><p>This is
some text.</p></div>

      </div>
      <div id="footer">
         <button id="save">Save Locally</button>
         <button id="load">Load Locally</button>
         -
         <button class="loadExample" id="example1">Clear</button>

      </div>
      <div id="floatingControls" class="closable">
      <button id="closeFloatingControls"></button>
      <button id="deleteObject"></button>
         <div id="variableControl"></div>
      </div>
   <div id="error">
      <p>
```

```
        Your browser is a little out of date to view this demo.
Please update your browser or try with the latest versions of any
of the following :
        <br><br><br><br>
        <a href="http://www.google.com/chrome/">Chrome</a><br>
        <a href="http://www.mozilla.com/en-
US/firefox/fx/">Firefox</a><br>
        <a href="http://www.apple.com/safari/">Safari</a><br>
        <a href="http://windows.microsoft.com/en-US/internet-
explorer/products/ie/home">Internet Explorer</a>
      </p>
      <button onclick="$(this).closest('#error').hide();">Close
Anyway</button>
   </div>
</div>
</html>
```

2. Το αρχείο **particleController.js**

Στο αρχείο particleController.js αρχικοποιείται η εφαρμογή μας. Εδώ αποδίδεται στα στοιχεία <canvas> με id:display / displayCurrent / ionDisplayCurrent τα αντικείμενα Display / DisplayCurrent, όπου περιγράφονται γενικές ιδιότητες σχεδιασμού και κίνησης γραφικών. Στη συνέχεια δημιουργούνται τα αντικείμενα CurrentSystem, IonCurrentSystem, ParticleSystem, όπου υπάρχουν εξειδικευμένες ιδιότητες και λειτουργίες κίνησης. Ακόμα, δημιουργούνται αντικείμενα αλληλεπίδρασης, όπως τα κουμπιά Save, Clear, Load, και αντιστοιχίζονται εργασίες σε συμβάντα όπως το πάτημα πλήκτρων

```
var display = null;
var displayCurrent=null;
var ionDisplayCurrent=null;
var particleSystem = null;
var currentSystem = null;
var globalTime;
var examples = {
example1 : '0,basic:Sv1(1000|1000|1|0|0|1|1)'
```

```
};
var particleStyle = 'basic';
var programmaticUpdate = false;
var currentSystem = new CurrentSystem();
var ionCurrentSystem = new IonCurrentSystem();
var newGlobalDate = new Date();
var   newGlobalTime = newGlobalDate.getTime();
$(function(){
    display = new Display(document.getElementById('display'));
      display.init();
      displayCurrent = new
DisplayCurrent(document.getElementById('current'));
      displayCurrent.init();
      ionDisplayCurrent = new
DisplayCurrent(document.getElementById('ionCurrent'));
      ionDisplayCurrent.init();
      particleSystem = new ParticleSystem().init(display);

      var listeners = {
      /*onObjectBlur : hideFloatingControls,
      onObjectClick : objectClicked,
      onObjectMouseIn : objectMouseIn,
      onObjectMouseOut : objectMouseOut,
      onObjectFinishMove : updateHash,*/
      onError : onError
      }
   particleSystem.addListener('error',listeners);
  /* particleSystem.addListener('objectBlur',listeners);
   particleSystem.addListener('objectClick',listeners);
   particleSystem.addListener('objectMouseIn',listeners);
   particleSystem.addListener('objectMouseOut',listeners);
   particleSystem.addListener('objectFinishMove',listeners);
*/
   display.start();
   displayCurrent.start();
   ionDisplayCurrent.start();

      bindKeys();
      registerButtons();
      try {
            addEmitter();
     addFieldLine();
     addNegativeFieldLine();
      loadFromHash();
   } catch(e) {
      addEmitter();
     addFieldLine();
     addNegativeFieldLine();
   }

   $(window).bind('hashchange',loadFromHash);
});

function loadFromHash() {
```

```
    if (!programmaticUpdate) {
        loadState(location.hash.substr(1));
    }
    programmaticUpdate = false;
}

function updateHash() {
    programmaticUpdate = true;
    location.hash = getState();
}

function bindKeys() {
      $(window).keypress(function(evt) {
          switch (evt.which) {
              case 32 : displayPause();return false /*prevents page
scrolling*/; break;
              case 115 : displayStep(); break;
              case 101 : addEmitter(); break;
              case 102 : addFieldLine(); break;
              case 110 : addNegativeFieldLine();break;
                case 105 : toggleInfo(); break;
              case 111 : toggleObjects(); break;
              case 112 : toggleParticles(); break;
              case 118 : toggleVelocities(); break;
              case 97 : toggleAccelerations(); break;
              case 99 : displayClear(); break;

          }
      });
}

function registerButtons() {
    $('button').button();
    $('.showHideControls').click(function(){$
(this).siblings('.controls').slideToggle()});
    $('#clear').click(displayClear);
    $('#addEmitter').click(addEmitter);
    $('#addField').click(addField);
    $('#addNegativeField').click(addNegativeField);
    $('#addNegativeFieldCurrent').click(addNegativeFieldCurrent);
    $('#addFieldLine').click(addFieldLine);
    $('#addNegativeFieldLine').click(addNegativeFieldLine);
    $('#save').click(saveState);
    $('#load').click(function(){loadState()});
    $('.loadExample').click(function(){loadState(examples[$
(this).attr('id')])});
    $('#closeFloatingControls').button({icons:{primary:"ui-icon-
close"},text:false}).click(function(){$
(this).closest('.closable').css('display','none');});
    $('#deleteObject').button(          {icons:{primary:'ui-icon-
trash'},text:false});
    $('#clickBehavior').buttonset();
    $('#clickBehavior1').click(function()
{particleSystem.mouseFieldStrength = -200});
```

```
   $('#clickBehavior2').click(function()
{particleSystem.mouseFieldStrength = 200});
   $('#clickBehavior3').click(function()
{particleSystem.mouseFieldStrength = 0});
   $('#strings').click(function(){
      display.draw.info = false;
        displayCurrent.draw.info = false;
        displayCurrent.clear();
      display.clear();
      display.draw.continuous = !display.draw.continuous;
   });

   $('#startStop').click(displayPause);
   $('#step').click(displayStep);
   $('#objects').click(toggleObjects);
   $('#info').click(toggleInfo);
   $('#accelerations').click(toggleAccelerations);
   $('#velocities').click(toggleVelocities);
   $('#particles').click(toggleParticles);
   $('#maxParticles > button').click(function()
{particleSystem.maxParticles = $(this).val();updateHash();});
   $('#particleStyle > button').click(function()
{changeParticleDrawStyle($(this).val());updateHash();});
   $('#fbShare').click(function(){
         var url = location;
         var encodedUrl = encodeURI(url);
         console.log(encodedUrl);
         var shareUrl = "http://www.facebook.com/share.php?u=";
         window.open (shareUrl + encodedUrl,
"fbShare","status=1,toolbar=0,location=1,menubar=0,directories=0,r
esizable=1,scrollbars=0,height=250,width=500");
   });
}
function addEmitter() {
   particleSystem.addEmitter(new Point(display.width / 6,
display.height ),new Vector(0.01,0));
   currentSystem.pulseParticle();
   ionCurrentSystem.pulseIon();
   updateHash();
}


function addFieldLine() {
       particleSystem.addFieldLine(new Point(0   , 0),1000,
display.height);
     updateHash();
}

function addNegativeFieldLine() {

   particleSystem.addNegativeFieldLine(new Point(display.width   ,
0),1000, display.height);
   updateHash();
}
```

```
function toggleInfo(evt)          {toggleDrawable(display,"info")}
function toggleObjects(evt)
{toggleDrawable(particleSystem,"objects")}
function toggleParticles(evt)
{toggleDrawable(particleSystem,"particles")}
function toggleVelocities(evt)
{toggleDrawable(particleSystem,"velocities")}
function toggleAccelerations(evt)
{toggleDrawable(particleSystem,"accelerations")}

function toggleDrawable(obj,drawable) {
   obj.draw[drawable] = !obj.draw[drawable];
   updateLabels();
}

function updateLabels() {
   var i = -1, drawable =
['accelerations','objects','particles','velocities'];
   while(drawable[++i]) {
      $('#' + drawable[i]).button("option","label",
(particleSystem.draw[drawable[i]] ? "Hide " : "Show ") +
drawable[i]);
   }
   $('#info').button("option","label", display.draw.info ? "Hide
info" : "Show info");
   $("#startStop").button("option","label", display.paused ?
"Start" : "Stop");
   updateHash();
}
function displayPause() {
   display.togglePause();
   updateLabels();
}
function displayStep() {
   display.step()
   updateLabels();
}
function displayClear(){
   display.clear();
   particleSystem.particles=[];
   particleSystem.ions=[];
}

function changeParticleDrawStyle(style) {
   particleStyle = style || 'basic';
   var styles = {
      basic : function() {
            Particle.GLOBAL_DRAW_COLOR = [255,80,40,255];
         Particle.prototype.draw = Particle.prototype.drawBasic;
      },
      variable : function() {
//          Particle.GLOBAL_DRAW_COLOR = [66,167,222,255];
            Particle.GLOBAL_DRAW_COLOR = [66,167,180,255];
```

