Σαράντος Ψυχάρης

ΑΣΠΑΙΤΕ

Ιανουάριος 2014

Οδηγός Χρήσης του Easy Java Simulations(Ejs) Tool

Περιεχόμενα

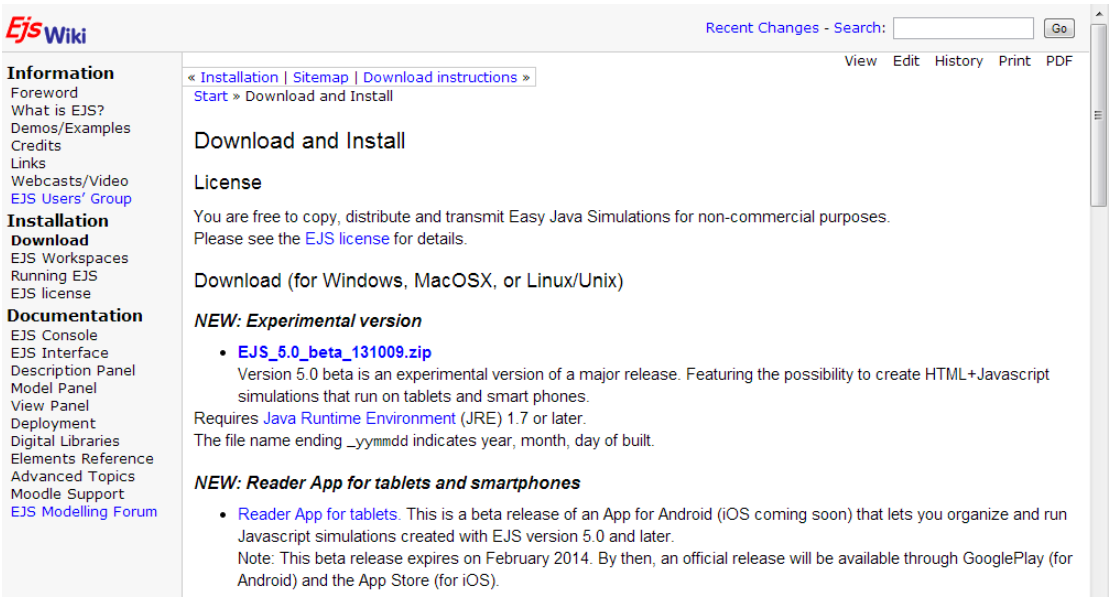
Κεφάλαιο 1. Εισαγωγή

Το EJS είναι ένα εργαλείο συγγραφής που έχει δημιουργηθεί για την ανάπτυξη αλληλεπιδραστικών προσομοιώσεων με χρήση της γλώσσας προγραμματισμού Java. Για την χρήση του Easy Java Simulations(EJS) δεν απαιτείται η γνώση της γλώσσας προγραμματισμού Java αν και η στοιχειώδης γνώση της Java θα βοηθούσε να καταλάβουμε καλύτερα με ποιο τρόπο το EJS υλοποιεί τις έννοιες της Java.

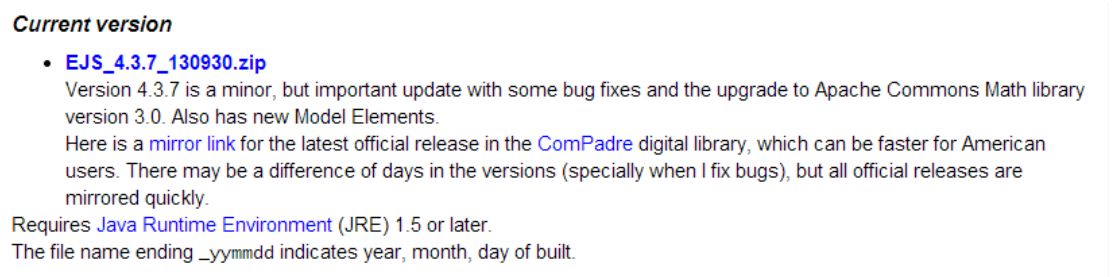
Η δημιουργία αυτού του εργαλείου ξεκίνησε από την ανάγκη να απλοποιηθούν τεχνικά θέματα που έχουν σχέση με τη γνώση της Java καθώς και με εννοιολογικά-διδακτικά θέματα που σχετίζονται με την ανάπτυξη διερευνητικών και εκφραστικών μοντέλων προσομοίωσης. Από τεχνικής πλευράς το EJS απλοποιεί θέματα σχετικά με τα «γραφικά» της προσομοίωσης, δηλαδή αυτά που συνήθως χρειάζονται προχωρημένο προγραμματισμό για να δημιουργηθούν(advanced level of technical know-how on programming computer graphics) αλλά και θέματα που σχετίζονται με την ανάπτυξη μοντέλων από τον εκπαιδευόμενο.
Από πλευράς ανάπτυξης εννοιολογικών μοντέλων προσομοίωσης, το EJS, παρέχει τη δυνατότητα μιας απλοποιημένης δομής για να δημιουργηθούν εκφραστικά μοντέλα προσομοίωσης συνεπή με την επιστημονική περιγραφή του φαινομένου υπό μελέτη, ενώ οι αριθμητικές μέθοδοι που χρησιμοποιεί είναι τέτοιες που επιτρέπουν την ανάπτυξη αλγορίθμων που οδηγούν σε αριθμητικές λύσεις πολύ κοντά στις επιστημονικές τιμές.
Το Ejs παρέχει τη δυνατότητα για μια απλοποιημένη δομή του φαινομένου που μελετάμε καθώς παρέχει τη δυνατότητα συγγραφής απλού κώδικα που περιλαμβάνει αλγόριθμους και δίνει τη δυνατότητα στον εκπαιδευόμενο να «ελέγχει» ο ίδιος το μοντέλο.
Το υλικό που περιέχεται έχει γραφτεί από τον Francisco Esquembre, September 2005, Easy Java Simulations, The Manual, Version 3.4 - September 2005

1.1 Η εγκατάσταση του EJS

Το Easy Java Simulations μπορεί να τρέξει σε οποιοδήποτε λειτουργικό σύστημα που υποστηρίζει την Java Virtual Machine(JVM) τρέχει με τον ίδιο τρόπο σε όλες τις περιπτώσεις. Υλικό σχετικό με την εγκατάσταση του EJS μπορεί να βρεθεί στις διευθύνσεις http://fem.um.es/Ejs και www.opensourcephysics.org

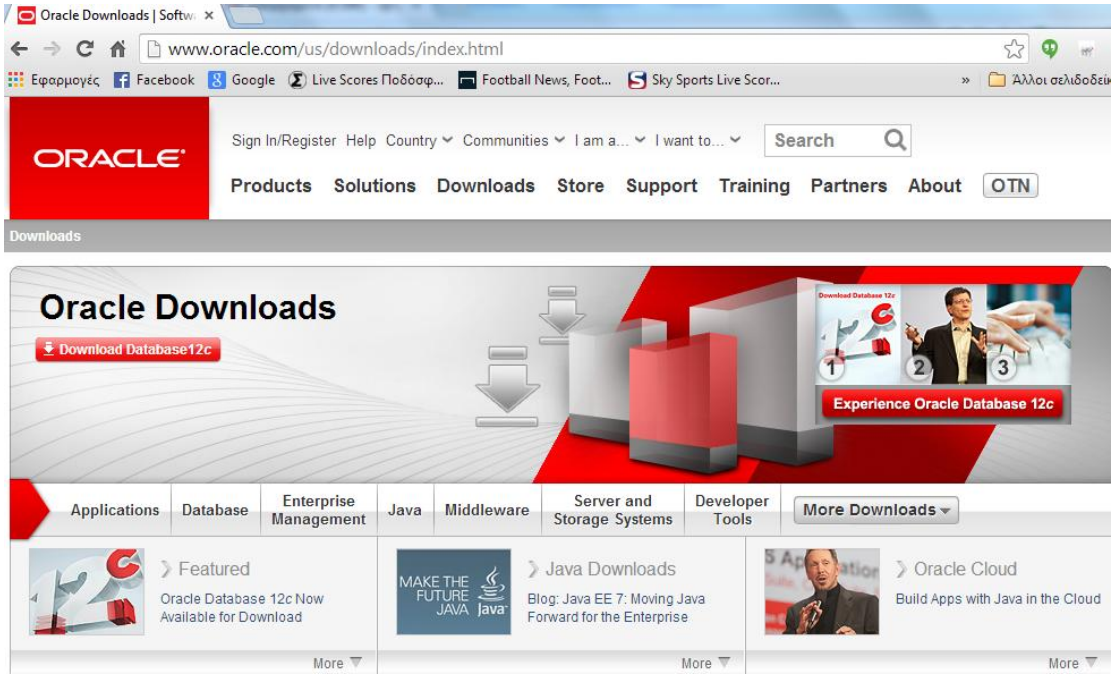


Εικόνα: η ιστοσελίδα για να κατεβάσουμε το Ejs



Εικόνα: η τρέχουσα έκδοση για να κατεβάσουμε το Ejs

Για να τρέξει το Ejs χρειάζεται η εγκατάσταση της Java, η οποία γίνεται από την ιστοσελίδα www.sun.com(www.oracle.com)
Το Easy Java Simulations απαιτεί την εγκατάσταση του Java Development Kit για να τρέξει.
Δεν θα πρέπει να συγχέουμε το Java Development Kit με το Java Runtime Environment (JRE), το οποίο είναι ένα βοηθητικό -plug-in της Java και το οποίο επιτρέπει τον φυλλομετρητή—browser- να τρέχει μικρο-εφαρμογές-applets- της Java που τρέχουν μέσα από HTML σελίδα. Έτσι το JRE δεν περιέχει μεταγλωττιστή και ενώ μόνο με τη χρήση του JRE μπορούμε να τρέχουμε εφαρμογές που έχουν δημιουργηθεί με το **EJS** , εν τούτοις δεν μπορούμε να δημιουργήσουμε εφαρμογές με τη χρήση του .



Εικόνα: η ιστοσελίδα (www.oracle.com) για να εγκατασταθεί η Java

Εναλλακτικά μπορείτε να πάτε απ ευθείας στην ιστοσελίδα

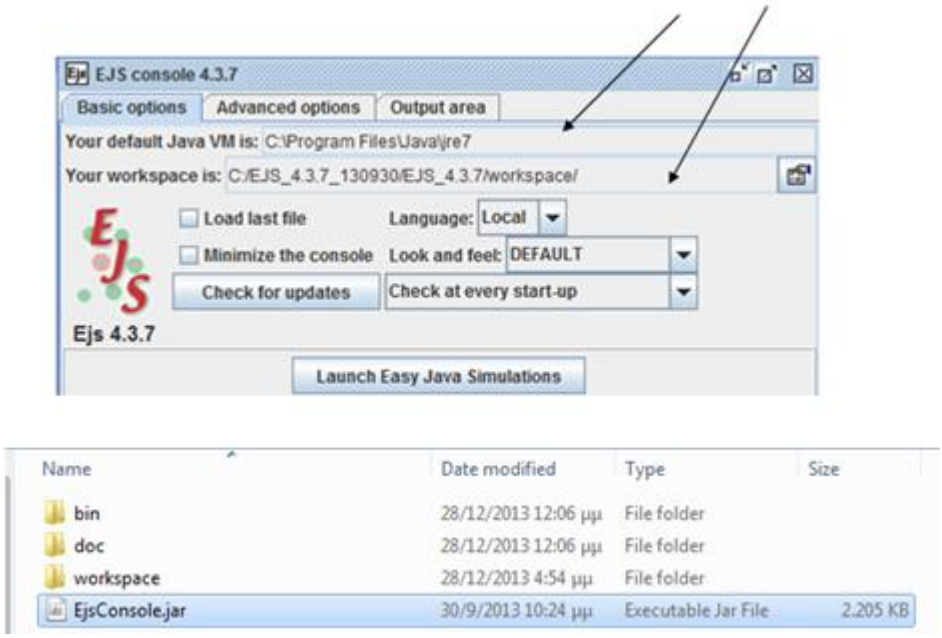
http://www.oracle.com/technetwork/java/javase/downloads/index.html

και να κατεβάσετε την έκδοση (28-12-2013) Java Platform (JDK) 7u45

.

Εικόνα: Η ιστοσελίδα
http://www.oracle.com/technetwork/java/javase/downloads/index.html
για να εγκατασταθεί η Java

Μετά την εγκατάσταση του **Ejs** και της Java, με διπλό κλίκ στο Ejsconsole.jar, θα έχουμε την παρακάτω εικόνα.



Εικόνα: Η θέση της Java στο Ejs και ο χώρος εργασίας των προσομοιώσεων

Επίσης μπορείτε να συμβουλευθείτε τα παρακάτω:
Introduction to EJS Video Tutorial
Για την χρήση αυτού του Video απαιτείται το λογισμικό Flash.

http://www.compadre.org/OSP/tutorials/EJS_Tutorial/

EJS Translation Facilities Video Tutorial

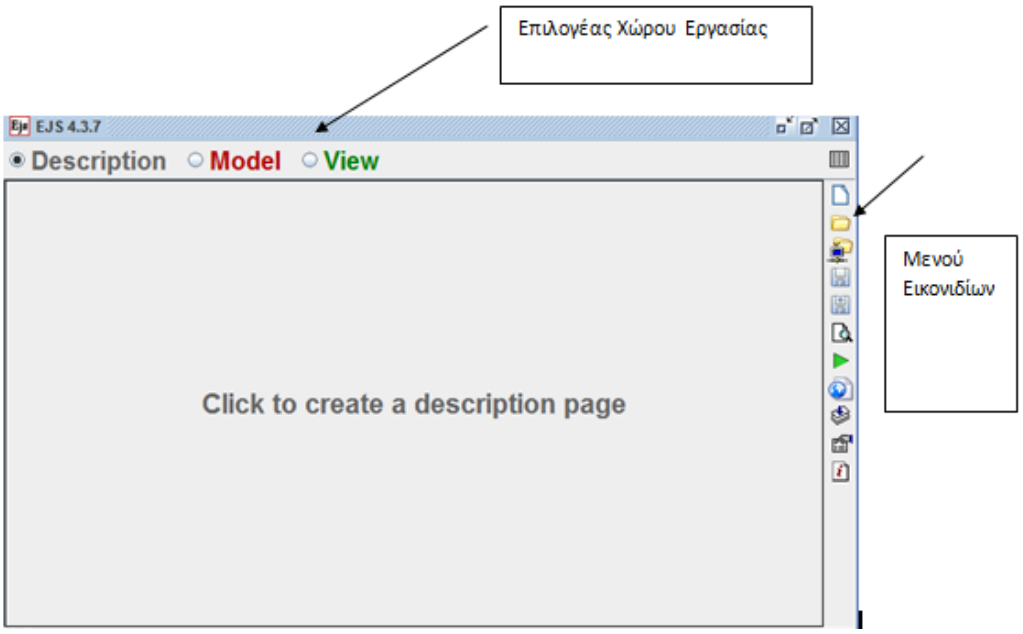
Το παρακάτω video tutorial δείχνει με ποιο τρόπο τα μοντέλα που έχουν αναπτυχθεί με το EJS μπορούν να τρέξουν σε άλλες γλώσσες εκτός από τα Αγγλικά.

http://www.compadre.org/OSP/tutorials/EJS_Translation/

http://www.phy.ntnu.edu.tw/ntnujava/swf/kepler_e.swf

1.2 Η διεπαφή του Ejs

Μετά το διπλό κλίκ στο Ejsconsole.jar η διεπαφή του Ejs είναι η παρακάτω.



Εικόνα: Η διεπαφή του Ejs

Στο δεξί μέρος του παραθύρου, βρίσκεται το μενού που περιλαμβάνει τα παρακάτω:

Εικόνα: Το μενού του Ejs

 . Πατώντας το εικονίδιο αυτό, κλείνουν όλες οι προσομοιώσεις(αν τρέχουν ήδη) και το Ejs επιστρέφει στην αρχική του κατάσταση ώστε να δημιουργηθεί μια νέα προσομοίωση.

 . Πατώντας το εικονίδιο αυτό ανοίγει μια ήδη υπάρχουσα προσομοίωση

 . Πατώντας το εικονίδιο αυτό ανοίγουν ψηφιακές βιβλιοθήκες του Ejs

Εικόνα: Οι ψηφιακές βιβλιοθήκες του Ejs

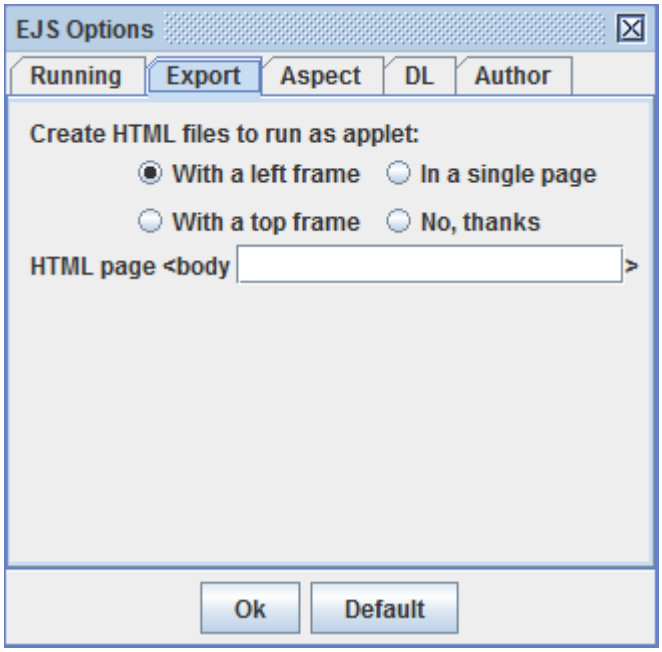
 Πατώντας το εικονίδιο σώζεται η τρέχουσα προσομοίωση στον ίδιο φάκελο από τον οποίο προήλθε.

 Πατώντας το εικονίδιο σώζεται η τρέχουσα προσομοίωση αλλά σε φάκελο που εμείς επιθυμούμε να δημιουργήσουμε

 Πατώντας το εικονίδιο , το Ejs δημιουργεί και τρέχει την τρέχουσα προσομοίωση. Στη συνέχεια αλλάζει το χρώμα από πράσινο σε κόκκινο και επιστρέφει στο πράσινο μόλις εξέλθουμε από την προσομοίωση

 Πατώντας το εικονίδιο, μπορούμε να μεταφράσουμε σε άλλη γλώσσα την διεπαφή της προσομοίωσης(βλ. παρακάτω)

 Πατώντας το εικονίδιο, έχουμε διάφορες επιλογές όπως εμφανίζονται στην παρακάτω εικόνα, σχετικά με π.χ. το τρέξιμο της εφαρμογής ως applet, τα στοιχεία του συγγραφέα κλπ.



Εικόνα: επιλογές για το Ejs

Πατώντας το εικονίδιο «πακετάρουμε» την εφαρμογή αυτή ή και άλλες εφαρμογές

Πατώντας το εικονίδιο, παίρνουμε πληροφορίες για την τρέχουσα εφαρμογή

Κεφάλαιο 2-Εργασία με υπάρχουσες προσομοιώσεις

2.1 Εργασία με μια προσομοίωση

2.2 Η δημοσίευση της εφαρμογής

2.3 Τροποποιώντας μια προσομοίωση

2.3.1. Οι μεταβλητές του μοντέλου

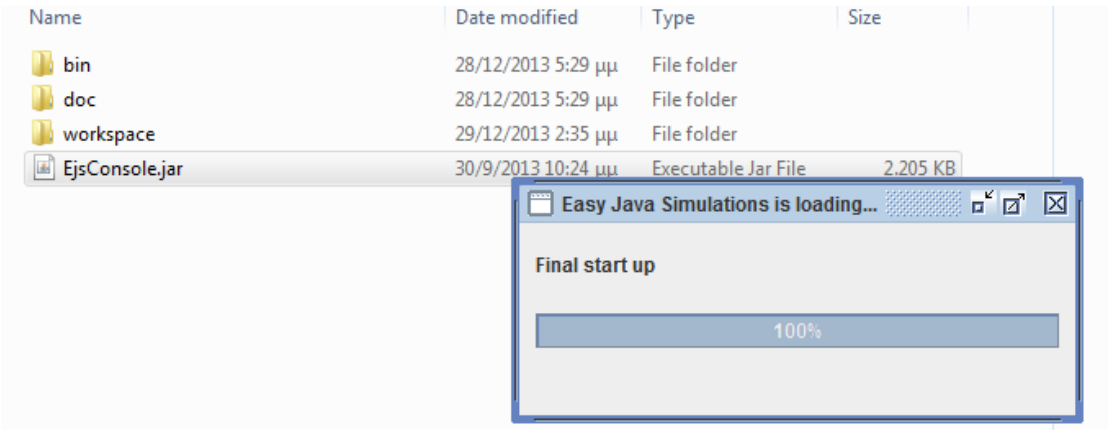
2.3.2 Η εξέλιξη του μοντέλου

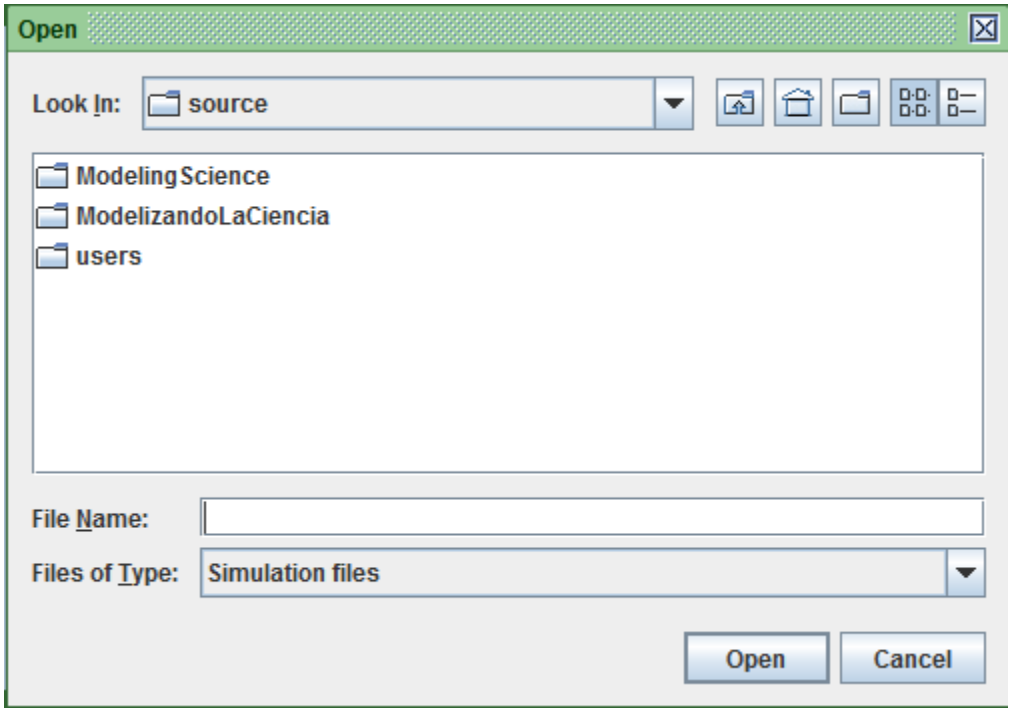
2.3.3 Εξέταση της θέασης του μοντέλου

2.1 Εργασία με μια προσομοίωση

Έχοντας εξοικειωθεί με το περιβάλλον του Ejs θα ασχοληθούμε με μία τρέχουσα προσομοίωση για να μελετήσουμε το μοντέλο της απλής αρμονικής ταλάντωσης.

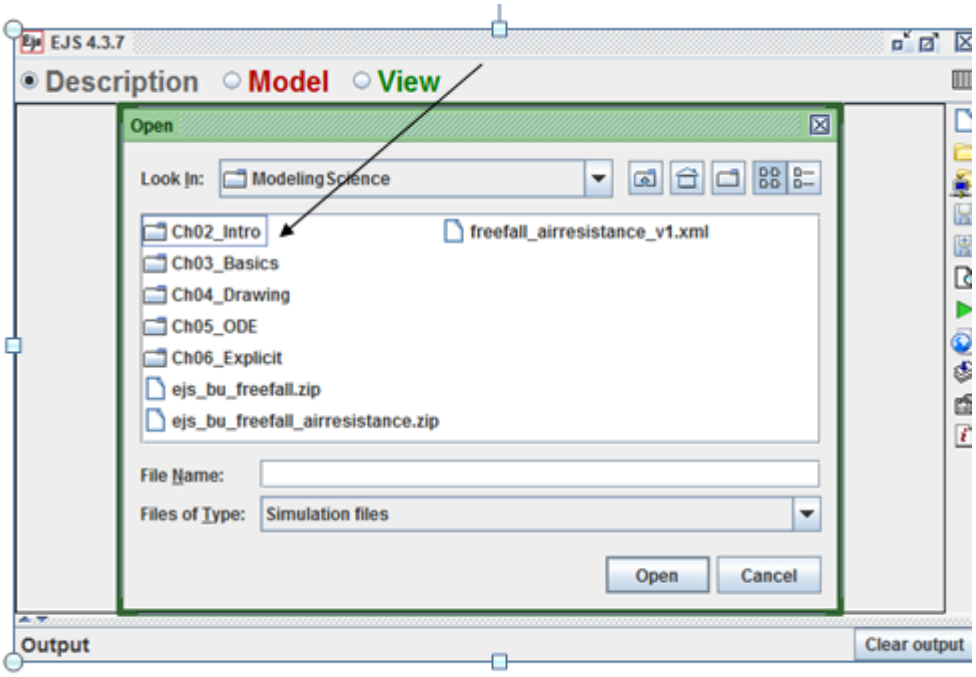
Αφού πατήσουμε στο Ejsconsole.jar και οδηγηθούμε στην διεπαφή του Ejs, πατάμε το εικονίδιο  για να ανοίξουμε μια υπάρχουσα εφαρμογή.



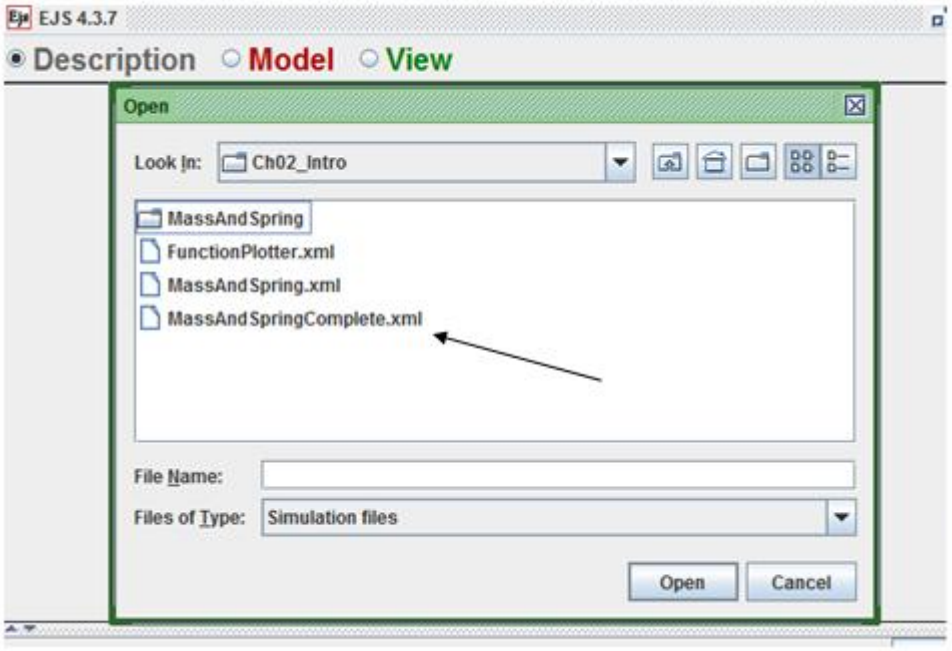


Εικόνα: Εύρεση μιας προσομοίωσης από την βιβλιοθήκη του Ejs

Από την επιλογή Modelling Science επιλέγουμε τον φάκελο Ch02_intro.

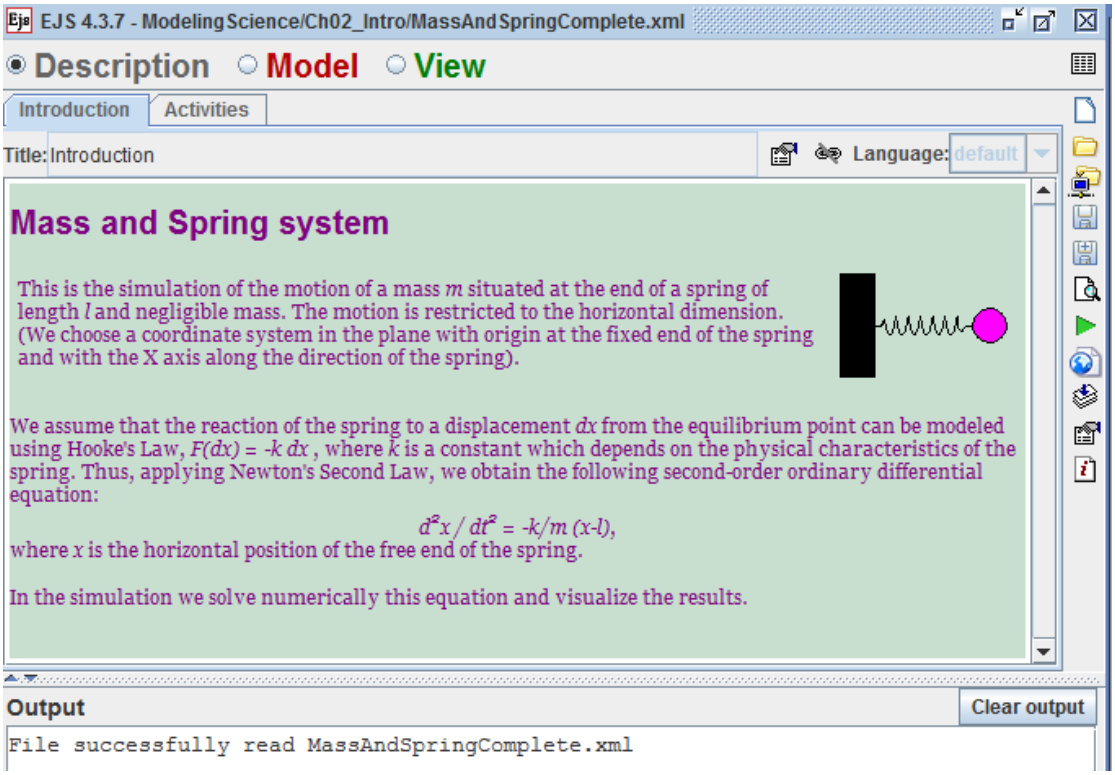


Εικόνα: Εύρεση της προσομοίωσης από τον φάκελο Ch02 από την βιβλιοθήκη του Ejs



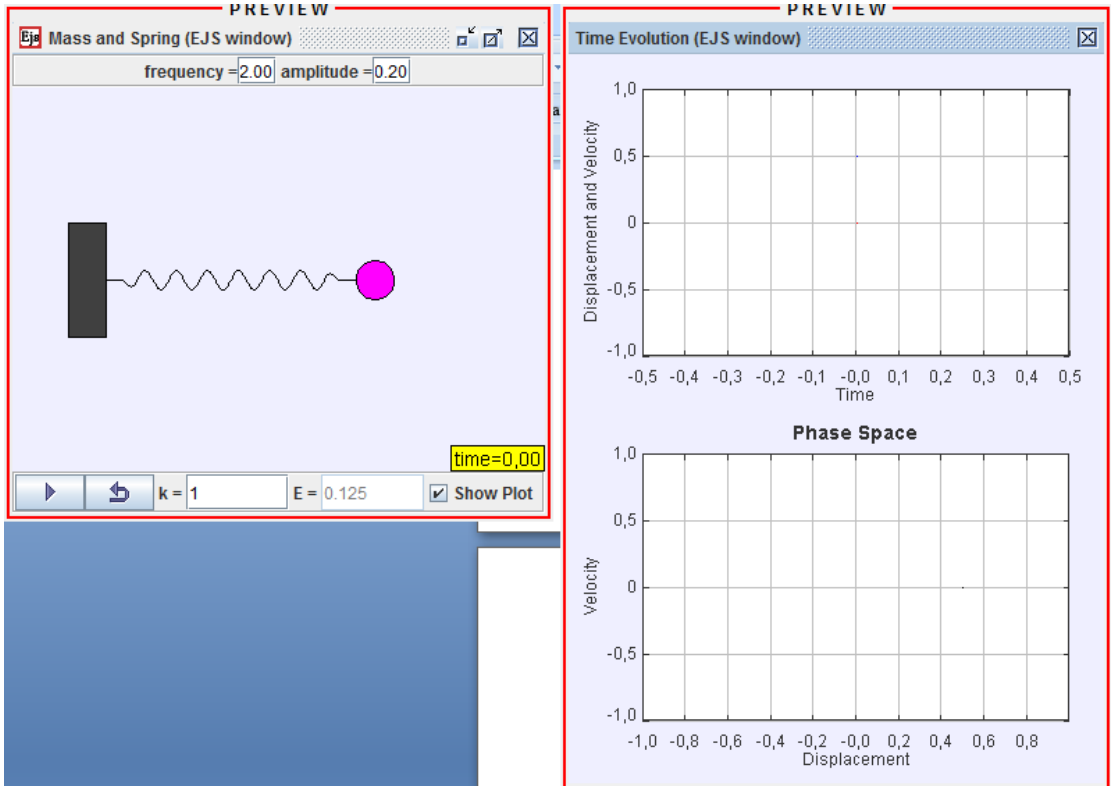
Εικόνα: Το αρχείο .xml της απλής αρμονικής ταλάντωσης

Μέσα στον φάκελο Ch02_intro, επιλέγουμε το αρχείο MassandSpringComplete.xml, όποτε εμφανίζεται(με διπλό κλίκ) η παρακάτω διεπαφή.



