Γραφική προσομοίωση πλανητικών τροχιών

Λίγγας Ραφαήλ A.E.M : 13731
Επιβλέπων καθηγητής: Δρ. Γ. Βουγιατζής

Τμήμα Φυσικής , Σχολή θετικών επιστημών,
Αριστοτέλειο πανεπιστήμιο Θεσσαλονίκης
Ιούνιος 2017

Aristotle University of
Thessaloniki

Περίληψη

Σκοπός της εργασίας αυτής είναι η κατασκευή κώδικα με τη χρήση OpenGL που θα αναπαριστάνει τις τροχιές πλανητών ή άλλων σωμάτων στον τρισδιάστατο χώρο λαμβάνοντας υπόψιν το πρόβλημα των N-σωμάτων. Η ολοκλήρωση των διαφορικών εξισώσεων του συστήματος γίνεται με συμπλεκτικό ολοκληρωτή 6ης τάξης. Οι αρχικές συνθήκες των πλανητών ορίζονται με τη χρήση των Κεπλεριανών τροχιακών στοιχείων τους. Στο πρόγραμμα αναπαριστώνται το ηλιακό μας σύστημα, τρία εξωπλανητικά συστήματα (τα στοιχεία των οποίων πάρθηκαν από τις αστρονομικές βάσεις δεδομένων openexoplanetcatalogue.com και exoplanet.eu) και ένα παράδειγμα ασταθούς συστήματος. Ο χρήστης μπορεί να εισάγει το δικό του πλανητικό σύστημα κατασκευάζοντας ένα ASCII αρχείο αρχικών συνθηκών και κάνοντας κάποιες μικρές αλλαγές στον κώδικα όπως περιγράφεται στην εργασία.

Abstract

The purpose of this project is the construction of a C-code using OpenGL that depicts the motion of planets or other bodies in three dimensional space taking into account the N-body gravitational problem. The integration of the system's differential equations is being done with a 6th order symplectic integrator. The initial conditions of the planets are defined through the use of their orbital elements. The program depicts our solar system , three exoplanet systems (the details of which were taken from the astonomical databases openexoplanetcatalogue.com and exoplanet.eu) and one unstable system. The user can insert his/her own planetary system by constructing an ASCII file with the initial conditions and making slight changes in the code as it is described in the project.

Contents

1 Εισαγωγή.

1.1 Το ηλιακό σύστημα.

Χιλιάδες χρόνια πριν, οι αρχαίοι ανεπτυγμένοι πολιτισμοί προσπαθούσαν να εξηγήσουν με ορθολογιστικό τρόπο τη κίνηση των ουράνιων σωμάτων προτείνοντας διάφορα μοντέλα για την ερμηνεία τους.

Ο Αρίσταρχος ο Σάμιος (310-230 π.Χ.) ήταν ο πρώτος αστρονόμος που υποστηριξε το ηλιοκεντρικό μοντέλο. Ισχυρίστηκε ότι ο Ήλιος και όλοι οι αστέρες παραμένουν διαρκώς ακίνητοι ενώ η Γη, την οποία τοποθέτησε σωστά ως τον τρίτο πλανήτη από τον Ήλιο, κάνει τόσο μια ετήσια περιστροφή γύρω από αυτόν όσο και μια ημερήσια περιστροφή περί τον άξονά της.

Κατά την αναγέννηση ο Nicolaus Copernicus (1473-1543) το 1543 δημοσίευσε το έργο με τίτλο De Revolutionibus Orbium Celestium (Περί της περιστροφής των Ουράνιων Σφαιρών) συνεχίζοντας τη θεωρία του ηλιοκεντρικού συστήματος. Στο μοντέλο αυτό ο Ήλιος κατέχει το κέντρο του συστήμα- τος ως ακλόνητο σώμα, η Γη περιστρέφεται γύρω από τον άξονα της και περιφέρεται γύρω απο τον Ήλιο, όπως και οι υπόλοιποι πλανήτες, αλλά με κυκλικές τροχιές.

Ο Johannes Kepler (1571–1630) από παρατηρήσεις ουράνιων σωμάτων διατύπωσε τους τρεις νόμους που προσδιορίζουν την κίνηση των πλανητών. Στο μοντέλο αυτό οι τροχιές των πλανητών είναι ελλειπτικές και όχι κυκλικές, με τη μία εστία της έλλειψης να τη κατέχει ο Ήλιος (1 ος νόμος). Το 1609 δημοσίευσε το έργο Astronomia Nova στο οποίο διατυπώνει τους δύο πρώτους νόμους, ενώ το 1619 δημοσιεύει το Harmonices Mundi στο οποίο διατυπώνει τον τρίτο νόμο.

Το 1687 ο Sir Isaac Newton (1643-1727) δημοσίευσε το έργο του Philosophiae Naturalis Principia Mathematica, στο οποίο διατυπώνει τους 3 νόμους, και βασιζόμενος στους νόμους του Kepler διατύπωσε το νόμο της παγκόσμιας έλξης, με τον οποίο ερμήνευσε την κίνηση των πλανητών. Η κίνησή τους περιγράφεται ως το αποτέλεσμα της βαρυτικής δύναμης η οποία είναι αντιστρόφως ανάλογη του τετραγώνου της απόστασης, και ανάλογη του γινομένου των μαζών τους.