```
            Particle.prototype.draw =
Particle.prototype.drawVariable;

        },
        fancy : function() {
              Particle.GLOBAL_DRAW_COLOR = [166,67,0,255];
            Particle.prototype.draw = Particle.prototype.drawSoft;
        }
    };
    if (styles[particleStyle]) {
        styles[particleStyle]();
    } else {
        styles.basic();
    }
    updateHash();
}

function updateAngle(object,angle) {
    object.velocity = Vector.fromAngle(angle * Math.PI /
180,object.velocity.getMagnitude());
}

function updateSpread(object,angle) {
    object.spread = angle * Math.PI / 180;
}

function updateVelocity(object, magnitude) {
    magnitude = magnitude || .1;
    object.velocity =
Vector.fromAngle(object.velocity.getAngle(),magnitude);
}

function updateMass(object, mass) {
    object.setMass(mass);
    object.cacheColor = object.drawColor;
}

function objectMouseIn(evt,thrower) {
    focusObject(evt.particleTarget);
    display.canvas.style.cursor = 'pointer !important';
}

function objectMouseOut(evt,thrower) {
    focusObject(evt.particleTarget,true);
    display.canvas.style.cursor = null;
}

function focusObject(object,unfocus) {
    if (unfocus) {
        object.drawColor = object.cacheColor ||
object.constructor.drawColor;
        $(object).animate({size : object.cacheSize},100,'linear');
        object.size = object.cacheSize;
    } else {
```

```
      if (!object.cacheSize) {
         object.cacheSize = object.size;
      }
      if (!object.cacheColor) {
         object.cacheColor = object.drawColor;
      }
      $(object).animate({size : 15},100,'linear');
      object.drawColor = "#990";
   }
}

function deleteObject(object) {
   if (object.constructor === Field) {
      particleSystem.removeField(object);
   } else if (object.constructor === ParticleEmitter) {
      particleSystem.removeEmitter(object);
   } else if (object.constructor === NegativeField) {
      particleSystem.removeNegativeField(object);
   }
   //updateHash();
}

function objectClicked(evt,thrower) {
   var object = evt.particleTarget;
   if (object.constructor === ParticleEmitter) {
      var html = ''+
         "<div>Angle</div>" +
         "<div class='angleSlider slider'></div>" +
         "<div>Speed</div>" +
         "<div class='velocitySlider slider'></div>" +
         "<div>Spread</div>" +
         "<div class='spreadSlider slider'></div>";
      $('#variableControl').html(html);
      $('.angleSlider').each(function(i,el) {
         var angle = object.velocity.getAngleDegrees().toFixed(0)
* Math.PI / 180;
         console.log(angle);
         $(el).slider({
            value : angle.toFixed(0),
            min : -180,
            max : 180,
            stop : updateHash,
            slide : function(evt,ui){updateAngle(object,ui.value)}
         });
      });
      $('.velocitySlider').each(function(i,el) {
         var magnitude =
object.velocity.getMagnitude().toFixed(0);
         $(el).slider({
            value : magnitude,
            min : 0,
            max : 3,
            step : .2,
            stop : updateHash,
```

```
            slide : function(evt,ui)
{updateVelocity(object,ui.value)}
         });
      });
      $('.spreadSlider').each(function(i,el) {
         var angle = object.spread / Math.PI * 180;
         $(el).slider({
            value : angle,
            min : 0,
            max : 180,
            stop : updateHash,
            slide : function(evt,ui)
{updateSpread(object,ui.value)}
         });
      });   } else if (object.constructor === Field||
object.constructor === NegativeField) {
      var html = ''+
         "<div>Strength</div>" +
         "<div class='massSlider slider'></div>";
      $('#variableControl').html(html);
      $('.massSlider').each(function(i,el) {
         $(el).slider({
            value : object.mass,
            min : -1000,
            max : 1000,
            stop : updateHash,
            create : function(evt,ui){
               $(this)
                  .removeClass('fieldRepel fieldAttract')
                  .addClass(object.mass >= 0 ? 'fieldAttract' :
'fieldRepel');
            },
            slide : function(evt,ui){
               $(this)
                  .removeClass('fieldRepel fieldAttract')
                  .addClass(ui.value >= 0 ? 'fieldAttract' :
'fieldRepel');
               updateMass(object,ui.value);
            }
         });
      });
   }
   $('#floatingControls').css('left' ,display.canvas.offsetLeft +
object.position.x+ object.size/2).css('top',
display.canvas.offsetTop + object.position.y +
object.size/2).show();
   $('#deleteObject').unbind().click(function(evt)
{deleteObject(object);hideFloatingControls();});
}

function hideFloatingControls() {
   $('#floatingControls').hide();
}
```

```
function saveState() {
   localStorage.setItem('systemState',getState());
}

function getState() {
   stateString = getCustomState() + ':' +
particleSystem.toString();
   return stateString;
}

function loadState(stateString) {
   stateString = stateString ||
localStorage.getItem('systemState') || loadState(examples[0]);
   var separatorPos = stateString.indexOf(':');
   loadCustomState(stateString.substr(0,separatorPos));
   stateString = stateString.substr(separatorPos+1);
   particleSystem.fromString(stateString);
   displayClear();
   display.start();
   updateLabels();
}

function loadCustomState(string) {
   var parts = string.split(',');
   display.draw.continuous = parts[0] === '1' ? true : false;
   changeParticleDrawStyle(parts[1]);
}
function getCustomState() {
   var parts = [
      (display.draw.continuous ? '1' : '0'),
      particleStyle
   ];
   return parts.join(',');
}
function onError() {
   $('#error').show();
}
```

3. Το αρχείο **ParticleSystem.js**

Εδώ, δημιουργείται το σύστημα σωματιδίων, ορίζονται οι μέθοδοι προσθήκης(και αφαίρεσης) σωματιδίων, ηλεκτρονίων και θετικών ιόντων στην εφαρμογή μας, κάποιες μέθοδοι αλληλεπίδρασης με το χρήστη, μέθοδοι σχεδιασμού των σωματιδίων/ηλεκτρονίων/φορτισμένων άπειρων γραμ-

μών. Ακόμα, ορίζεται η καταγραφή/αποθήκευση των παραμέτρων της εφαρμογής στον σύνδεσμο URL , αλλά και η αρχικοποίηση της εφαρμογής βάση των στοιχείων που αναγράφονται στο URL.

```
function ParticleSystem(){
   var self = this;
   self.maxParticles = 100;
   self.maxIons = self.maxParticles;
   self.startTime    = 0;
   self.draw = {
      objects       : true,
      accelerations : false,
      velocities    : false,
      particles     : true,
        ions      : true
   };
   self.particles = [];
   self.ions = [];
   self.emitters  = [];
   self.fields    = [];
   self.negative_fields    = [];
   self.positiveFieldLines = [];
   self.negativeFieldLines = [];
   self.listeners  = {};
   self.elapsed   = 0;
   self.lastEmitter = 0;
   self.mouseCoords = new Point(0,0);
   self.mouseFieldStrength = -140;
   self.mouseField;


   self.init = function(display) {
      display.addListener('draw',self);
      display.addListener('afterDraw',self);
      display.addListener('beforeUpdate',self);
      display.addListener('update',self);
      display.addListener('mouseUp',self);
      display.addListener('mouseDown',self);
      display.addListener('mouseMove',self);
      return self;
   }

      self.addListener = function(eventName, object) {
         if (!self.listeners[eventName]) {
            self.listeners[eventName] = [];
         }
         self.listeners[eventName].push(object);
      }

      self.fireEvent = function(eventName,evt) {
         if (self.listeners[eventName] &&
```

```
self.listeners[eventName].length > 0) {
         var eventMethod = "on" +
eventName.substr(0,1).toUpperCase() + eventName.substr(1);
            for (var i = 0; i < self.listeners[eventName].length;
i++) {
               if (self.listeners[eventName][i][eventMethod]) {
                  self.listeners[eventName][i]
[eventMethod].call(self.listeners[eventName][i],evt,self);
               }
            }
         }
      }

   self.addEmitter = function(point,velocity){
      var emitter = new ParticleEmitter(point, velocity);
      self.emitters.push(emitter);
      self.fireEvent('newObject', {particleTarget : emitter});
        emitter.move();
      var globalDate = new Date();
   globalTime= globalDate.getTime();