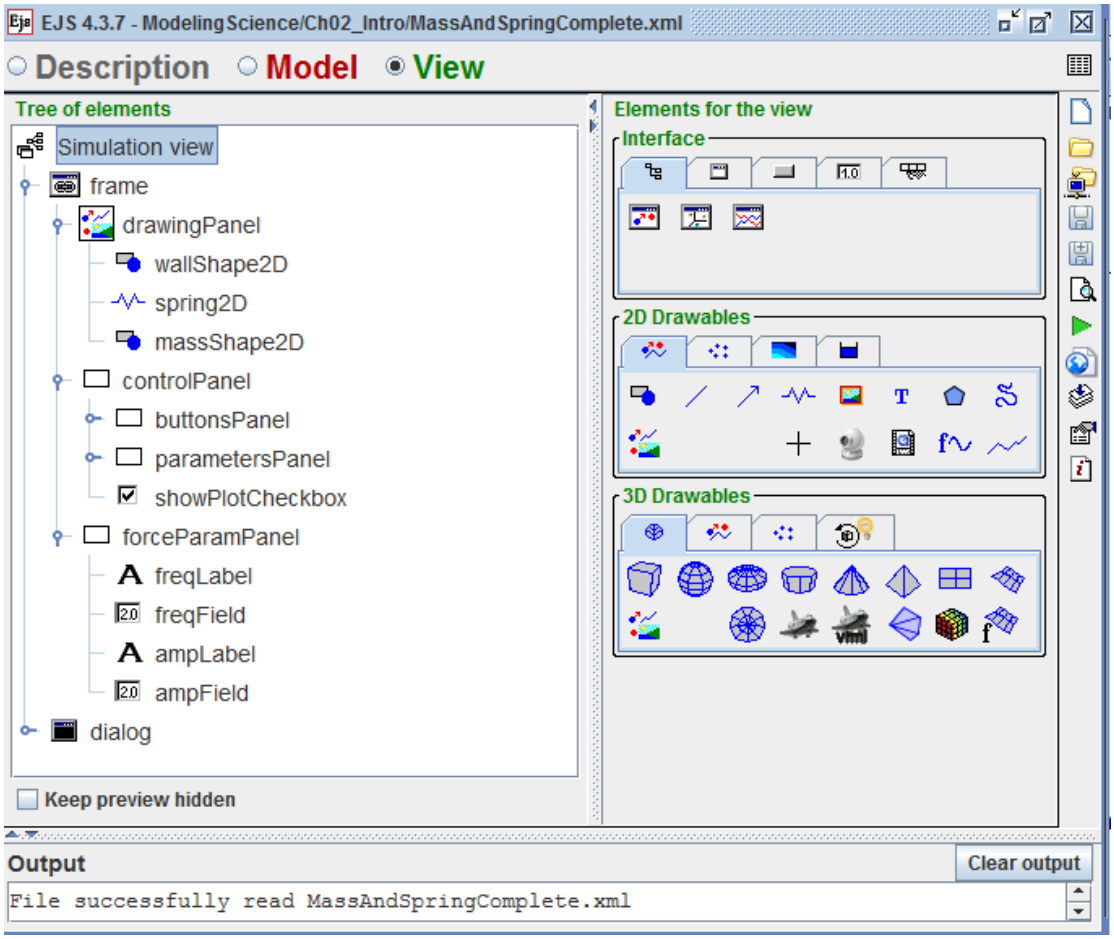
Εικόνα: Η προσομοίωση μιας απλής αρμονικής ταλάντωσης

Παρατηρούμε ότι μαζί με την απλή εισαγωγή εμφανίζονται και δυο ακόμα παράθυρα.



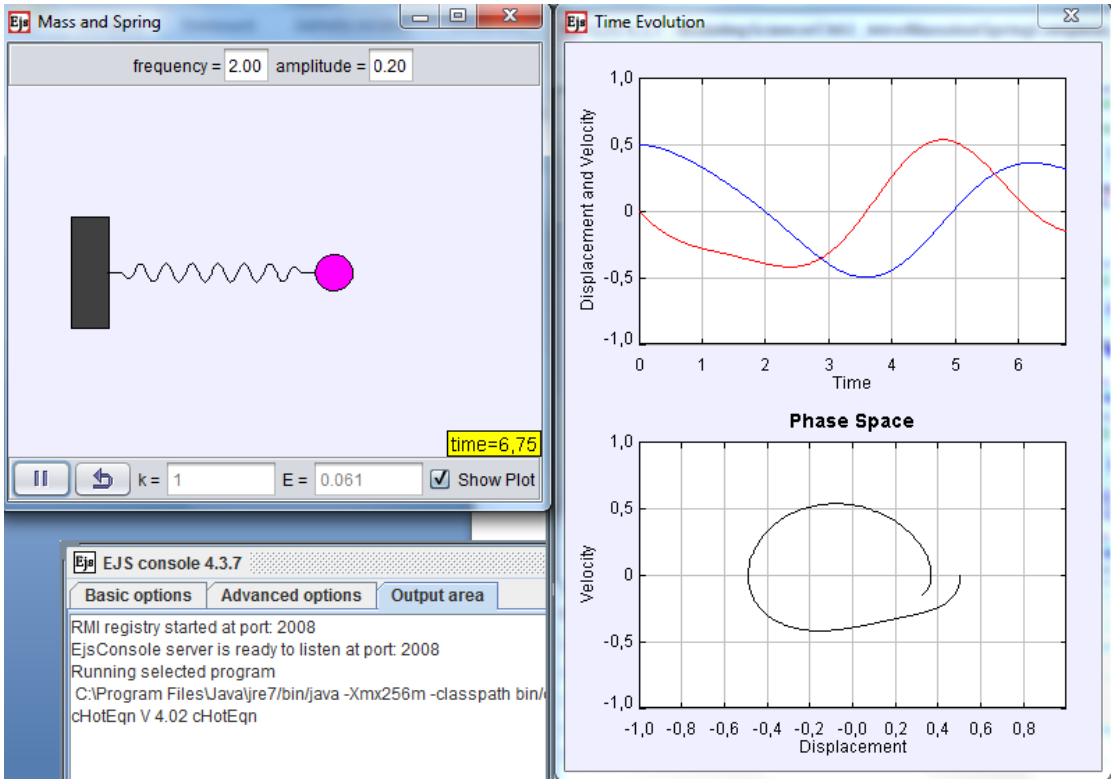
Εικόνα: Τα παράθυρα της θέασης

Τα δυο αυτά παράθυρα μπορείτε να τα μελετήσετε πατώντας το κουμπί View-Θέαση, όπου μπορείτε να μελετήσετε την δημιουργία των γραφικών του συγκεκριμένου μοντέλου προσομοίωσης.

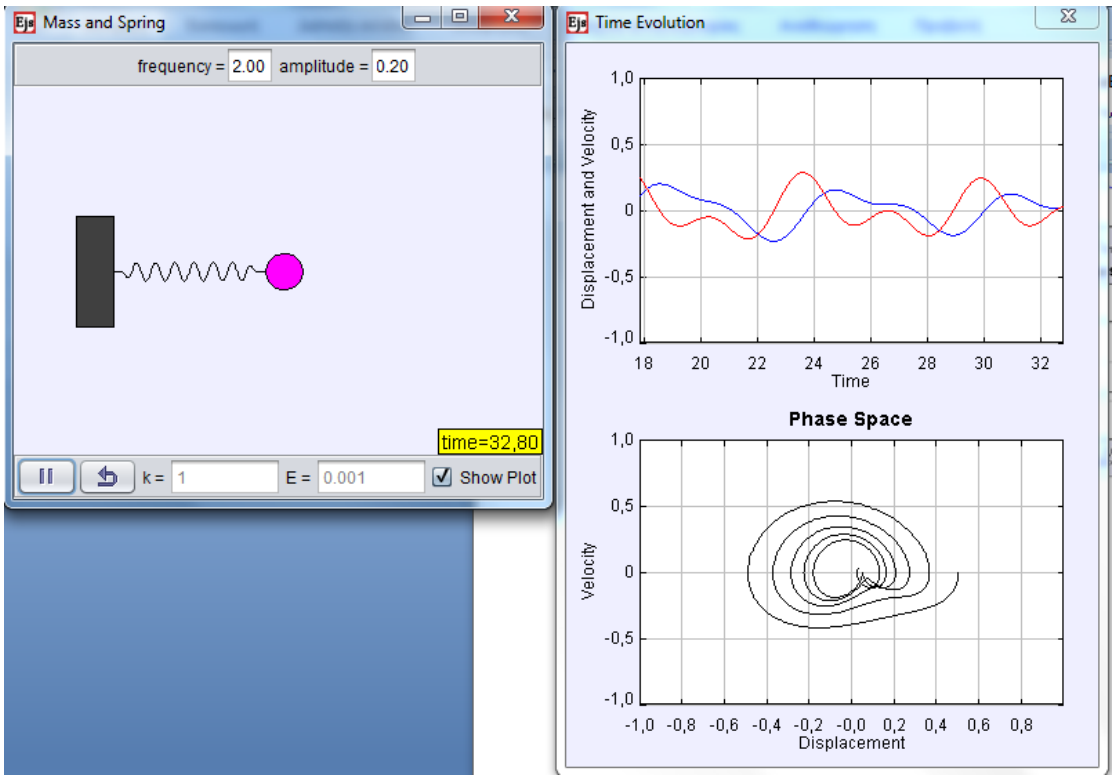


Εικόνα: τα σχεδιαστικά στοιχεία της εφαρμογής από το παράθυρο της θέασης

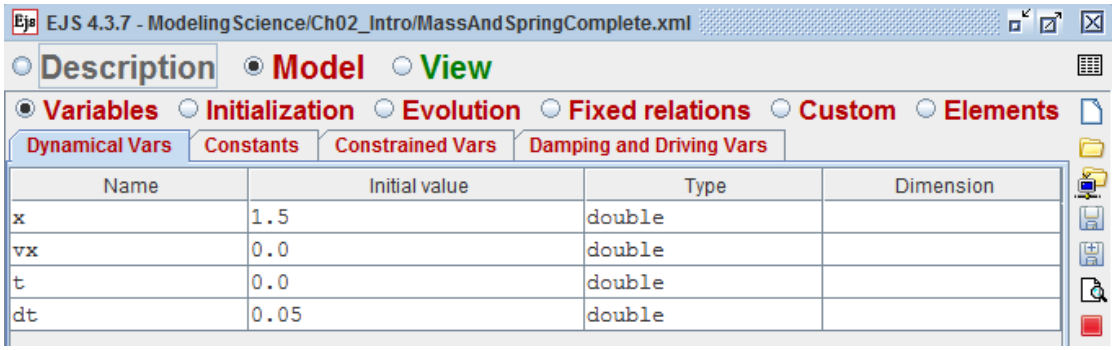
Κάνοντας κλίκ στο κουμπί «Play» θα αρχίσει να παίζει η προσομοίωση.



Εικόνα: οι γραφικές παραστάσεις στα παράθυρα της θέασης



Εικόνα: γραφικές παραστάσεις του μοντέλου με την εξέλιξη του χρόνου



Εικόνα: οι μεταβλητές μοντέλου

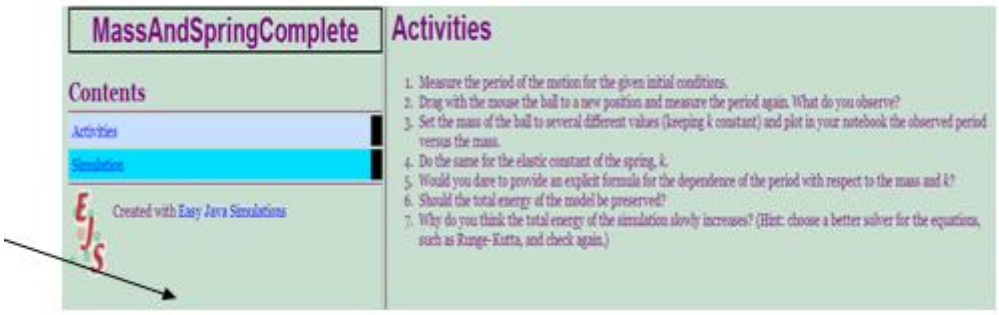
Μπορείτε να πειραματισθείτε με την παραπάνω προσομοίωση, εξετάζοντας π.χ. την σχέση της περιόδου με τις παραμέτρους του συστήματος και τις αρχικές συνθήκες.

Επίσης μπορείτε να αλλάξετε-χρησιμοποιώντας τα πεδία εισόδου- τις τιμές της μάζας και της σταθεράς του ελατηρίου.

2.2 Η δημοσίευση της εφαρμογής

Μπορούμε να ανεβάσουμε την εφαρμογή σε έναν Web Server.Το πλεονέκτημα του Ejs είναι ότι επειδή βασίζεται στην Java όλη η δουλειά έχει ήδη γίνει.

Έτσι μπορείτε να πατήσετε το κουμπί  και να εξάγετε την τρέχουσα εφαρμογή σε μορφή HTML(όπως θα δούμε παρακάτω αυτή αποθηκευεται στον φάκελο export).. Κάνοντας αυτό θα πρέπει να θυμηθείτε να συμπεριλάβετε τον φάκελο _library. Επίσης μπορούμε να εξάγουμε το αρχείο σε συμπιεσμένη μορφή .jar.



Εικόνα: Εξαγωγή του μοντέλου σε μορφή applet και σε μορφή .jar.

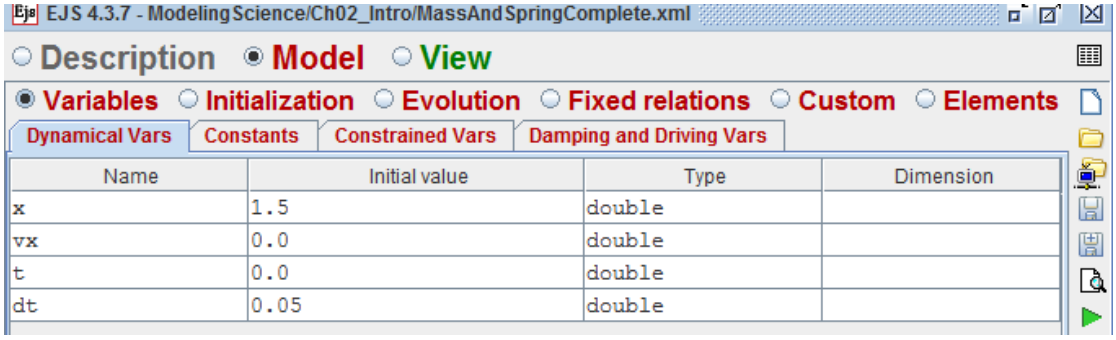
2.3 Τροποποιώντας μια προσομοίωση

Μπορούμε να τροποποιήσουμε υπάρχοντα μοντέλα ανοίγοντας το κουμπί «μοντέλο» και διερευνώντας τον τύπο των μεταβλητών και τις εξισώσεις-εκφράσεις του μοντέλου.

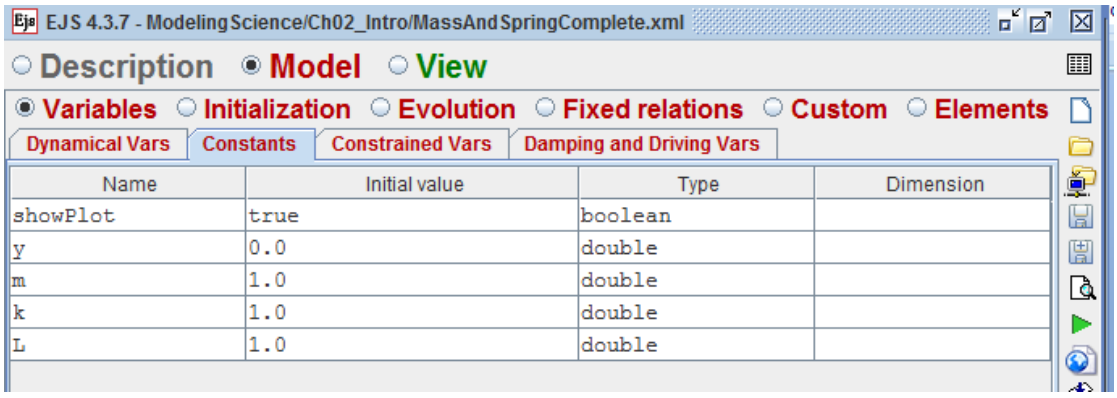
2.3.1. Οι μεταβλητές του μοντέλου

Θα ασχοληθούμε τώρα με το «έτοιμο» μοντέλο της ταλάντωσης ενός σώματος που είναι δεμένο σε ελατήριο. Θα χρησιμοποιούμε τον όρο «μεταβλητές» τόσο για τις παραμέτρους, όσο και για τις καταστατικές μεταβλητές καθώς και για τις εισόδους και εξόδους του μοντέλου. Κάθε αριθμητική τιμή(οποιουδήποτε τύπου, π.χ. λογική(boolean) ή «κειμένου») καλείται επίσης μεταβλητή ακόμα και όταν η τιμή της παραμένει σταθερή κατά τη διάρκεια της προσομοίωσης.

Για το παράδειγμα της ταλάντωσης που μελετάμε, οι μεταβλητές είναι οι:



Εικόνα: Οι μεταβλητές του μοντέλου



Εικόνα: Οι μεταβλητές του μοντέλου

Παρατηρούμε ότι μερικές μεταβλητές αντιστοιχούν σε παραμέτρους του συστήματος(οι m,K,l) άλλες περιγράφουν την κατάσταση του συστήματος (x,y,vx) και άλλες χρειάζονται για την προσομοίωση(t και dt).

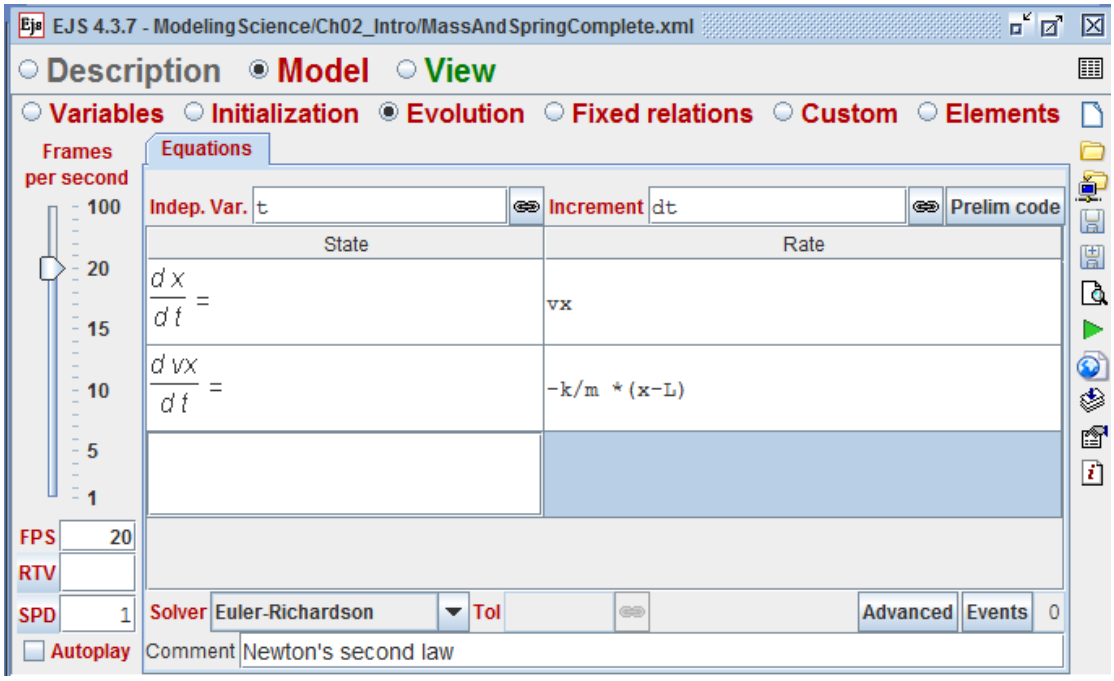
Αρχικοποίηση του μοντέλου

Οι μεταβλητές που θα χρησιμοποιηθούν πρέπει να αρχικοποιηθούν. Για το λόγο αυτό στον στην δεύτερη στήλη στις παραπάνω εικόνες παρατηρούμε αρχικές τιμές των μεταβλητών.

Παρατήρηση: Σε άλλες πολύ σύνθετες περιπτώσεις απαιτείται πρώτα υπολογισμός των αρχικών τιμών

2.3.2 Η εξέλιξη του μοντέλου

Στο panel με τίτλο «εξέλιξη» ουσιαστικά περιέχεται η προσομοίωση.
Το panel περιέχει δυο τύπους σελίδων: ο ένας τύπος αναφέρεται στον κώδικα Java που μπορεί να γραφεί και ο οποίος υλοποιεί τον αλγόριθμο του φαινομένου. Ο άλλος τύπος αναφέρεται στις διαφορικές εξισώσεις που θα χρησιμοποιηθούν.
Στο συγκεκριμένο παράδειγμα που αναφερόμαστε της ταλάντωσης, υλοποιείται ο δεύτερος τύπος, δηλαδή αναγράφεται η διαφορική εξίσωση του φαινομένου.
Στην παρακάτω εικόνα εμφανίζονται οι εξισώσεις του μοντέλου, που προέρχονται από τον δεύτερο νόμο του Netwton και τον Νόμο του Hooke.
Για το πρόβλημά μας επιλέγουμε ένα σύστημα συντεταγμένων με την αρχή στο σημείο που είναι δεμένο το ελατήριο και τον άξονα x κατά μήκος του ελατηρίου.



Εικόνα:οι διαφορικές εξισώσεις του μοντέλου

Αντί να γράψουμε μια διαφορική εξίσωση δευτέρας τάξης επιλέξαμε να γράψουμε 2 πρώτης τάξης, κάτι που έγινε για να είναι πιο οικείες σε μαθητές-φοιτητές.

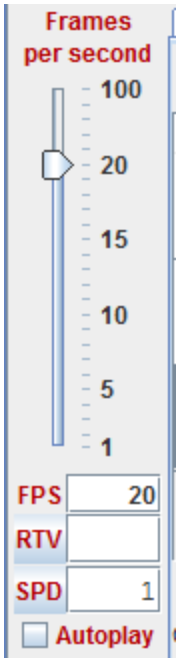
Έχουμε επίσης επιλέξει το t ως ανεξάρτητη μεταβλητή και έχουμε ζητήσει από το πρόγραμμα να παρέχει την λύση των εξισώσεων για αύξηση του χρόνου κατά dt.

Έχοντας δώσει όλη αυτή την πληροφόρηση, το πρόγραμμα θα δημιουργήσει όλο τον κώδικα που είναι απαραίτητος για την αριθμητική ανάλυση και τον υπολογισμό τα λύσης των διαφορικών εξισώσεων.

Για αυτό το σχετικά απλό πρόβλημα , έχουμε επιλέξει την μέθοδο Euler–Richardson method, η οποία παρέχει λογικές τιμές για το πηλίκο ταχύτητα/ακρίβεια.

Παρατήρηση: η αριθμητική επίλυση διαφορικών εξισώσεων είναι ένα υψηλού επιπέδου καθήκον-διεργασία και παρέχεται με έναν πολύ εύκολο τρόπο από το EJS. Βασικά το Ejs προσφέρει ακριβώς αυτό: **γράφουμε τις εξισώσεις και αυτό αυτόματα γεννά τον αντίστοιχο κώδικα.**

Τέλος θα πρέπει να προσέξουμε ότι αριστερά στην οθόνη υπάρχει ένα frame που αναφέρεται στα frames.

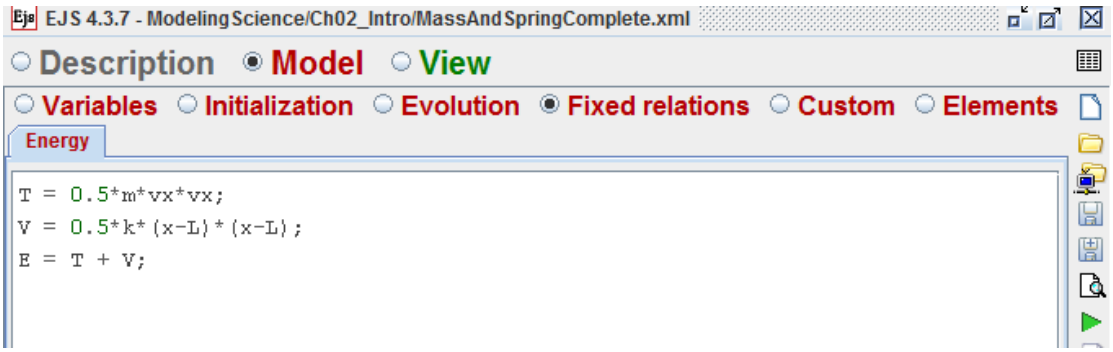


Εικόνα: η επιλογή των frames/sec

Έχοντας επιλέξει 20 βήματα ανά δευτερόλεπτο , αυτό συνεπάγεται ένα βήμα κάθε 0,05 δευτερόλεπτα. Αν σκεφθείτε ότι έχουμε επιλέξει για την μεταβλητή dt=0,05 αυτό σημαίνει ότι καταλήγουμε σε μια προσομοίωση η οποία τρέχει περίπου σε πραγματικό χρόνο, αν και αυτό δεν είναι αναγκαίο για κάθε προσομοίωση.
Παρατηρήστε επίσης ότι το κουμπί "Autoplay" δεν έχει επιλεγεί. Αυτό θα επιλέγεται κάθε φορά που θέλουμε η προσομοίωση να τρέχει αυτόματα .

Σύνδεσμοι ανάμεσα στις μεταβλητές

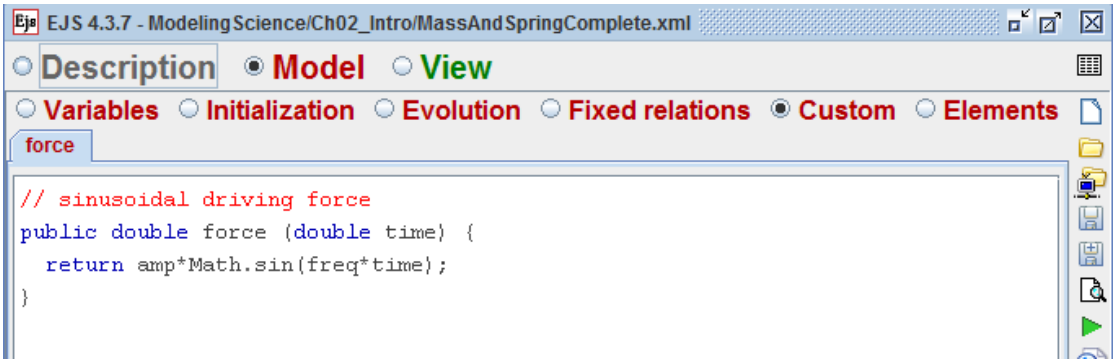
Το panel με τίτλο «Fixed relations» χρησιμοποιείται για να γραφούν οι σχέσεις μεταξύ των μεταβλητών του συστήματος.
Οι σχέσεις αυτές είναι δεδομένες και δεν αλλάζουν, όπως π.χ. οι «δεσμοί» των διαφόρων μορφών ενέργειας.



Εικόνα: οι «Fixed relations» του μοντέλου

Οι μέθοδοι Custom (Προσαρμογής)

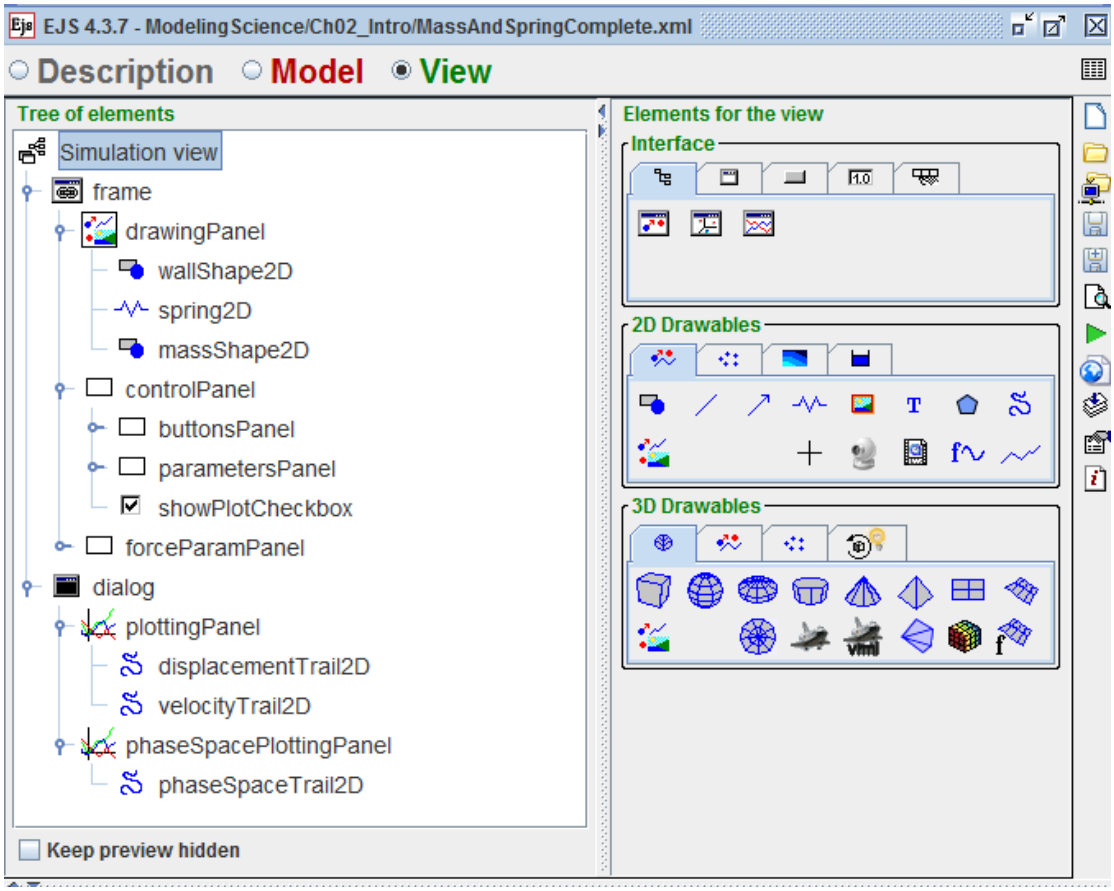
Αυτές οι μέθοδοι είναι αυτές που γράφει ο προγραμματιστής.
Για το παράδειγμα που μελετάμε έχει γραφεί μια μέθοδος (με το όνομα force και με παράμετρο την time) , η οποία επιστρέφει την τιμή «πλάτος. ημ(συχνότητα. χρόνος)».
Η μέθοδος αυτή δεν χρησιμοποιείται για την απλή αρμονική ταλάντωση αλλά για την εξαναγκασμένη, όπως θα δούμε τροποποιώντας το μοντέλο που έχουμε περιγράψει έως τώρα.