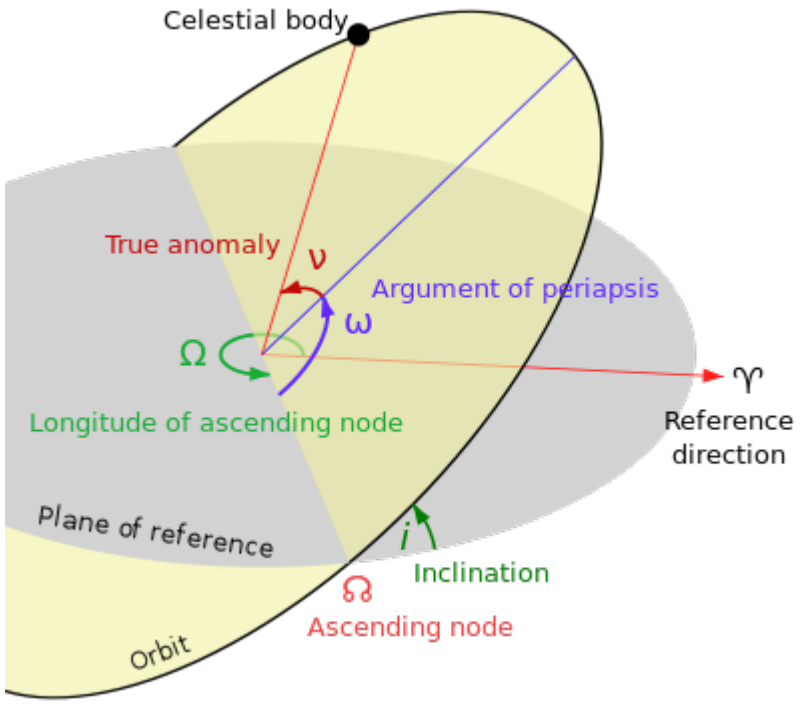
1.2 Εξωπλανητικά συστήματα.

Εξωηλιακός πλανήτης ή εξωπλανήτης ονομάζεται κάθε πλανήτης που δεν ανήκει στο δικό μας Ηλιακό Σύστημα. Ο πρώτος εξωπλανήτης βρέθηκε το 1988 και επιβεβαιώθηκε το 1992. Από τότε έχουν βρεθεί (μέχρι τον Ιούνιο του 2017) 3610 εξωπλανήτες σε 2704 εξωπλανητικά συστήματα. Πληροφορίες για αυτά τα συστήματα υπάρχουν στις αστρονομικές βάσεις δεδομένων: exoplanet.eu και openexoplanetcatalogue.com .

1.3 Τροχιές πλανητών.

Στην εργασία αυτή το σύστημα αναφοράς είναι το εκάστοτε αστέρι του συστήματος που αναπαριστάται.

Για να περιγραφεί μια συγκεκριμένη και μοναδική τροχιά χρειάζονται 6 χαρακτηριστικά που λέγονται Κεπλέρια στοιχεία.



Τα δύο κύρια στοιχεία που ορίζουν το σχήμα και το μέγεθος της τροχιάς.

1)Μεγάλος ημιάξονας (***semimajor axis*** $\boldsymbol{\alpha}$) : Το ήμισυ του μήκους του κύριου άξονα της έλλειψης.

2)Εκκεντρότητα (***eccentricity*** $\boldsymbol{e}$): προσδιορίζει τη μορφή της έλλειψης. Είναι ένα μέτρο που προσδιορίζει πόσο η κωνική τομή "απέχει" από το να είναι τέλειος κύκλος.

Τα επόμενα δύο στοιχεία καθορίζουν τον προσανατολισμό του τροχιακού επιπέδου.

3)Κλίση (***inclination*** $\boldsymbol{i}$) : Η γωνία μεταξύ του επιπέδου που ορίζει η τροχιά σε σχέση με το επίπεδο αναφοράς.

4)Μήκος αναβιβάζοντος συνδέσμου (***Longtitude of the ascending node*** $\boldsymbol{\Omega}$) : είναι η γωνία, που βρίσκεται στο επίπεδο αναφοράς, και προσδιορίζει το προσανα- τολισμό του αναβιβάζοντος συνδέσμου της έλλειψης ως προς τη κατεύθυνση αναφοράς.

5)Το όρισμα (γωνία) του περικέντρου (***Argument*** of ***pericenter*** $\boldsymbol{\omega}$) : είναι η γωνία, η οποία ορίζεται πάνω στο επίπεδο της τροχιάς και ορίζεται ανάμεσα στο περίκεντρο της τροχιάς και τον αναβιβάζοντα σύνδεσμο με φορά τη διεύθυνση της κίνησης.

6)Αληθής Ανωμαλία, (***True Anomaly*** $\boldsymbol{v}$) : Η πραγματική γωνία που υποδεικνύει την ακριβή θέση του σώματος στην τροχιά μετά από το τελευταίο πέρασμά του από το περίκεντρο.