   };

   self.removeEmitter = function(index){
      if (typeof index.constructor !== Number)  index =
self.emitters.indexOf(index);
      var success = self.emitters.splice(index,1);
      if (success) {
         self.fireEvent('deleteObject',{particleTarget:success});
      }
   };


   self.addFieldLine = function(point,mass){
      var field = new FieldLine(point, mass);
      self.positiveFieldLines.push(field);
      self.fireEvent('newObject',{particleTarget : field});
   };

   self.addNegativeFieldLine = function(point,mass, height){
      var field = new NegativeFieldLine(point, mass, height);
      self.negativeFieldLines.push(field);
      self.fireEvent('newObject',{particleTarget : field});
   };

   self.removeFieldLine = function(point,mass){
      if (typeof index.constructor !== Number)  index =
self.positiveFieldLines.indexOf(index);
      var success = self.positiveFieldLines.splice(index,1);
      if (success) {
         self.fireEvent('deleteObject',{particleTarget:success});
      }
   };
```

```
   self.removeNegativeFieldLine = function(point,mass){
      if (typeof index.constructor !== Number)  index =
self.negativeFieldLines.indexOf(index);
      var success = self.negativeFieldLines.splice(index,1);
      if (success) {
         self.fireEvent('deleteObject',{particleTarget:success});
      }
   };


   self.onBeforeUpdate = function(evt, display) {
      if (self.draw.accelerations)
self.drawAccelerations(display);
      if (self.draw.velocities)    self.drawVelocities(display);
   };

   self.onUpdate = function(evt,display) {
      self.elapsed++;
        self.plotEmitters(display.width,display.height);
      self.addNewParticles();
        self.addNewIons();
        self.plotParticles(display.width,display.height);
        self.plotIons(display.width,display.height);
   };

   self.onDraw = function(evt,display) {
      if (self.draw.particles) {
         self.drawParticles(display);
      }
        if (self.draw.ions) {
         self.drawIons(display);
      }
      if (self.draw.objects) {
         self.drawFields(display);
             self.drawNegativeFields(display);
         self.drawEmitters(display);
             self.drawFieldLines(display);
             self.drawNegativeFieldLines(display);
      }
   };

   self.onAfterDraw = function(evt, display) {
      if (display.draw.info) {
         display.fillStyle("white");
         display.drawText("Particles : " + self.getParticleCount()
+" Ions : "+self.getIonCount()+" Emitters :
"+self.getEmitterCount(), new Point(100, display.height-10), 100);
      }
   };

   self.onMouseDown = function(evt,display) {
      var object = self.getObjectAtPoint(self.mouseCoords);
      if (self.selected) {
         evt.particleTarget = self.selected;
```

```
            self.fireEvent('objectBlur',evt);
            self.selected = undefined;
         }
         if (object) {
            self.clicked = object;
            evt.particleTarget = object;
            self.fireEvent('objectMouseDown');
         } else {
           // self.mouseField = new FieldLine(self.mouseCoords,
self.mouseFieldStrength);
            //self.mouseField.size = 0;
            //self.positiveFieldLines.push(self.mouseField);
         }
      };

      self.onMouseUp = function(evt) {
         var currentObject = self.getObjectAtPoint(self.mouseCoords);
         if (self.mouseField) {
            self.removeFieldLine(self.mouseField);
            self.mouseField = undefined;
         } else if (self.clicked) {
            evt.particleTarget = self.clicked;
            if (currentObject === self.clicked) {
               if(self.clicked.moved) {
                  self.fireEvent('objectFinishMove',evt);
               } else {
                  self.selected = self.clicked;
                  self.fireEvent('objectClick',evt);
                  self.fireEvent('objectFocus',evt);
               }
               delete self.clicked.moved;
               self.clicked = undefined;
            }
         }
      };

      self.onMouseMove = function(evt,display) {
         self.mouseCoords = new Point(evt.offsetX || (evt.layerX -
display.canvas.offsetLeft), evt.offsetY || (evt.layerY -
display.canvas.offsetTop));
         if (self.mouseField) {
            self.mouseField.moveTo(self.mouseCoords);
         } else if (self.clicked) {
            self.clicked.moved = true;
            self.clicked.moveTo(self.mouseCoords);
         } else { // not over anything
            var object = self.getObjectAtPoint(self.mouseCoords);
            if (self.objectMouseOver !== object) { // if we're over
something different
               if (self.objectMouseOver) {         // if we were over
something before
                  evt.particleTarget = self.objectMouseOver;
                  self.fireEvent('objectMouseOut',evt);
                  self.objectMouseOver = undefined;
```

```
            } else {                                // we're in
*something* new, even if it's nothing
               evt.particleTarget = object;
               self.fireEvent('objectMouseIn',evt);
               self.objectMouseOver = object;
            }
         }
      }
   }

   self.addNewParticles = function() {
         for (var i = 0, emitter; emitter = self.emitters[i]; i++)
{
            for (var j = 0; j < emitter.emissionRate; j++) {
               self.particles.push(emitter.addParticle());
            }
         }

   }

   self.addNewAvalancheParticles = function(particle) {

                  self.particles.push(particle.addParticle());


   }

   self.addNewIons = function() {
         for (var i = 0, emitter; emitter = self.emitters[i]; i++)
{
            for (var j = 0; j < emitter.emissionRate; j++) {
               self.ions.push(emitter.addIon());
            }
         }

   }

   self.addNewAvalancheIons = function(particle) {

                              self.ions.push(particle.addIon());

   }

   self.plotEmitters = function(boundsX,boundsY){
         var oldEmitters = self.emitters;
         var updatedEmitters = [];
         var emitter;
            while(emitter = oldEmitters.pop()){
                  if(emitter.ttl>0){
                        if(++emitter.lived >= emitter.ttl) {
                        continue; // emitter dies.
                              }
                        }
```

```
                        emitter.move();
        var p = emitter.position;
            if ( p.x < 0 || p.x > boundsX || p.y < 0 || p.y >
boundsY) {
                // goodbye particle
                } else {
                   updatedEmitters.push(emitter);
                }
            }
            self.emitters = updatedEmitters;
   };


   self.plotParticles = function(boundsX,boundsY) {
      var oldParticles = self.particles;
      var fields = self.positiveFieldLines;
      var updatedParticles = [];
      var particle;
      while(particle = oldParticles.pop()) {
            if (particle.ttl > 0) {
                  if (++particle.lived >= particle.ttl) {
                        continue; // particle dies.
                   }
            }
      particle.submitToFields(fields);
      particle.move();

      if((particle.velocity.getMagnitude()>3)&&Math.random()>0.95)
      {
            self.addNewAvalancheParticles(particle);
            self.addNewAvalancheIons(particle);
      }
         var p = particle.position;
             if ( p.x < 0 || p.x > boundsX || p.y < 0 || p.y >
boundsY) {
                // goodbye particle
                } else {
                   updatedParticles.push(particle);
                }
            }
            self.particles = updatedParticles;
   };

   self.plotIons = function(boundsX,boundsY) {
      var oldIons = self.ions;
      var fields = self.negativeFieldLines;
      var updatedIons = [];
      var ion;
            while(ion = oldIons.pop()) {
                  if (ion.ttl > 0) {
                     if (++ion.lived >= ion.ttl) {
                        continue; // particle dies.
                     }
                  }
```

```
            ion.submitToFields(fields);
                    ion.move();
            var p = ion.position;
                if ( p.x < 0 || p.x > boundsX || p.y < 0 || p.y >
boundsY) {
               // goodbye particle
                    } else {
                       updatedIons.push(ion);
                    }
               }
               self.ions = updatedIons;
      };

      self.drawParticles = function(display) {
         var imageData =
display.context.getImageData(0,0,display.width,display.height);
         var pixels = imageData.data;
         var width = display.width;
         var particle, i = -1;
         while(particle = self.particles[++i]){
               particle.draw(pixels,display.width,display.height);
         }
               display.context.putImageData(imageData,0,0);
      };

      self.drawIons = function(display) {
         var imageData =
display.context.getImageData(0,0,display.width,display.height);
         var pixels = imageData.data;
         var width = display.width;
         var ion, i = -1;
         while(ion = self.ions[++i]){
                  ion.draw(pixels,display.width,display.height);
         }
               display.context.putImageData(imageData,0,0);
      };

      self.drawAccelerations = function(display) {
         display.strokeStyle("red");
         display.context.beginPath();
               for( var i = 0, l = self.particles.length; i < l; i+
+ ){
                     var particle = self.particles[ i ];
                     display.context.moveTo(particle.position.x,
particle.position.y);
                     display.context.lineTo(particle.position.x +
particle.acceleration.x, particle.position.y +
particle.acceleration.y);
               }
               display.context.stroke();
      };

      self.drawVelocities = function(display) {
         display.strokeStyle("blue");
```