Εικόνα: οι μέθοδοι Custom

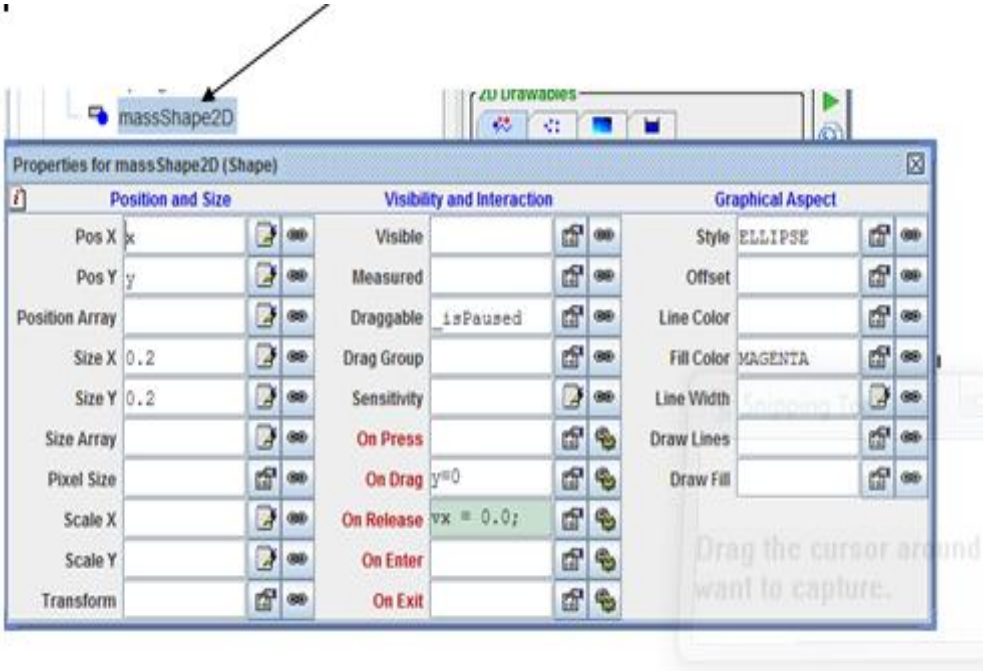
2.3.3 Εξέταση της Θέασης του μοντέλου

Στο πρώτο στάδιο της θέασης δημιουργούμε το δέντρο της προσομοίωσης



Εικόνα: η θέαση του μοντέλου

Στο δεύτερο στάδιο εξετάζουμε τις ιδιότητες των στοιχείων της θέασης.
Σε αυτό το στάδιο μπορεί να γίνει διαχείριση των ιδιοτήτων κάθε στοιχείου.
Οι ιδιότητες είναι εσωτερικές τιμές που προσδιορίζουν με ποιο τρόπο εμφανίζεται αλλά και συμπεριφέρεται το στοιχείο στην οθόνη.
Έτσι μπορεί να ενδιαφερόμαστε στην τροποποίηση των ιδιοτήτων σε σχέση με στατικά γραφικά(π.χ. χρώμα, γραμματοσειρά κλπ) ή να καθοδηγήσουμε την ιδιότητα για να μεταβληθούν δυναμικά χαρακτηριστικά (π.χ. θέση , μέγεθος) στην οθόνη.
Αυτή η δεύτερη δυνατότητα καθιστά την προσομοίωση πραγματικά δυναμική και αλληλεπιδραστική .

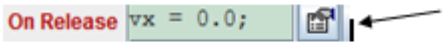


Εικόνα: οι ιδιότητες του στοιχείου massShape

Το παράθυρο δείχνει τον Πίνακα ιδιοτήτων αυτού του στοιχείου. Οι στατικές τιμές δείχνουν την εμφάνιση που θα έχει το σωματίδιο, δηλαδή το χρώμα του και το σχήμα του (έλλειψη μεγέθους 0.2 και 0.2 , άρα κύκλος) και επίσης ότι αυτό το στοιχείο ενεργοποιείται , δηλαδή αποκρίνεται στην αλληλεπίδραση όταν ο χρήστης κάνει κλίκ με το ποντίκι του.
Άλλες ιδιότητες του στοιχείου είναι πιο ενδιαφέρουσες από τις παραπάνω και αναφέρονται στην συσχέτιση αυτών με μεταβλητές του μοντέλου ή με χρήση εκφράσεων που χρησιμοποιούν τις μεταβλητές του μοντέλου. Για παράδειγμα, η θέση του στοιχείου σχετίζεται με το σημείο με συντεταγμένες(x,y) όπου αυτές είναι οι αντίστοιχες μεταβλητές του μοντέλου.
Αυτή η συσχέτιση είναι αμφίδρομη. Από τη μια κατεύθυνση σημαίνει ότι κάθε φορά που αλλάζουν οι τιμές των μεταβλητών, τότε το στοιχείο ειδοποιείται και οι αντίστοιχες ιδιότητες λαμβάνουν νέα τιμές. Στην άλλη κατεύθυνση κάθε φορά που ο χρήστης με το μπαλάκι αλλάζοντας την θέση του, τότε υπάρχει αυτόματη αλλαγή των τιμών των μεταβλητών x,y.
Τέλος, ο τρόπος που το μπαλάκι «αντιδρά» όταν αλληλεπιδρά με τον χρήστη καθορίζεται από τις λεγόμενες «action properties» του στοιχείου, οι οποίες χρησιμοποιούν μεταβλητές του μοντέλου ή μεθόδους αυτού.

Για παράδειγμα, πατώντας το κουμπί της παρακάτω εικόνας, εμφανίζεται ο κώδικας

```
vx = 0.0;            // sets the velocity to zero
_view.resetTraces(); // clears all plots
```



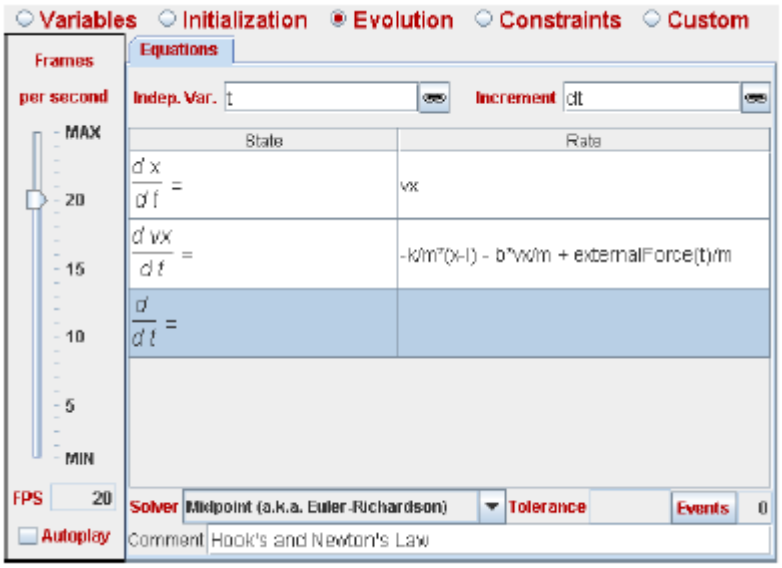
Εικόνα: οι «ενέργειες» του στοιχείου massShape

Θα τροποποιήσουμε τώρα το μοντέλο ώστε να περιλαμβάνει τριβή και εξαναγκασμένη ταλάντωση.

Η εξίσωση είναι η:

$$\ddot{x} = -\frac{k}{m}(x - l) - \frac{b}{m}\dot{x} + \frac{1}{m}f_e(t)$$

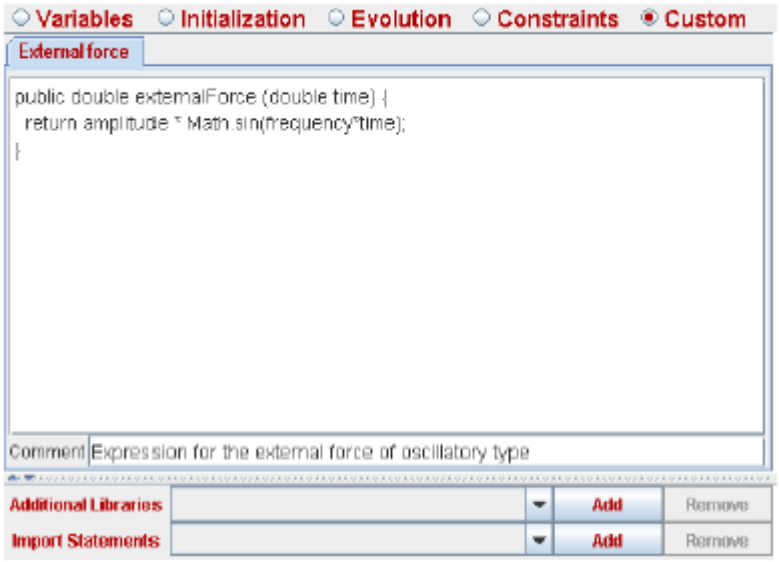
Όπου b ο συντελεστής απόσβεσης και fe(t) είναι μια εξωτερική δύναμη που εφαρμόζεται στο σύστημα.
Εστω ότι η εξωτερική δύναμη είναι της μορφής fe(t) = A sin(f.t), η οποία μας επιτρέπει να μελετήσουμε φαινόμενα συντονισμού.
Οι διαφορικές εξισώσεις του μοντέλου αναγράφονται στο panel «μοντέλο» ως εξής:



Εικόνα: οι εξισώσεις του μοντέλου με απόσβεση και εξωτερική δύναμη

Για να ολοκληρώσουμε το μοντέλο μας χρειάζεται να γράψουμε μια έκφραση για την εξωτερική δύναμη . Για να γίνει αυτό πηγαίνουμε στο subpanel "Custom" και κλικάρουμε πάνω του για να δημιουτγήσουμε μια σελίδα που καλείται "External force" και στην οποία γράφουμε τον παρακάτω κώδικα.

```
public double externalForce (double time) {
return amplitude * Math.sin(frequency*time);
}
```



Εικόνα: η μέθοδος για την εξωτερική δύναμη

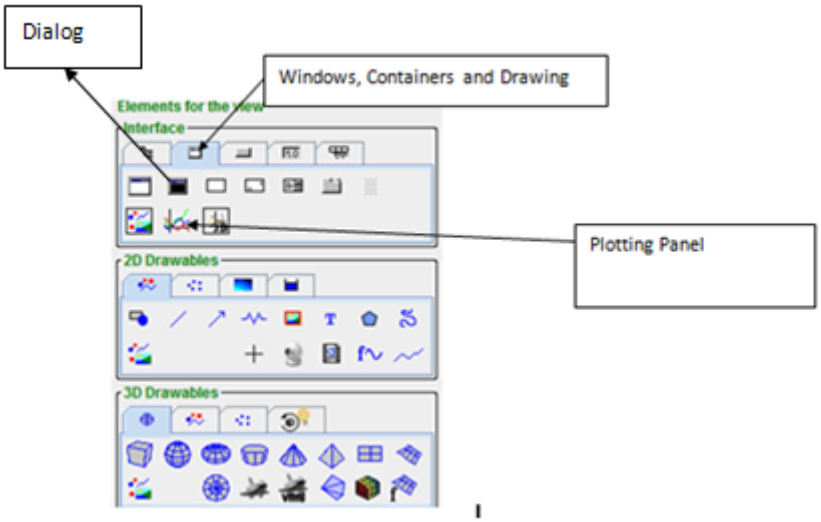
Παρατήρηση: Ο κώδικας περιέχει μια μέθοδο που επιστρέφει μια αριθμητική τιμή. **Η έκφραση Math.sin** αντιστοιχεί στην κλήση της συνάρτησης του ημιτόνου χρησιμοποιώντας την μαθηματική βιβλιοθήκη της Java. Παρατηρήστε ότι αυτή η μέθοδος δέχεται ως παράμετρο την μεταβλητή time, η οποία χρησιμοποιείται μόνο μέσα στην μέθοδο και θεωρείται ως τοπική μεταβλητή, ενώ χρησιμοποιεί τις global μεταβλητές πλάτος και συχνότητα(amplitude, frequency).
Η χρήση της παραμέτρου time είναι σημαντική **και θα ήταν λάθος να γράφαμε τον παρακάτω κώδικα που χρησιμοποιεί απευθείας την καθολική μεταβλητή t.**

```
public double externalForce () {
return amplitude * Math.sin(frequency*t);
}
```

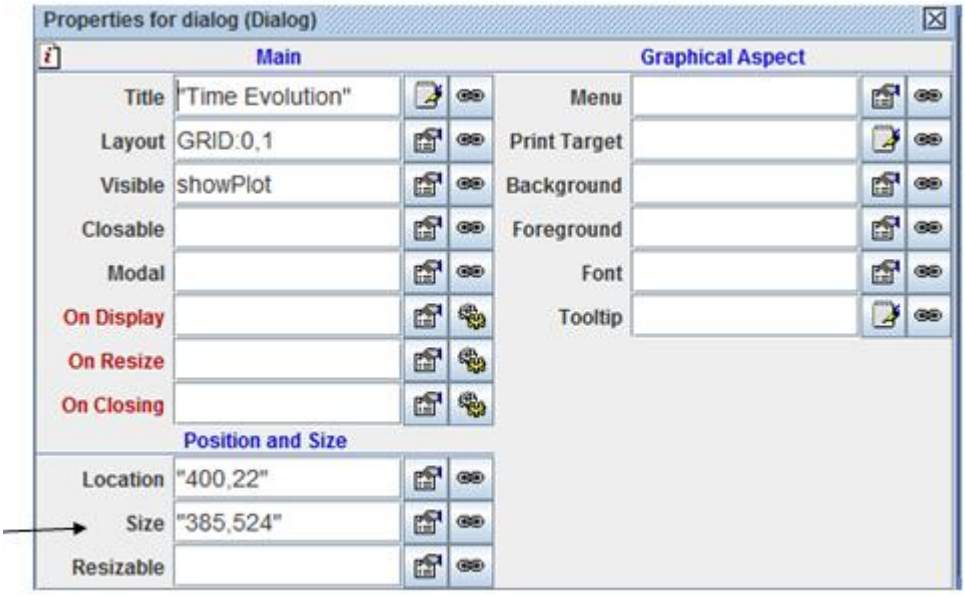
Η μεταβλητή t αλλάζει κατά την διάρκεια της διαδικασίας της λύσης της διαφορικής εξίσωσης και αυτό αλλάζει την ακρίβεια του αριθμητικού αλγόριθμου.

Αν τώρα θελήσουμε να εμπλουτίσουμε την θέαση με π.χ. το διάγραμμα φάσης θέσης-ταχύτητας, η διαδικασία που ακολουθούμε είναι η εξής:

Στην δεξιά πλευρά της οθόνης εμφανίζεται η παρακάτω εικόνα, από την οποία επιλέγουμε το κουμπί  (Windows, Containers and Drawing Panels) και από αυτό το κουμπί «Dialog» μέσα στο οποίο θα βάλουμε το «PlottingPanel».

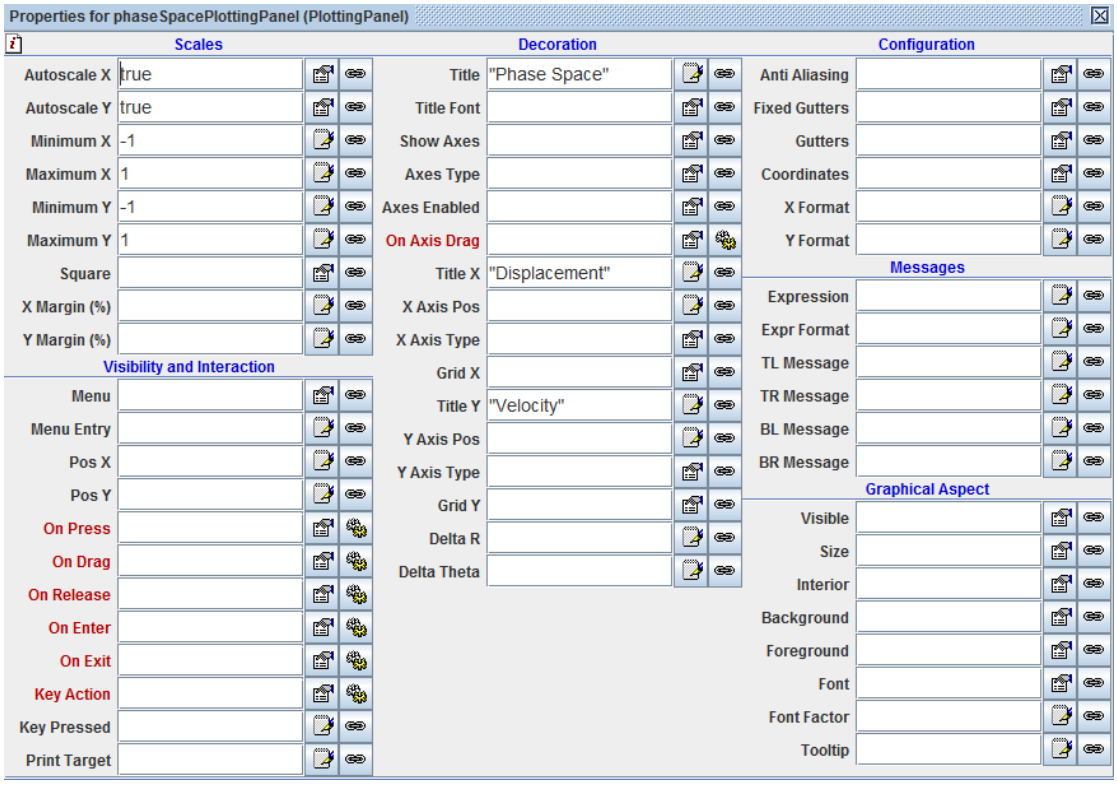


Εικόνα: Εισαγωγή γραφικών στην θέαση του μοντέλου



Εικόνα: Τα χαρακτηριστικά του στοιχείου «Dialog».

Αν θέλουμε να αλλάξουμε τα μέγεθος του παραθύρου «íDialog» αυτό γίνεται από την επιλογή Size.

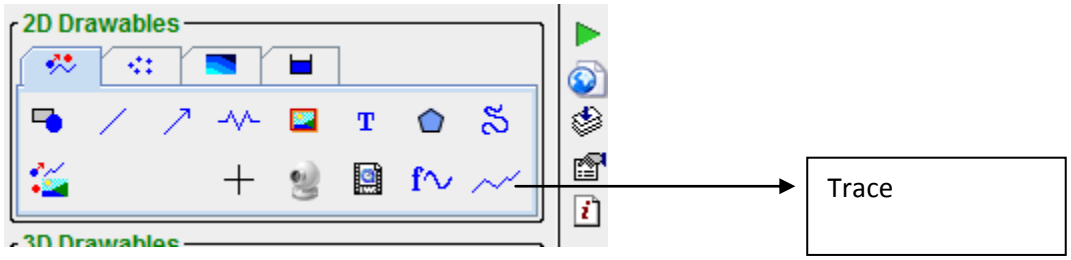


Εικόνα: Τα χαρακτηριστικά του Panel

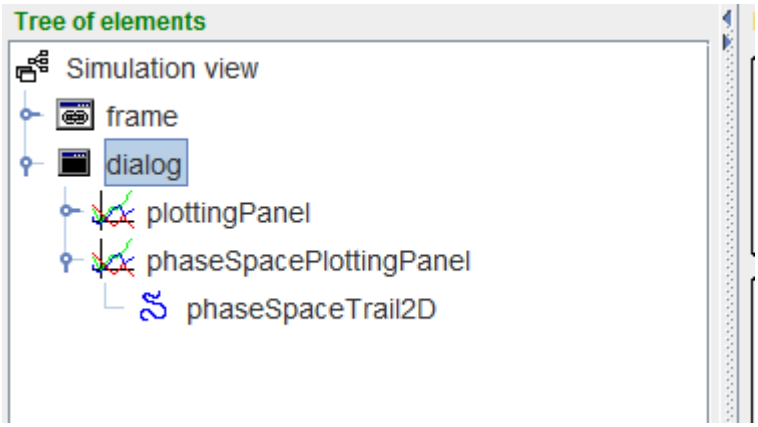
Παρατήρηση: Αν προσπαθήσετε να τροποποιήσετε κάποιο στοιχείο κειμένου, το Ejs θα βγάλει μια ένδειξη με κίτρινο χρώμα που υποδηλώνει ότι για να γίνει αποδεκτή η αλλαγή σας θα πρέπει να πατήσετε το κουμπί «return». Μια εξαίρεση αυτού του κανόνα αφορά τις λεγόμενες ιδιότητες «action». Αυτά τα πεδία είναι ειδικά και κάθε τροποποίηση αυτών γίνεται άμεσα αποδεκτή. Η μόνη προειδοποίηση από αυτά τα στοιχεία(με πράσινο χρώμα) εμφανίζεται όταν επεκτείνονται σε περισσότερο από μια γραμμή χώρο.

Το plotting panel που προσθέσαμε λειτουργεί μόνο ως υποδοχέας για την γραφική παράσταση που επιθυμούμε να προσθέσουμε.

Για την οπτικοποίηση της γραφικής παράστασης χρειαζόμαστε ένα στοιχείο τύπου "Trace" το οποίο μπορείτε να βρείτε από τα δισδιάστατα σχεδιαστικά στοιχεία(2D Drawables) από το δεξιό μέρος της οθόνης.

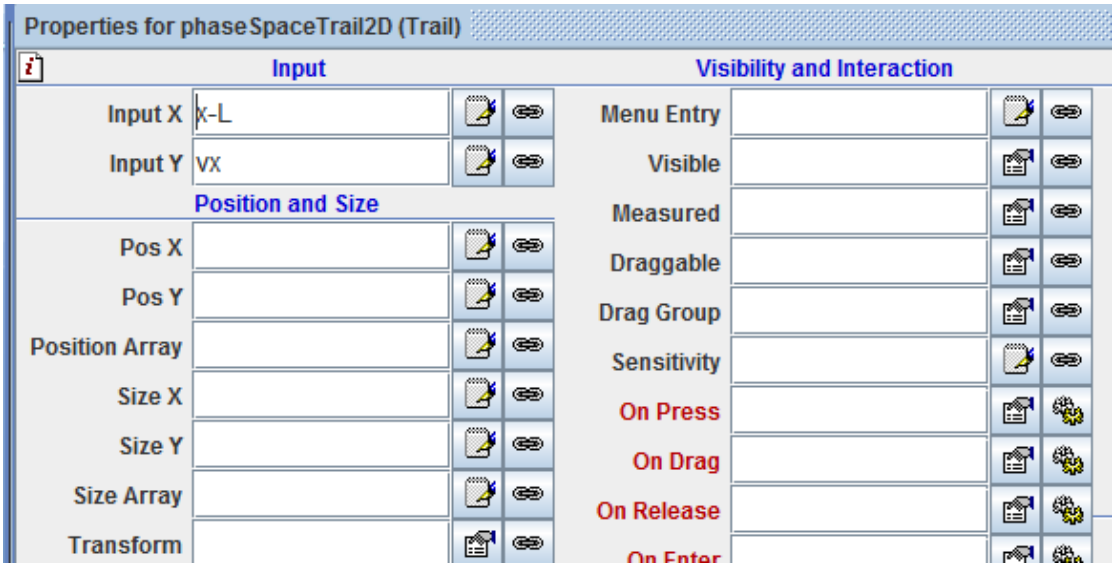


Εικόνα: Το panel με τα 2D Drawables



Εικόνα: η εισαγωγή του Trace στο δέντρο της προσομοίωσης

Μέσα στο Trace της προσομοίωσης τροποποιούμε τις εισόδους X,Y σε x-L και vx αντίστοιχα.



Εικόνα: Το περιεχόμενο του Trace στο δέντρο της προσομοίωσης

Άσκηση.

Να προσθέσετε ένα υποδοχέα Dialog και μέσα σε αυτόν, ακολουθώντας τις προηγούμενες διαδικασίες, να εισάγετε τρία Trace όπου με διαφορετικά χρώματα να γίνεται η γραφική παράσταση της κινητικής, της δυναμικής και της ολικής ενέργειας.

Κεφάλαιο 3. Η δημιουργία Μοντέλων στο Ejs

3.1 Ο Ορισμός του μοντέλου- Η κατάσταση του συστήματος

3.1.1 Η Εξέλιξη του συστήματος-Οι εξισώσεις συνδέσμων

3.1.2 Το «τρέξιμο» του Μοντέλου

3.2 Η Διεπαφή του μοντέλου στο EJS

3.3 Η Δήλωση μεταβλητών στο μοντέλο

3.4 Η Αρχικοποίηση του μοντέλου

3.5 Η Εξέλιξη του μοντέλου

3.5.1 Εισαγωγή

3.5.2 Οι εξισώσεις εξέλιξης του μοντέλου

3.5.3 Η επιλογή του αριθμητικού αλγορίθμου

3.5.4 Τα γεγονότα μιας διαφορικής εξίσωσης

3.6 Οι Εξισώσεις συνδέσμων

3.7 Οι Μέθοδοι προσαρμογής

3.7.1 Εισαγωγή

3.7.2 Γραφή μεθόδων προσαρμογής

3.7.3 Μέθοδοι ορισμένες από το Ejs

3.1 Ο ορισμός του Μοντέλου-Η κατάσταση του συστήματος

Όταν λέμε ότι ορίζουμε το μοντέλο(εννοιολογικό μοντέλο) εννοούμε ότι ορίζουμε τις καταστατικές μεταβλητές ή τις παραμέτρους ή οποιαδήποτε άλλη σχετική ποσότητα εισόδου , θέτουμε τις αρχικές τιμές τους και γράφουμε τους νόμους εξέλιξής τους.

Η κατάσταση του μοντέλου προσδιορίζεται από την γνώση των τιμών των μεταβλητών σε μια δεδομένη χρονική στιγμή.

Η κατάσταση του συστήματος μπορεί να αλλάξει για δύο λόγους:

1. Λόγω της εσωτερικής δυναμικής της προσομοίωσης, όπου σε αυτή την περίπτωση αναφερόμαστε σε εξέλιξη του συστήματος

2. Την επίδραση εξωτερικών παραγόντων λόγω της αλληλεπίδρασης π.χ. του χρήστη με την προσομοίωση, όπου ο χρήστης αλλάζει την τιμή μιας η περισσοτέρων παραμέτρων.

Οι αλλαγές που προκαλούνται από τους παραπάνω λόγους μπορεί να προκαλέσουν άλλες αλλαγές με έμμεσο τρόπο. Αυτό συμβαίνει για παράδειγμα όταν υπάρχουν μεταβλητές στο μοντέλο των οποίων οι τιμές εξαρτώνται εκπεφρασμένα από τις τιμές των μεταβλητών που μεταβάλλονται. Σε αυτή την περίπτωση θεωρούμε ότι υπάρχουν σύνδεσμοι (constraints) ανάμεσα στις μεταβλητές.

Συνοψίζοντας: για να προσδιορίσουμε το μοντέλο μιας προσομοίωσης, θα πρέπει;

- να ορίσουμε τις μεταβλητές του μοντέλου
- την αρχική κατάσταση του συστήματος
- τις εξισώσεις εξέλιξης
- τις εξισώσεις συνδέσμων

Για τον ορισμό των μεταβλητών υποθέτουμε ότι οι μεταβλητές καλούνται x1, x2, . . . , xn. . Για να είναι καλύτερα κατανοητό το μοντέλο μας, είναι προτιμότερο να δίνονται στις μεταβλητές ονόματα-ακρωνύμια που από αυτά είναι εύκολο ο χρήστης του μοντέλου να καταλάβει τι αντιπροσωπεύει η μεταβλητή.

Για τον προσδιορισμό της αρχικής κατάστασης του συστήματος θα πρέπει να δοθούν οι σωστές τιμές στις μεταβλητές. Αυτό μπορεί να γίνει γράφοντας απλά την σωστή τιμή . Ωστόσο υπάρχουν περιπτώσεις όπου η αρχική τιμή μιας μεταβλητής λαμβάνεται μετά από υπολογισμούς που οδηγούν στη τιμή αυτής της μεταβλητής.

3.1.1 Η εξέλιξη του συστήματος-Οι εξισώσεις συνδέσμων

Το σύστημα μπορεί να εξελιχθεί από μια τρέχουσα κατάσταση $x_1, x_2, \ldots, x_n$, σε μία νέα κατάσταση. Η νέα κατάσταση περιγράφεται από μια εξίσωση της μορφής

$x^*_i = fi(x_1, x_2, \ldots, x_n)$.

Ορισμένες φορές οι νόμοι εξέλιξης έχουν μια άμεση μαθηματική έκφραση όπως συμβαίνει στα λεγόμενα διακριτά συστήματα.
Σε άλλες περιπτώσεις οι νόμοι εξέλιξης προέρχονται από συνεχή μοντέλα που περιγράφονται από διαφορικές εξισώσεις.
Στις πιο συνηθισμένες περιπτώσεις ωστόσο οι νόμοι εξέλιξης προέρχονται από ένα συνδυασμό αριθμητικών τεχνικών και αλγορίθμων που καταλήγουν σε έναν αλγόριθμο που υλοποιείται στον υπολογιστή.
Συμπερασματικά, η προσομοίωση της χρονικής εξέλιξης ενός συστήματος συνίσταται των τιμών των x^*_i μεταβλητών , χρησιμοποιώντας τις τιμές $x_1, x_2, \ldots, x_n$..Στη συνέχεια η διαδικασία επαναλαμβάνεται αναδρομικά κοκ.

Το τελευταίο βήμα για την δημιουργία ενός μοντέλου είναι η συγγραφή των εξισώσεων συνδέσμων. Όπως αναφέρθηκε προηγουμένως, οι αλλαγές στις μεταβλητές που προκαλούνται από την χρονική εξέλιξη, επηρεάζουν έμμεσα άλλες μεταβλητές. Αυτές οι έμμεσες αλλαγές προσδιορίζονται από τις εξισώσεις-συνδέσμους.

Αυτές οι εξισώσεις έχουν την γενική μορφή expressions of the form $x_i = g_i(x_1, x_2, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$ **όπου η μεταβλητή θα πρέπει να εμφανίζεται μόνο στο ένα μέρος της εξίσωσης.**

Οι παραπάνω εξισώσεις υποδεικνύουν ότι αν μια η περισσότερες μεταβλητές του δεξιού μέλους αλλάξουν τιμή, τότε η μεταβλητή του αριστερού μέλους αλλάζει τιμή. Όπως και στην περίπτωση των εξισώσεων εξέλιξης, απαιτείται η συγγραφή έξυπνων αλγορίθμων που θα περιέχουν και αριθμητικές τεχνικές.
Παρατήρηση: Μια συνηθισμένη πρακτική-αλλά άσχημη πρακτική!- είναι να θεωρούνται οι εξισώσεις για τους συνδέσμους ως μέρος των εξισώσεων εξέλιξης.
Ο λόγος που θα πρέπει να αποφεύγεται να αναγράφονται οι εξισώσεις συνδέσμων ως εξισώσεις εξέλιξης είναι ότι οι σύνδεσμοι πρέπει να ισχύουν ακόμα και όταν ο χρήστης αλλάζει κάποια μεταβλητή ενώ η προσομοίωση έχει σταματήσει.

Ένα χρήσιμο κριτήριο για να διακρίνουμε τους δυο τύπους εξισώσεων είναι να εξετάσουμε την τιμή μιας μεταβλητής μια ορισμένη χρονική στιγμή, και αν η αυτή η τιμή εξαρτάται από την τρέχουσα τιμή της ίδιας μεταβλητής, τότε το πιο πιθανό σενάριο είναι να γραφεί εξίσωση εξέλιξης.
Αν αντίθετα η τιμή της μεταβλητής εξαρτάται από την τιμή άλλων μεταβλητών, τότε θα πρέπει να γραφεί εξίσωση ως συνδέσμου.

3.1.2 Το «τρέξιμο» του Μοντέλου

Έχοντας ολοκληρώσει τα παραπάνω στάδια έχουμε προσδιορίσει πλήρως το μοντέλο. Τρέχοντας την προσομοίωση θα εμφανισθούν τα παρακάτω γεγονότα:

1. δημιουργούνται οι μεταβλητές και οι τιμές τους σύμφωνα με την αρχικοποίηση που έχουμε ορίσει
2. υπολογίζονται οι εξισώσεις συνδέσμων καθώς η αρχική τιμή κάποιων μεταβλητών επηρεάζει τις άλλες.