Ένα ακόμα στοιχείο που χρειάζεται στην συγκεκριμένη εργασία είναι η Μέση Ανωμαλία (***Mean Anomaly*** $\boldsymbol{M}$) : είναι η γωνία κατά την οποία θα είχε κινηθεί το δευτερεύον σώμα από τη χρονική στιγμή της τελευταίας του διέλευσης από το περίκεντρο, υποθέτοντας ότι θα κινούνταν με σταθερή ταχύτητα σε μία κυκλική τροχιά με περίοδο ίση με αυτήν της πραγματικής ελλειπτικής του τροχιάς. Ορίζεται στις 0 μοίρες στο περίκεντρο και στις 180 μοίρες στο απόκεντρο. Η μέση ανωμαλία συνηθίζεται να χρησιμοποιείται ως το 6°Κεπλέριο στοιχείο αντί της αληθούς ανωμαλίας.

1.4 OpenGL.

Η Open Graphics Library (OpenGL) είναι η καθιερωμένη Διεπαφή Προγραμματισμού Εφαρμογών(Application programming interface, API) για την απόδοση 2D και 3D Γραφικών. Είναι ανεξάρτητη λειτουργικού συστήματος και συνήθως αλληλεπιδρά με την κάρτα γραφικών ώστε να πετύχει ταχύτερη απόδοση γραφικών. Χρησιμοποιείται εκτενώς στα εργαλεία σχεδίασης μέσω υπολογιστή, στα βιντεοπαιχνίδια, στην εικονική και επαυξημένη πραγματικότητα, σε εξομοιωτές πτήσης, στις επιστημονικές οπτικοποιήσεις και έχει έναν μεγάλο αριθμό ενσωματωμένων δυνατοτήτων. Η OpenGL είναι ορισμένη ως ένα σύνολο μεθόδων (ρουτίνων) και ένα σύνολο ακεραίων σταθερών που μπορούν να κληθούν ώστε να αποδοθούν τα γραφικά. Αν και οι ορισμοί των μεθόδων είναι παρόμοιοι με αυτούς της γλώσσας προγραμματισμού C, είναι ανεξάρτητοι γλώσσας, το οποίο επιτρέπει την χρήση των μεθόδων αυτών από διάφορες γλώσσες προγραμματισμού.

Η OpenGL αναπτύχθηκε το 1991 από την Silicon Graphics (SGI) και πλέον αναπτύσσεται από την μη-κερδοσκοπική κοινοπραξία Khronos Group.

Στο πρόγραμμα χρησιμοποιείται η GLUT (graphics library utility toolkit) που είναι ένα εργαλείο για την κατασκευή κώδικα OpenGL. Η GLUT κάνει πολύ πιο εύκολη την χρήση και την εξερεύνηση των OpenGL προγραμμάτων και παρέχει ενα φορητό API για την κατασκευή OpenGL προγραμμάτων που λειτουργούν σε ολά τα λειτουργικά συστήματα.

Ο κώδικας κατασκευάστηκε στο Visual Studio που είναι το περιβάλλον ανάπτυξης (IDE) της Microsoft.

Τα αρχεία που χρησιμοποιήθηκαν είναι: glut32.lib, glu32.lib, glut32.dll, glut.h .

2 Πρόβλημα των N-σωμάτων.

Με την ανακάλυψη του Νεύτωνα έγινε η πρώτη επιτυχημένη προσπάθεια να μελετηθεί η κίνηση δύο σωμάτων που αλληλεπιδρούν βαρυτικά (πρόβλημα 2 σωμάτων). Ο Bernoulli έδειξε ότι η τροχιά που μπορεί να διαγράψει ένα σώμα, μπορεί να είναι είτε ελλειπτική, είτε παραβολική είτε υπερβολική. Ο Euler έδειξε πως το πρόβλημά μπορεί να αναχθεί σε πρόβλημα της κίνησης ενός σώματος, με μάζα την ανηγμένη μάζα του συστήματος $\mu = \dfrac{m_1 m_2}{m_1 + m_2}$.

Μετά από τις παραπάνω ανακαλύψεις, ξεκίνησε η προσπάθεια για τη μελέτη πιο πολύπλοκων συστημάτων, δηλαδή των 3 σωμάτων και άνω. Ο Newton ασχολήθηκε με το lunar problem (Ηλιος-Γη-Σελήνη) στο Principia χωρίς όμως κάποιο αποτέλεσμα. Ο J. D' Alembert εισήγαγε τον όρο το πρόβλημα των 3 σωμάτων και προσπάθησε να λύσει το πρόβλημα χρησιμοποιώντας διαφορικές εξισώσεις. Στη συνέχεια ο Euler το 1772 εισήγαγε την ειδική περίπτωση του περιορισμένου προβλήματος των 3 σωμάτων (restricted 3 body problem), ενώ ο Joseph-Louis Lagrange κατάφερε να προσδιορίσει τα σημεία ισορροπίας στο συγκεκριμένο πρόβλημα.