```
      display.context.beginPath();
            for( var i = 0, l = self.particles.length; i < l; i+
+ ){
                  var particle = self.particles[ i ];
                  display.context.moveTo(particle.position.x,
particle.position.y);
                  display.context.lineTo(particle.position.x +
particle.velocity.x, particle.position.y + particle.velocity.y);
            }
            display.context.stroke();
   };

   self.drawFields = function(display) {

            for( var i = 0, l = self.fields.length; i < l; i++ ){
                  self.drawCircularObject(display,self.fields[i]);
            }
   };

   self.drawNegativeFields = function(display) {

            for( var i = 0, l = self.negative_fields.length; i <
l; i++ ){

      self.drawCircularObject(display,self.negative_fields[i]);
            }
   };

   self.drawEmitters = function(display) {
            for( var i = 0, l = self.emitters.length; i < l; i++ )
{

      self.drawCircularObject(display,self.emitters[i]);
            }
   };
   self.drawFieldLines = function(display) {
            for( var i = 0, l = self.positiveFieldLines.length; i
< l; i++ ){

      self.drawLinearObject(display,self.positiveFieldLines[i]);
            }
   };
   self.drawNegativeFieldLines = function(display) {
            for( var i = 0, l = self.negativeFieldLines.length; i
< l; i++ ){

      self.drawLinearObject(display,self.negativeFieldLines[i]);
            }
   };
   self.drawCircularObject = function(display,object) {
      var halfSize = object.size >> 1;
      var gradient = display.context.createLinearGradient(
         object.position.x - halfSize,
         object.position.y - halfSize,
```

```
            object.position.x + halfSize,
            object.position.y + halfSize
         );
         gradient.addColorStop(0, object.drawColor ||
object.constructor.drawColor);
         gradient.addColorStop(1, object.drawColor2 ||
object.constructor.drawColor2);
         display.fillStyle(gradient);
         display.drawCircle(object.position, halfSize);
      };
      self.drawLinearObject = function(display,object) {
         var halfSize = object.size >> 1;
           var endPoint = new Point(object.position.x,
display.height)
        display.drawLine(object.position, endPoint);
      };

      self.getObjectAtPoint = function(point) {
               for( var i = 0; i < self.emitters.length; i++ ){
                     var emitter = self.emitters[i];
                     if (point.withinBounds(emitter.position,
emitter.size)) {
                        return emitter;
                     }
               }

               for( var i = 0; i < self.positiveFieldLines.length; i+
+ ){
                     var fieldLine = self.positiveFieldLines[ i ];
                     if (point.withinBounds(fieldLine.position,
fieldLine.size)) {
                        return fieldLine;
                     }
               }

               for( var i = 0; i < self.negativeFieldLines.length; i+
+ ){
                     var negativeFieldLine = self.negativeFieldLines[
i ];
                     if
(point.withinBounds(negativeFieldLine.position,
negativeFieldLine.size)) {
                        return negativeFieldLine;
                     }
               }
      };

      self.getParticleCount    = function() { return
         self.particles.length;   };
      self.getIonCount    = function() { return self.ions.length;   };
      self.getEmitterCount     = function() {
         return self.emitters.length;   };
      self.getFieldCount       = function() {
         return self.fields.length;   };
```

```
   self.getNegativeFieldCount      = function() {
      return self.negative_fields.length;   };


   self.toString  = function() {
      var stateVersion = 1;
      var coreAttributes = [
         self.maxParticles,
             self.maxIons,
         self.draw.objects ? 1 : 0,
         self.draw.accelerations ? 1 : 0,
         self.draw.velocities ? 1 : 0,
         self.draw.particles ? 1 : 0,
             self.draw.ions ? 1 : 0

      ];
      for (var i = 0; i < self.emitters.length; i++) {
         coreAttributes.push(self.emitters[i].toString());
      }
      for (var i = 0; i < self.fields.length; i++) {
         coreAttributes.push(self.fields[i].toString());
      }
        for (var i = 0; i < self.negative_fields.length; i++) {
         coreAttributes.push(self.negative_fields[i].toString());
      }
        for (var i = 0; i < self.negativeFieldLines.length; i++) {
         coreAttributes.push(self.negativeFieldLines[i].toString()
);
      }
        for (var i = 0; i < self.positiveFieldLines.length; i++) {
         coreAttributes.push(self.positiveFieldLines[i].toString()
);
      }
      return 'Sv' + stateVersion + '(' + coreAttributes.join('|')
+ ')';
   }
   // Sv#(string)
   self.fromString = function(string) {
      var versions = {
         Sv1 : self.loadStateV1
      }
      var matches = string.match(/^([^(]+)\((.*)\)$/);
      if (matches && matches.length == 3) {
         if (versions[matches[1]]) {
            versions[matches[1]](matches[2]);
         }
      }

   }
   // maxP|draw.obj|draw.acc|draw.vel|draw.part|emitter|emitter|
field|field
  self.loadStateV1 = function (string) {
      var parts = string.split('|');
      self.maxParticles       = parseInt(parts.shift());
```

```
        self.maxIons      = parseInt(parts.shift());
      self.draw.objects       = parts.shift() === "1" ? true :
false;
      self.draw.accelerations = parts.shift() === "1" ? true :
false;
      self.draw.velocities    = parts.shift() === "1" ? true :
false;
      self.draw.particles     = parts.shift() === "1" ? true :
false;
      self.draw.ions     = parts.shift() === "1" ? true : false;
      self.emitters = [];
      self.fields = [];
      self.negative_fields = [];
      self.negativeFieldLines = [];
      self.positiveFieldLines = [];
      var object;
      while (objectString = parts.shift()) {
         if (objectString.charAt(0) == 'E') {
            self.emitters.push(ParticleEmitter.fromString(objectSt
ring));
         } else if(objectString.charAt(0) == 'n') {
            self.negativeFieldLines.push(NegativeFieldLine.fromStr
ing(objectString));
            }else if(objectString.charAt(0) == 'f') {
            self.positiveFieldLines.push(FieldLine.fromString(obje
ctString));
         }
      }
   }
}
```

4. Το αρχείο **Display.js**

Όπως δηλώνει το όνομα του, στο αρχείο Display.js βρίσκονται οι βασικές ιδιότητες εμφάνισης και μέθοδοι σχεδιασμού(π.χ. Σχεδιασμός κύκλου, ευθείας γραμμής), καθώς και το γενικό πλαίσιο δημιουργίας κινούμενων γραφικών που περιγράφεται στην παράγραφο 2.3, όσον αφορά το στοιχείο **<canvas>** με id:display.

```
function Display(canvas) {
    var self = this;
      self.canvas     = canvas;
```

```
      self.context    = undefined;
      self.framerate  = 100;
      self.numFrames  = 0;
      self.paused     = true;
      self.nextRedraw = 0;
      self.scale      = 1;
      self.listeners  = {};
      self.draw = {
            continuous : false,
            info   : true
      };

      self.info  = {
         fps                : 0,
         lastFrameTime      : 0,
         runningFrameTime   : 0
   };

      self.init = function(){
            if(!self.canvas.getContext){
         self.error("No Context");
                  return;
            }

            self.context = self.canvas.getContext( "2d" );
            self.context.scale(self.scale,self.scale);
            self.width = canvas.width / self.scale;
            self.height = canvas.height / self.scale;

      self.canvas.onmousedown = function(evt)
{self.fireEvent('mouseDown',evt);return false;};
      self.canvas.onmouseup   = function(evt)
{self.fireEvent('mouseUp',evt);return false;};
      self.canvas.onmouseover = function(evt)
{self.fireEvent('mouseOver',evt)};
      self.canvas.onmousemove = function(evt)
{self.fireEvent('mouseMove',evt)};

      window.requestAnimFrame = window.requestAnimationFrame
||
         window.webkitRequestAnimationFrame ||
         window.mozRequestAnimationFrame    ||
         window.oRequestAnimationFrame      ||
         window.msRequestAnimationFrame;

      self.addListener('draw', self);
      self.addListener('afterDraw', self);
      self.addListener('newFrame', self);
        self.main();
      };

      self.addListener = function(eventName, object) {
         if (!self.listeners[eventName]) {
            self.listeners[eventName] = [];
```