Σε αυτό το σημείο το μοντέλο βρίσκεται στην αρχική του κατάσταση **και η προσομοίωση «αναμένει» είτε το επόμενο βήμα εξέλιξης , είτε την αλληλεπίδραση με το χρήστη.**

Παρατήρηση:Στην πρώτη περίπτωση υπολογίζονται οι εξισώσεις εξέλιξης και μετά οι εξισώσεις συνδέσμων και το σύστημα βρίσκεται σε μια νέα κατάσταση.
Στην δεύτερη περίπτωση , όταν ο χρήστης αλλάζει μια μεταβλητή, υπολογίζονται οι εξισώσεις συνδέσμων και έτσι προσδιορίζεται η νέα κατάσταση του συστήματος **την ίδια χρονική στιγμή.**

Παρατήρηση: Ορισμένες φορές είναι δυνατό να αλληλεπιδρά ο χρήστης με την προσομοίωση ενώ η προσομοίωση βρίσκεται ήδη στην εξέλιξη της(για την ακρίβεια ενώ εκτελείται ένα βήμα εξέλιξης).
Αν και αυτό δεν είναι άσχημο, ορισμένες φορές μπορεί να προκαλέσει ανεπιθύμητα αποτελέσματα όπως την αλλαγή της τιμής μιας παραμέτρου. Στην περίπτωση αυτή είναι προτιμότερο να ζητηθεί από τον χρήστη να σταματήσει την προσομοίωση χρησιμοποιώντας κάποιο κουμπί ελέγχου που έχει κατασκευασθεί.

3.2 Η διεπαφή του Μοντέλου στο EJS

Το panel του Ejs που χρησιμοποιείται για την ανάπτυξη μοντέλων εμφανίζεται στην παρακάτω εικόνα. Αυτή περιλαμβάνει αυτά που έχουμε αναφέρει (radio buttons) καθώς και ένα κουμπί επιπλέον, που ονομάζεται Custom.

Εικόνα: Το panel για την συγγραφή του μοντέλου

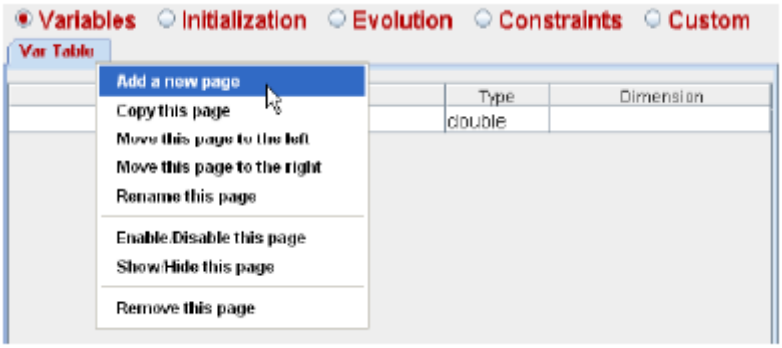
Κάθε φορά που θέλετε να δημιουργήσετε μια νέα σελίδα , το λογισμικό σας προτρέπει μέσω του μηνύματος για νέα σελίδα.

Εικόνα: Εισαγωγή νέας σελίδας

Μπορούμε να δημιουργήσουμε μεταβλητές πατώντας δεξί κλίκ στο παραπάνω panel οπότε εμφανίζεται η παρακάτω εικόνα.

Εικόνα: Το panel για την εισαγωγή μεταβλητών

Πολλές φορές είναι χρήσιμο να οργανώνουμε την προσομοίωση δημιουργώντας πάνω από μια σελίδες στο ίδιο panel, γιατί έτσι ομαδοποιούμε τις μεταβλητές ή τους αλγορίθμους που έχουν το ίδιο περιεχόμενο.



Εικόνα: popup menu μιας σελίδας

Από τις επιλογές που εμφανίζονται, μια που χρειάζεται ιδιαίτερη προσοχή είναι η Enable/Disable the page.
Αυτή η επιλογή χρησιμοποιείται όταν θέλουμε το Ejs να μην χρησιμοποιεί μια σελίδα που έχομε δημιουργήσει αλλά όμως δεν θέλουμε να την διαγράψουμε.
Αυτή η διαδικασία είναι χρήσιμη όταν θέλουμε να κάνουμε τεστ με διαφορετικούς αλγορίθμους(για το ίδιο μοντέλο)και θέλουμε να συγκρίνουμε αυτούς ενεργοποιώντας ή απενεργοποιώντας αυτούς.
Η επιλογή show/hide μια σελίδα χρησιμοποιείται ορισμένες φορές όταν θέλουμε να αποκρύψουμε σελίδες που περιέχουν δύσκολους αλγόριθμους και δεν επιθυμούμε να είναι ορατοί σε πρώτη φάση από τον εκπαιδευόμενο.

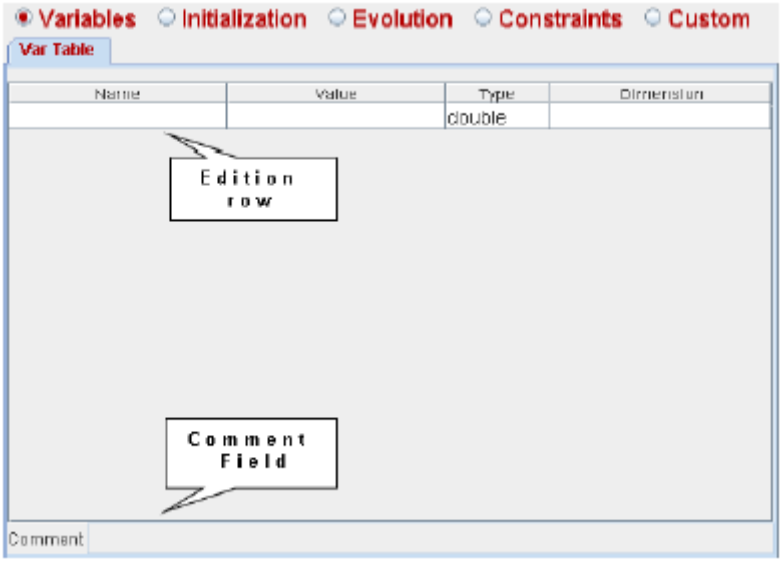
3.3 Η δήλωση μεταβλητών στο Μοντέλο

Η δήλωση μεταβλητών είναι το πρώτο βήμα για τη δημιουργία του μοντέλου.
Για τον ορισμό των μεταβλητών, θα πρέπει αν προσδιορίσουμε το τύπο τους και για την περίπτωση που οι μεταβλητές είναι πίνακες ή διανύσματα, θα πρέπει να προσδιορίσουμε και την διάστασή τους.
Προαιρετικά μπορούμε να δώσουμε και αρχική τιμή στις μεταβλητές.
Στην Java χρησιμοποιούμε κυρίως τους παρακάτω τύπους μεταβλητών:

• boolean, για μεταβλητές που παίρνουν λογικές τιμές(true ή false).
• byte, short, int, και long για μεταβλητές που κρατούν ακέραιες τιμές δεδομένων
• float και double για μεταβλητές που κρατούν δεδομένα κινητής υποδιαστολής.
• char and String για μεταβλητές που κρατούν δεδομένα τύπου χαρακτήρα και δεδομένα αλφαριθμητικά αντίστοιχα.
Παρατήρηση: Ως μια αντικειμενοστραφής γλώσσα προγραμματισμού, η Java εισαγάγει ένα νέο τύπο δεδομένων που καλείται **Object** , ως τη βάση αυτών που καλούνται κλάσεις της γλώσσας.

Παρατήρηση: Αν και το Ejs δεν δημιουργήθηκε για επαγγελματίες προγραμματιστές , εν τούτοις αφήνει μια πόρτα ανοικτή για να χρησιμοποιήσουμε

αντικειμενοστραφή προγραμματισμό. Για παράδειγμα μπορούμε να δημιουργήσουμε χρώματα ως αντικείμενα της κλάσης java.awt.Color και τα οποία μπορούν να αλλάζουν ανάλογα με την κατάσταση του μοντέλου.



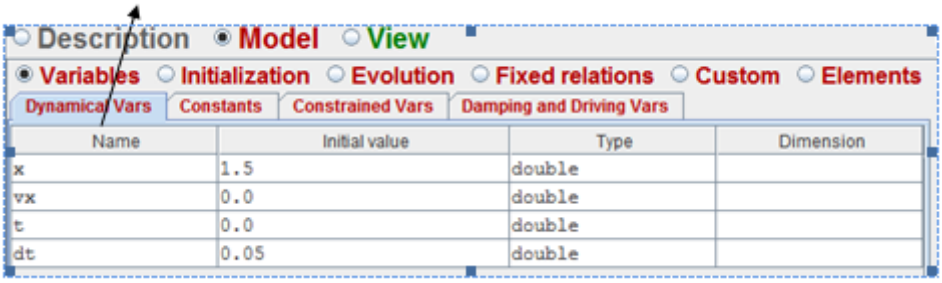
Εικόνα: η δημιουργία μεταβλητών και ο χώρος για σχόλια για αυτές

Αν η μεταβλητή είναι Πίνακας, τότε πρέπει να προσδιορίσουμε την διάστασή του. Αν το πεδίο 'Διάσταση»-Dimension- είναι κενό, τότε θα δημιουργηθεί μια μόνο μεταβλητή, δηλαδή μια μεταβλητή του τύπου που επιθυμούμε. Αν όμως γράψουμε στο πεδίο αυτό ένα ή περισσότερους ακέραιους ανάμεσα σε [], τότε το Ejs θα δημιουργήσει ένα σύνολο από μεταβλητές του τύπου που επιθυμούμε, όπου ο κάθε αριθμός ανάμεσα στα [], θα δηλώνει μια διάσταση.
Για παράδειγμα, αν γράψουμε [50] στο πεδίο της «διάστασης», τότε θα κρατηθούν στην μνήμη 50 μεταβλητές. Αν γράψουμε [10][100], θα δημιουργηθεί ένας δισδιάστατος πίνακας με 10 γραμμές και 100 στήλες. Ο ακέραιος που χρησιμοποιείται για να προσδιορίσει την διάσταση, μπορεί να είναι είτε μια σταθερά είτε μεταβλητή τύπου «ακέραιος»(int).

Παρατήρηση:
Όταν η διάσταση ενός πίνακα προσδιορίζεται από μια ακέραιη μεταβλητή, ο πίνακας θα δημιουργηθεί χρησιμοποιώντας τη τιμή της μεταβλητής όταν αυτή αρχικοποιήθηκε.
Κάθε μετέπειτα αλλαγή στη τιμή της μεταβλητής δεν θα μεταβάλλει την διάσταση του πίνακα.

Παρατήρηση: Μια μεταβλητή μπορεί να αρχικοποιηθεί γράφοντας μια σταθερή τιμή ή μια μαθηματική έκφραση στο αντίστοιχο πεδίο "Value" .

Εξ ορισμού το Ejs δίνει σε κάθε μεταβλητή μια τιμή (π.χ. τη τιμή μηδέν για αριθμητικές μεταβλητές) τις οποίες μπορούμε να τροποποιήσουμε αν επιθυμούμε. Επίσης το πεδίο αυτό μπορούμε να το αφήσουμε και κενό ενώ η αρχικοποίηση της τιμής μιας μεταβλητής μπορεί να γίνει κατόπιν υπολογισμού λόγω των συνδέσμων.



Εικόνα: οι δυναμικές μεταβλητές του μοντέλου

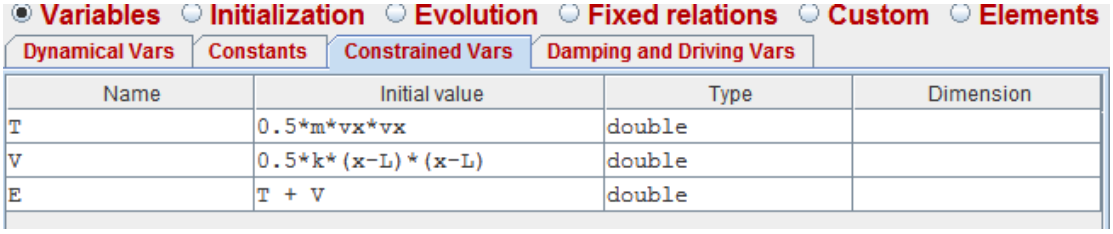
Για παράδειγμα, οι δυναμικές μεταβλητές (Dynamical Vars) παρουσιάζονται στην παραπάνω εικόνα.

Μπορούμε να επεκτείνουμε τις μεταβλητές (βλ. Constants)

Name	Initial value	Type	Dimension
showPlot	true	boolean	
y	0.0	double	
m	1.0	double	
k	1.0	double	
L	1.0	double	

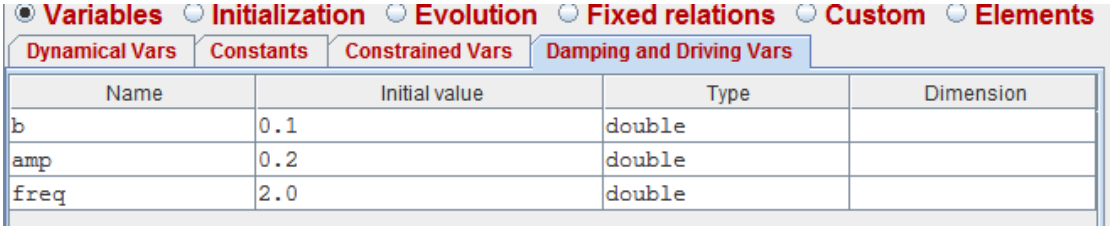
Εικόνα: οι σταθερές μεταβλητές του μοντέλου

Επίσης μπορούμε να έχουμε μεταβλητές που προκύπτουν από συνδέσμους όπως στην παρακάτω εικόνα.



Εικόνα: οι μεταβλητές που ορίζονται μέσω συνδέσμων του μοντέλου

Τέλος μπορούμε να εισάγουμε και μεταβλητές που να περιγράφουν την φθίνουσα και την εξαναγκασμένη ταλάντωση.

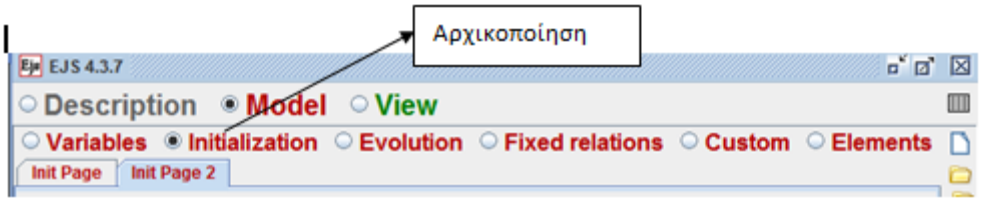


Εικόνα: οι μεταβλητές που περιγράφουν την φθίνουσα και εξαναγκασμένη ταλάντωση

3.4 Η Αρχικοποίηση του Μοντέλου

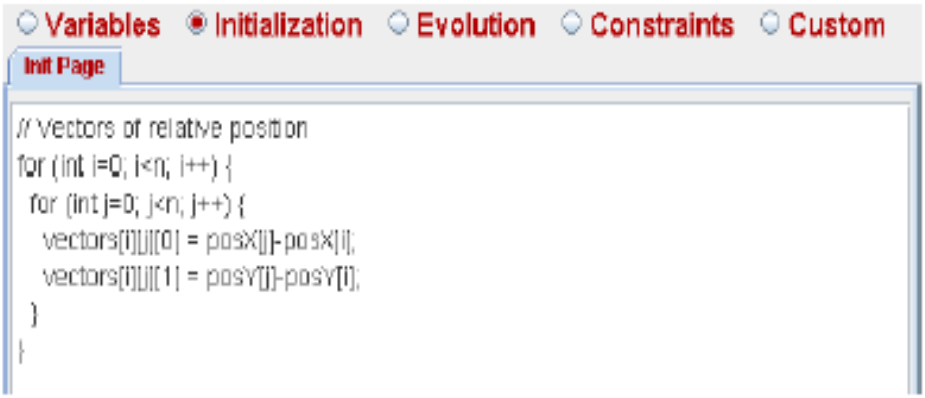
Μπορούμε να αρχικοποιήσουμε τις μεταβλητές του μοντέλου με σταθερές τιμές ή με απλές εκφράσεις στην στήλη "Value" του πίνακα των μεταβλητών. Ωστόσο, ορισμένες φορές, χρειαζόμαστε πιο σύνθετες αρχικοποιήσεις, όπως για παράδειγμα στην περίπτωση ενός πίνακα ή ενός διανύσματος.

Σε αυτές τις περιπτώσεις πηγαίνουμε στο κουμπί «Αρχικοποίηση»- "Initialization" και ανοίγουμε σελίδες για να αρχικοποιήσουμε τις μεταβλητές.



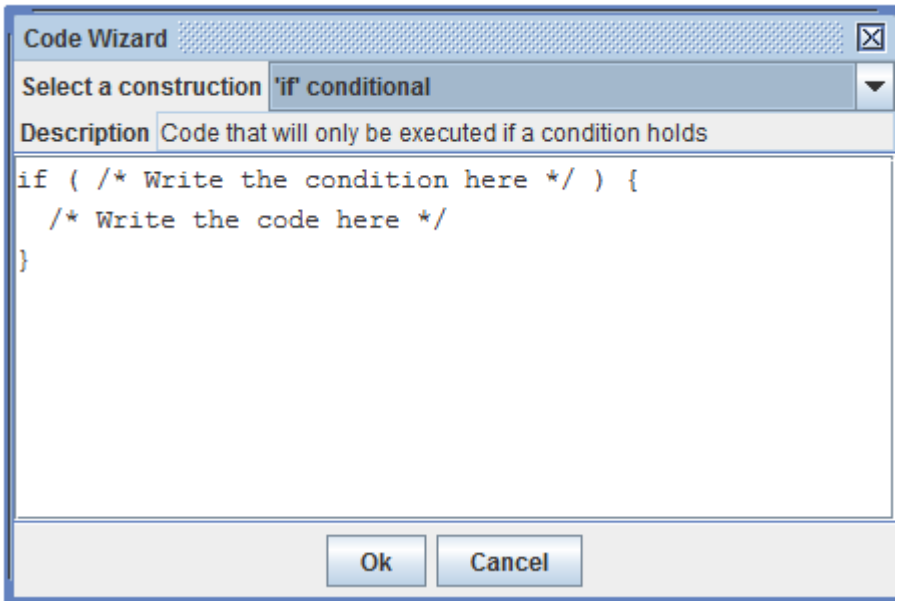
Εικόνα: δημιουργία σελίδων αρχικοποίησης

Μπορούμε επίσης στις σελίδες αυτές, να γράψουμε επίσης κώδικα Java που θα εκτελεί υπολογισμούς. Αυτός ο κώδικας θα εκτελείται μόνο μια φορά όταν θα ξεκινά η προσομοίωση.
Για παράδειγμα, μπορούμε σε μια από τις σελίδες να γράψουμε τον παρακάτω κώδικα για την αρχικοποίηση των στοιχείων ενός πίνακα.

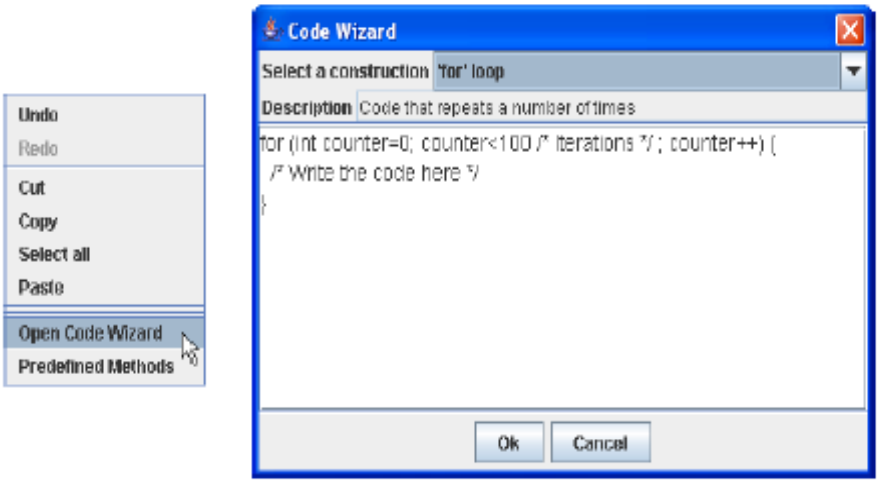


Εικόνα: αρχικοποίηση των στοιχείων ενός πίνακα με τη χρήση κώδικα στη γλώσσα Java

Το παράθυρο της αρχικοποίησης παρέχει επίσης της δυνατότητα για εισαγωγή standard Java εκφράσεων που περιέχουν επαναληπτικές δομές, συνθήκες κλπ.. Για να χρησιμοποιήσετε αυτή τη δυνατότητα κάνετε κλίκ **στον λευκό χώρο που διατίθεται στην ετικέττα «Custom».**



Εικόνα: The code Wizard: Ο μάγος του κώδικα για αρχικοποίηση μεταβλητών



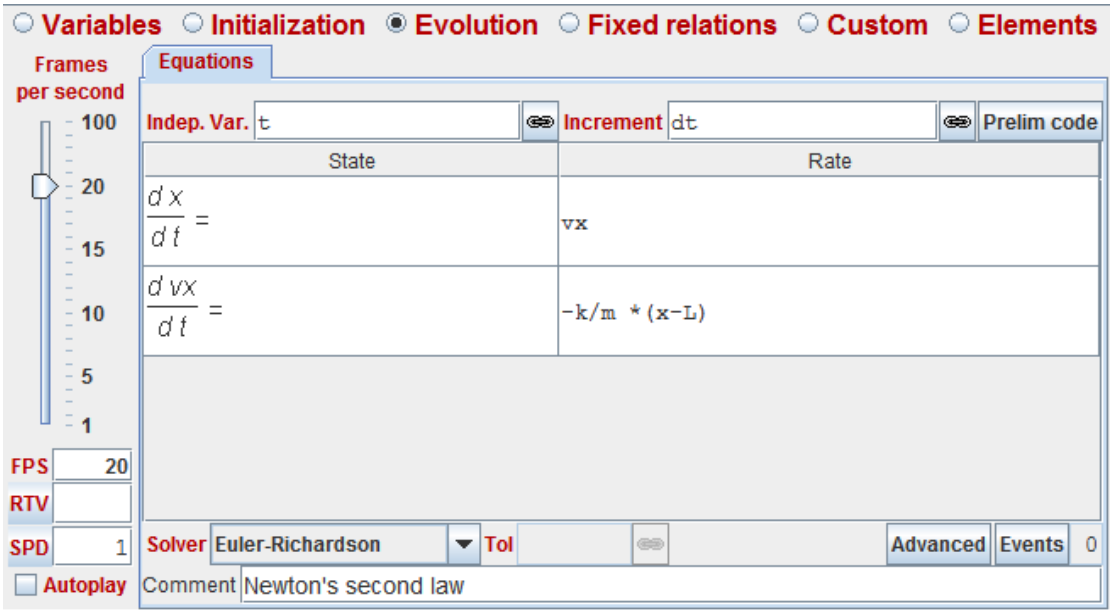
Εικόνα: Αρχικοποίηση μεταβλητών με τη βοήθεια του Code wizard

3.5 Η Εξέλιξη του Μοντέλου

3.5.1 Εισαγωγή

Εικόνα: Η καρτέλα «Εξέλιξη-Evolution" του μοντέλου

Η εξέλιξη των μοντέλων πραγματοποιείται με δυο τρόπους. Ο ένας υλοποιείται με την συγγραφή διαφορικών εξισώσεων(ODE-Ordinary Differential Equations). Στο παράδειγμα που αναφερόμαστε, η παρακάτω εικόνα δείχνει τις διαφορικές εξισώσεις που έχουν γραφεί για την περιγραφή της κίνησης στο παράδειγμα του απλού αρμονικού ταλαντωτή που αναφερόμαστε..



Εικόνα: οι διαφορικές εξισώσεις που περιγράφουν το μοντέλο

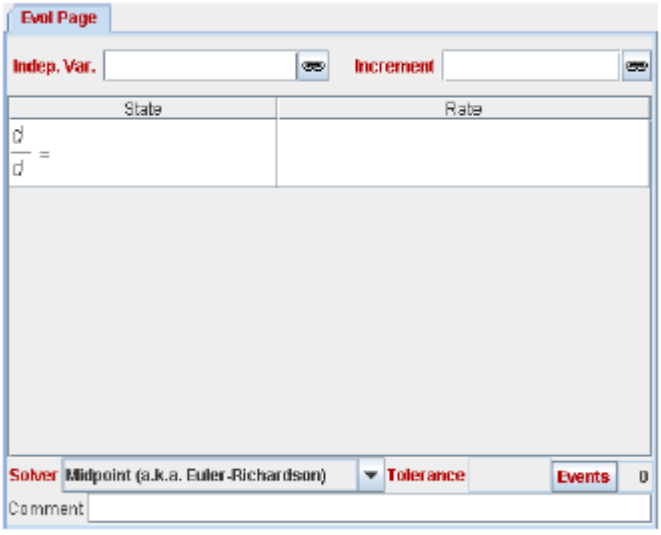
Στα αριστερά του παραθύρου βλέπουμε ένα slider που προκαλεί αλλαγή των Frames/sec(FPS). Στην παραπάνω εικόνα η τιμή του είναι 20.
Η τιμή του FPS μας πληροφορεί για το πόσες φορές ανά δευτερόλεπτο θα εκτελεσθούν οι εξισώσεις εξέλιξης. Η τιμή του FPS πρέπει να είναι σε αντιστοιχία με το βήμα χρόνου που θα χρησιμοποιηθεί για την εξέλιξη των διαφορικών εξισώσεων. Για παράδειγμα, αν η τιμή του FPS=10, και η τιμή του βήματος χρόνου dt=0,1, αυτό σημαίνει ότι γίνεται ένας υπολογισμός ανά δευτερόλεπτο, δηλαδή η προσομοίωση τρέχει σε πραγματικό χρόνο.
Κάτω από το FPS υπάρχει το κουμπί "Autoplay". Αυτό όταν είναι ενεργοποιημένο, δείχνει ότι η εξέλιξη ξεκινά αυτόματα όταν η προσομοίωση τρέχει.
Όταν θέλουμε να συμβαίνει αυτό, τότε το κουμπί "Autoplay" είναι ενεργοποιημένο. Ωστόσο σε πολλές περιπτώσεις, προτιμάμε η εξέλιξη να μην ξεκινά ταυτόχρονα με την προσομοίωση. Αυτό μπορεί να συμβαίνει όταν θέλουμε οι χρήστες της προσομοίωσής μας να μπορούν να διαχειρίζονται την διεπαφή τα προσομοίωσης και να καθορίζουν μόνοι τους τις αρχικές συνθήκες.
Για το σκοπό αυτό κρατάμε απενεργοποιημένο το κουμπί "Autoplay" και, όπως θα δούμε παρακάτω, μπορούμε να δημιουργήσουμε μια μέθοδο τύπου play().

3.5.2 Οι εξισώσεις εξέλιξης του μοντέλου

Το panel για την συγγραφή του μοντέλου διαιρείται σε δυο μέρη. Το πρώτο μας προτρέπει να γράψουμε κώδικα Java, ενώ το κάτω να γράψουμε διαφορικές εξισώσεις.
Αυτό σημαίνει ότι η εξέλιξη του μοντέλου μπορεί να γραφεί με δυο τρόπους.
Η εξέλιξη του μοντέλου συνίσταται ουσιαστικά στον μετασχηματισμό των εξισώσεων $x^*_i = f_i(x_1, x_2, \ldots, x_n)$ σε κώδικα Java. Αυτό ουσιαστικά συμβαίνει όταν έχουμε διακριτά μοντέλα στα οποία οι μεταβλητές αλλάζουν τιμές σε προκαθορισμένα χρονικά βήματα.
Ωστόσο, πολλές διαδικασίες που μελετάμε βασίζονται σε συνεχή μοντέλα στα οποία ο χρόνος θεωρείται συνεχής μεταβλητή.

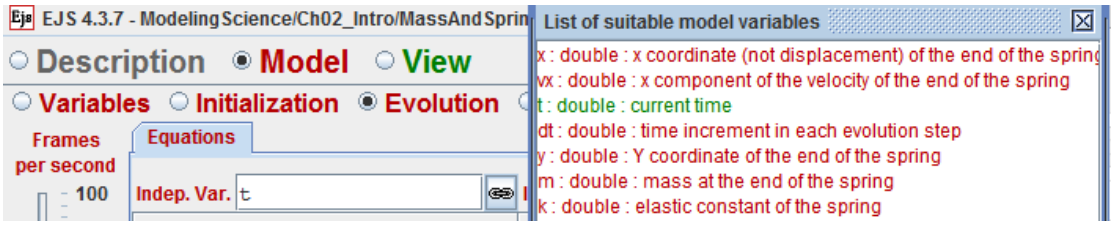
Σε αυτές τις περιπτώσεις ,καθώς ο ΗΥ είναι διακριτή μηχανή, παράγονται εξισώσεις από μεθόδους της αριθμητικής ανάλυσης(που διακριτικοποιούν τις εξισώσεις) και συνήθως απαιτούνται έξυπνοι αλγόριθμοι.

Ο editor του Ejs για να γράφουμε συνήθεις διαφορικές εξισώσεις παρουσιάζεται στην παρακάτω εικόνα.



Εικόνα: Ο Editor του Ejs για την γραφή διαφορικών εξισώσεων

Ο editor του Ejs μας επιτρέπει να εισαγάγουμε συστήματα συνήθων διαφορικών εξισώσεων πρώτης τάξης, δηλαδή διαφορικών εξισώσεων όπου εκφράζουμε την πρώτη παράγωγο μιας μεταβλητής ως συνάρτηση της ίδιας της μεταβλητής καθώς και άλλων μεταβλητών.
Το γεγονός ότι χρησιμοποιούμε διαφορικές εξισώσεις πρώτης τάξης δεν είναι ουσιαστικός περιορισμός, καθώς κάθε σύστημα εξισώσεων υψηλότερης τάξης μπορεί να γραφεί ως σύστημα διαφορικών εξισώσεων πρώτου βαθμού, χρησιμοποιώντας βοηθητικές μεταβλητές.
Για προφανείς λόγους οι μεταβλητές πρέπει να είναι τύπου double.
Πριν την αναγραφή αυτών των εξισώσεων στο Ejs, θα πρέπει να επιλέξουμε την ανεξάρτητη μεταβλητή. Συνήθως η ανεξάρτητη μεταβλητή είναι ο χρόνος.



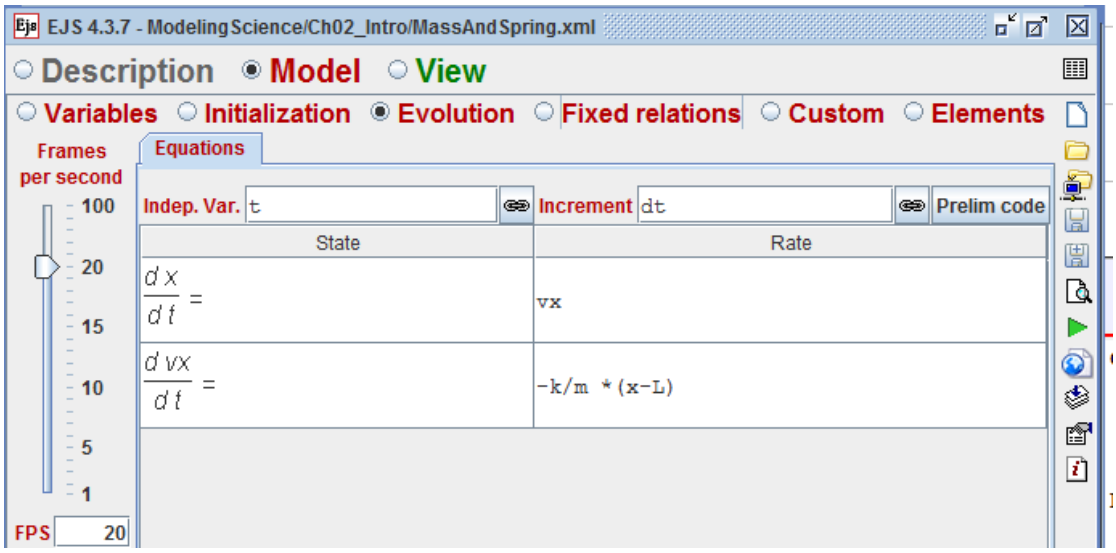
Εικόνα: η εισαγωγή της ανεξάρτητης μεταβλητής

Μπορούμε να γράψουμε την ανεξάρτητη μεταβλητή ή να επιλέξουμε την ανεξάρτητη μεταβλητή από το κουμπί  .
Στη συνέχεια θα πρέπει να εισαγάγουμε το βήμα ολοκλήρωσης που θα χρησιμοποιήσει ο editor του Ejs για να λύσει την διαφορική εξίσωση.