Αρκετοί ακόμη μεγάλοι επιστήμονες ασχολήθηκαν με το πρόβλημα των 3 σωμά- των, όπως ο Carl Gustav Jacob Jacobi, ο George William Hill, ο Charles-Eugène Delaunay συνεισφέροντας αρκετά στην μελέτη του προβλήματος χωρίς όμως να βρεθεί η γενική του λύση. Το 1887 ο βασιλιάς της Σουηδίας, με αφορμή τα εξηκοστά του γενέθλια, διοργάνωσε ένα διαγωνισμό απονέμοντας το βραβείο σε όποιον μπορούσε να δώσει τη λύση στο πρόβλημα των N-σωμάτων. Ο Γάλλος Jules Henri Poincaré κέρδισε το βραβείο αποδεικνύοντας ότι το πρόβλημα των N-σωμάτων για $N > 2$ δεν έχει αναλυτική λύση, και ήταν ο πρώτος που εξέφρασε τη βασική αρχή της θεωρίας του χάους, ότι δηλαδή "μικρές διαταραχές στις αρχικές συνθήκες προκαλούν μεγάλες διαφορές στο τελικό αποτέλεσμα".

Το πρόβλημα των N-σωμάτων διατυπώθηκε πρώτα απο τον Νεύτωνα.

Θα κατασκευάσουμε τις εξισώσεις κίνησης N-σωμάτων με μάζες $m_i (i = 1, 2, ...N)$ όπου τα διανύσματα θέσης R_i ορίζονται από ένα σταθερό σημείο O, και η μεταξύ τους διανυσματική απόσταση είναι r_{ij} όπου

$$\boldsymbol{r}_{ij} = \boldsymbol{R}_j - \boldsymbol{R}_i \qquad (2.1)$$

Σημειώνεται ότι το r_{ij}σημαίνει ότι το διάνυσμα ανάμεσα στο m_i και στο m_j έχει κατεύθυνση από το m_i στο m_j . Οπότε:

$$\boldsymbol{r}_{ij} = -\boldsymbol{r}_{ji} \qquad (2.2)$$

Από τους νόμους του Νεύτωνα και το νόμο της παγκόσμιας έλξης, έχουμε:

$$m_i \ddot{\boldsymbol{R}}_i = G \sum_{j=1}^{N} \frac{m_i m_j}{r_{ij}^3} \boldsymbol{r}_{ij}, \quad (j \neq i, i = 1, 2, ...N) \qquad (2.3)$$

2.1 Τα 10 ολοκληρώματα της κίνησης.

Από την (2.3) αν πάρουμε το άθροισμα για όλα τα σώματα βρίσκουμε:

$$\sum_{i=1}^{N} m_i \ddot{\boldsymbol{R}}_i = 0$$

γιατί παρουσιάζονται όροι r_{ij} και r_{ji} που είναι αντίθετοι.
ολοκληρώνοντας μία φορά βρίσκουμε

$$\sum_{i=1}^{N} m_i \dot{\boldsymbol{R}}_i = \boldsymbol{a} \qquad (2.4)$$

και ολοκληρώνοντας άλλη μία

$$\sum_{i=1}^{N} m_i \boldsymbol{R}_i = \boldsymbol{a}t + \boldsymbol{b} \qquad (2.5)$$

όπου $\boldsymbol{a}, \boldsymbol{b}$ σταθερές.

Από τον ορισμό του κέντρου μάζας ενός συστήματος από σημειακές μάζες, και ορίζοντας ως διάνυσμα θέσης του κέντρου μάζας R_{cm}:

$$M\boldsymbol{R}_{cm} = \sum_{i=1}^{N} m_i \boldsymbol{R}_i,$$

όπου

$$M = \sum_{i=1}^{N} m_i,$$

βρίσκουμε

$$\boldsymbol{R}_{cm} = \frac{\boldsymbol{a}t + \boldsymbol{b}}{M} \qquad (2.6)$$

και

$$\dot{\boldsymbol{R}}_{cm} = \frac{\boldsymbol{a}}{M} \qquad (2.7)$$

Οι εξισώσεις (2.6) και (2.7) δείχνουν οτι το κέντρο μάζας του συστήματος κινείται στο χώρο με σταθερή ταχύτητα. Αν ξαναγραφτούν οι εξισώσεις (2.6) και (2.7) ως προς ένα σταθερό τρισορθογώνιο σύστημα αναφοράς με κέντρο το O, τότε τα έξι ολοκληρώματα είναι a_x, a_y, a_z , $b_x, b_y b_z$.

Παίρνοντας το εξωτερικό γινόμενο των $\boldsymbol{R}_i$ και $\ddot{\boldsymbol{R}}_\iota$ για κάθε μια από τις εξισώσεις της (2.3) και αθροίζοντας κατά μέλη, παρατηρούμε ότι

$$\sum_{i=1}^{N} m_i \boldsymbol{R}_i \times \ddot{\boldsymbol{R}}_i = G \sum_{i=1}^{N} \sum_{j=1}^{N} \frac{m_i m_j}{r_{ij}^3} \boldsymbol{R}_i \times \boldsymbol{r}_{ij}, \quad (j \neq i) \qquad (2.8)$$

Αλλά

$$\boldsymbol{R}_i \times \boldsymbol{r}_{ij} = \boldsymbol{R}_i \times (\boldsymbol{R}_j - \boldsymbol{R}_i) = \boldsymbol{R}_i \times \boldsymbol{R}_j$$

και

$$\boldsymbol{R}_j \times \boldsymbol{r}_{ij} = \boldsymbol{R}_j \times \boldsymbol{R}_i = -\boldsymbol{R}_i \times \boldsymbol{R}_j$$