```
         }
         self.listeners[eventName].push(object);
      }

      self.fireEvent = function(eventName,evt) {
         if (self.listeners[eventName] &&
self.listeners[eventName].length > 0) {
         var eventMethod = "on" +
eventName.substr(0,1).toUpperCase() + eventName.substr(1);
            for (var i = 0; i < self.listeners[eventName].length;
i++) {
               if (self.listeners[eventName][i][eventMethod]) {
                  self.listeners[eventName][i]
[eventMethod].call(self.listeners[eventName][i],evt,self);
               }
            }
         }
      }

      self.main = function(){
         if (!self.paused) {
            self.nextFrame();
         }
         if (window.requestAnimFrame) {
         window.requestAnimFrame(function(){ self.main(); });
      } else {
         self.nextRedraw = setTimeout( function(){ self.main(); },
1000/self.framerate );
      }
      };

      self.nextFrame = function() {
         self.fireEvent('newFrame');
         self.fireEvent('beforeUpdate');
         self.fireEvent('update');
         self.fireEvent('afterUpdate');
         self.fireEvent('beforeDraw');
         self.fireEvent('draw');
         self.fireEvent('afterDraw');
      }

      self.error = function (msg) {
         self.fireEvent('error', { text : msg });
   };

      self.getFps = function() {
         return (1000/self.info.runningFrameTime).toFixed(1);
      };

      self.onDraw = function(){
      self.tick();
      };

      self.onAfterDraw = function() {
```

```
         if (self.draw.info) {
            self.drawStats();
         }
      }

   self.drawStats = function() {
      self.fillStyle("white");
      self.drawText("FPS : " + self.getFps(), new Point(10,
self.height-10),80);
   };

   self.onNewFrame = function() {
         if (!self.draw.continuous) {
         self.clear();
      }
   }

   self.drawLine = function(startPoint, endPoint) {
            var context=self.context;
            self.context.beginPath();
            self.context.moveTo(startPoint.x,startPoint.y);
            self.context.lineTo(endPoint.x, endPoint.y);

            context.lineWidth = 20;
            context.lineCap = "round";
            context.lineJoin = "round";
            context.strokeStyle = "gold";
            context.stroke();
            context.closePath();
   }

   self.drawText = function(txt,point,width){
      self.context.fillText(txt, point.x, point.y, width);
   }

   self.drawCircle = function(point, radius) {
      self.context.beginPath();
      self.context.arc(point.x, point.y, radius, 0, Math.PI*2);
      self.context.closePath();
      self.context.fill();
   }


   self.fillStyle = function(fill) { self.context.fillStyle =
fill; }
   self.strokeStyle = function(fill) { self.context.strokeStyle =
fill; }

      self.tick = function() {
         self.numFrames++;
         if (!self.info.lastFrameTime) {
            self.info.lastFrameTime = new Date().getTime();
         } else {
            var now = new Date().getTime();
```

```
            var timeToDraw = now - self.info.lastFrameTime;
            self.info.runningFrameTime =
self.info.runningFrameTime * .8 + timeToDraw * .2;
            self.info.lastFrameTime = now;
        }
    };

    self.clear = function() {
    self.context.clearRect( 0, 0, self.width, self.height);
    };
    self.start        = function() { self.paused =
false;       };
    self.stop         = function() { self.paused =
true;        };
    self.togglePause  = function() { self.paused = !
self.paused;};
    self.unpause      = self.start;
    self.pause        = self.stop;
    self.step         = function() { self.stop();
self.nextFrame(); };
};
```

5. Τα αρχεία **DisplayCurrent.js & IonDisplayCurrent.js**

Παρόμοια με το αρχείο Display.js, τα DisplayCurrent.js & IonDisplayCurrent.js εφοδιάζουν τα στοιχεία <canvas> με id:displayCurrent & ionDisplayCurrent με ιδιότητες εμφάνισης&σχεδιασμού. Για λόγους οικονομίας παρατίθεται στη συνέχεια μόνο το DisplayCurrent.js.

```
function DisplayCurrent(canvas) {
    var self = this;
      self.canvas     = canvas;
      self.context    = undefined;
      self.framerate  = 100;
      self.numFrames  = 0;
      self.paused     = true;
      self.nextRedraw = 0;
      self.scale      = 1;
      self.listeners  = {};
      self.draw = {
            continuous : true,
            info    : true
```

```
                        };

      self.info = {
         fps                : 0,
         lastFrameTime      : 0,
         runningFrameTime   : 0
                        };

      self.init = function(){
            if(!self.canvas.getContext){
         self.error("No Context");
                  return;
            }

            self.context = self.canvas.getContext( "2d" );
            self.context.scale(self.scale,self.scale);

            self.width = canvas.width / self.scale;
            self.height = canvas.height / self.scale;

            self.canvas.onmousedown = function(evt)
{self.fireEvent('mouseDown',evt);return false;};
            self.canvas.onmouseup   = function(evt)
{self.fireEvent('mouseUp',evt);return false;};
            self.canvas.onmouseover = function(evt)
{self.fireEvent('mouseOver',evt)};
      self.canvas.onmousemove = function(evt)
{self.fireEvent('mouseMove',evt)};

       window.requestAnimFrame = window.requestAnimationFrame
||
         window.webkitRequestAnimationFrame ||
         window.mozRequestAnimationFrame    ||
         window.oRequestAnimationFrame      ||
         window.msRequestAnimationFrame;

       self.addListener('draw', self);
            self.addListener('afterDraw', self);
       self.addListener('newFrame', self);
            self.main();
      };

      self.addListener = function(eventName, object) {
         if (!self.listeners[eventName]) {
            self.listeners[eventName] = [];
         }
         self.listeners[eventName].push(object);
      }

      self.fireEvent = function(eventName,evt) {
         if (self.listeners[eventName] &&
self.listeners[eventName].length > 0) {
         var eventMethod = "on" +
eventName.substr(0,1).toUpperCase() + eventName.substr(1);
```

```
            for (var i = 0; i < self.listeners[eventName].length;
i++) {
               if (self.listeners[eventName][i][eventMethod]) {
                  self.listeners[eventName][i]
[eventMethod].call(self.listeners[eventName][i],evt,self);
               }
            }
         }
      }

      self.main = function(){
         if (!self.paused) {
            self.nextFrame();
         }
         if (window.requestAnimFrame) {
         window.requestAnimFrame(function(){ self.main(); });
      } else {
         self.nextRedraw = setTimeout( function(){ self.main(); },
1000/self.framerate );
      }
      };

      self.nextFrame = function() {
         self.fireEvent('newFrame');
         self.fireEvent('beforeUpdate');
         self.fireEvent('update');
         self.fireEvent('afterUpdate');
         self.fireEvent('beforeDraw');
         self.fireEvent('draw');
         self.fireEvent('afterDraw');
      };

      self.error = function (msg) {
         self.fireEvent('error', { text : msg });
    };

      self.getFps = function() {
         return (1000/self.info.runningFrameTime).toFixed(1);
      };

      self.getSegments = function(){
            return currentSystem.noOfSegments;
      };

      self.getIonSegments = function(){
            return ionCurrentSystem.noOfSegments;
      };

      self.getGlobalTime = function(){
            return newGlobalTime;
      };

      self.drawTime = function (){
```

```
                        self.context.beginPath();
                        self.drawText(self.getGlobalTime(),new
Point(canvas.width*0.5,canvas.height*0.50),120);
                        self.drawText("Time in nanoseconds",new
Point(canvas.width*0.5,canvas.height*0.50 +10),120);
                        self.context.lineWidth = 2;
                self.context.lineCap = "round";
                self.context.lineJoin = "round";
                self.context.strokeStyle = "gold";
                self.context.stroke();
                        self.context.closePath();

      };

      self.onDraw = function(){
      self.tick();
      };

      self.onAfterDraw = function() {
         if (self.draw.info) {
            //self.drawTime();
         }
      }
   self.drawStats = function() {
      self.fillStyle("white");
      self.drawText("FPS : " + self.getFps(), new Point(10,
self.height-10),80);
        self.drawText("Segments : " + self.getSegments(), new
Point(60, self.height-10),80);
   };

   self.onNewFrame = function() {
         if (!self.draw.continuous) {
         self.clear();
      }
   }

   self.drawLine = function(startPoint, endPoint) {
            self.context.beginPath();
            self.context.moveTo(startPoint.x,startPoint.y);
            self.context.lineTo(endPoint.x, endPoint.y);
            self.context.stroke();
   }

   self.drawText = function(txt,point,width){
      self.context.fillText(txt, point.x, point.y, width);
   }

   self.drawCircle = function(point, radius) {
      self.context.beginPath();
      self.context.arc(point.x, point.y, radius, 0, Math.PI*2);
      self.context.closePath();
      self.context.fill();
   }
```