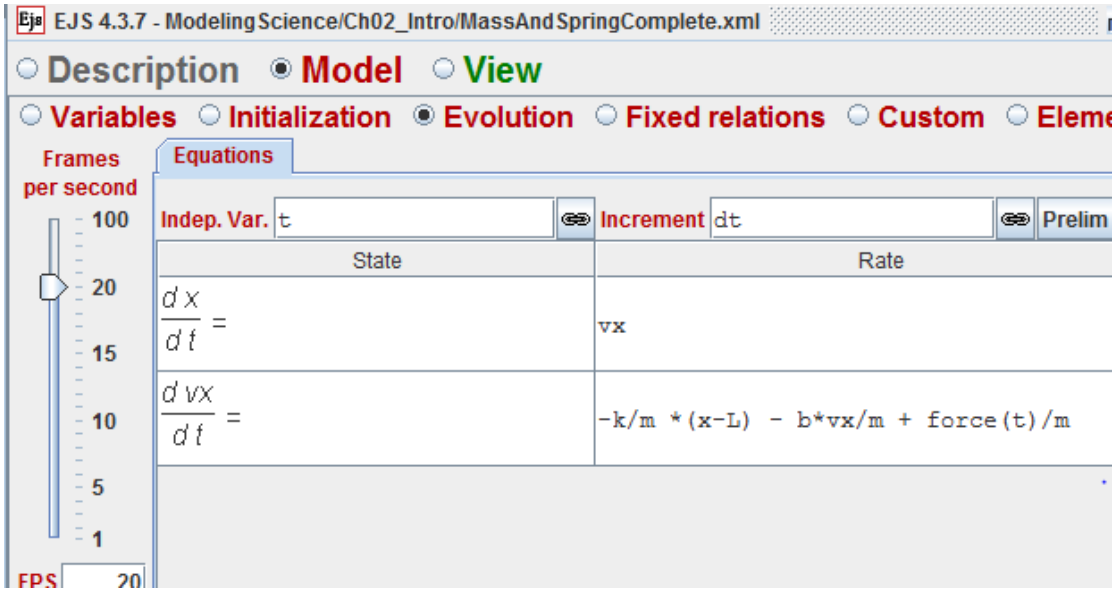
Η τιμή της αύξησης μπορεί να είναι σταθερή ή μεταβλητή (τύπου όμως double) και δείχνει την αύξηση που θα λάβει η μεταβλητή σε κάθε βήμα της εξέλιξης.

Εικόνα: η αύξηση της μεταβλητής στο βήμα εξέλιξης

Το βήμα της αύξησης μπορεί επίσης να γραφεί από την αρχή στην δήλωση των μεταβλητών.
Με βάση τα παραπάνω , οι διαφορικές εξισώσεις για το μοντέλο που μελετάμε, παρουσιάζονται στην παρακάτω εικόνα για το απλό μοντέλο(χωρίς απόσβεση).
Οι εξισώσεις για το μοντέλο με απόσβεση παρουσιάζονται στην παρακάτω εικόνα.

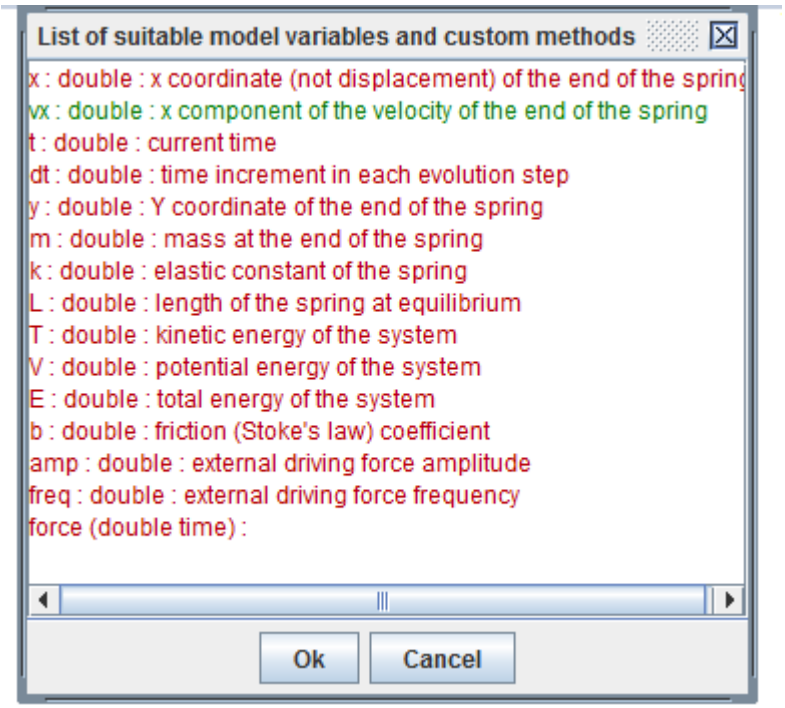


Εικόνα: οι εξισώσεις του απλού αρμονικού ταλαντωτή όταν δεν υπάρχει απόσβεση



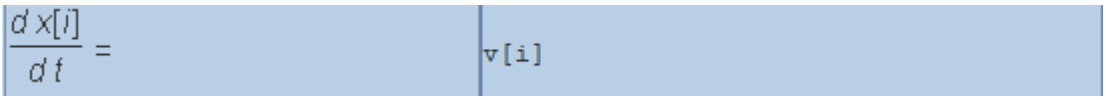
Εικόνα: οι εξισώσεις του αρμονικού ταλαντωτή όταν υπάρχει απόσβεση και διεγέρτης

Παρατήρηση: Κάνοντας δεξί κλίκ στα κελιά που γράφουμε τις διαφορικές εξισώσεις, ο editor μας βγάζει διάφορες επιλογές, μια εκ των οποίων είναι ιδιαίτερα χρήσιμη καθώς μας παρουσιάζει όλες τις μεταβλητές του μοντέλου, οπότε μπορούμε να επιλέξουμε κάποια για να εισαχθεί στη διαφορική εξίσωση.



Εικόνα: η λίστα των μεταβλητών που έχουμε χρησιμοποιήσεικάνοντας δεγί κλίκ σε ένα κελί των διαφορικών εξισώσεων

Παρατήρηση. Η παράγωγος διανύσματος: Η γραφή τα παραγώγου διανυσμάτων είναι παρόμοια, αλλά με μια μικρή διαφορά. Καθώς ο editor μπορεί για τεχνικούς λόγους να παραγωγίζει απλές μεταβλητές, θα πρέπει να εισαγάγουμε, και από αριστερά και από δεξιά, τον δείκτη του διανύσματος στην διαφορική εξίσωση. Για τον δείκτη του διανύσματος θα πρέπει να προσέξουμε να μην είναι κάποια καθολική μεταβλητή του μοντέλου. Συνήθως χρησιμοποιούμε το αγγλικό γράμμα i που την θεωρούμε τοπική μεταβλητή. Για παράδειγμα, αν θέλαμε να παραγωγίσουμε μια συνιστώσα ενός διανύσματος, τότε η παρακάτω εικόνα δείχνει τον τρόπο τον οποίο θα πρέπει να ακολουθήσουμε.



Εικόνα: η παράγωγος διανύσματος

Παρατήρηση- Γράφοντας διαφορικές εξισώσεις χρησιμοποιώντας μεθόδους προσαρμογής.
Τα παραδείγματα που έχουμε χρησιμοποιήσει έως τώρα είναι σχετικά απλά όσον αφορά το δεξιό μέλος των διαφορικών εξισώσεων.
Ορισμένες φορές όμως χρειαζόμαστε στο δεξιό μέλος πιο πολύπλοκες εκφράσεις που προκύπτουν από αλγορίθμους. Στην περίπτωση αυτή είναι προτιμότερο να χρησιμοποιήσουμε στο δεξί μέλος μεθόδους σε γλώσσα Java που θα γραφτούν στην καρτέλα Custom.Περισσότερα πάνω στο ζήτημα αυτό θα αναδειχθούν μέσω παραδειγμάτων.

3.5.3 Η επιλογή του αριθμητικού αλγορίθμου

Ακριβώς κάτω από τον πίνακα που γράφουμε τις εξισώσεις εξέλιξης μπορούμε να δούμε έναν επιλογέα που μας δηλώνει τον αριθμητικό αλγόριθμο που θα χρησιμοποιήσουμε για να λυθούν οι εξισώσεις εξέλιξης.

Οι αλγόριθμοι που χρησιμοποιούνται συνήθως είναι οι:

Euler: ο αλγόριθμος έχει χαμηλή ακρίβεια και χρησιμοποιείται κυρίως για παιδαγωγικούς λόγους καθώς είναι εύκολα κατανοητός από τους μαθητές

Euler–Richardson; αυτός ο αλγόριθμος είναι κατάλληλος για απλά προβλήματα. Συνήθως μπορούμε να αρχίσουμε με αυτόν τον αλγόριθμο και αν παρατηρήσουμε ότι τα αποτελέσματά μας δεν έχουν ακρίβεια, τότε θα πάμε σε μια καλύτερη μέθοδο.

Runge–Kutta: αυτός είναι η κλασσική μέθοδος Runge–Kutta τέταρτης τάξης και είναι πολύ δημοφιλής αλγόριθμος ενώ έχει πολύ ικανοποιητική ακρίβεια.

Runge–Kutta–Fehlberg: είναι μία μέθοδος με προσαρμοστικό βήμα και αποτελεί μια εξέλιξη του προηγουμένου όταν θέλουμε να έχουμε λάθος μικρότερο από κάποια τιμή. Στους αλγόριθμους του Ejs που υπάρχει αυτή η δυνατότητα, υπάρχει το κουμπί tolerance δίπλα στον αλγόριθμο.

Τέλος σε κάθε σελίδα υπάρχει η δυνατότητα γραφής σχολίων(Comments)

3.5.4 Τα γεγονότα μιας διαφορικής εξίσωσης

Ορισμένες φορές όταν υλοποιούμε την εξέλιξη μιας προσομοίωσης μέσω της λύσης ενός συστήματος διαφορικών εξισώσεων, θέλουμε το πρόγραμμα μας να ανιχνεύσει αν έχει συμβεί μια συνθήκη(η οποία εξαρτάται βέβαια από τις μεταβλητές). Με τον τρόπο αυτό θα μπορούμε να κάνουμε τις απαραίτητες διορθώσεις.

Για παράδειγμα, ας υποθέσουμε ότι προσομοιώνουμε την πτώση που κάνει ένα ελαστικό μπαλάκι. Οι εξισώσεις μας είναι:

$$\begin{aligned} \dot{y}(t) &= v_y(t) \\ \dot{v_y}(t) &= -g \end{aligned}$$

Η αριθμητική λύση των παραπάνω εξισώσεων δεν λαμβάνει υπόψη της το γεγονός ότι η εξέλιξη του συστήματος οδηγήσει τελικά το μπαλάκι σε μία θέση όπου αυτό θα είναι κάτω από το δάπεδο, δηλαδή η τιμή y(t) < 0 θα γίνει αρνητική.
Καθώς η μέθοδος της εξέλιξης του συστήματος προχωρά με σταθερά βήματα, είναι πολύ πιθανό η ακριβής χρονική στιγμή που το μπαλάκι χτυπά στο έδαφος να μην συμπίπτει με μια ένα από τα βήματα του αλγόριθμου, οπότε στη περίπτωση αυτή αναφερόμαστε σε μία μη επιτρεπτή κατάσταση του συστήματος.
Αυτό που θέλουμε από το πρόγραμμά μας είναι να ανιχνεύει το πρόβλημα και να σταματά στιγμιαία την πτώση που κάνει το μπαλάκι, να εφαρμόζει τον κώδικα για την αναπήδηση και να συνεχίζεται με αυτό τον τρόπο η προσομοίωση.

Αυτή η διαδικασία είναι αυτό που καλούμε **«γεγονός».**

Ακριβέστερα ορίζουμε ως γεγονός την αλλαγή προσήμου μιας πραγματικής συνάρτησης των καταστατικών μεταβλητών και των ανεξάρτητων μεταβλητών μιας διαφορικής εξίσωσης. Συνήθως τα γεγονότα που προκαλούνται μόνο από την ανεξάρτητη μεταβλητή καλούνται «γεγονότα χρόνου», ενώ αυτά που προκαλούνται από άλλες μεταβλητές καλούνται «γεγονότα κατάστασης».

Για να έχουμε ένα λειτουργικό ορισμό του γεγονότος ας καλέσουμε h(t) την συνάρτηση η οποία αλλάζει πρόσημο σχετικά με το γεγονός που μας ενδιαφέρει.

Για να βρούμε την χρονική στιγμή που συμβαίνει το γεγονός θα πρέπει να βρούμε τη λύση της εξίσωσης h(t) = 0. Θα υποθέσουμε ότι η συνήθης κατάσταση του συστήματος συμβαίνει όταν h(t) > 0, και το γεγονός συμβαίνει όταν h(t+δt) < 0, όπου δt η αύξηση της ανεξάρτητης μεταβλητής της διαφορικής εξίσωσης.

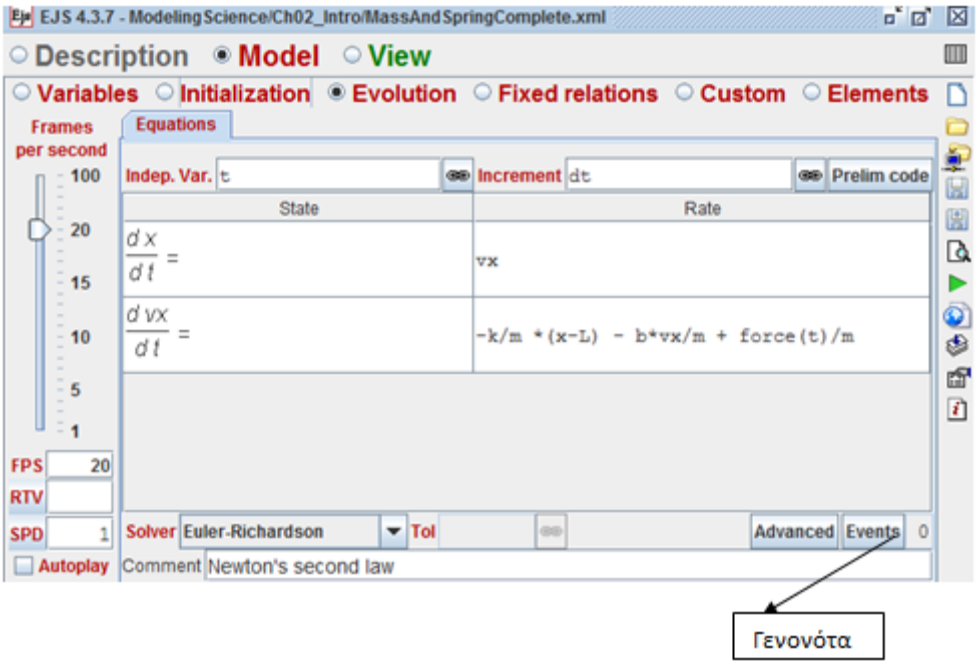
Πρακτικά υπάρχουν προβλήματα σε αυτή την θεώρηση καθώς μικρά λάθη στρογγυλοποίησης κατά τη διάρκεια των υπολογισμών μπορούν να προκαλέσουν αρνητικές τιμές , ώστε h(t) <0, και παρόλα αυτά η κατάσταση να είναι επιτρεπτή.

Έτσι θα πρέπει να επαναορίσουμε την έννοια ου γεγονότος ώστε αν είναι λειτουργικός και ένας τρόπος για να συμβεί αυτό είναι να προκύψει από την ακόλουθη διαδικασία:

1. κάθε επιτρεπτή κατάσταση του συστήματος τη χρονική στιγμή t συνδέεται με μια μη αρνητική τιμή της h(t). Εξαιτίας πιθανών λαθών στρογγυλοποίησης, μια κατάσταση θα θεωρείται επιτρεπτή ακόμα και όταν h(t) > −ε, όπου ε ένας πολύ μικρός θετικός αριθμός.
2. Αν το βήμα εξέλιξης του συστήματος οδηγεί το σύστημα σε μια κατάσταση όπου h(t+δt) > −ε, τότε η μέθοδος αριθμητικής επίλυσης θα θεωρήσει ότι έχει πραγματοποιηθεί ένα γεγονός στο χρονικό διάστημα [t,t+δt]. Στην περίπτωση αυτή η αριθμητική μέθοδος θα αναζητήσει ένα άλλο σημείο t', ώστε h(t') > −ε. Σε αυτή την αναζήτηση , το «πρώτο» t' θα είναι η ζητούμενη χρονική στιγμή. Τέλος έχοντας βρει αυτή τη χρονική στιγμή, θα πρέπει να έχει δημιουργηθεί μια μέθοδος από τον δημιουργό της προσομοίωσης, η οποία θα επαναφέρει το σύστημα στη κατάσταση h(t+_δt) > −ε.

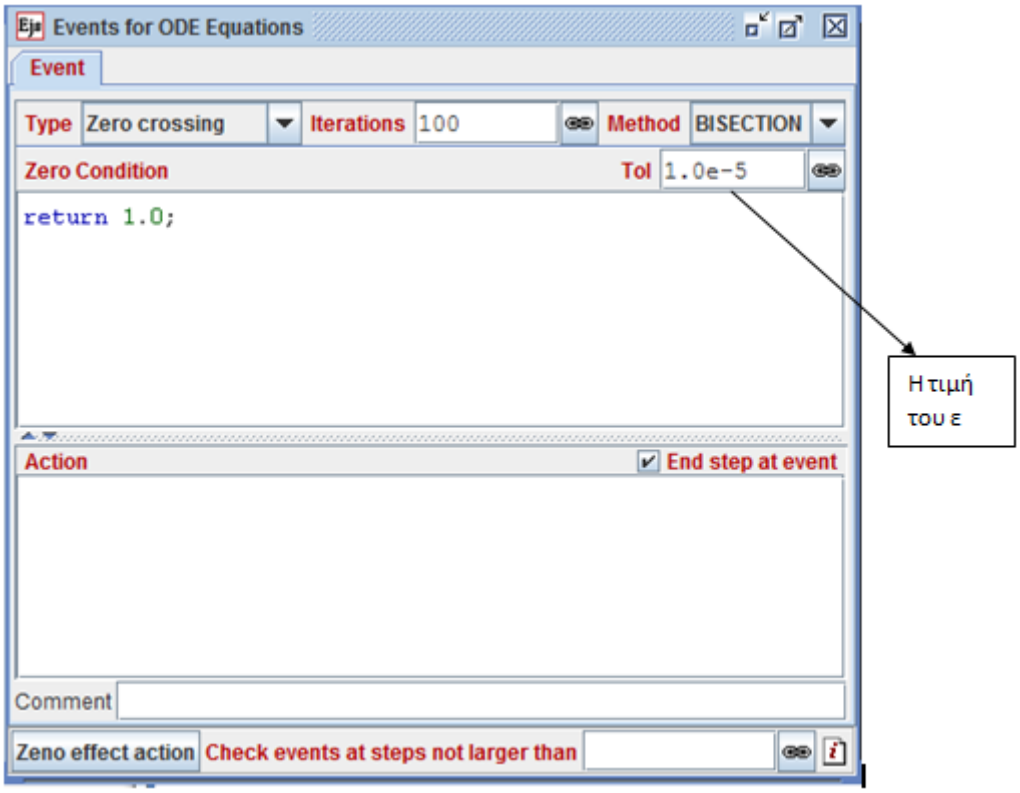
Δημιουργία γεγονότων:Για να δημιουργήσουμε γεγονότα σε μια διαφορική εξίσωση ακολουθούμε την παρακάτω διαδικασία.

Στον editor του Ejs υπάρχει μία καρτέλα με την ένδειξη events. Κάνοντας κλίκ στην καρτέλα αυτή θα ανοίξει ένα ανεξάρτητο παράθυρο στο οποίο μπορούμε να δημιουργήσουμε σελίδες γεγονότων.



Εικόνα: η καρτέλα events-γεγονότα μιας διαφορικής εξίσωσης.

Κάνοντας κλίκ στην καρτέλα "events"αυτή θα ανοίξει ένα ανεξάρτητο παράθυρο στο οποίο μπορούμε να δημιουργήσουμε σελίδες γεγονότων. Το παράθυρο αυτό περιέχει κάποια στοιχεία τα οποία θα συζητηθούν σε εφαρμογές.



Εικόνα: το παράθυρο για δημιουργία γεγονότων

Το παράθυρο των events χωρίζεται σε δυο τμήματα. Το πάνω τμήμα χρησιμοποιείται για την ανίχνευση των γεγονότων. Αυτό πραγματοποιείται με την χρήση κώδικα ο οποίος υπολογίζει τη συνάρτηση h(t) από τις τιμές των καταστατικών μεταβλητών και την ανεξάρτητη μεταβλητή της διαφορικής εξίσωσης. Ο κώδικας αναγκαστικά τελειώνει επιστρέφοντας μια τιμή τύπου double. Για να μας βοηθήσεις ο editor του Ejs γράφει την εντολή

return 1.0 ;

Επίσης στο πεδίο Tolerance γράφουμε τη τιμή του ε που συζητήσαμε παραπάνω. Το κάτω τμήμα χρησιμοποιείται για να δοθεί εντολή στον ΗΥ τι ενέργειες να γίνουν όταν ανιχνευθεί ένα γεγονός. Αυτή η ενέργεια θα πρέπει να «λύσει» τη κατάσταση που δημιούργησε το γεγονός.

Το κουμπί "stop at event", χρησιμοποιείται για να βρεθεί ο χρόνος του γεγονότος.

Πιο συγκεκριμένα, στο παράδειγμά μας με την πτώση που κάνει ένα ελαστικό μπαλάκι, στο πάνω τμήμα θα γράψουμε

return y; για να δηλώσουμε ότι το γεγονός θα συμβεί όταν το μπαλάκι φθάσει στο έδαφος και τότε y=0. Δεχόμενοι την τιμή του ε που αναγράφεται, η ενέργεια –action- που πρέπει να γραφεί στο κάτω τμήμα είναι η

vy = -vy;

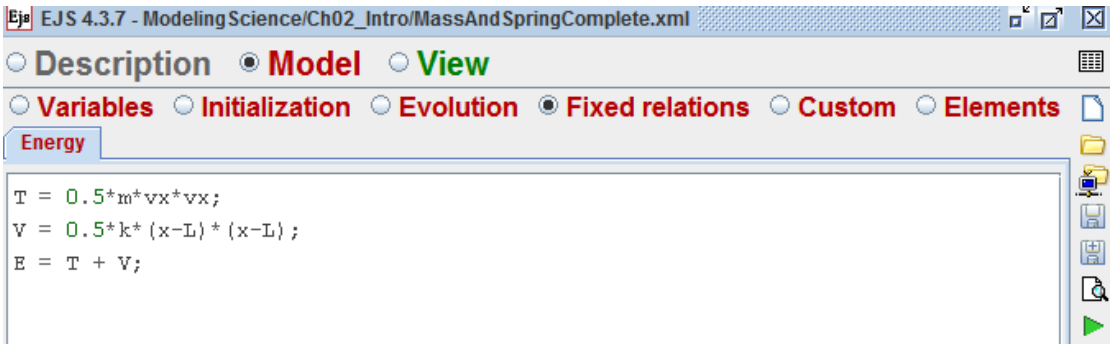
Έχοντας ενεργοποιήσει το “Stop at event”, το σύστημα οπτικοποιεί τη χρονική στιγμή που το μπαλάκι αναπηδά

Παρατήρηση: Ο event editor μας επιτρέπει να ορίζουμε περισσότερα από ένα γεγονότα για την ίδια διαφορική εξίσωση. Όταν υπάρχουν περισσότερα από ένα γεγονότα, ο editor θα ερευνήσει για αυτά που συμβαίνουν στη χρονική διάρκεια [t, t+δt] και από αυτά θα επιλέξει εκείνο το οποίο θα συμβεί πρώτο, εκτελώντας την αντίστοιχη με αυτό ενέργεια.

Παρατήρηση: Για τα λεγόμενα “zeno-type problems ”(από τον Αρχαίο μας πρόγονο Ζήνωνα).. Αυτά συμβαίνουν όταν το σύστημα φθάνει σε μια κατάσταση στην οποία συμβαίνει ένας μεγάλος αριθμός γεγονότων στα οποία η χρονική διαφορά εμφάνισης είναι πολύ μικρή. Τότε το πρόγραμμα αντί να προχωρά την προσομοίωση καταναλώνει πολύ χρόνο για την επίλυση αυτών των συμβάντων.

3.6 Οι Εξισώσεις συνδέσμων

Το παράθυρο για τη δημιουργία των εξισώσεων συνδέσμων συμπεριφέρεται με τον ίδιο τρόπο όπως το παράθυρο της αρχικοποίησης. Το μόνο που έχουμε να κάνουμε είναι να γράψουμε τις εξισώσεις συνδέσμων, όπως για παράδειγμα στην παρακάτω εικόνα.



Εικόνα: οι εξισώσεις περιορισμών

3.7 Οι Μέθοδοι προσαρμογής

3.7.1 Εισαγωγή

Το τελευταίο κουμπί του μοντέλου είναι το “Custom”. Ο κύριος σκοπός αυτού είναι να δίνει τη δυνατότητα στο χρήστη να δημιουργεί τις δικές του μεθόδους οι οποίες θα χρησιμοποιηθούν σε άλλα τμήματα της προσομοίωσης ες(οι μέθοδοι είναι οι αντίστοιχες συναρτήσεις ή υπορουτίνες που χρησιμοποιούνται σε άλλες γλώσσες). Μια άλλη χρήση του “Custom”. Είναι να δίνει τη δυνατότητα σε προχωρημένους χρήστες να δημιουργούν τις δικές τους βιβλιοθήκες σε μορφή JAR ή ZIP. Οι μέθοδοι προσαρμογής χρησιμοποιούνται για να ομαδοποιούν τμήματα του κώδικα ώστε αυτά να χρησιμοποιηθούν από άλλα τμήματα του μοντέλου(ή της θέασης) είτε γιατί το τελικό μοντέλο είναι ευκολότερο να κατανοηθεί μέσω αυτών, είτε γιατί επαναχρησιμοποιεί τμήματα του κώδικα που εμφανίζονται σε διαφορετικά τμήματα του μοντέλου.

Ο τρόπος χρήσης του panel “Custom” είναι ο ίδιος με τα άλλα panel που χρησιμοποιήσαμε για την αρχικοποίηση και τους συνδέσμους.

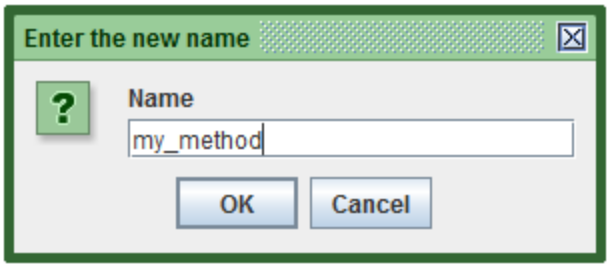
Ωστόσο υπάρχουν δυο σημαντικές διαφορές.

Η μια διαφορά είναι ότι στον τρόπο που χρησιμοποιούμε αυτό που γράφουμε σε αυτές τις σελίδες. Συγκεκριμένα το Ejs δεν χρησιμοποιεί άμεσα αυτές τις σελίδες αλλά μόνο όταν οι μέθοδοι σε αυτές τις σελίδες κληθούν από άλλο τμήμα της προσομοίωσης.

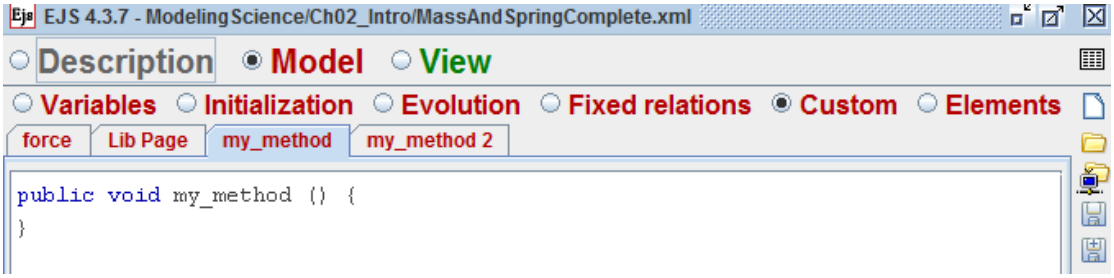
Η δεύτερη διαφορά είναι ότι σε αυτές τις σελίδες μπορεί να γραφεί κώδικας Java

3.7.2 Γραφή μεθόδων προσαρμογής

Όταν δημιουργούμε μια νέα σελίδα προσαρμογής, το Ejs μας βοηθά να συμπεριλάβουμε στην σελίδα τον αρχικό σκελετό της μεθόδου. Για παράδειγμα, αν η μέθοδός μας έχει το όνομα my_method, τότε η διεπαφή γράφει τον παρακάτω κώδικα



Εικόνα: νέα σελίδα με το όνομα της μεθόδου



Εικόνα: έναρξη της νέας μεθόδου

Γράφοντας μια μέθοδο σε Java απαιτείται να γράψουμε κώδικα, το τύπο της επιστρεφόμενης τιμής, την προσβασιμότητα της μεθόδου και τις παραμέτρους που θα περιέχει. Η προσβασιμότητα αναφέρεται στο ποια τμήματα του κώδικα μπορούν να χρησιμοποιούν την μέθοδο. Αν η μέθοδος π.χ. δηλώνεται ως public, τότε η μέθοδος είναι ορατή από κάθε μέρος της εφαρμογής. Επίσης είναι «ορατή», από τμήματα εκτός της εφαρμογής μέσω της χρήσης JavaScript. Αν η μέθοδος δηλωθεί ως private (η δεν έχει τεθεί προσδιοριστής μπροστά από το όνομά της), τότε η μέθοδος είναι ορατή μόνο μέσα στο συγκεκριμένο μοντέλο.

Η τιμή που επιστρέφει μια μέθοδος εξαρτάται από το αν η μέθοδος επιστρέφει μια τιμή (όπως οι συναρτήσεις), ή δεν επιστρέφει (όπως οι υπορουτίνες).

Αν δεν επιστρέφει τιμή τότε χρησιμοποιείται η λέξη void, ειδάλλως μπορεί να επιστραφεί οποιοσδήποτε τύπος τιμής και η μέθοδος κλείνει με τη λέξη return;

Για παράδειγμα, η παρακάτω μέθοδος επιστρέφει το εκθετικό του συνιμιτόνου

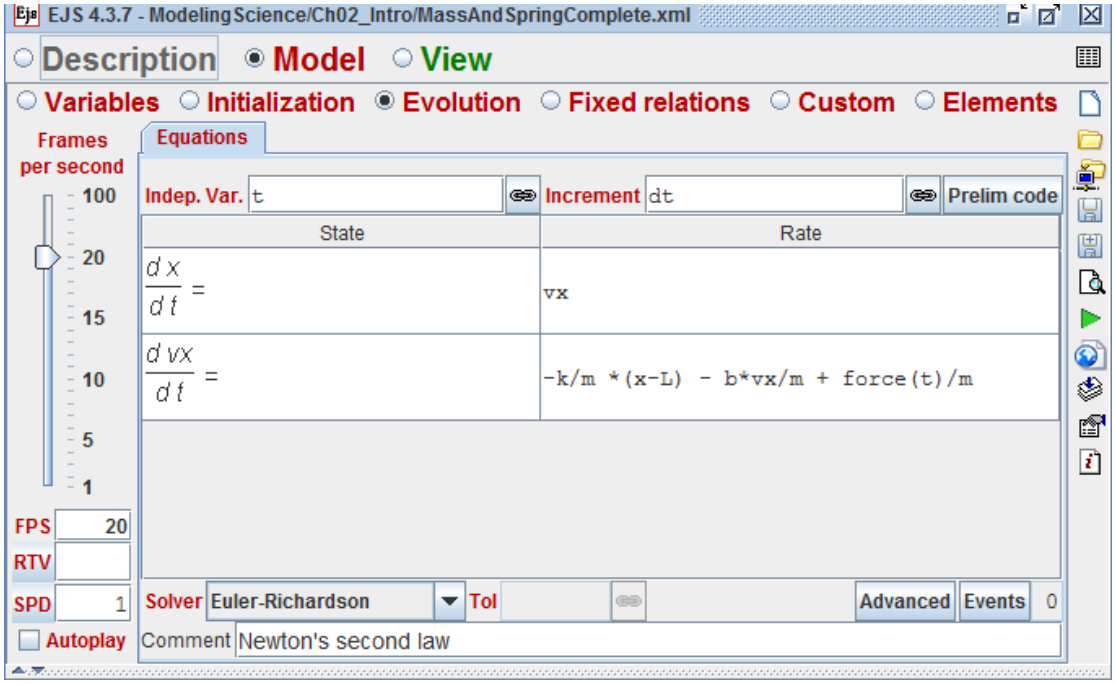
```
  double expCos (double x) {

return

 Math.exp (Math.cos(x));

}
```

Οι παράμετροι της μεθόδου δηλώνονται βάζοντας το κόμμα ανάμεσά τουςκαθώς και το τύπο των παραμέτρων.