Οπότε στο δεξί μέλος της (2.8) οι όροι του αθροίσματος είναι ανά ζεύγη αντίθετοι, οπότε

$$\sum_{i=1}^{N} m_i \boldsymbol{R}_i \times \ddot{\boldsymbol{R}}_\iota = 0$$

και ολοκληρώνοντας

$$\sum_{i=1}^{N} m_i \boldsymbol{R}_i \times \dot{\boldsymbol{R}}_i = C \qquad (2.9)$$

Η σχέση (2.9) δηλώνει ότι η συνολική ποσότητα της στροφορμής του συστήματος παραμένει σταθερή. Παρομοίως αν ξαναγραφτεί η (2.9) ως προς ένα αδρανειακό τρισορθογώνιο σύστημα συντεταγμένων, τότε

$$\sum_{i=1}^{N} m_i (x_i \dot{y}_i - y_i \dot{x}_i) = C_1$$

$$\sum_{i=1}^{N} m_i (y_i \dot{z}_i - z_i \dot{y}_i) = C_2$$

$$\sum_{i=1}^{N} m_i (z_i \dot{x}_i - x_i \dot{z}_i) = C_3$$

όπου

$$C^2 = C_1^2 + C_2^2 + C_3^2$$

Άρα η συνολική στροφορμή C^2και οι C_1, C_2, C_3 παραμένουν σταθερές, δίνοντας τρία ακόμη ανεξάρτητα ολοκληρώματα της κίνησης.

Το 10°ολοκλήρωμα προκύπτει από την (2.3) πολλαπλασιασμένη με το διάνυσμα της ταχύτητας $\dot{\boldsymbol{R}}_i$.

$$\sum_{i=1}^{N} m_i \dot{\boldsymbol{R}}_i \cdot \ddot{\boldsymbol{R}}_i = G \sum_{i=1}^{N} \sum_{j=1}^{N} \frac{m_i m_j}{r_{ij}^3} \dot{\boldsymbol{R}}_i \cdot \boldsymbol{r}_{ij} \quad (j \neq i) \qquad (2.10)$$

$$\dot{\boldsymbol{R}}_i \cdot \boldsymbol{r}_{ij} = \dot{\boldsymbol{R}}_i (\boldsymbol{R}_j - \boldsymbol{R}_i) \qquad (2.11)$$

και

$$\dot{\boldsymbol{R}}_j \cdot \boldsymbol{r}_{ji} = \dot{\boldsymbol{R}}_j (\boldsymbol{R}_i - \boldsymbol{R}_j) \qquad (2.12)$$

Αθροίζοντας τις (2.11) , (2.12)

$$\dot{\boldsymbol{R}}_i \cdot \boldsymbol{r}_{ij} + \dot{\boldsymbol{R}}_j \cdot \boldsymbol{r}_{ji} = -(\dot{\boldsymbol{R}}_j - \dot{\boldsymbol{R}}_i) \cdot (\boldsymbol{R}_j - \boldsymbol{R}_i)$$

Χρησιμοποιώντας την (2.1) , η (2.10) απο ολοκλήρωση, δίνει

$$\frac{1}{2}\sum_{i=1}^{N} m_i \dot{\boldsymbol{R}}_i \dot{\boldsymbol{R}}_i - \frac{1}{2} G \sum_{i=1}^{N}\sum_{j=1}^{N} \frac{m_i m_j}{r_{ij}} = E \quad (j \neq i) \qquad (2.13)$$

η ταχύτητα της i-οστής μάζας θα είναι

$$V_i^2 = \dot{\boldsymbol{R}}_i \dot{\boldsymbol{R}}_i$$

επίσης

$$U = \frac{1}{2} G \sum_{i=1}^{N}\sum_{j=1}^{N} \frac{m_i m_j}{r_{ij}}$$

και η (2.13) γίνεται

$$T - U = E \qquad (2.13)$$

όπου

$$T = \frac{1}{2}\sum_{i=1}^{N} m_i V_i^2$$

Ο πρώτος όρος στη (2.13) είναι η κινητική ενέργεια και το -U είναι η δυναμική ενέργεια. Η (2.13) λοιπόν δηλώνει ότι η συνολική ενέργεια του συστήματος των N-σωμάτων είναι σταθερή, που είναι το 10°ολοκλήρωμα της κίνησης. [A.E Roy]

2.2 Οι εξισώσεις της κίνησης.

$$U = \frac{1}{2} G \sum_{i=1}^{N} \sum_{j=1}^{N} \frac{m_i m_j}{r_{ij}}$$

Αν i, j, k τα μοναδιαία διανύσματα στους άξονες O_x, O_y, O_z τότε

$$\nabla U \equiv gradU = \boldsymbol{i} \frac{\partial U}{\partial x} + \boldsymbol{j} \frac{\partial U}{\partial y} + \boldsymbol{k} \frac{\partial U}{\partial z}$$

επίσης

$$\ddot{\boldsymbol{R}}_i = \boldsymbol{i} \ddot{x}_i + \boldsymbol{j} \ddot{y}_i + \boldsymbol{k} \ddot{z}_i$$

άρα

$$m_i \ddot{\boldsymbol{R}}_i = grad_i U \qquad (2.14)$$

$$grad_i U = \boldsymbol{i} \frac{\partial U}{\partial x_i} + \boldsymbol{j} \frac{\partial U}{\partial y_j} \boldsymbol{k} \frac{\partial U}{\partial z_k}$$

και οι εξισώσεις της κίνησης θα είναι

$$m_i \ddot{x}_i = \frac{\partial U}{\partial x_i}$$

$$m_i \ddot{y}_i = \frac{\partial U}{\partial y_i} \qquad (2.15)$$

$$m_i \ddot{z}_i = \frac{\partial U}{\partial z_i}$$

2.3 Αριθμητική ολοκλήρωση με τη συμπλεκτική μέθοδο.