```
   self.drawWire = function(point, width, height) {
      self.context.beginPath();
      self.context.rect(point.x,point.y,width,height);
      self.context.closePath();
      self.context.fill();
   }

   self.fillStyle = function(fill) { self.context.fillStyle =
fill; }

   self.strokeStyle = function(fill) { self.context.strokeStyle =
fill; }

      self.tick = function() {
         self.numFrames++;
         if (!self.info.lastFrameTime) {
            self.info.lastFrameTime = new Date().getTime();
         } else {
            var now = new Date().getTime();
            var timeToDraw = now - self.info.lastFrameTime;
            self.info.runningFrameTime =
self.info.runningFrameTime * .8 + timeToDraw * .2;
            self.info.lastFrameTime = now;
         }
      };

      self.clear = function() {
      self.context.clearRect( 0, 0, self.width, self.height);
      };

      self.start        = function() { self.paused =
false;       };

      self.stop         = function() { self.paused =
true;        };

      self.togglePause  = function() { self.paused = !
self.paused;};

      self.unpause      = self.start;
      self.pause        = self.stop;
      self.step         = function() { self.stop();
self.nextFrame(); };
};
```

6. Τα αρχεία **CurrentSystem.js & IonCurrentSustem.js**

Πρόκειται για τα αντίστοιχα αρχεία με το ParticleSystem.js, όπου περιγράφεται ακριβώς ο τριγωνικός παλμός που θέλουμε να δημιουργηθεί στα αριστερά μικρότερα πλαίσια (CurrentSysrem & IonCurrentSystem αντίστοιχα), όταν τα ηλεκτρόνια και τα θετικά ιόντα δημιουργήσουν ηλεκτρικό ρεύμα κατά την πρόσπτωσή τους στις ηλεκτρικά φορτισμένες γραμμές. Για λόγους οικονομίας παρατίθεται μόνο το CurrentSystem.js

```
var newGlobalDate = new Date();
var   newGlobalTime = newGlobalDate.getTime();
function CurrentSystem(){

      var self = this;
      self.noOfSegments;
      var allowAnimation=true;
      var segmentLength=5;
      var lastUpdateTime= 0;
      var numSegments= 1000;
                  // moving to the left
      var direction={
                  x: -1,
                  y: -0
                  };
      var ground = {
                              x: 0,
                              y: 0

                      };

      var pulse = {
                  segmentLength: 1,
                  lastUpdateTime: 0,
                  numSegments: 1000,
                  // moving to the left
                  direction: {
                      x: -1,
                      y: -0
                  },
                  segments: []
              };

      window.requestAnimFrame = window.requestAnimationFrame
||
         window.webkitRequestAnimationFrame ||
         window.mozRequestAnimationFrame    ||
```

```
        window.oRequestAnimationFrame      ||
        window.msRequestAnimationFrame;



    function updateSnake(canvas, snake){
                var maxVariance =0;
                var snakeSpeed = 100; //px / s
                var segmentsPerSecond = snakeSpeed /
snake.segmentLength;
                var segments = snake.segments;
                var date = new Date();
                var time = date.getTime();
                var timeDiff = (time - snake.lastUpdateTime);
                if (timeDiff > 1000 / segmentsPerSecond) {
                    var head = segments[segments.length - 1];
                    var neck = segments[segments.length - 2];
                    var tail = segments[0];

                    var direction = snake.direction;
                    var newHeadX = head.x + direction.x *
snake.segmentLength;
                    var newHeadY = head.y + direction.y *
snake.segmentLength;
                            ground = {
                            x: canvas.width,
                            y: canvas.height*0.8
                                    };
                            for (var n = 0; n < segments.length;
n++) {
                    var segment = segments[n];
                            segment.x = segment.x + direction.x
* snake.segmentLength;
                            }
                            segments.unshift({
                        x: tail.x+segmentLength,
                        y: ground.y
                            });
                    // remove segments if thy are no more visible
                    if ( head.x < -30) {
                        segments.pop();

                    }
                            segments2 =[{
                        x: tail.x+30,
                        y: tail.y
                            }];


                    if (segments.length > snake.numSegments) {
                        segments.pop();
                    }
```

```
                    var variance = ((maxVariance / 2) -
Math.random() * maxVariance);

                    direction.x += variance;
                    direction.y -= variance;

                    // update direction vector
                    if (direction.x > 1) {
                        direction.x = 1;
                    }
                    if (direction.x < -1) {
                        direction.x = -1;
                    }
                    snake.lastUpdateTime = time;
                              self.noOfSegments=segments.length;

                }
                        pulse.segments=snake.segments;
                              }

            function animate(canvas, snake){
                var context = canvas.getContext("2d");

                // update
                        displayCurrent.drawTime();
                        if(allowAnimation==true){
                updateSnake(canvas, snake);

                // clear
                context.clearRect(0, 0, canvas.width,
canvas.height);

                     // draw
                        drawSnake(context, snake);
                        drawTime(context,canvas);

                // request new frame
                        if (window.requestAnimFrame) {

      window.requestAnimFrame(function(){ animate(canvas,
snake);});
                  } else {
                                        self.nextRedraw = setTimeout(
function(){ animate(canvas, snake); }, 1000/self.framerate );
                  }

                        }
            }

                 function getGlobalTime(){

                        return Math.round(-(globalTime-
displayCurrent.info.lastFrameTime)/1000);
                        }
```

```
            function drawTime(context,canvas,point){
                        context.beginPath();

      displayCurrent.drawText(getGlobalTime(),new
Point(canvas.width*0.5,canvas.height*0.50),120);
                        displayCurrent.drawText("Time in
nanoseconds",new Point(canvas.width*0.5,canvas.height*0.50
+10),120);
                        context.lineWidth = 2;
                context.lineCap = "round";
                context.lineJoin = "round";
                context.strokeStyle = "gold";
                context.stroke();
                        context.closePath();
                        }

            function drawSnake(context, snake){
                var segments = snake.segments;
                var tail2 = segments[segments.length-1];
                        var head2 = segments[0];
                context.beginPath();
                context.moveTo(tail2.x, tail2.y);

                for (var n = 1; n < segments.length+1; n++) {
                    var segment = segments[segments.length-n];
                    context.lineTo(segment.x, segment.y);
                             if(segment.time>0){
                             var style1 = context.fillstyle;
                             context.fillStyle =
context.strokeStyle;
                             //console.log(segment.time);

      displayCurrent.context.fillText(segment.time,segment.x-
segmentLength, ground.y+10,segmentLength*4,"bold 120px sans-
serif");
                                       }
                }

                context.lineWidth = 2;
                context.lineCap = "round";
                context.lineJoin = "round";
                context.strokeStyle = "gold";
                context.stroke();
                        context.closePath();


            }

                   self.pulseParticle = function (){
                        var canvas =
document.getElementById("current");
                var segmentLength = 5; // px
                var headX=canvas.width;
```

```
            var headY=canvas.height*0.8;
            allowanimation=false;
snake = {
    segmentLength: 5,
    lastUpdateTime: 0,
    numSegments: 1000,
    // moving to the left
    direction: {
        x: -1,
        y: -0
    },
    segments: [{
        // tail
        x: headX ,
        y: headY
    },{
        // head
        x: headX - segmentLength,
        y: headY
    }]
};

            var pulse2 = pulse;
            if(pulse.segments.length>2){
            pulse.segments.unshift({
            // head
            x: pulse.segments[0].x+2*segmentLength,
            y: pulse.segments[0].y-10,
                        time: getGlobalTime()
    });

            }else{
                  pulse=snake;
                  }
     allowAnimation=true;
animate(canvas, pulse);

            };

            }
```

7. Τα αρχεία **FieldLine.js & NegativeFieldLine.js**

Τα αρχεία αυτά αρχικοποιούν τις ηλεκτρικά φορτισμένες γραμμές. Προσδίδονται αρχικές τιμές για το χρώμα, το μέγεθος, λειτουργίες μετακίνησης

των γραμμών/ηλ.πεδίων, καθώς και λειτουργίες αποθήκευσης των παραμέτρων τους στην URL αλλά και φόρτωσης των παραμέτρων από την URL. Λόγω της ομοιότητάς τους παρατίθεται στη συνέχεια μόνο το FieldLine.js

```
function FieldLine(point, mass) {
   this.position   = point;
   this.size       = 16;
   this.setMass(mass);
}

FieldLine.drawColor = "rgb(210,105,30)";
FieldLine.drawColor2 = "rgb(0,0,0)";


FieldLine.prototype.setMass = function(mass) {
   this.mass = mass;
   this.drawColor = mass < 0 ? "#900" : "#090";
   return this;
}

FieldLine.prototype.glow = function(){
      this.drawColor = "rgb(23,20,45)";
      return this;
}

FieldLine.prototype.restore = function(){
      mass=this.mass;
      this.drawColor = mass < 0 ? "#900" : "#090";
   var t= window.setTimeout("return this",200);
}

FieldLine.prototype.moveTo = function(point) {
   this.position = point;
}

FieldLine.prototype.toString = function() {
   var coreAttributes = [
      this.position.toString(),
      this.mass
   ];
   return 'f' + coreAttributes.join(':');
}
FieldLine.fromString = function(string) {
   var parts = string.substr(1).split(':');
   var fieldLine = new
FieldLine(Point.fromString(parts.shift()),parseInt(parts.shift()))
;
   return fieldLine;}
```

8. Το αρχείο **ParticleEmitter.js**

Πρόκειται για το σωματίδιο που διαπερνά τον ανιχνευτή με ταχύτητα κοντά στην ταχύτητα του φωτός. Εδώ ορίζεται η εξίσωση κίνησης και οι μέθοδοι εκπομπής ηλεκτρονίων και θετικών ιόντων (addParticle() & addIon()). Ακόμα, προστίθενται στην URL οι παράμετροι του σωματιδίου.