Οι παράμετροι δηλώνονται μέσα στη παρένθεση ακριβώς μετά το όνομα της μεθόδου. Όταν καλούμε τη μέθοδο, η κλήση της θα πρέπει να γίνεται πάντα χρησιμοποιώντας τον ακριβή αριθμό και τον τύπο των παραμέτρων. Όταν η μέθοδος δεν έχει παραμέτρους, τότε αφήνουμε άδεια την λίστα των παραμέτρων.

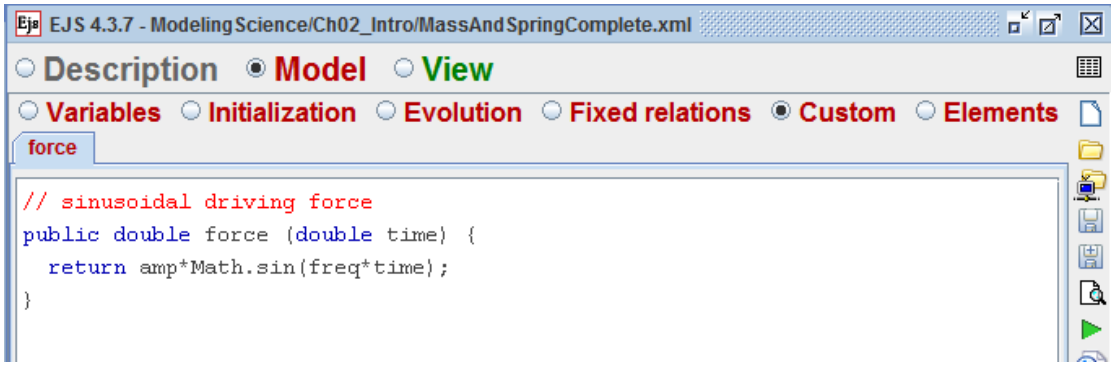
Μπορούμε στο μοντέλο μας και στην εξέλιξή το να έχουμε στην διαφορική εξίσωση, στο δεξιό μέλος, την μέθοδο που έχουμε δημιουργήσει στο παράθυρο “Custom”.

Για παράδειγμα, στην παρακάτω διαφορική εξίσωση για την επιτάχυνση, στο δεξιό μέλος έχουμε κλήση της μεθόδου force(t), που έχουμε δημιουργήσει.



Εικόνα: εισαγωγή μεθόδων στην διαφορική εξίσωση εξέλιξης

Η μέθοδος force(t) στο μοντέλο του απλού αρμονικού ταλαντωτή με απόσβεση και διεγέρτη υπολογίζει το μεταβλητό πλάτος της ταλάντωσης(τα // χρησιμοποιούνται για σχόλια).



Εικόνα: Γραφή μεθόδων προσαρμογής

Methods that control the execution of the simulation
`void _play()` Starts the evolution of the simulation.
`void _pause()` Stops the evolution of the simulation.
`void _step()` Executes one step of the evolution of the simulation.
`void _setFPS(int fps)` Sets the number of frames per second for the simulation (which will modify the settings of the slider of the evolution panel) to the integer value `fps`.
`void _setDelay(int delay)` This method is, so to say, the reciprocal of the previous one. It allows to control the number of milliseconds that *Ejs* must wait when it finishes one step of the evolution, before executing the next one. Obviously, this affects the number of frames per second of the simulation.
`void _reset()` This initializes all model variables to the values set in the tables of variables, it also executes the pages of initialization code (followed by those of constraints) and clears the view. This brings back the simulation to its exact initial state.
`void _initialize()` This method can be considered as a soft reset. It executes the initialization pages (plus constraints), but doesn't use the initialization values of the tables of variables. This can be useful when we want to initialize the simulation while respecting the values of some variables that the user has modified using the interface of the simulation.
`boolean _isPlaying()` This method returns a true boolean if the evolution is playing and a false one if it is not. This method is typically used to enable and disable some view elements depending on whether the simulation is playing or not.
`boolean _isPaused()` This method returns a true boolean if the evolution is paused and a false one if it is not. It is the opposite of the previous one and its use is similar to that one.

Methods specific for the view
`void _resetView()` It resets all the view elements to their original state. The precise meaning of this depends on each type of element. For instance, the result of this method on an element of type "Trace" will clear all the points drawn so far, including the memory (if it has memory).
`void _clearView()` Clears all the elements of the view. Again, the precise meaning of this depends on each type of elements. In general, this method can be considered as a softer version of the previous one. For instance, the result of this method on an element of type "Trace" will clear the points drawn so far in the current trace, but will respect the traces in memory (if it has memory).

`void _print(String text)`
Prints the provided text in the view of the simulation if this contains an element of the type "TextArea". If this is not the case, the message will appear on the console from which the program was launched or on the Java console of the Web browser that is running the applet.

`void _println(String text)`
Prints the indicated text, followed by a (so-called) carriage return and a new line, so that the next message will appear at the beginning of a new line. The text will appear as with the method `_print`.

`void _println()`
Prints a blank line, followed by a carriage return and a new line. Works similarly to the method `_print`.

`void _clearMessages()`
Clears the text area of the view. If the view has no such element, the method has no effect at all.

`String _format(double value, String format)`
Returns a String with the provided numerical value written according to the format indicated. The possible values for the format are the same as those for the "Format" properties that several view elements exhibit. See for instance the HTML reference page for the element of type "NumberField".

`void _alert(String element, String title, String message)`
Displays a dialog, with the given title, that shows the given message. The `element` parameter must be the name of an element of the view on top of which the dialog will appear. If `null` the dialog will appear on top of the element set using the `_setParentComponent()` method or, if this is not set, at the center of the computer screen. The view can't be accessed until the message is acknowledged.

`void _setParentComponent(String element)`
Sets the element of the view on top of which subsequent alert dialogs will appear.

`org.opensourcephysics.ejs.control.ControlElement _view.getElement(String element)`
Gets the ControlElement object for the given element of the view. Consult the JavaDoc documentation for `ControlElement` for more information.

`java.awt.Component _view.getVisual(String element)`
Gets the `java.awt.Component` wrapped by a given element of the view. See `_view.getComponent()` for an explanation.

`java.awt.Component _view.getComponent(String element)`
Gets the `java.awt.Component` wrapped by a given element of the view. Usually, the component and the visual of an element are the same. There are occasions in which they are different. For example, for a TextArea, the visual is the JTextArea and the component a ScrollPane. It is wise to check the class before using it.

`java.awt.Container _view.getContainer(String element)`
Gets the `java.awt.Container` wrapped by a given container element of the view. Usually, the container and the component of a container element are the same. There are occasions in which they are different. For example, for a Frame, the component is the JFrame and the container is its content pane. It is wise to check the class before using the object returned.

`Object _view.getObject(String element, String objectName)`
Some view elements define particular objects in its interior (the function of a ControlFunction is a good example). This method gets the object with the given name defined inside the given element of the view. Elements should document which objects they contain.

`org.opensourcephysics.ejs.Function`
`_view.getFunction(String element, String functionName)`

Some view elements define `org.opensourcephysics.ejs.Function` objects in its interior (the function of a ControlFunction is a good example). This method gets the object with the given name defined inside the given element of the view. Elements should document which functions they define.

Methods for input and output
`boolean _isApplet()` Whether the simulation is running as an applet.
`String _getParameter(String name)` When the simulation runs as an applet, this method can be used to retrieve the value of a given `<param>` tag of the applet. The name of the parameter is given by `name`. If the parameter of the applet is not set, this method returns `null`. If the simulation runs as an application, this method looks for a pair of arguments (of the Java start-up command for the application) of the form `"-name value"` (where `name` is the argument of this method). If found, the method return returns the value of `value`. This method can therefore be used to read parameters from either the `<applet>` HTML tag or the start-up command, using exactly the same program logic for both, applet and application form.
`String[] _getArguments(String name)` When the simulation runs as an application, this method returns the array of arguments that was specified at the Java start-up command. If the simulation runs as an applet, this method returns `null`.

`boolean _saveText(String filename, String text)`

Saves the given text to the specified file. The method returns `true` if everything went well, and `false` if there was any problem (then you might also get an error message in the console from which you run *Ejs*).

If the filename starts with the prefix "ejs:", then the text is written to a temporary memory location. This means that data can be later read as if they had been stored in a file (using the `readText` method), but only during the same working session of the simulation, since data will be lost when the simulation ends. This possibility of saving temporarily to memory is useful when the application is running as an applet, since (for security reasons) an (unsigned) applet can not save to the hard disk. However, this possibility of saving to memory can be used without problems.

If any other case, the text is written to a file on the hard disk. Security restrictions apply. This means that, again, if the simulation is running as an applet, then the applet needs to be signed or the method will trigger a security exception. You'll need to consult a specialized book to learn how to sign Java applets (*Ejs* can't -for security reasons- do this for you, sorry.)

`boolean _saveText(String filename, StringBuffer textBuffer)`

A variant of the previous method that uses a `StringBuffer`. (`StringBuffer` is a Java class that works faster when you need to handle many small texts.)

`String _readText(String filename)`

It reads text from the file `filename`. Returns `null` if it was not successful.

If the name of the file starts with the prefix "url:", then the method will interpret the rest of the name as a location on the Internet relative to the location of the simulation itself and will try to connect to this location and read the data from the corresponding Web server. If, moreover, the rest of the name starts with the prefix "http:", then the Internet location will be considered as an absolute URL. These options are specially interesting when we call this method from an HTML page using JavaScript (see Subsection 4.4.1).

Finally, if the name starts with the prefix "ejs:", *Ejs* will try to read the data from the specified memory location (see `_saveText()`).

`boolean _saveImage(String filename, String element)`

Saves the image of a given element of the view to the specified file. The method returns `true` if everything went well, and `false` if there was any problem (then you might also get an error message in *Ejs*' console).

The `filename` parameter is used as in `_saveText()` above. Moreover, if the filename has an extension, this extension is used as the format in which to save the image file, and the system tries to do so. Different systems may support different graphic formats. Currently, only PNG and JPEG formats are guarantied to be supported by all Java Virtual Machines (JPEG is used by default). We have added support for GIF format, but it only works well with images with less than 256 different colors on it.

`boolean _saveState(String filename)`

Saves the state of all model variables in the specified file. If successful, it returns a true boolean; if it encounters any problem, it returns `false`. The parameter `filename` indicates the name of the file where to store the data, and is used as in `_saveText()` above. The file is created as a binary file. To create text files, use the method `_saveText()` described above.

`boolean _readState(String filename)`

It reads the value of all model variables from the file `filename`. This file must have been created using a previous call to the `_saveState()` method from the same simulation, that is, with exactly the same variables, declared in the same exact order. Returns a true boolean if successful and a false if it is not.

The `filename` parameter is used as in `_readText()` above.

`boolean _saveVariables(String filename, String variablesList)`

Similar to `_saveState()`, but only saves the variables specified in the given list. This must be a String with the names of the variables you want to save, separated by either spaces, commas, or semicolons. The parameter `filename` indicates the name of the file where to store the data, and is used as in `_saveText()` above. The file is created as a binary file. To create text files, use the method `_saveText()` described above.

`boolean _saveVariables(String filename, java.util.ArrayList variablesList)`

Similar to `_saveState()`, but only saves the variables specified in the given list. The parameter `filename` indicates the name of the file where to store the data, and is used as in `_saveText()` above. The file is created as a binary file. To create text files, use the method `_saveText()` described above.

`boolean _readVariables(String filename, String variablesList)`

Similar to `_readState()`, but only reads the variables specified in the given list. The data file must have been created with a matching call to `_saveVariables()`. The list of variables must be a String with the names of the variables you want to read, separated by either spaces, commas, or semicolons. The parameter `filename` indicates the name of the file where to read the data from, and is used as in `_readText()` above.

`boolean _readVariables(String filename, java.util.ArrayList variablesList)` Similar to `_readState()`, but only saves the variables specified in the given list. The data file must have been created with a matching call to `_saveVariables()`. The parameter `filename` indicates the name of the file where to read the data from, and is used as in `_saveText()` above.

Methods to access variables using JavaScript
`boolean _setVariables(String command)` This method is used to set the value of one or more variables of the model. The parameter `command` must consist on a semicolon-separated list of instructions of the type `variable=value`. As in, for example, `_setVariables("x = 1.0; y = 0.0")`. If the variable is a unidimensional array (a vector), it is still possible to use this method, separating the values of the different array elements by commas, as in `_setVariables("posX = -0.5,0.0,0.5")`. This method is not designed to be used from within *Ejs*, but from HTML pages using JavaScript. See Subsection 4.4.2.
`boolean _setVariables(String command, String sep, String arraySep)` This is a variant of the previous methods in which it is possible to specify the characters that separate variables and array elements. This may be necessary when you want to assign a value to variables of type String which contain commas or semicolons..
`String _getVariable(String variable)` This method is the counterpart of `_setVariables`. Given the name of a variable, it returns a String with its value. The user must extract the value of the variable (using the right type) from this String.

Μέθοδοι από την μαθηματική βιβλιοθήκη της Java

Κάθε μια από τις παρακάτω μεθόδους όταν καλείται θα πρέπει μπροστά να μπαίνει το Math.

Έτσι για παράδειγμα, για να υπολογίσουμε το ημίτονο γράφουμε Math.sin(0.5)

Methods of the mathematical library of Java
`double abs(double x)` Returns the absolute value of `x`.
`double acos(double x)` Returns the inverse function of the cosine, $\cos^{-1}(x)$, from 0 to π.
`double asin(double x)` Returns the inverse function of the sine, $\sin^{-1}(x)$, from $-\pi/2$ to $\pi/2$.
`double atan(double x)` Returns the inverse function of the tangent, $\tan^{-1}(x)$, from $-\pi/2$ to $\pi/2$.
`double ceil(double x)` Returns the smallest integer greater than or equal to `x`.

`double cos(double x)` Returns the cosine of `x`.
`double exp(double x)` Returns the exponential of `x` with basis e, e^x.
`double floor(double x)` Returns the greatest integer smaller than or equal to `x`.
`double log(double x)` Returns the natural logarithm (with basis e) of `x`.
`double max(double x, double y)` Returns the greatest of the numbers `x` and `y`.
`int max(int x, int y)` Returns the greatest of the integer numbers `x` and `y`.
`double min(double x, double y)` Returns the smallest of the numbers `x` and `y`.
`int min(int x, int y)` Returns the smallest of the integer numbers `x` and `y`.
`double pow(double x, double y)` Returns `x` to the power of `y`. More precisely, $e^{y \log x}$.
`double random()` Returns a random number between 0.0 and 1.0 (0.0 is excluded).
`double rint(double x)` Returns the integer number (in form of `double`) closest to `x`.
`long round(double x)` Returns the integer number (in form of long integer) closest to `x`.
`double sin(double x)` Returns the sine of `x`.
`double sqrt(double x)` Returns the square root of `x`.
`double tan(double x)` Returns the tangent of `x`.
`double atan2(double x, double y)` Returns the argument of the complex number `x+yi`.

3.7.3 Μέθοδοι ορισμένες από το Ejs

Το Ejs παρέχει τις δικές του μεθόδους (predefined) μεθόδους για χρήση από τον δημιουργό της προσομοίωσης.
Ένας ειδικός τύπος μεθόδων είναι αυτές που ελέγχουν(control) την εκτέλεση της προσομοίωσης.
Για παράδειγμα, για να κάνουμε pause ή play της εξέλιξης, για να τροποποιήσουμε τη ταχύτητα με την οποία τρέχει η προσομοίωση, για να παίξει η προσομοίωση βήμα-βήμα, ή για να επιστρέψει η προσομοίωση στην αρχική της κατάσταση χρησιμοποιούμε προκαθορισμένες μεθόδους.
Επίσης μια πολύ χρήσιμη μέθοδος είναι αυτή που επιτρέπει την αποθήκευση της κατάστασης της προσομοίωσης σε ένα δίσκο και την ανάγνωσή τους αργότερα από το Διαδίκτυο.
Παρακάτω παρουσιάζονται οι μέθοδοι ομαδοποιημένες ανά κατηγορία, ενώ περιγράφονται και τα αποτελέσματα κάθε μεθόδου.

Κεφάλαιο 4. Η εργαλειοθήκη View-Θέαση του Ejs

4.1 Οι γραφικές διεπαφές

4.1.1 Μια πρώτη κατηγοριοποίηση των στοιχείων

4.1.2 Οι διαχειριστές διάταξης- Layout Properties

4.1.3 Τα βασικά στοιχεία

4.1.4 Τα σχεδιαστικά στοιχεία

4.2 Η διεπαφή του Easy Java Simulations

4.2.1 Τύποι στοιχείων

4.2.2 Εισαγωγή στοιχείων στην Θέαση

4.3. Επεξεργασία των ιδιοτήτων των στοιχείων

4.4 Συσχέτιση μεταβλητών και ιδιοτήτων

4. Η εργαλειοθήκη View-Θέαση του Ejs

4.1 Οι γραφικές διεπαφές

Εισαγωγή: Η εργαλειοθήκη View-Θέαση του EJS

Ένα γραφικό στοιχείο, ή απλά ένα στοιχείο είναι ένα τμήμα της διεπαφής που κατέχει και θέση στην οθόνη και εκτελεί ένα συγκεκριμένο καθήκον για το οποίο έχει δημιουργηθεί. Υπάρχουν στοιχεία διαφόρων τύπων, όπως panels, labels, buttons, particles, arrows κλπ.

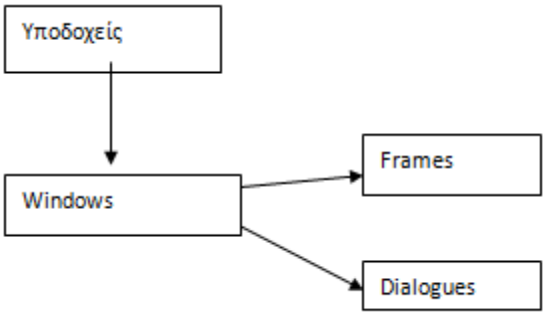
Κάθε στοιχείο έχει ένα σύνολο από εσωτερικά χαρακτηριστικά που καλούνται ιδιότητες τα οποία μπορούν να τροποποιηθούν ώστε το στοιχείο να εμφανίζεται με τον τρόπο που θέλουμε αλλά και να συμπεριφέρεται με τις ενέργειες που έχουμε αποφασίσει για αυτό.
Οι ιδιότητες μπορεί να έχουν στατικές τιμές αλλά υπάρχει η δυνατότητα και για δυναμική αλλαγή αυτών των τιμών.
Πολύ σημαντικός είναι ένας ειδικός τύπος ιδιοτήτων που καλούνται «ενέργειες»- actions- που επιτρέπουν στον χρήστη την αλληλεπίδραση με την διεπαφή, όταν αυτός χρησιμοποιεί αυτά τα στοιχεία(για παράδειγμα όταν κάνει κλίκ στο ποντίκι ή πληκτρολογεί στο πληκτρολόγιο).

4.1.1 Μια πρώτη κατηγοριοποίηση των στοιχείων

Μια πρώτη κατηγοριοποίηση των στοιχείων είναι η διάκριση σε: υποδοχείς, βασικά στοιχεία και σχεδιαστικά στοιχεία.

Οι Υποδοχείς

Ένα υποδοχέας είναι ένα γραφικό στοιχείο που «φιλοξενεί» άλλα στοιχεία της οθόνης. Ο υποδοχέας είναι ο χώρος που θα τοποθετηθούν τα συστατικά του γραφικού περιβάλλοντος διασύνδεσης(GUI).
Στην περίπτωση αυτή ο υποδοχέας καλείται parent και το στοιχείο που περιέχεται σε αυτόν child(θυγατρικό στοιχείο). Ένας υποδοχέας μπορεί να είναι ταυτόχρονα και parent και child, με αποτέλεσμα να μπορούμε να δομήσουμε μια πλήρη ιεραρχική δομή με γραφικά στοιχεία με τη μορφή ενός δέντρου στην λεγόμενη «Θέαση Προσομοίωσης»- Simulation View.Ως πρώτο child συνήθως δημιουργούμε το λεγόμενο κύριο παράθυρο—main window- το οποίο είναι το παράθυρο στην οθόνη όταν τρέχουμε την εφαρμογή ή το πρώτο παράθυρο στην HTML σελίδα όταν τρέχουμε την εφαρμογή ως applet.
Υπάρχουν δυο τύποι windows, τα frames(πλαίσια) και οι dialogs(πλάισια διαλόγου).



Εικόνα: Οι τύποι των Windows

Οι οικογένειες των υποδοχέων είναι δύο: οι υποδοχείς που περιέχουν άλλους υποδοχείς και οι υποδοχείς που περιέχουν σχεδιαστικά στοιχεία.
Η πρώτη ομάδα χρησιμοποιείται μαζί με βασικά στοιχεία για να σχεδιαστεί ο σκελετός της προσομοίωσης παρέχοντας στον χρήστη στοιχεία για να ελέγχει-control- την προσομοίωση και να μεταβάλλουν την τιμή των μεταβλητών του μοντέλου.
Σε αυτή την κατηγορία ανήκουν τα windows, toolbars, standard panels, split panels, and tabbed panels.
Η δεύτερη ομάδα χρησιμοποιείται μαζί με σχεδιαστικά στοιχεία για να δημιουργήσει animations ή να οπτικοποιήσει γραφικές παραστάσεις προσομοιωμένων φαινομένων. Αυτή η ομάδα περιέχει δισδιάστατα drawing panels, plotting panels (2D panels with coordinate axes), and three-dimensional drawing panels.

4.1.2 Οι διαχειριστές διάταξης- Layout Properties

Ισως η πιο σημαντική ιδιότητα των υποδοχέων πρώτου είδους είναι ο διαχειριστής διάταξης.
Όταν ένα στοιχείο προστίθεται στον υποδοχέα, ο «γονέας» παρέχει στο θυγατρικό στοιχείο μια κατάλληλη θέση και μέγεθος σύμφωνα με τον διαθέσιμο χώρο.
Οι υποδοχείς δευτέρου είδους ορίζουν το δικό τους σύστημα συνταγμένων που επιτρέπει στα θυγατρικά στοιχεία (τα οποία είναι βέβαια σχεδιαστικά στοιχεία), να προσδιορίζουν την θέση τους χρησιμοποιώντας συντεταγμένες από τον χρήση. Έτσι αυτά τα στοιχεία **δεν έχουν διαχειριστή διάταξης.**

Flow

Κατανέμει τα θυγατρικά στοιχεία- children- σε μια γραμμή από αριστερά προς τα δεξιά. Τα θυγατρικά στοιχεία μπορεί να είναι στα αριστερά, στα δεξιά ή στο κέντρο.

Border
Ίσως η πιο συχνά εμφανιζόμενη ιδιότητα διαχείρησης, όπου κατανέμει τα θυγατρικά στοιχεία σε μια από τις πέντε(5) δυνατές θέσεις : up, down, left, right and center, όπου κάθε θυγατρικό στοιχείο μπορεί να επιλέξει μια διαφορετική θέση.

Grid

Αυτή είναι η δεύτερη πιο δημοφιλής ιδιότητα και τοποθετεί τα θυγατρικά στοιχεία σε τετραγωνικό πίνακα με όσο περισσότερες γραμμές και στήλες ίσου μεγέθους

Horizontal box
Μοιάζει με το \Grid αλλά με μια γραμμή

Vertical box

Μοιάζει με το \Grid αλλά με μια στήλη
Για όλες τις παραπάνω ιδιότητες διαχείρησης μπορούν να οριστούν επιπλέον παράμετροι που προσδιορίζουν την απόσταση των θυγατρικών στοιχείων οριζόντια και κατακόρυφα.

4.1.3 Τα βασικά στοιχεία

Τα βασικά στοιχεία αποτελούνται από ένα σύνολο από στοιχεία διεπαφής που μπορούν να χρησιμοποιηθούν για να σχεδιασθεί η διεπαφή, να οπτικοποιηθούν και να επεξεργασθούν οι μεταβλητές του μοντέλου καθώς και να κληθούν ενέργειες(actions) , δηλαδή μέθοδοι που θα διαχειρισθούν το μοντέλο.
Τα στοιχεία αυτής της ομάδας είναι πολύ οικεία στους χρήστες προγραμμάτων καθώς εμφανίζονται πολύ συχνά σε όλα τα προγράμματα υπολογιστών αι περιλαμβάνουν κουμπιά, ετικέτες, πεδία κειμένων κλπ.Αυτά τα στοιχεία μπορούν **να προστεθούν σε οποιοδήποτε υποδοχέα της πρώτης ομάδας,** αλλά δεν μπορούν να δεχθούν άλλα στοιχεία. Υπάρχουν μόνο δυο εξαιρέσεις στα παραπάνω. Η καρτέλα

«Μενού» περιέχει όλα τα στοιχεία που χρειάζεται για να δημιουργήσουμε «μενού».Σε αυτή τη καρτέλα υπάρχουν τα ‘MenuBar” και τα “Menu-types” που είναι ουσιαστικά υποδοχείς.

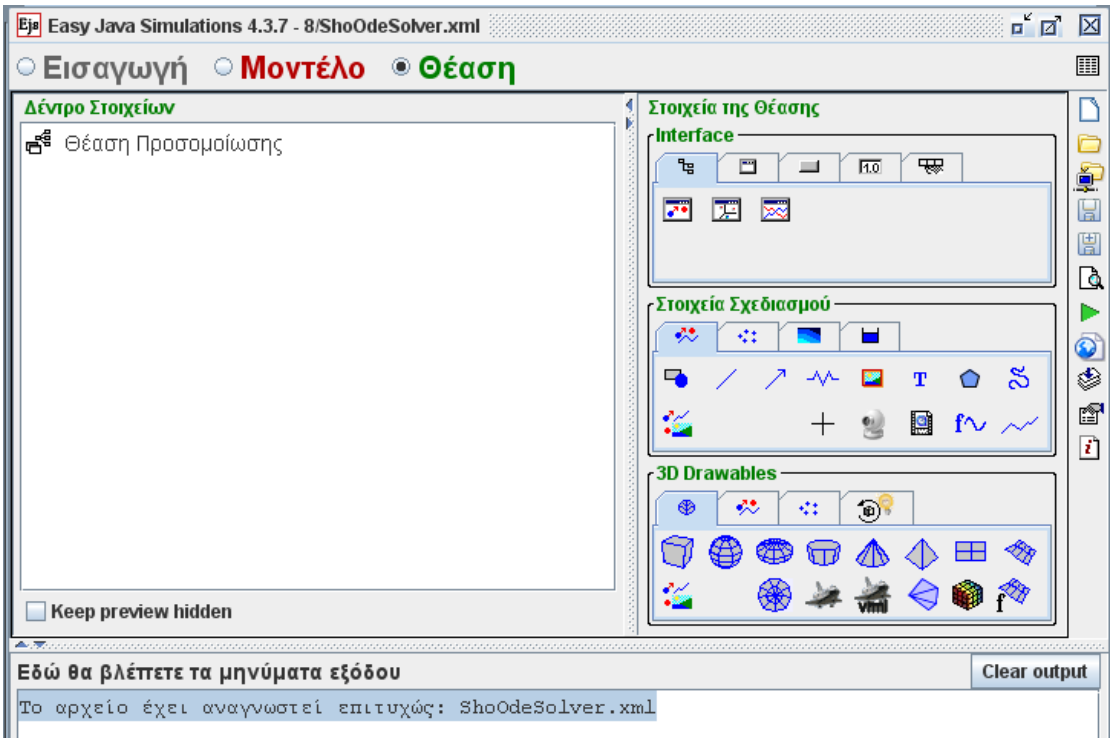
4.1.4 Τα σχεδιαστικά στοιχεία

Τα σχεδιαστικά στοιχεία εισάγονται σε υποδοχείς της δεύτερης κατηγορίας, δηλαδή σε αυτούς που περιέχουν σχεδιαστικά στοιχεία μόνο.

Οι υποδοχείς δευτέρου είδους ορίζουν από μόνοι τους το σύστημα συντεταγμένων που επιτρέπει στα θυγατρικά στοιχεία να προσδιορίσουν μόνα τους και για αυτό το λόγο αυτοί οι υποδοχείς δεν έχουν ιδιότητες διαχείρισης (layout properties)
Τα γραφικά με κίνηση(animated graphics) ποικίλουν από απλά έως πιο πολύπλοκα και μπορούν να σχεδιασθούν σε 2 ή 3 διαστάσεις ενώ μπορεί να αποτελούνται από σωματίδια, διανύσματα, εικόνες , γεωμετρικά σχήματα, χάρτες κλπ.

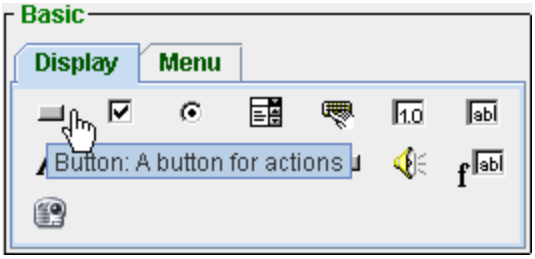
4.2 Η διεπαφή του Easy Java Simulations

Έτσι η δημιουργία της θέασης αποτελείται από μια δομή(δέντρο) από στοιχεία τα οποία οπτικοποιούν την κατάσταση του μοντέλου, στοιχεία αλληλεπίδρασης με την προσομοίωση του μοντέλου, και υποδοχείς που περιέχουν όλα τα υπόλοιπα στοιχεία.



Εικόνα: Η θέαση με τα στοιχεία

Η αριστερή πλευρά του χώρου εργασίας του EJS αποτελείται από ένα panel που αντιστοιχεί στο δέντρο των στοιχείων και το οποίο αρχικά είναι κενό(η θέαση προσομοίωσης).
Στη συνέχεια θα προστεθούν νέα στοιχεία και το panel θα περιέχει θα γεμίσει από τα στοιχεία που θα προστεθούν.
Η δεξιά πλευρά του χώρου εργασίας περιέχει τρείς τομείς σε κάθετη διαμόρφωση. Κάθε κάθετος τομέας περιέχει ένα σύνολο από στοιχεία ομαδοποιημένα σύμφωνα με την κατηγοριοποίηση που αναφέραμε προηγουμένως, δηλαδή σχεδιαστικά και μη σχεδιαστικά στοιχεία.
Η εξερεύνηση των στοιχείων μπορεί να πραγματοποιηθεί τοποθετώντας το ποντίκι πάνω σε κάθε στοιχείο και αναμένοντας για περίπου ένα second, μέχρις ότου ένα μήνυμα εμφανιστεί με την περιγραφή της λειτουργίας του στοιχείου.



Εικόνα: Περιγραφή του κουμπιού Button

4.2.1 Τύποι στοιχείων

Α. Υποδοχείς

Frame **Το** κύριο χαρακτηριστικό των Frames είναι ότι «γνωρίζουν» με ποιο τρόπο θα διαχειρισθούν το παραθυρικό περιβάλλον του λειτουργικού συστήματος. Ειδικότερα μπορούν να ελαχιστοποιηθούν ή να μεγεθυνθούν ενώ σε αυτά τοποθετούνται άλλα στοιχεία **ενώ η θέαση πρέπει να έχει ένα τουλάχιστον τέτοιο στοιχείο για να τοποθετηθούν άλλα αντικείμενα. Το πρώτο Frame καλείται και κύριο παράθυρο((main window).**

Dialog Τα παράθυρα τύπου Dialog είναι παράθυρα **τα οποία δεν μπορούν να ελαχιστοποιηθούν ενώ μπορεί να γίνει απόκρυψή τους.**
Μπορούν να εμφανίζονται πάνω από το Frame |που προηγείται με αποτέλεσμα να είναι χρήσιμα σε προσομοιώσεις με περισσότερα από ένα παράθυρα.