Για να μπορέσουμε να προσομοιώσουμε το πρόβλημα των N-σωμάτων, θα πρέπει να ολοκληρώσουμε τις διαφορικές εξισώσεις με μία αριθμητική μέθοδο

Οι συμπλεκτικοί ολοκληρωτές έχουν σχεδιαστεί για να λύνουν αριθμιτικά τις εξισώσεις του Hamilton

$$\dot{\boldsymbol{p}} = -\frac{\partial H}{\partial \boldsymbol{q}}, \quad \dot{\boldsymbol{q}} = \frac{\partial H}{\partial \boldsymbol{p}} \qquad (2.16)$$

όπου το $\boldsymbol{q}$ συμβολίζει τις συντεταγμένες θέσης και το $\boldsymbol{p}$ της γενικευμένης ορμής.

Έστω η Χαμιλτονιανή συνάρτηση

$$H(\boldsymbol{q},\boldsymbol{p}) = T(\boldsymbol{p}) + V(\boldsymbol{q}) \qquad (2.17)$$

όπου Τ είναι η κινητική ενέργεια και V η δυναμική.

Για να μελετήσουμε ενα Χαμιλτονιανό σύστημα θα πρέπει να λάβουμε υπόψιν δύο βασικά κριτήρια τα οποία δεν πρέπει να παραβιάζονται, η συμπλεκτική ιδιότητα και η διατήρηση της ενέργειας. Η συμπλεκτική ιδιότητα αναφέρεται στην διατήρηση του εμβαδού στο χώρο των φάσεων (θεώρημα Liouville). Διανυσματικά εκφράζεται ως:

$$d\boldsymbol{p} \times d\boldsymbol{q} = d\boldsymbol{p}' \times d\boldsymbol{q}'$$

ενώ αλγεβρικά ως:

$$\begin{pmatrix} \boldsymbol{q}' \\ \boldsymbol{p}' \end{pmatrix} = A \begin{pmatrix} \boldsymbol{q} \\ \boldsymbol{p} \end{pmatrix}$$

όπου Α πίνακας με $det(A) = 1$

Η διατήρηση της ενέργειας εκφράζεται ως

$$H(\boldsymbol{q},\boldsymbol{p}) = H(\boldsymbol{q}',\boldsymbol{p}') = constant$$

Αν προσπαθήσουμε να λύσουμε τις εξισώσεις (2.16) με κάποια αριθμητική μέθοδο (π.χ. Runge-Kutta) θα δούμε ότι με τη πάροδο του χρόνου τα σφάλματα συσσωρεύονται και η ενέργεια δεν διατηρείται. Επίσης μπορεί κάποια μέθοδος να κρατά την ενέργεια σταθερή αλλά να παραβιάζεται η συμπλεκτική συνθήκη, έτσι ώστε να μας δίνει μια τροχιά η οποία δεν ανταποκρίνεται στη πραγματικότητα.

Κατά τους Ge και Marsden (1988) δεν υπάρχει κάποια μέθοδος σε μη ολοκληρώσιμα χαμιλτονιανά συστήματα, η οποία να τηρεί επακριβώς τα δύο παραπάνω κριτήρια. Υπάρχει όμως η κατηγορία των συμπλετικών ολοκληρωτών, οι οποίοι εκ κατασκευής τους απαιτούν να τηρείται στο ακέραιο η συμπλεκτική συνθήκη και η ενέργεια να μεταβάλλεται φραγμένα με ένα σφάλμα ως προς την αρχική ενέργεια. Το σφάλμα αυτό προσδιορίζει την τάξη του συμπλεκτικού ολοκληρωτή ως προς το βήμα.

Έχουν αναπτυχθεί δύο βασικές μέθοδοι συμπλεκτικών ολοκληρωτών, η έμμεση (Implicit) και η άμεση (Explicit). Η έμμεση μέθοδος στηρίζεται στη λογική της εύρεσης μιας άλλης Χαμιλτονιανής με τη βοήθεια μιας κατάλληλα επιλεγμένης γενέτειρας συνάρτησης και με τη χρήση του κανονικού μετασχηματισμού.[Ο.Κότσας]

Στο πρόγραμμα χρησιμοποιείται η άμεση μέθοδος η οποία στηρίζεται στην κατάλληλη σύνθεση συμπλεκτικών απεικονίσεων των "χαμιλτονιανών" $T(p), V(q)$. Η (2.17) αποτελείται από δύο ολοκληρώσιμα τμήματα. Έτσι, αν αγνοήσουμε την $\mathrm{H} = T(p)$, η λύση θα ήταν απλά

$$\dot{p} = 0 \rightarrow p' = p,$$

$$\int \dot{q}dt = \int (\frac{\partial H}{\partial p})_{p'=p} dt \rightarrow q' = q + \tau \frac{\partial T}{\partial p}$$