```
function ParticleEmitter(point,velocity) {
      this.position     = point;
      this.velocity     = velocity;
      this.size         = 2;
      this.particleLife = -1;
      this.ionLife = -1;
      this.spread       = Math.PI ;
      this.emissionRate = 1;

      this.moveTo = function(point) {
      this.position = point;
      }



ParticleEmitter.prototype.move = function() {
   //this.velocity.x += this.acceleration.x;
   //this.velocity.y += this.acceleration.y;
   this.position.x += 30      //+0.02*this.velocity.x;
   this.position.y += -60//-0.5*this.velocity.y;

};

this.addParticle = function(){
      var particle = new Particle(
                this.position.copy(),
                Vector.fromAngle(this.velocity.getAngle() +
this.spread - (Math.random() * this.spread * 0.2),
this.velocity.getMagnitude()/10)
      );
      particle.ttl = this.particleLife;
            return particle;
      };

this.addIon = function(){
                  var ion = new Ion(
                this.position.copy(),
                Vector.fromAngle(this.velocity.getAngle() +
this.spread - (Math.random() * this.spread * 0.2),
this.velocity.getMagnitude()/10)
      );
```

```
      ion.ttl = this.ionLife;
            return ion;
      };

this.toString = function() {
      var coreAttributes = [
         this.position.toString(),
         this.velocity.toString(),
         this.size,
         this.particleLife,
             this.ionLife,
         this.spread.toFixed(2),
         this.emissionRate
      ];
      return 'E' + coreAttributes.join(':');
   }
}

ParticleEmitter.drawColor  = "#999";
ParticleEmitter.drawColor2 = "#000";
ParticleEmitter.jitter     = .05;

ParticleEmitter.fromString = function(string) {
   var parts = (string.substr(1).split(':'));
   var emitter = new ParticleEmitter();
   emitter.position     = Point.fromString(parts.shift());
   emitter.velocity     = Vector.fromString(parts.shift());
   emitter.size         = parseInt(parts.shift());
   emitter.particleLife = parseInt(parts.shift());
   emitter.ionLife = parseInt(parts.shift());
   emitter.spread       = parseFloat(parts.shift());
   emitter.emissionRate = parseInt(parts.shift().valueOf());
   return emitter;
}
```

9. Τα αρχεία **Particle.js & Ion.js**

Εδώ δημιουργούνται τα αντικείμενα που αναπαριστούν τα ηλεκτρόνια και τα θετικά ιόντα της εφαρμογής μας. Εκτός από ιδιότητες σχετικά με τα χρώματα, υπάρχουν μέθοδοι κίνησης και αντιστοίχησης στις φορτισμένες γραμμές. Ακόμα, εδώ βρίσκονται μέθοδοι προσθήκης νέων ηλεκτρονίων και ιόντων(φαινόμενο Avalanche). Για λόγους οικονομίας παρατίθεται μόνο το αρχείο Particle.js που σχετίζεται με τα (πορτοκαλί) ηλεκτρόνια.

```
var displayCurrent = null;
var currentSystem = null;
var ionCurrentSystem = null;
var currentWidth = 0;
var headX;
var headY;

function Particle(point,velocity){
      this.position     = point;
    this.velocity     = velocity;
      this.acceleration = new Vector(0,0);
      this.ttl          = -1;
      this.lived        = 0;
      this.particleLife = -1;
      this.ionLife = -1;
      this.spread       = Math.PI ;
      this.emissionRate = 1;
      this.rndm = Math.random();
}

Particle.GLOBAL_DRAW_COLOR = [255,69,0,255];
//Particle.GLOBAL_DRAW_COLOR = [166,67,0,255];

Particle.prototype.submitToFields = function(fields) {
   var totalAccelerationX = 0;
   var totalAccelerationY = 0;

   for (var i = 0; i < fields.length; i++) {
      var field = fields[i];

      // inlining what should be Vector object methods for
performance reasons
var vectorX = field.position.x - this.position.x;
   var vectorY = 0;//field.position.y - this.position.y;
            var force = field.mass /(Math.pow((vectorX*vectorX+
(display.height)*2/4),0.5));
      totalAccelerationX += force/vectorX ;
      totalAccelerationY += vectorY * force;
        if((vectorX*vectorX)<field.size*field.size){
              this.touch(); //particle dies
               currentWidth+=1;
               headX +=-5;
               headY +=0;
               currentSystem.pulseParticle();// electric pulse is
created

              }
   }
   this.acceleration = new
Vector(totalAccelerationX,totalAccelerationY);
```

```
};

Particle.prototype.move = function() {
   this.velocity.x += this.acceleration.x;
   this.velocity.y += this.acceleration.y;
   this.position.x += this.velocity.x;
   this.position.y += this.velocity.y;

};

Particle.prototype.touch = function() {
   this.velocity.x = 0;
   this.velocity.y = 0;
   this.position.x = -1000;
   this.position.y = -1000;


};

Particle.prototype.addParticle = function(){

            var particle = new Particle(
               this.position.copy(),
               Vector.fromAngle(/*this.velocity.getAngle() +
*/this.spread - (Math.random() * this.spread * 0.2),
this.velocity.getMagnitude()/10)
                  );
      particle.ttl = this.particleLife;
            return particle;
      };

Particle.prototype.addIon = function(){
            var ion = new Ion(
               this.position.copy(),
               Vector.fromAngle(/*this.velocity.getAngle() +*/
this.spread - (Math.random() * this.spread * 0.2),
this.velocity.getMagnitude()/10)
      );
      ion.ttl = this.ionLife;
            return ion;
      };

// 3 drawing methods, drawBasic will be used
Particle.prototype.drawVariable = function(pixels,width,height) {
   var baseIndex = 4 * (~~this.position.y * width +
~~this.position.x);
   var velocity = this.velocity.getMagnitude();
   var r = Particle.GLOBAL_DRAW_COLOR[0] * velocity;
   var g = Particle.GLOBAL_DRAW_COLOR[1];
   var b = Particle.GLOBAL_DRAW_COLOR[2] * .5/velocity;
   var a = Particle.GLOBAL_DRAW_COLOR[3];
   pixels[baseIndex]      += r;
   pixels[baseIndex + 1]  += g;
   pixels[baseIndex + 2]  += b;
```

```
    pixels[baseIndex + 3]   = a;
}

Particle.prototype.drawBasic = function(pixels,width,height) {
    var baseIndex = 4 * (~~this.position.y * (width) +
~~this.position.x);
    var r = Particle.GLOBAL_DRAW_COLOR[0];
    var g = Particle.GLOBAL_DRAW_COLOR[1];
    var b = Particle.GLOBAL_DRAW_COLOR[2];
    var a = Particle.GLOBAL_DRAW_COLOR[3];
    pixels[baseIndex - 4]  += r;
    pixels[baseIndex - 3]  += g;
    pixels[baseIndex - 2]  += b;
    pixels[baseIndex - 1]   = a;
    pixels[baseIndex]      += r;
    pixels[baseIndex + 1]  += g;
    pixels[baseIndex + 2]  += b;
    pixels[baseIndex + 3]   = a;
    pixels[baseIndex + 4]  += r;
    pixels[baseIndex + 5]  += g;
    pixels[baseIndex + 6]  += b;
    pixels[baseIndex + 7]   = a;
    baseIndex += width * 4;
    pixels[baseIndex - 4]  += r;
    pixels[baseIndex - 3]  += g;
    pixels[baseIndex - 2]  += b;
    pixels[baseIndex - 1]   = a;
    pixels[baseIndex]      += r;
    pixels[baseIndex + 1]  += g;
    pixels[baseIndex + 2]  += b;
    pixels[baseIndex + 3]   = a;
    pixels[baseIndex + 4]  += r;
    pixels[baseIndex + 5]  += g;
    pixels[baseIndex + 6]  += b;
    pixels[baseIndex + 7]   = a;
    baseIndex += width * 4;
    pixels[baseIndex - 4]  += r;
    pixels[baseIndex - 3]  += g;
    pixels[baseIndex - 2]  += b;
    pixels[baseIndex - 1]   = a;
    pixels[baseIndex]      += r;
    pixels[baseIndex + 1]  += g;
    pixels[baseIndex + 2]  += b;
    pixels[baseIndex + 3]   = a;
    pixels[baseIndex + 4]  += r;
    pixels[baseIndex + 5]  += g;
    pixels[baseIndex + 6]  += b;
    pixels[baseIndex + 7]   = a;
}