Panel . Είναι υποδοχείς που χρησιμοποιούνται **για να ομαδοποιήσουμε άλλα στοιχεία**

SplitPanel  Είναι υποδοχείς που έχουν δυο διακριτές περιοχές που μπορούν να αλλάζουν μέγεθος .

DrawingPanel  Βασικός υποδοχέας για δισδιάστατες απεικονίσεις.

PlottingPane  Βασικός υποδοχέας για δισδιάστατες απεικονίσεις με καρτεσιανές ή πολικές συντεταγμένες

DrawingPanel3D  Η τρισδιάστατη εκδοχή του DrawingPanel.

Β. Βασικά Στοιχεία

Button Element  **:** κουμπί (διαταγής)για να καλεί ενέργειες ώστε να συμβεί ένα γεγονός

CheckBox  **-πλαίσιο ελέγχου-:** κουμπί που επιτρέπει να επιλέξουμε μεταξύ δυο δυνατών «λογικών» καταστάσεων(αληθής-ψευδής). Χρησιμοποιείται για την ενεργοποίηση ή απενεργοποίηση επιλογών

RadioButton  -Ραδιοπλήκτρο : ίδιο με το προηγούμενο μόνο μια από τις επιλογές μπορεί να επιλεχθεί.

ComboBox  :ένα στοιχείο στη μορφή λίστας που επιτρέπει την επολογή μιας η περισσότέρων επιλογών

Slider  : χρησιμοποιείται για την εισαγωγή και τροποποίηση μιας αριθμητικής τιμής μέσα σε ένα πεδίο τιμών

NumberField  : χρησιμοποιείται για την εισαγγωγή αριθμητικών τιμών από το πληκτρολόγιο

TextField  : χρησιμοποιείται για την εισαγωγή κειμένου από το πληκτρολόγιο

Label  -ετικέτα: αυτή είναι ένα κομάτι κειμένου το οποίο δεν μπορεί να τροποποιηθεί από το χρήστη.

Γ. Σχεδιαστικά στοιχεία

Particle  : σχεδιάζει μια έλλειψη ή ορθογώνιο δεδομένων διαστάσεων σε μια θέση

Arrrow  : σχεδιάζει ένα αλληπιδραστικό τόξο

Spring  **: δείχνει ένα αλληλεπιδραστικό ελατήριο**

Image  : δείχνει μια αλληλεπιδραστική εικόνα GIF

Trace  **: Σχεδιάζει μια πολυγωνική γραμμή από διαδοχικά σημεία**

Polygon  **: σχεδιάζει μια πολυγωνική γραμμή από τις συντεταγμένες της κορυφής**

Surface  **: σχεδιάζει μια παραμετρική επιφάνεια**

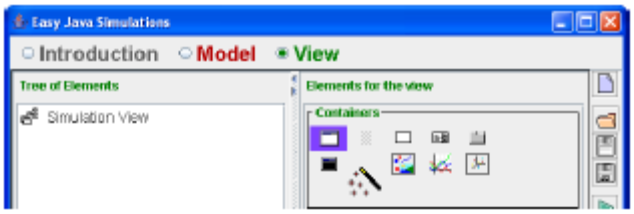
GridPlot  **: χρησιμοποιείται για οπτικοποίηση βαθμωτών πεδίων**

VectorField  **:χρησιμοποιείται για διανυσματικά πεδία**

SurfacePlot  **:σχεδιάζει 3-διάστατες γραφικές παραστάσεις βαθμωτών πεδίων**

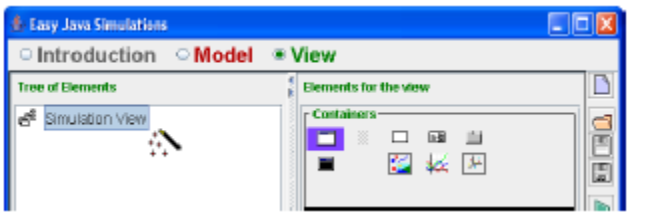
4.2.2 Εισαγωγή στοιχείων στην Θέαση

Για την εισαγωγή του στοιχείου αρχικά επιλέγουμε αυτό. Επιλέγοντας το στοιχείο το εικονίδιο του στοιχείου αλλάζει χρώμα. Επιπλέον ο cursor αλλάζει και εμφανίζεται το μαγικό ραβδί(magic wind)  .



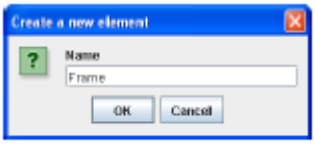
Εικόνα: η επιλογή ενός στοιχείου τύπου "Frame"

Η τοποθέτηση του νέου στοιχείου θα γίνει βάζοντας το μαγικό ραβδί στον υποδοχέα που θα φιλοξενήσει το στοιχείο που επιλέξαμε. **Αν το στοιχείο που έχουμε επιλέξει είναι "frame" ή "dialog" τότε δεν χρειάζεται «γονέας» και το στοιχείο προστίθεται απ ευθείας στο "Simulation View".**



Εικόνα: η επιλογή «γονέα» για το νέο στοιχείο

Κάθε στοιχείο χρειάζεται ένα όνομα και το Ejs προτείνει ένα βασιζόμενο στον τύπο του στοιχείου.

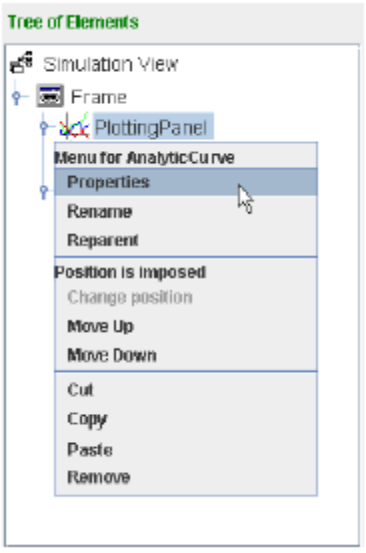


Εικόνα: εισαγωγή ονόματος

Για το νέο στοιχείο θα ζητηθεί ο προσδιορισμός της θέσης του(όταν ο «γονέας» χρησιμοποιεί το "border") οπότε θα εμφανισθεί ένα πλαίσιο που θα ζητά την επιλογή της θέσης του στοιχείου.

Εικόνα: η επιλογή της θέσης του στοιχείου

Έχοντας θέσει όλα τα στοιχεία που χρειαζόμαστε, είναι πιαθνό να θέλουμε να αλλάξουμε κάποια στοιχεία ή να τροποποιήσουμε τη δομή που έχουμε ήδη φτιάξει. Σε αυτή τη περίπτωση το Ejs μας προσφέρει την εξής δυνατότητα. Κάθε στοιχείο έχει ένα popup menu που μπορούμε να το καλέσουμε κάνοντας δεξί κλίκ πάνω του.

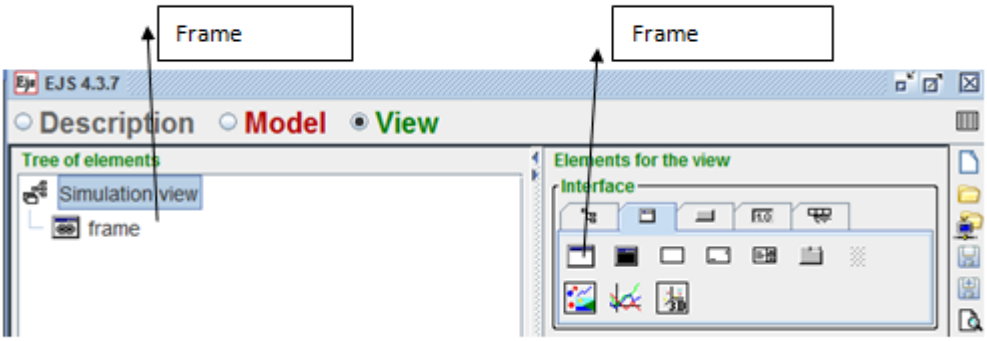


Εικόνα: popup menu για στοιχεία

Το πρώτο στοιχείο που εμφανίζεται στο popupmenu είναι οι « ϊδιότητες», και ίσως είναι το πιο σημαντικό στοιχείο καθώς μας επιτρέπει να αλλάζουμε τις ιδιότητες του στοιχείου. Μια άλλη σημαντική επιλογή που εμφανίζεται είναι η διαχείριση της θέσης του στοιχείου μέσα στον «γονέα» του. Η θέση του στοιχείου μπορεί να έχει προσδιορισθεί από εμάς ή και από την «πολιτική» που εφαρμόζει ο γονέας όταν τοποθετεί τα στοιχεία.

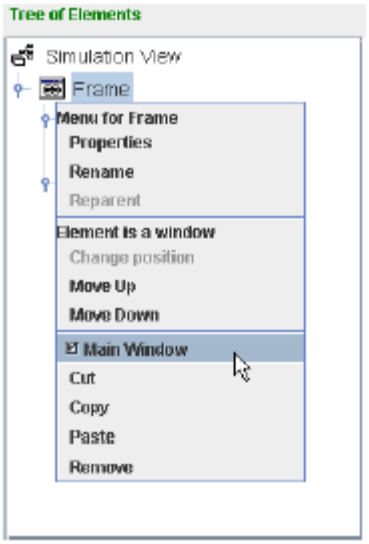
Παρατήρηση: Ειδικά για το στοιχείο Frame θα πρέπει να προσέξουμε το εξής:

Ενώ όταν επιλέγουμε το Frame εμφανίζεται η εικόνα  , **όταν τοποθετούμε το στοιχείο στο «δέντρο της προσομοίωσης» εμφανίζεται η εξής εικόνα**  **.**



Εικόνα: Το στοιχείο Frame

Αυτό συμβαίνει για τον εξής λόγο: όταν η προσομοίωση τρέχει στη μορφή ενός applet, δηλαδή μέσα ε μια HTML σελίδα, τότε το κύριο παράθυρο «main window" θα εμφανισθεί μέσα στην σελίδα και το εικονίδιο  προσδιορίζει αυτή την επιλογή.

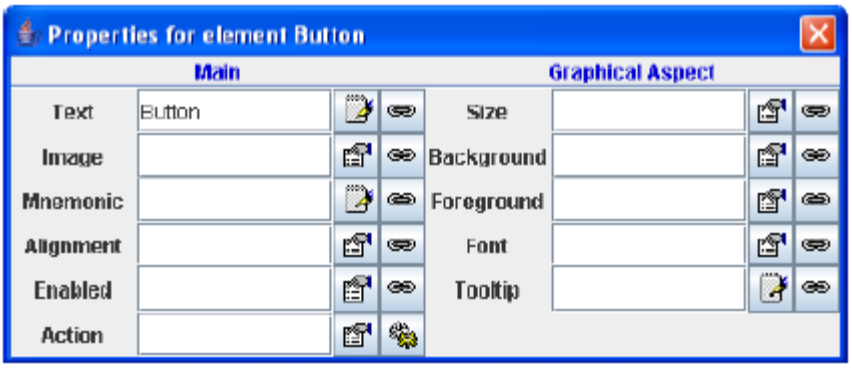


Εικόνα: Λεπτομέρειες για την επιλογή «main window».

4.3. Επεξεργασία των ιδιοτήτων των στοιχείων

Οι ιδιότητες είναι «εσωτερικές τιμές» που προσδιορίζουν με ποιο τρόπο εμφανίζονται και συμπεριφέρονται τα στοιχεία. Μπορούμε να τροποποιήσουμε αυτές τις ιδιότητες χρησιμοποιώντας το panel που αντιστοιχεί στις ιδιότητες κάθε στοιχείου. Αυτό μπορεί να γίνει από το popup menu του στοιχείου επιλέγοντας τις «ιδιότητες», η με διπλό κλίκ πάνω στο στοιχείο.

Για παράδειγμα για το στοιχείο τύπου «Κουμπί εντολών-Button», το panel εμφανίζεται στην παρακάτω εικόνα.

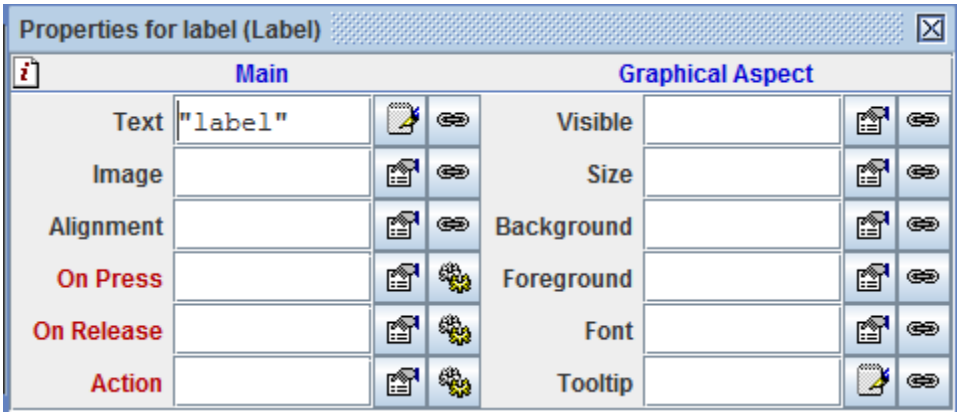


Εικόνα: το panel με τις ιδιότητες για το στοιχείο τύπου «Κουμπί εντολών-Button»

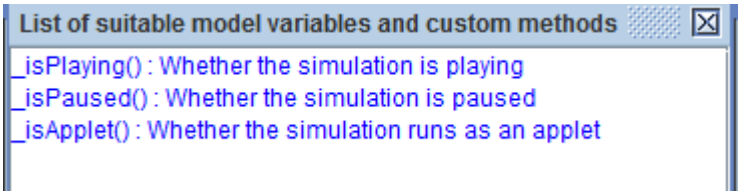
Όταν δίπλα στο στοιχείο εμφανίζεται το εικονίδιο  **τότε δεν υπάρχει editor που να τον χρειαζόμαστε καθώς το μόνο που έχουμε να κάνουμε είναι**

να γράψομε μια αριθμητική τιμή ή ένα κείμενο. Αντίθετα όταν δίπλα στο στοιχείο εμφανίζεται το εικονίδιο , τότε το Ejs μας παρέχει ένα editor για να επεξεργαστούμε αυτήν την ιδιότητα.

Για παράδειγμα για το στοιχείο Label-ετικέτα, στο πεδίο "Text" γράφουμε απλά ένα κείμενο, ενώ στο πεδίο "Visible", όταν πατήσουμε το , θα εμφανισθεί η παρακάτω εικόνα



Εικόνα: οι ιδιότητες που εμφανίζονται στο στοιχείο «ετικέτα» στο πεδίο "Text"



Εικόνα: οι ιδιότητες που εμφανίζονται στο στοιχείο «ετικέτα» στο πεδίο "Visible"

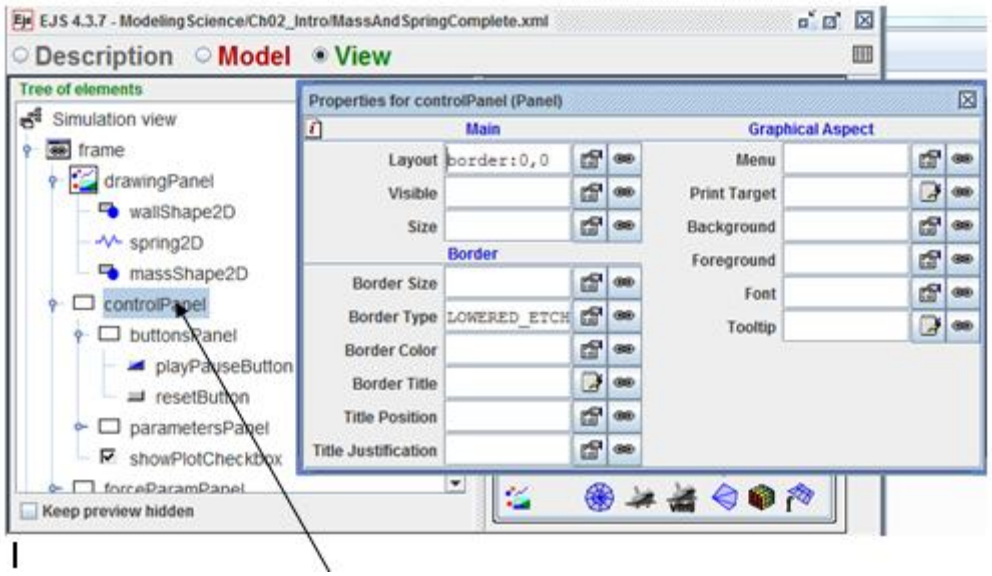
4.4 Συσχέτιση ιδιοτήτων με μεταβλητές και μεθόδους του μοντέλου

Οι ιδιότητες μπορούν να συσχετισθούν με τις μεταβλητές του μοντέλου χρησιμοποιώντας εκφράσεις της Java καθώς και με ενέργειες χρησιμοποιώντας προτάσεις της Java που περιλαμβάνουν μεθόδους του μοντέλου.
Αυτές οι συσχετίσεις είναι το βασικό στοιχείο για την δημιουργία αλληλεπίδρασης ενώ η ανάπτυξη αυτών των συσχετίσεων είναι μια σχετικά εύκολη διαδικασία η οποία πραγματοποιείται με την συμπλήρωση στο πεδίο της ιδιότητας της μεταβλητής ή της αντίστοιχης έκφρασης της Java.

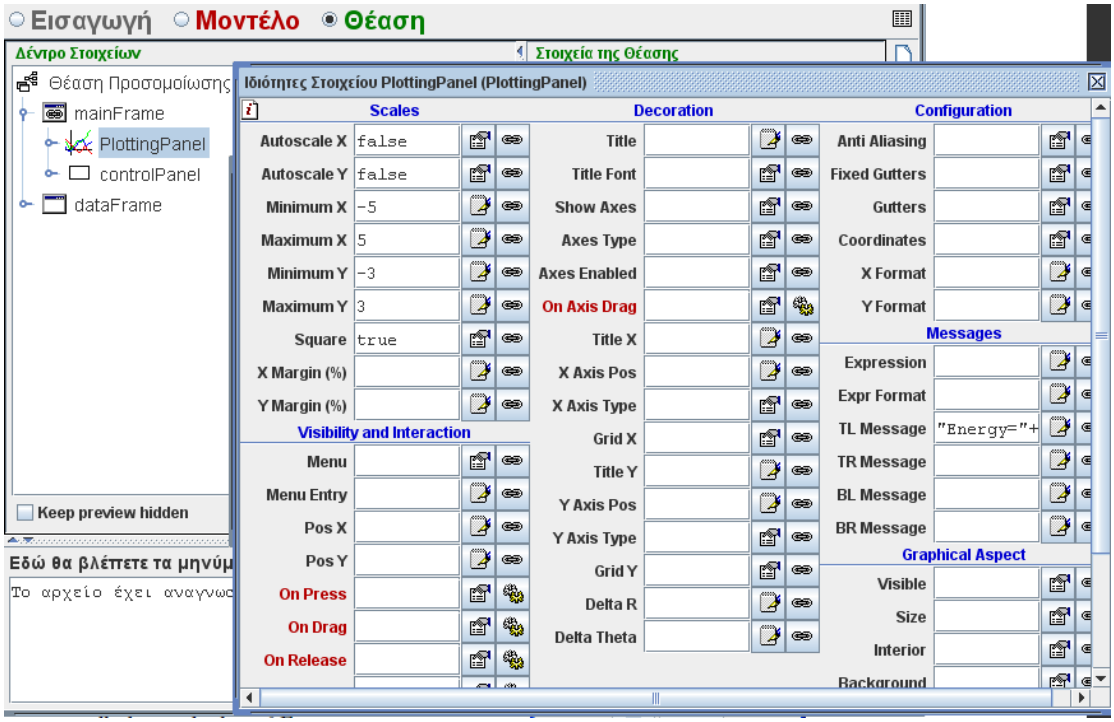
Για την διευκόλυνση αυτής της διαδικασίας υπάρχουν δυο στήλες για κάθε ιδιότητα του στοιχείου.

Η μια στήλη προέρχεται από το εικονίδιο  και ή άλλη στήλη από το εικονίδιο  .

Για παράδειγμα, για το στοιχείο controlPanel, παρατηρούμε στην παρακάτω εικόνα, τα δυο εικονίδια που αναφέραμε. Δοκιμάστε να πατήσετε σε καθένα από αυτά για κάθε ιδιότητα του στοιχείου για να εξερευνήσετε τις δυνατότητες που προσφέρει το Ejs.

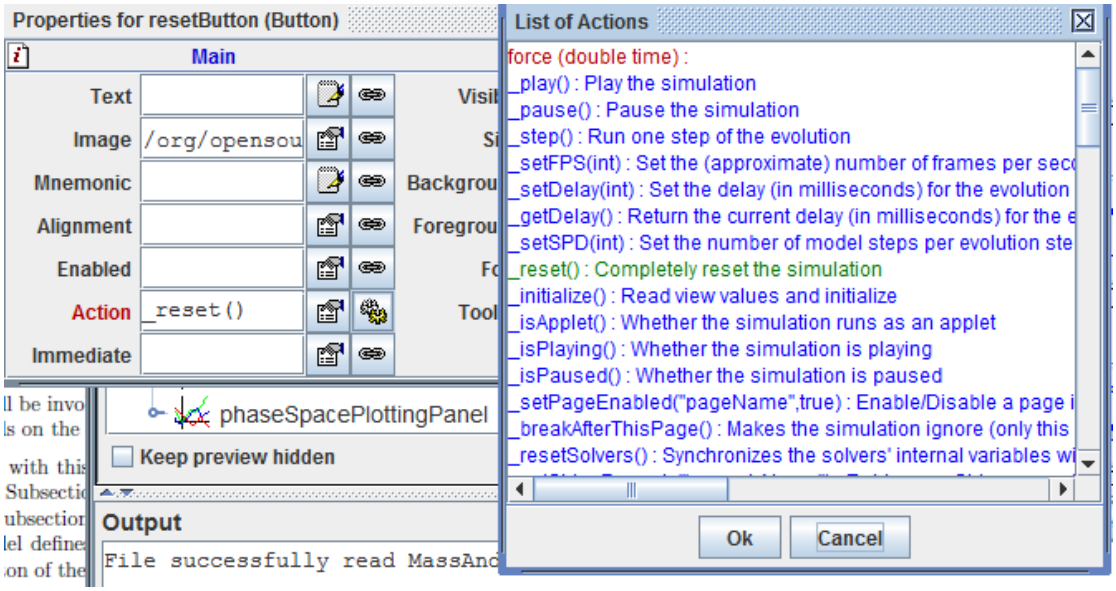


Εικόνα: Οι ιδιότητες του κουμπιού controlPanel και η διασύνδεσή τους με μεταβλητές



Εικόνα: Οι ιδιότητες του κουμπιού PlottingPanel και η διασύνδεσή τους με μεταβλητές

Το εικονίδιο  αντιστοιχεί σε ενέργειες , δηλαδή όταν το εικονίδιοαυτό εμφανίζεται δίπλα σε μια ιδιότητα, αυτό σημαίνει ότι αυτή η ιδιότητα είναι μια από της λεγόμενες ιδιότητες ενεργειών(action properties). Πρακτικά αυτό σημαίνει ότι ότι μπορεί να χρησιμοποιηθεί για να ορίσει μια ενέργεια η οποία θα κληθεί όταν ο χρήστης αλληλεπιδράσει με το στοιχείο. Στην παρακάτω εικόνα εμφανίζονται οι προσαρμοσμένες (custom) μέθοδοι που έχουμε δημιουργήσει στο μοντέλο μας αλλά και μέθοδοι που παρέχει το Ejs για να επιλέξουμε.



Εικόνα: οι μέθοδοι για την ενέργεια _reset() .

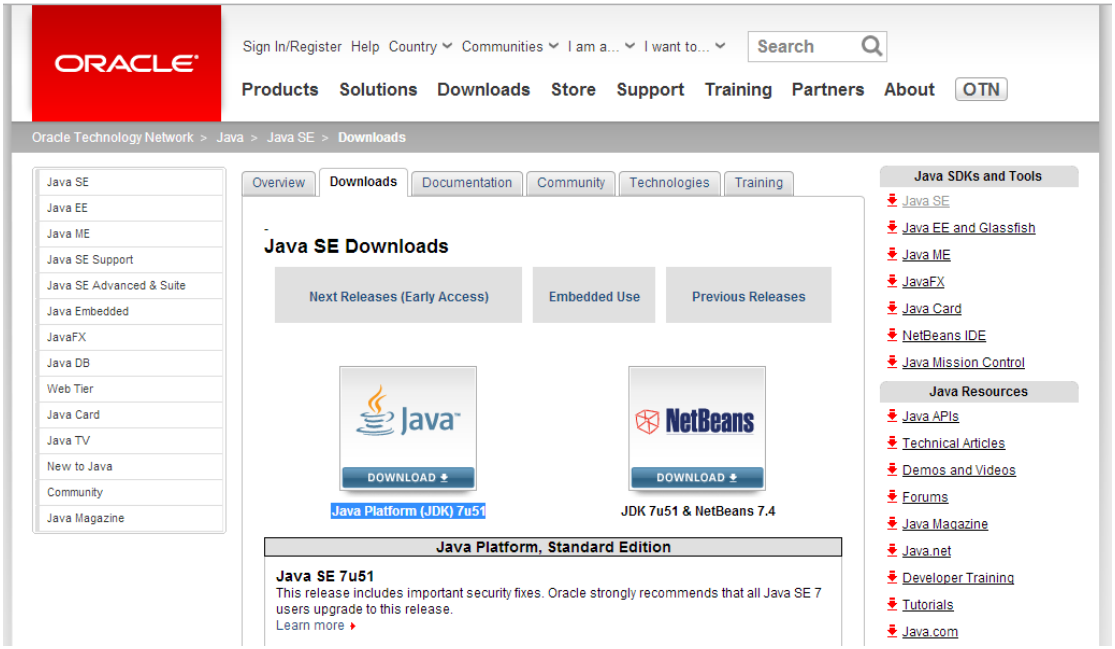
Παρατήρηση: Όταν εισάγουμε ένα στοιχείο και επιχειρούμε να συμπληρώσουμε το πεδίο που αντιστοιχεί σε μια ιδιότητα, τότε –στην περίπτωση που αυτή η ιδιότητα δε αντιστοιχεί σε ενέργεια- παρατηρούμε ότι μέσα στο πεδίο εμφανίζεται λευκό χρώμα. Όταν επεξεργαζόμαστε όμως αυτό το πεδίο, τότε αυτόματα το χρώμα αλλάζει σε κίτρινο, ώστε το Ejs να μας προειδοποιήσει ότι ναι με τροποποιούμε αυτό αλλά όμως αυτή η τροποποίηση δεν έχει ακόμα εφαρμοστεί στην ιδιότητα. Έτσι, μόνο όταν πατήσουμε enter το χρώμα ξαναγίνεται λευκό. Έτσι είναι σημαντικό να θυμάστε να μην αφήνετε τα πεδία με κίτρινο χρώμα. Αντίθετα για τις ιδιότητες τύπου «action», τα πράγματα είναι διαφορετικά. Σε αυτές οι τιμές χρησιμοποιούνται όταν δημιουργείται η προσομοίωση, και επομένως οτιδήποτε αναγράφεται σε αυτά χρησιμοποιείται αυτόματα, και ποτέ το χρώμα σε αυτές δεν θα είναι κίτρινο,

Παρατήρηση: Οι ιδιότητες τύπου String. Χρειάζεται προσοχή όταν χρησιμοποιούμε μεταβλητές τύπου String-οι λεγόμενες αλφαριθμητικές. Για παράδειγμα όταν θέλουμε να εισάγουμε ένα όνομα στο πεδίο «Title», ενός στοιχείου, τότε απαιτείται προσοχή στο εξής: αν γράψουμε π.χ. τη λέξη «συνάρτηση», τότε υπάρχουν οι εξής δυο εκδοχές. Αν γραφεί ως «συνάρτηση» αυτό σημαίνει ότι το στοιχείο θα χρησιμοποιήσει το κείμενο ως μια λέξη-ένα τίτλο. Αν όμως γραφεί ως **%συνάρτηση%, τότε το στοιχείο θα θεωρήσει το κείμενο ως το όνομα μιας μεταβλητής.**

Κεφάλαιο 5 . Εξαγωγή και χρήση των αρχείων του Ejs

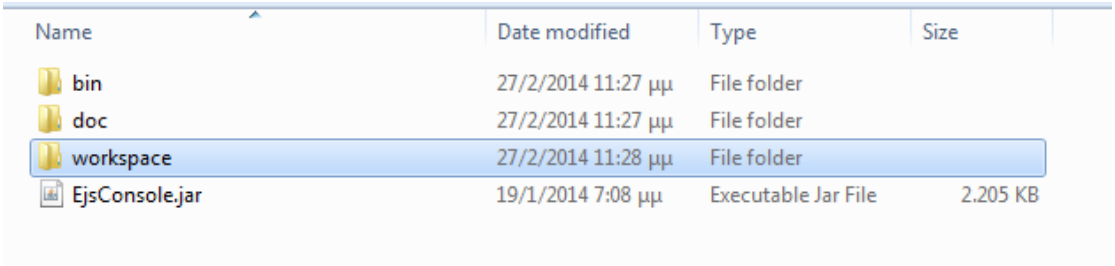
Το Easy Java Simulations απαιτεί για να τρέξει το the Java Development Kit , το οποίο δεν πρέπει να συγχέεται με το Java Runtime Environment (JRE), το οποίο ορισμένες φορές καλείται και Java
plug-in.
Το JRE είναι απλούστερο από το Java Development Kit και επιτρέπει στον φυλλομετρητή να τρέχει Java applets αλλά δεν περιλαμβάνει τον μεταγλωττιστή της Java .
Έτσι με το Java Runtime Environment (JRE), μπορούμε να τρέχουμε εφαρμογές που έχουν δημιουργηθεί με το Ejs, αλλά δεν μπορούμε να δημιουργήσουμε προσομοιώσεις χρησιμοποιώντας το Ejs.
Επομένως για να δημιουργηθούν προσομοιώσεις με τοEjs , θα πρέπει να κατεβάσουμε το JDK από την διεύθυνση
http://www.oracle.com/technetwork/java/javase/downloads/index.html

Η τρέχουσα(28-2-2014) έκδοση της Java είναι η Java Platform (JDK) 7u51 και η παρακάτω εικόνα εμφανίζει την διεπαφή όταν αναζητήσουμε να κατεβάσουμε την Java.

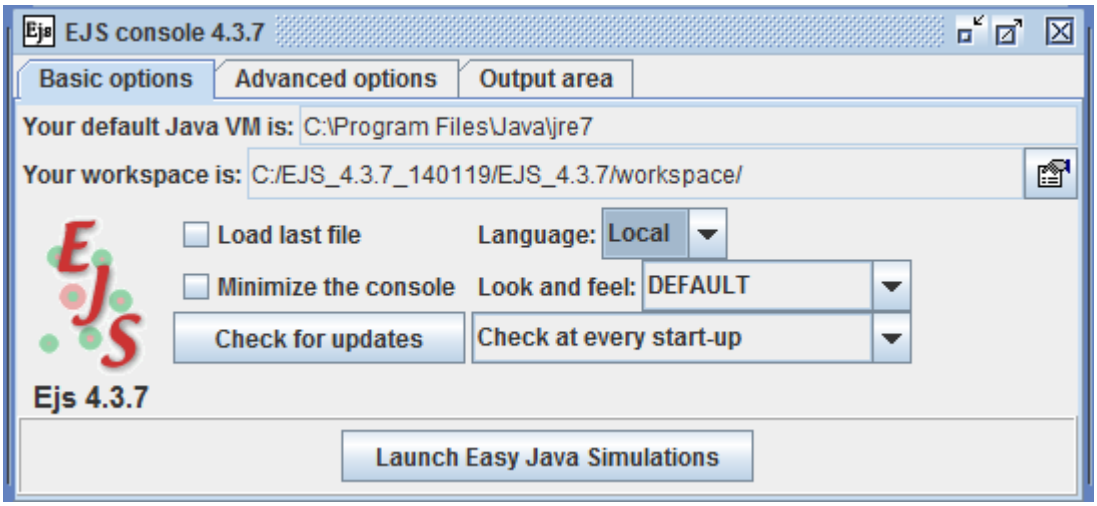


Εικόνα : η ιστοσελίδα για να κατεβάσουμε την Java.