Η λύση είναι ακριβής και βέβαια η απεικόνιση, $S_T(\tau)$, είναι συμπλεκτική. Αντίστοιχα, αν $\mathrm{H} = V(q)$, η απεικόνιση $S_V(\tau)$ θα έδινε επίσης την ακριβή και συμπλεκτική λύση

$$\dot{q} = 0 \rightarrow q' = q,$$

$$\int \dot{p}dt = -\int (\frac{\partial H}{\partial q})_{q'=q} dt \rightarrow p' = p - \tau \frac{\partial V}{\partial q}$$

Το ερώτημα είναι λοιπόν το εξής: αν συνθέταμε αυτές τις δύο απεικονίσεις (η σύνθεση θα δώσει κατ' ανάγκην επίσης συμπλεκτική απεικόνιση) – δηλαδή, λύναμε πρώτα την 1η εξίσωση και μετά τη 2η (ή ανάποδα) αλλά χρησιμοποιώντας στη 2η τις τιμές που προκύπτουν από την επίλυση της 1ης – θα παίρναμε την "πραγματική" λύση; Η απάντηση είναι δυστυχώς όχι, αλλά η σύνθεση των δύο στοιχειωδών απεικονίσεων θα μας έδινε μια λύση "κοντά" στην πραγματική, με ακρίβεια $O_{(\tau)}$. Ο λόγος για τον οποίο η σύνθεση

$$S_T(\tau) \circ S_V(\tau) \neq S_H(\tau)$$

δε δίνει την $S_H(\tau)$ μπορεί να γίνει κατανοητός με βάση τη "γλώσσα" της άλγεβρας Lie.

2.4 Μετατροπή τροχιακών στοιχείων σε καρτεσιανό σύστημα συντεταγμένων.

Για την εύρεση της θέσης και της ταχύτητας από τα τροχιακά στοιχεία, δλδ

$$a, e, i, \omega, \Omega, M \rightarrow X, Y, Z, \dot{X}, \dot{Y}, \dot{Z}$$

ακολουθείται η εξής διαδικασία:

Υπολογίζεται από την εξίσωση του Kepler η έκκεντρη ανωμαλία, EA, $(0^\circ \to 360^\circ)$

$$M = EA - e \cdot sinEA$$

η αληθής ανωμαλία, v, $(0^\circ \to 360^\circ)$

$$tan\frac{v}{2} = [\frac{(1+e)}{(1-e)}]^{\frac{1}{2}} \cdot tan\frac{EA}{2}$$

η ακτίνα, r,

$$r = a(1 - e \cdot cosEA) = \frac{a(1-e^2)}{1 + e \cdot cosv}$$

η στροφορμή, h,

$$h = [\mu a(1-e^2)]^{\frac{1}{2}}$$

και βρίσκεται η θεση

$$X = r[cos\Omega cos(\omega + v) - sin\Omega sin(\omega + v)cosi]$$

$$Y = r[sin\Omega cos(\omega + v) - cos\Omega sin(\omega + v)cosi]$$

$$Z = r[sini \cdot sin(\omega + v)]$$

και η ταχύτητα

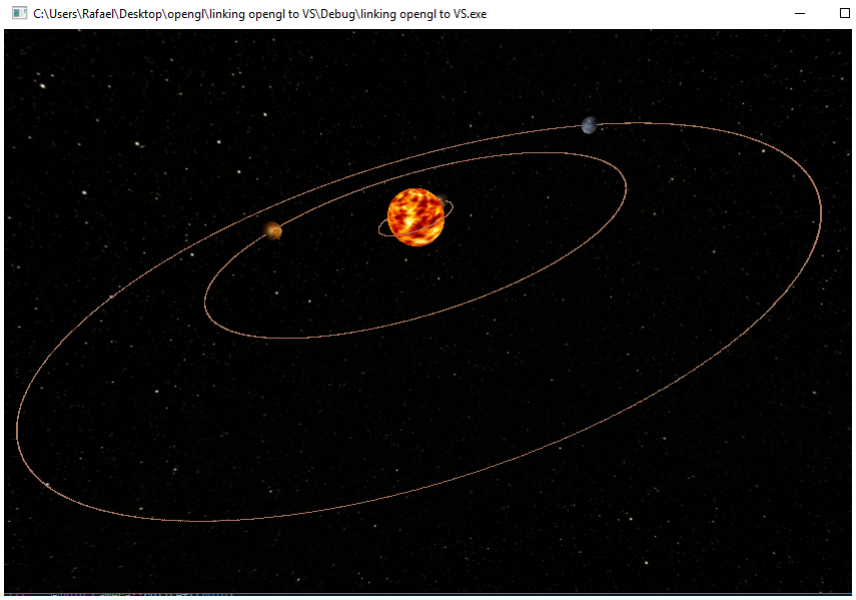
$$\dot{X} = \frac{Xhe}{ra(1-e^2)}sinv - \frac{h}{r}(cos\Omega sin(\omega + v) + sin\Omega cos(\omega + v)cosi)$$

$$\dot{Y} = \frac{Yhe}{ra(1-e^2)}sinv - \frac{h}{r}(sin\Omega sin(\omega + v) - cos\Omega cos(\omega + v)cosi)$$

$$\dot{Z} = \frac{Xhe}{ra(1-e^2)}sinv + \frac{h}{r}sinicos(\omega + v)$$

[CCAR]

3 Το πρόγραμμα.