Particle.prototype.drawSoft = function(pixels,width,height) {
    var baseIndex = 4 * (~~this.position.y * width +
~~this.position.x);
    var r = Particle.GLOBAL_DRAW_COLOR[0];
```

```
    var g = Particle.GLOBAL_DRAW_COLOR[1];
    var b = Particle.GLOBAL_DRAW_COLOR[2];
    var a = Particle.GLOBAL_DRAW_COLOR[3];
    pixels[baseIndex - 4]  += r*.80;
    pixels[baseIndex - 3]  += g*.80;
    pixels[baseIndex - 2]  += b*.80;
    pixels[baseIndex - 1]   = a;
    pixels[baseIndex]      += r*.80;
    pixels[baseIndex + 1]  += g*.80;
    pixels[baseIndex + 2]  += b*.80;
    pixels[baseIndex + 3]   = a;
    pixels[baseIndex + 4]  += r*.80;
    pixels[baseIndex + 5]  += g*.80;
    pixels[baseIndex + 6]  += b*.80;
    pixels[baseIndex + 7]   = a;
    baseIndex += width * 4;
    pixels[baseIndex - 4]  += r*.80;
    pixels[baseIndex - 3]  += g*.80;
    pixels[baseIndex - 2]  += b*.80;
    pixels[baseIndex - 1]   = a;
    pixels[baseIndex]      += r;
    pixels[baseIndex + 1]  += g;
    pixels[baseIndex + 2]  += b;
    pixels[baseIndex + 3]   = a;
    pixels[baseIndex + 4]  += r*.80;
    pixels[baseIndex + 5]  += g*.80;
    pixels[baseIndex + 6]  += b*.80;
    pixels[baseIndex + 7]   = a;
    baseIndex += width * 4;
    pixels[baseIndex - 4]  += r*.80;
    pixels[baseIndex - 3]  += g*.80;
    pixels[baseIndex - 2]  += b*.80;
    pixels[baseIndex - 1]   = a;
    pixels[baseIndex]      += r*.80;
    pixels[baseIndex + 1]  += g*.80;
    pixels[baseIndex + 2]  += b*.80;
    pixels[baseIndex + 3]   = a;
    pixels[baseIndex + 4]  += r*.80;
    pixels[baseIndex + 5]  += g*.80;
    pixels[baseIndex + 6]  += b*.80;
    pixels[baseIndex + 7]   = a;
}

Particle.prototype.draw = Particle.prototype.drawBasic;
```

Κεφάλαιο 4:
Η σημασία της εκλαΐκευσης της Φυσικής

Η διδασκαλία και παρουσίαση της Φυσικής είναι ουσιαστικά πιο δύσκολη από την έρευνα. Η τελευταία είναι μια επιστημονική δραστηριότητα, ενώ η διδασκαλία και η παρουσίαση είναι ένας συνδυασμός τέχνης και επιστήμης. Η εκλαΐκευση, δηλαδή η παρουσίαση εξειδικευμένων επιστημονικών θεωριών και αποτελεσμάτων σε ένα ευρύτερο κύκλο ανθρώπων, όχι απαραίτητα σχετικών με το θέμα, παραμένει ένα σοβαρό πρόβλημα, ως προς την ανάγκη παρουσίασης αυτής καθ' εαυτής αλλά και ως προς τα μέσα που πρέπει να χρησιμοποιηθούν για να επιτευχθεί.

Από τον 18ο αιώνα μέχρι τις αρχές του 20ου, οι επιστήμονες με διάφορους τρόπους (ανοικτές διαλέξεις, επιδείξεις πειραμάτων, ακόμα και κοινωνικές συναναστροφές), επιτελούσαν αυτό το έργο της εκλαΐκευσης. Όμως, οι ραγδαίες εξελίξεις στην επιστήμη οδήγησαν σε μια ολοένα αυξανόμενη πολυπλοκότητα, ενώ η "γλώσσα" που χρησιμοποιείται πλέον στην έρευνα (των θετικών επιστημών) είναι τόσο τεχνική που κάνει την εκλαΐκευση όλο και πιο δύσκολη, με αποτέλεσμα να "παραμελείται" από τους επιστήμονες ο τομέας της εκλαΐκευσης. Πολλοί την αντιμετωπίζουν ως μείωση του επιπέδου, εγκατάλειψη της επιστημονικής ακρίβειας.

Οι σύγχρονες θετικές επιστήμες, ιδιαίτερα η Φυσική και η Βιολογία, είναι ό,τι πιο θετικό έχει προσφέρει ο πολιτισμός του εικοστού αιώνα. Η κατανόηση της δομής της ύλης, του ατόμου, των μορίων, της συμπυκνωμένης ύλης, ή τα νέα τηλεσκόπια για την εξερεύνηση του διαστήματος σε ακτινοβολίες γ, X, UV, IR, μικροκυμάτων, αλλά και με ιδέες που ίσως δεν έχουν αποκτήσει ακόμη την τελική τους μορφή, π.χ. τη Μεγάλη Έκρηξη, την πρώιμη εξέλιξη της ύλης, τη διαστολή του Σύμπαντος. Το κοινό δεν ενδιαφέρεται γι'αυτά μόνο από επιστημονικής άποψης, αλλά επίσης από φιλοσοφικής και συχνά θρησκευτικής άποψης. Υπό αυτήν την έννοια, είναι

ασφαλώς πιο εύκολο να παρουσιάζονται στον αδαή αυτά τα ζητήματα, αφού ενδιαφέρεται τόσο πολύ γι'αυτά, αν και παράλληλα είναι πολύ δύσκολο να εξηγηθεί σωστά και απλά το περιεχόμενό τους.

Ακόμα, οι εκλαϊκευτικές παρουσιάσεις συμβάλλουν στην κατάρριψη της άποψης που θεωρεί τον επιστήμονα, ούτε λίγο ούτε πολύ, ως ένα είδος μάγου. Φαίνεται επιτακτική η ανάγκη των φυσικών επιστημόνων να παρουσιάσουν τις θεωρίες τους απλά και κατανοητά, όχι μόνο για να προσελκύσουν το ενδιαφέρον νέων ανθρώπων που πιθανώς θα αποτελέσουν τη νέα γενιά επιστημόνων, αλλά και για να αποφευχθούν παρανοήσεις που συχνά οδηγούν σε σενάρια τρόμου.

Ένα τέτοιο σενάριο προέκυψε το 2009 κατά τη διεξαγωγή πειραμάτων στο CERN, όταν μερίδα του παγκόσμιου Τύπου υιοθέτησε την ανησυχία δημιουργίας μιας μαύρης τρύπας που θα απορροφούσε τον πλανήτη, όπως φαίνεται στο άρθρο του ιστότοπου foxnews.com. Είναι προφανές ότι σενάρια τρομολαγνείας έχουν μεγαλύτερη απήχηση και ως εκ τούτου περισσότερο χρόνο προβολής στα μέσα ενημέρωσης, από ότι οι ουσιαστικές εξελίξεις στην επιστημονική κοινότητα.

Ένα άλλο παράδειγμα παρανόησης αποτελεί η προβολή της πρόσφατης επιβεβαίωσης της ύπαρξης του μποζονίου Χιγκς, σαν σωματίδιο του Θεού, από τα μέσα μαζικής ενημέρωσης.

ΒΙΒΛΙΟΓΡΑΦΙΑ - ΠΗΓΕΣ

http://www.physics.ntua.gr/POPPHYS/

http://public.web.cern.ch/public/en/research/Detector-en.html

http://jarrodoverson.com/static/demos/particleSystem/

http://www.html5canvastutorials.com

http://www.w3.org/