Συνήθως αποθηκεύουμε το Ejs στον C και με διπλό κλίκ στο EjsConsole.jar βγαίνει η παρακάτω εικόνα.



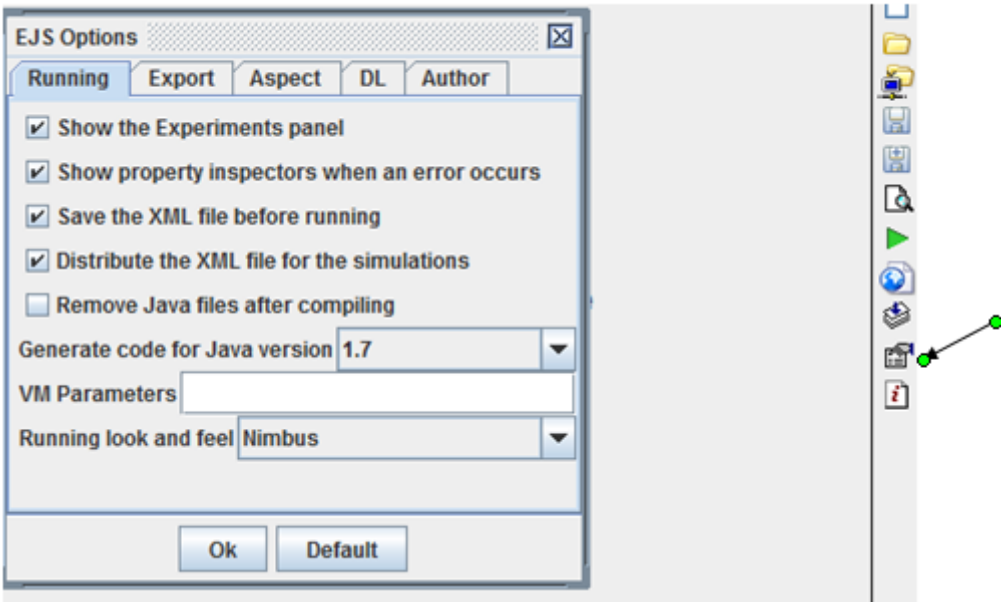
Εικόνα: η διεπαφή του Ejs



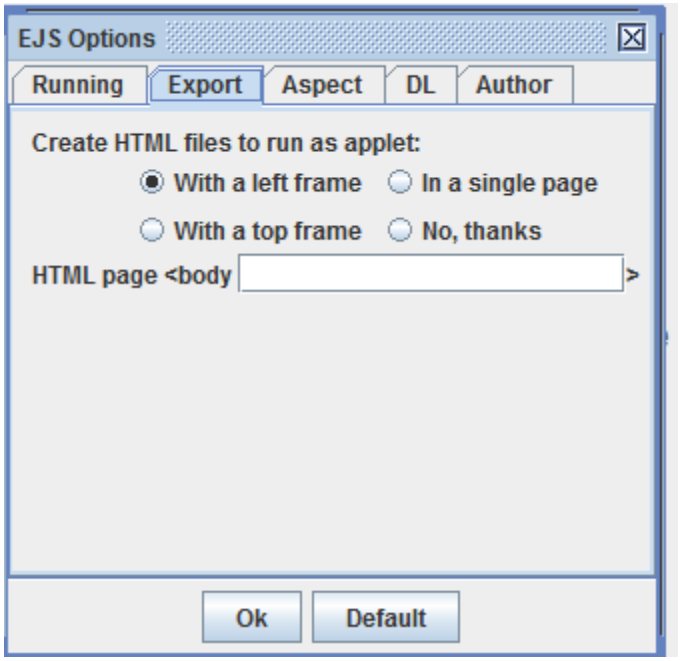
Εικόνα: Η διεπαφή μετά το διπλό κλίκ στο EjsConsole.jar

Όταν δημιουργούμε εφαρμογές, το Ejs(σε οποιοδήποτε λειτουργικό σύστημα) δημιουργεί και τους δικούς του φακέλους που περιέχουν βιβλιοθήκες και επομένως αυτοί οι φάκελοι δεν θα πρέπει να σβήνονται.
Οι προσομοιώσεις που δημιουργούνται με το Easy Java Simulations μπορεί να έχουν τρείς μορφές.
Η πρώτη χρησιμοποιεί απευθείας το Ejs , η δεύτερη «εκτελεί» την προσομοίωση ως "applet" και η τρίτη τρέχει την προσομοίωση ως αυτόνομη εφαρμογή της Java.

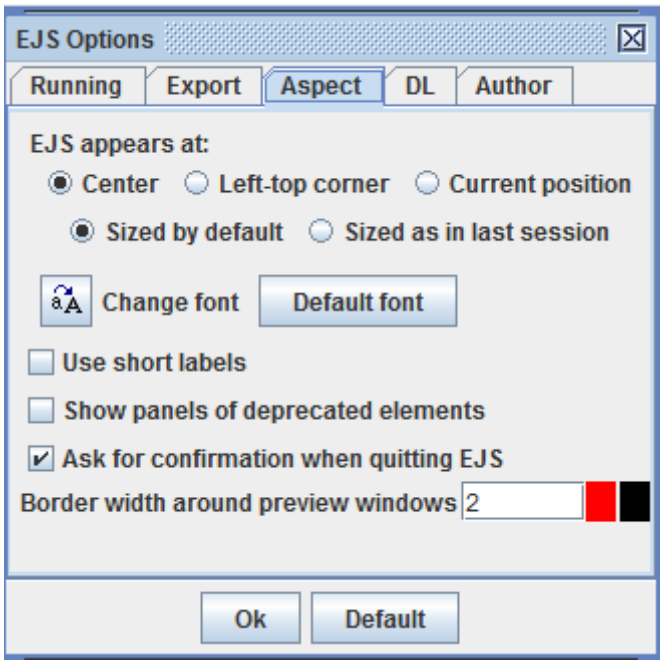
Πατώντας στο εικονίδιο  έχουμε την παρακάτω εικόνα



Εικόνα: Οι επιλογές για την εμφάνιση των αρχείων που έχουμε δημιουργήσει



Εικόνα: Οι επιλογές για την εξαγωγή των αρχείων που έχουμε δημιουργήσει σε μορφή HTML



Εικόνα: Οι επιλογές για την εξαγωγή των αρχείων που έχουμε δημιουργήσει σε μορφή .ejs

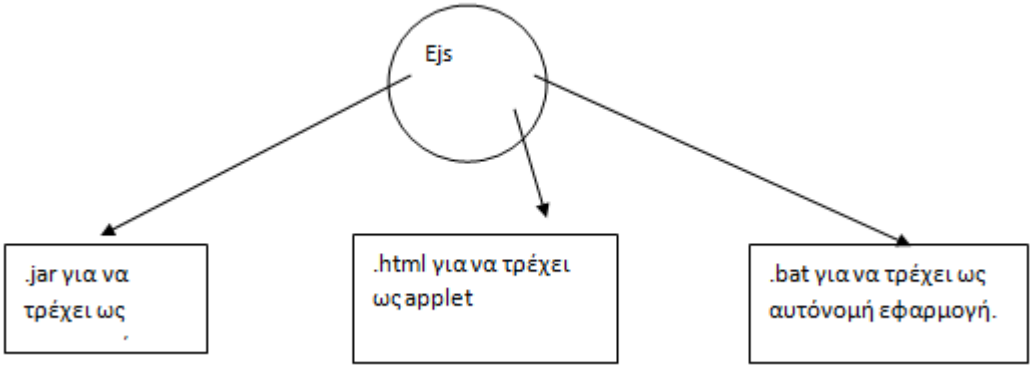
Παρατήρηση: Έτσι αν θέλουμε να εξαγάγουμε σε μορφή applet την εφαρμογή που έχουμε δημιουργήσει, πατάμε το κουμπί και επιλέγουμε «Export website with applets".

Εικόνα: Οι επιλογές για την αναγραφή των στοιχείων του συγγραφέα της προσομοίωσης

Η «διανομή» μιας προσομοίωσης που δημιουργήθηκε με το Ejsγίνεται απλά με την παροχή στον χρήστη του φακέλου της προσομοίωσης . Για παράδειγμα από τις έτοιμες εφαρμογές του Ejs μπορείτε να επιλέξετε την Spring.xml. Ο χρήστης θα πρέπει απλά να ξέρει πως να ξεκινήσει το Ejs και να φορτώσει αυτόν τον φάκελο. Να σημειωθεί εδώ ότι δημιουργώντας μια εφαρμογή με το Ejs, δημιουργείται αυτόματα και το αρχείο με την κατάληξη .xml.
Τέλος, τρέχοντας ο χρήστης την εφαρμογή με την κατάληξη .xml έχει τη δυνατότητα να δει και τον κώδικα-μοντέλο που έχει δημιουργήσει ο συγγραφέας της εφαρμογής.

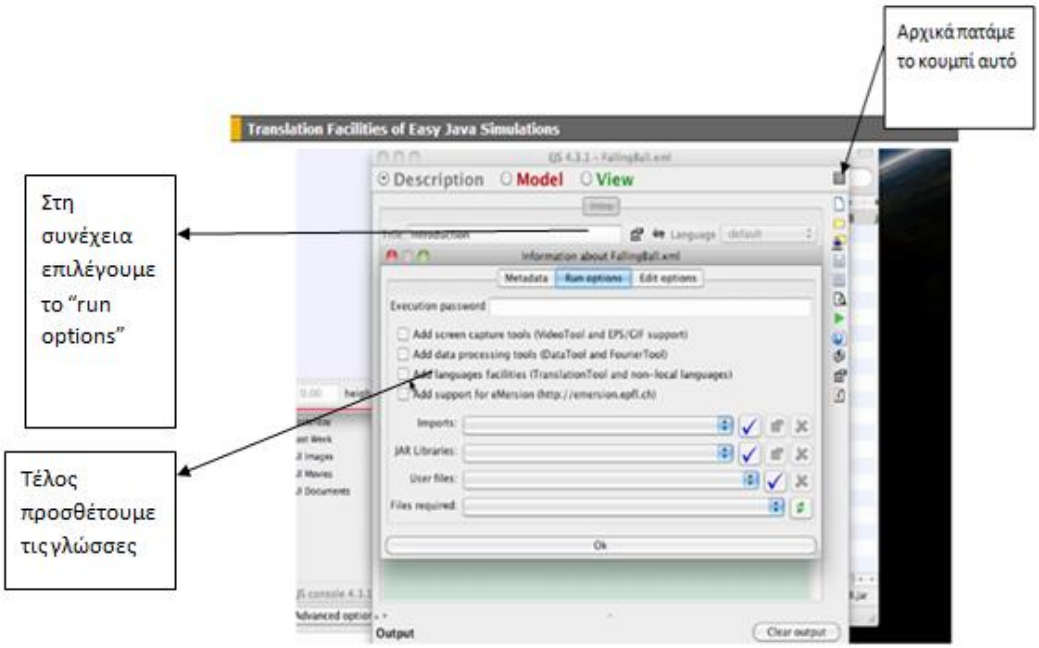
Όταν δημιουργούμε εφαρμογές με το Ejs, είναι σωστό να αναφέρουμε τον επίσημο Web server για το λογισμικό, Ejs, http://fem.um.es/Ejs.

Όταν δημιουργούμε HTML σελίδες , θα πρέπει να προσέχουμε μέσα στον φάκελο που τις περιέχει να υπάρχει και το .jar αρχείο καθώς αυτό χρησιμοποιείται από την HTML σελίδα.



Εικόνα: οι διάφορες μορφές που μπορεί να τρέξουν οι εφαρμογές που δημιουργούνται με το Ejs.

Από την παρακάτω διεύθυνση-
http://www.compadre.org/OSP/tutorials/EJS_Translation/
- μπορείτε να παρακολουθήσετε τον τρόπο μετάφρασης στο Ejs.



Εικόνα: η διαδικασία εισαγωγής γλώσσας στο Ejs-το αρχικό βήμα

Αφού ακολουθήσουμε την παραπάνω διαδικασία, στη συνέχεια πατάμε το κουμπί  **. Και από το περιβάλλον που εμφανίζεται γράφουμε στα ελληνικά π.χ. τον τίτλο, τα labels κλπ.**

Translation of Ejs-Psycharis/MonteCarloPi.ejs

Language: Ελληνικά

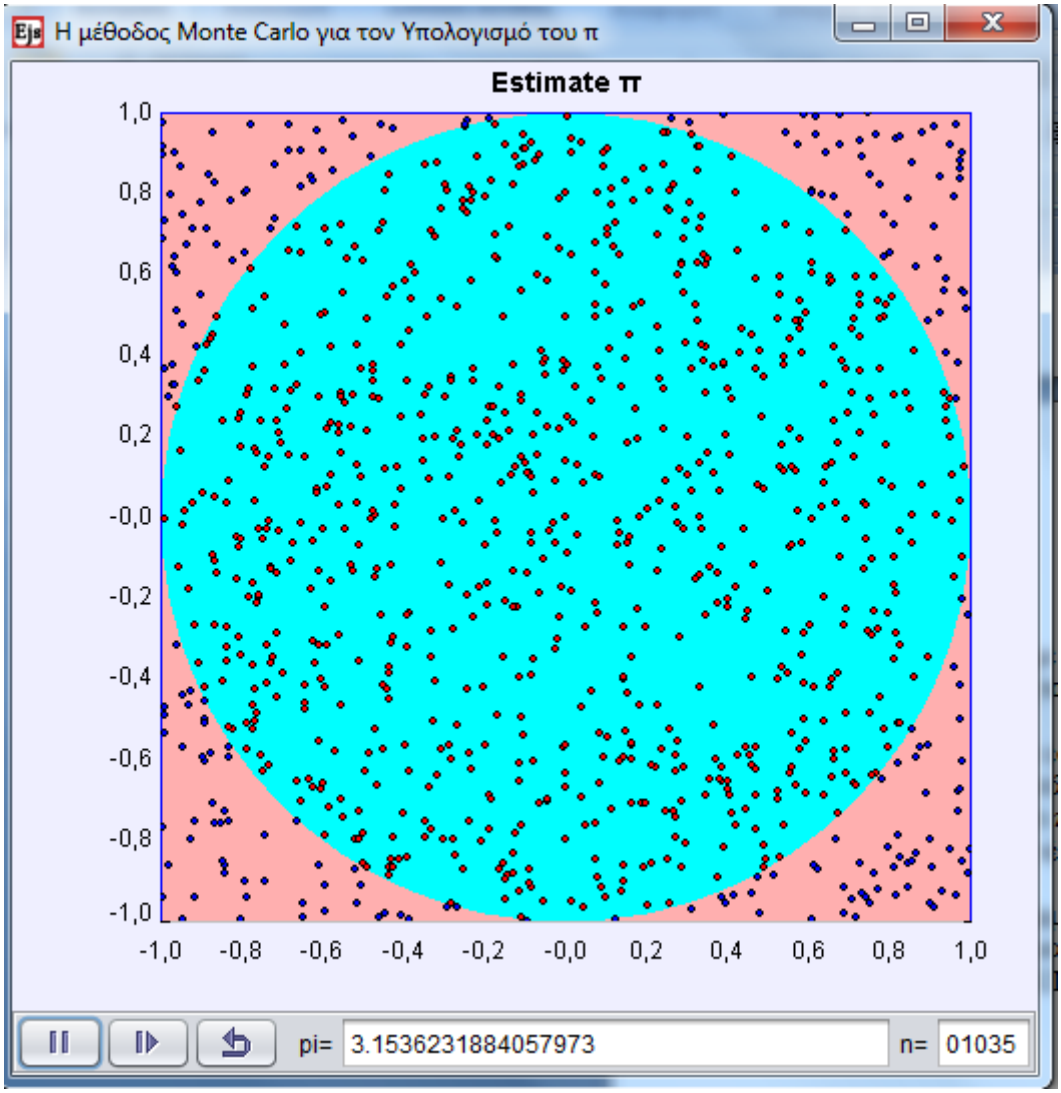
Property	Value
View.monteCarloFrame.size	512,506
View.monteCarloFrame.title	Η μέθοδος Monte Carlo για τον Υπολογισμό του π
View.nField.format	00000
View.nField.size	50,24
View.nField.tooltip	# of samples
View.nLabel.text	n=
View.nLabel.tooltip	# of samples
View.piField.format	0.0000000000000000
View.piField.size	80,24
View.piField.tooltip	estimate of pi
View.plottingPanel2.title	Estimate π
View.restButton.image	/org/opensourcephysics/resources/controls/images/r...
View.startStopButton2.imageOff	/org/opensourcephysics/resources/controls/images/p...
View.startStopButton2.imageOn	/org/opensourcephysics/resources/controls/images/p...
View.startStopButton2.textOff	
View.startStopButton2.textOn	
View.stepButton.image	/org/opensourcephysics/resources/controls/images/s...
View.λαβελ pi.text	pi=
View.λαβελ pi.tooltip	εκτίμηση του π

Εικόνα: η διαδικασία εισαγωγής της ελληνικής γλώσσας

Για να προσθέσουμε την γλώσσα που μας ενδιαφέρει βάζουμε τα δυο πρώτα αρχικά της γλώσσας, π.χ. για ελληνικά el.

Όταν τρέξουμε την εφαρμογή και έχοντας εισαγάγει την γλώσσα που θέλουμε, το πρόγραμμα μας ρωτά αν θέλουμε να αλλάξει το αρχείο με κατάληξη .xml σε κατάληξη .ejs, κάτι που πρέπει να αποδεχτούμε ώστε η εφαρμογή μας να μην έχει πρόβλημα με τους νέους χαρακτήρες(λόγω της γλώσσας) που θα εισαχθούν.

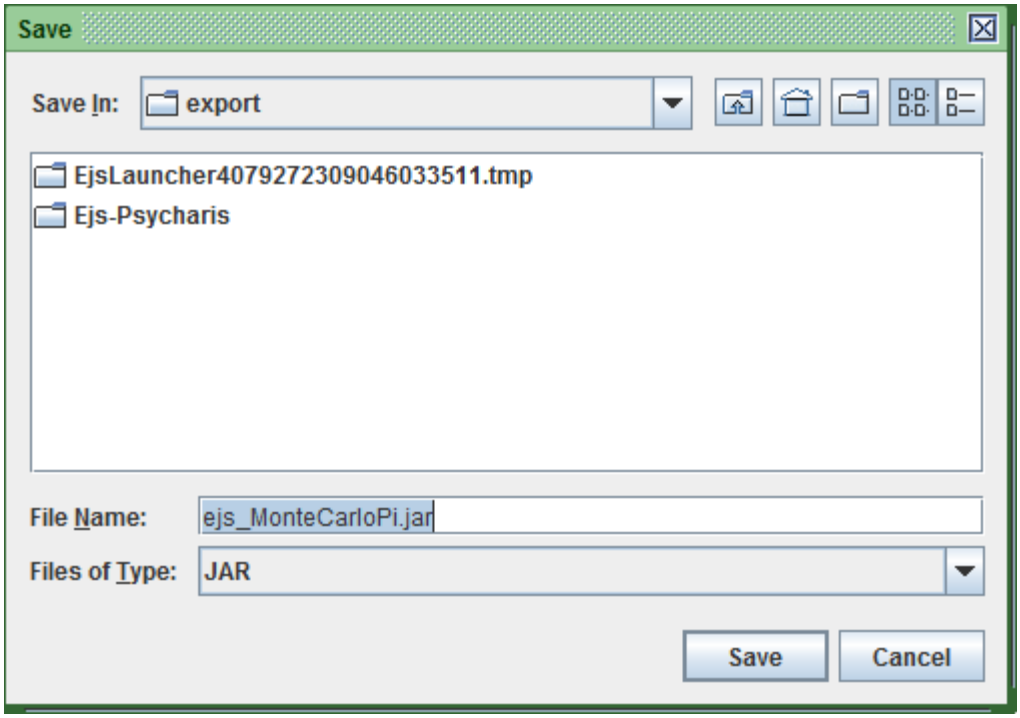
Τρέχοντας την εφαρμογή, θα πρέπει να προσέξουμε ότι αρχικά εμφανίζεται η αγγλική γλώσσα. Για να εμφανισθεί η ελληνική γλώσσα θα πρέπει να πατήσουμε δεξί κλίκ και από την επιλογή GUI Options , Επιλέγουμε language και στη συνέχεια την ελληνική γλώσσα.



Εικόνα: η οθόνη εισάγοντας την ελληνική γλώσσα

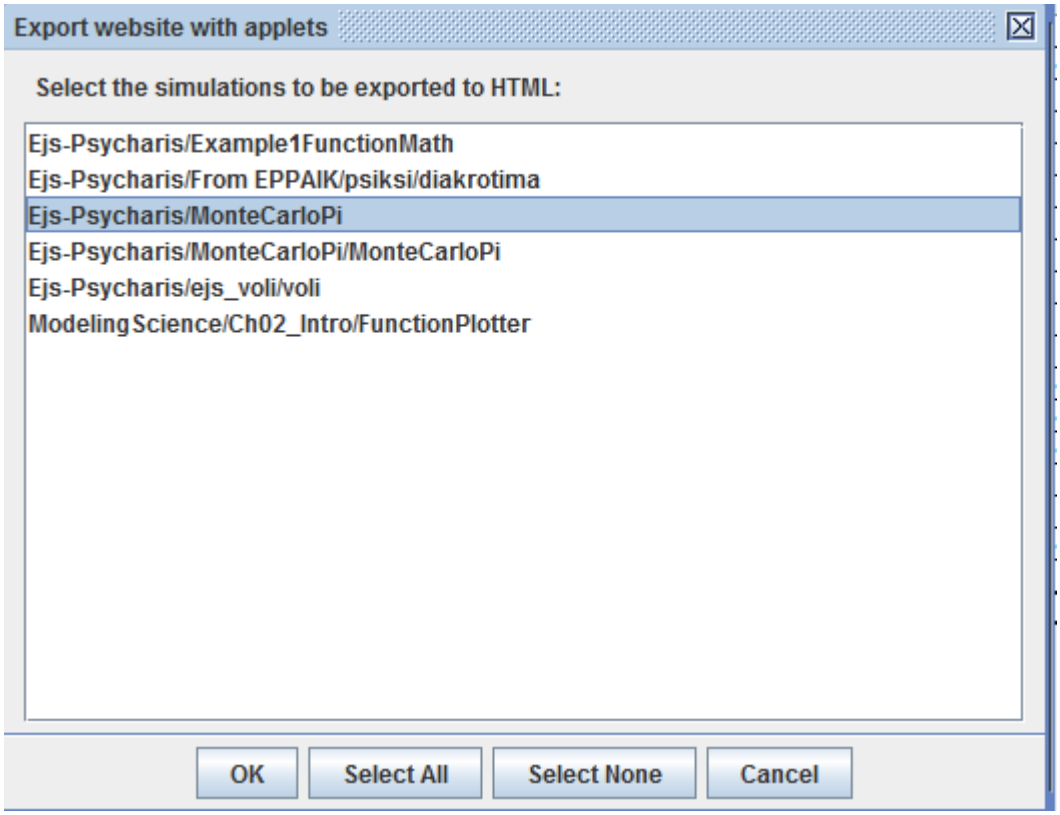
Τέλος θα ασχοληθούμε με το τρέξιμο της εφαρμογής ως applet.

Το σκοπό αυτό πατάμε το κουμπί  και μεταφέρουμε το αρχείο με τη μορφή .jar στον φάκελο export.



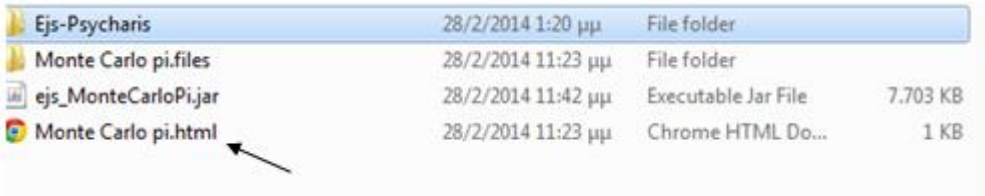
Εικόνα: μεταφορά του αρχείου στον φάκελο export.

Στη συνέχεια πατάμε πάλι το κουμπί  και επιλέγουμε export website with applets. Με αυτό τον τρόπο μετατρέπουμε την προσομοίωση σε applet που βρίσκεται στον φάκελο export.



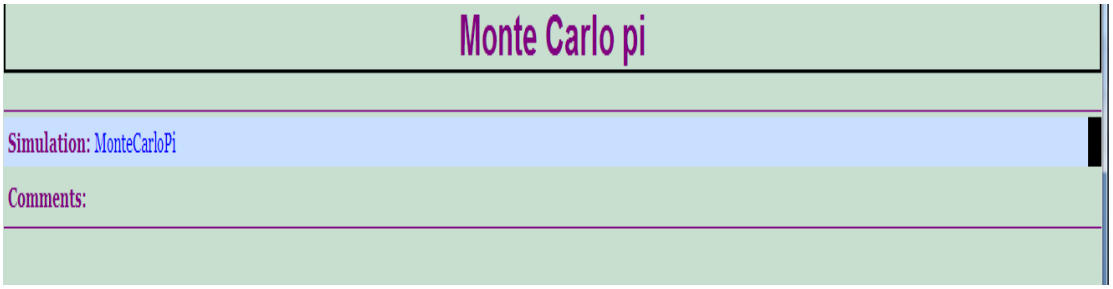
Εικόνα: μετατροπή του αρχείου σε applet

Στην παρακάτω εικόνα φαίνεται το αρχείο σε html μορφή.

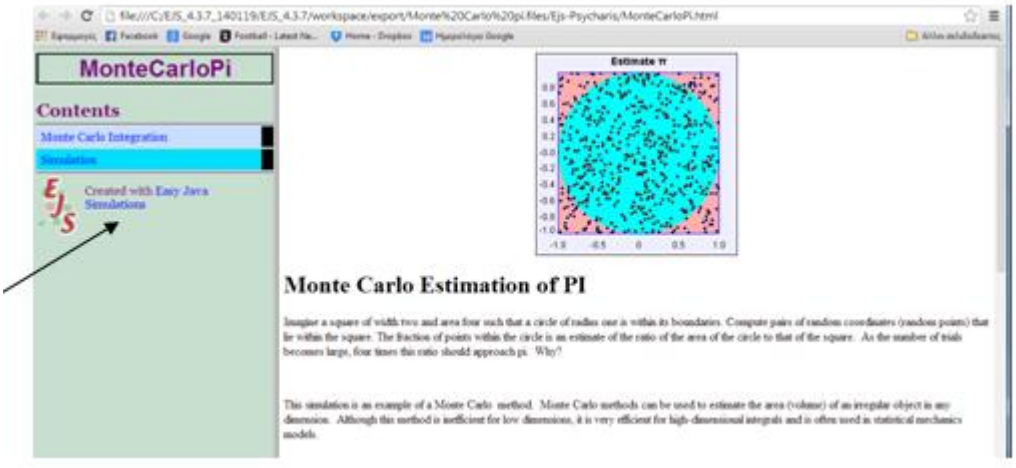


Εικόνα: το αρχείο σε html μορφή

Πατώντας πάνω στο αρχείο εμφανίζεται η παρακάτω εικόνα.

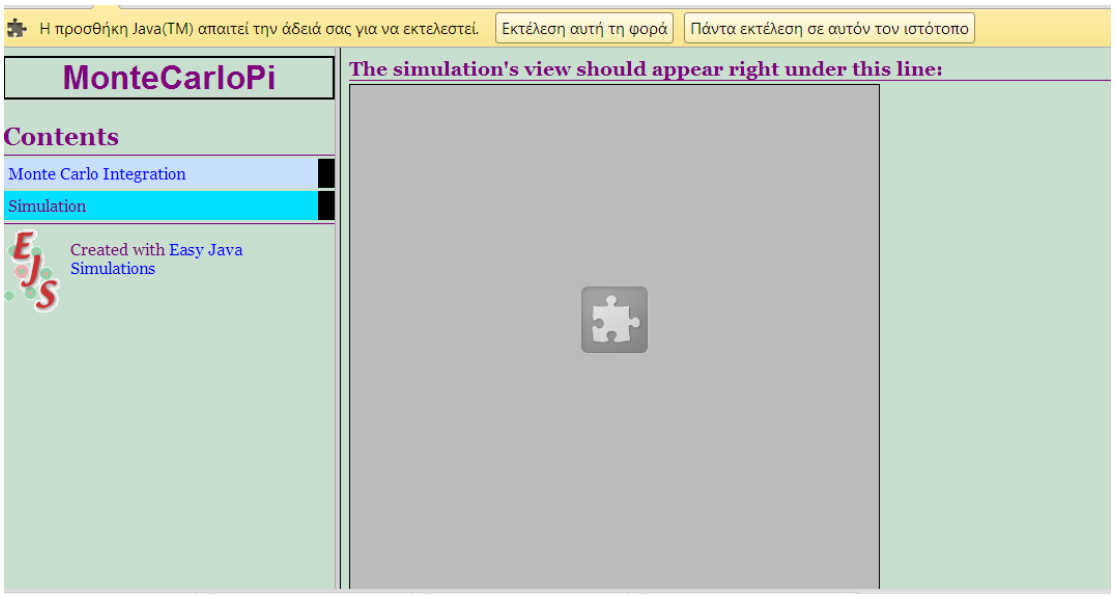


Εικόνα: το αρχείο προσομοίωσης σε html μορφή
Πατώντας πάνω στη λέξη simulation, εμφανίζεται η παρακάτω εικόνα

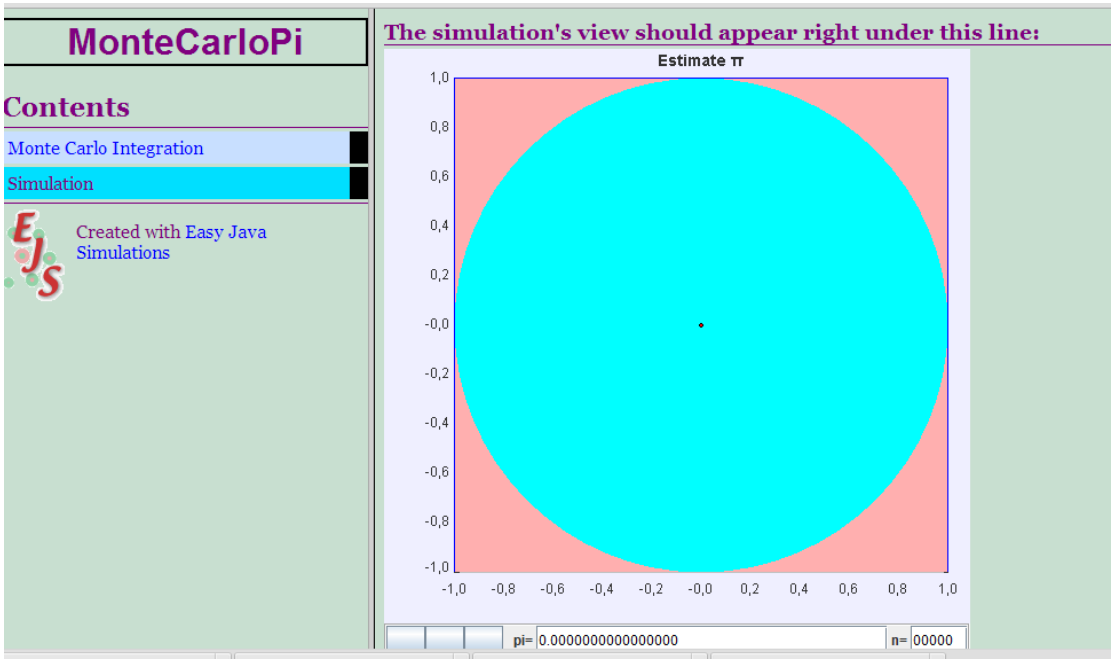


Εικόνα: το αρχείο προσομοίωσης στον φάκελο export

Πατώντας πάλι στην λέξη Simulation , εμφανίζεται η παρακάτω εικόνα όπου δεχόμαστε να τρέξει στον Υπολογιστή μας.



Εικόνα: η προετοιμασία για το τρέξιμο του applet



Εικόνα: το τρέξιμο του applet