3.1 Αρχικές συνθήκες.

Όλες οι πληροφορίες που χρειάζονται για να γίνει η αναπαράσταση είναι σε αρχεία txt με τη μορφή:

HD106315.txt - Notepad

File Edit Format View Help

```
//NBS project  GV March 2007
//configuration parameters + initial conditions
//-----------------------------------------------

Number of Bodies (without SUN) = 2
Internal Integration step dt = 0.0001
Main DT Step iterations N=100


Masses and Initial Conditions
Mass of Star   M = 1.0
-------------------------------------------------------------
Mass       a(AU)    ecc        incl      M     om     OM     r=
0.000025   0.900   0.3000     0.000   0.00    0.0    0.0     0.05
0.000063   1.500   0.2300     1.000   0.00    0.0    0.0     0.09


Period of 1st Planet in the list (Physical units) TP1=1
Number of DT1 iterations  NN=20000
```

Στο παράδειγμα αυτό υπάρχει ο ήλιος και δύο ακόμα σώματα. Η μόνη πληροφορία που χρειάζεται για τον ήλιο είναι η μάζα του (Mass of Star M = 1.0).

Το βήμα που χρησιμοποιείται: Internal Integration step dt καθορίζει πόσο "μακριά" θα είναι το επόμενο στιγμιότυπο. Ο Συμπλεκτικός Ολοκληρωτής ενεργεί και από το $\vec{x}(t_o)$ μας δίνει το $\vec{x}(t_0 + \Delta T)$ όπου $\Delta T = dt \cdot N$.

Τα στοιχεία για τον κάθε πλανήτη είναι με τη σειρά: "Mass" η μάζα, "a" ο μεγάλος ημιάξονας, "ecc" η εκκεντρότητα, "incl" η κλίση, "M" η μέση ανωμαλία, "om" το όρισμα του περικέντρου, "OM" το μήκος αναβιβάζοντος συνδέσμου, "r" η ακτίνα (της οποίας η τιμή δεν είναι απαραίτητα σε κλίμακα).

3.2 Περί του προγράμματος.

Τα πλανητικά συστήματα που αναπαριστώνται στην εργασία αυτή είναι:

α) το αστέρι HD 106315 το οποίο περιβάλλουν δύο πλανήτες και βρίσκεται σε απόσταση 107pc περίπου.

HD106315.txt - Notepad

File Edit Format View Help

```
//NBS project  GV March 2007
//configuration parameters + initial conditions
//-----------------------------------------------

Number of Bodies (without SUN) = 2
Internal Integration step dt = 0.0001
Main DT Step iterations N=100


Masses and Initial Conditions
Mass of Star   M = 1.0
-----------------------------------------------------------------
Mass       a(AU)    ecc        incl      M     om     OM
0.000025   0.900   0.3000     0.000   0.00    0.0    0.0
0.000063   1.500   0.2300     1.000   0.00    0.0    0.0


Period of 1st Planet in the list (Physical units) TP1=1
Number of DT1 iterations  NN=20000
```

β) το αστέρι Kepler-9 το οποίο περιβάλλουν τρεις πλανήτες και βρίσκεται σε απόσταση 650pc περίπου.

kepler-9.txt - Notepad

File Edit Format View Help

```
//NBS project  GV March 2007
//configuration parameters + initial conditions
//-----------------------------------------------

Number of Bodies (without SUN) = 3
Internal Integration step dt = 0.0001
Main DT Step iterations N=100


Masses and Initial Conditions
Mass of Star   M = 1.0
-----------------------------------------------------------------
Mass        a(AU)     ecc        incl      M     om     OM
0.000141    1.43    0.0626      0.00     0.00    0.0    0.0
0.0000975   2.29    0.0684      0.00     0.00    0.0    0.0
0.00091651  0.273    0.05       3.00     0.00    0.0    0.0


Period of 1st Planet in the list (Physical units) TP1=1
Number of DT1 iterations  NN=20000
```

γ) το αστέρι Gliese 876 το οποίο περιβάλλουν τέσσερις πλανήτες και βρίσκεται σε απόσταση 4.69pc περίπου.

gliese876.txt - Notepad

File Edit Format View Help

```
//NBS project  GV March 2007
//configuration parameters + initial conditions
//-----------------------------------------------

Number of Bodies (without SUN) = 4
Internal Integration step dt = 0.0001
Main DT Step iterations N=100


Masses and Initial Conditions
Mass of Star   M = 3.70
-------------------------------------------------------------
Mass        a(AU)     ecc        incl      M     om      OM
0.026700    2.186    0.0340      0.00     0.00    0.0    0.0
0.008430    1.360    0.2540      0.00     0.00    0.0    0.0
0.000220    0.218    0.1100      0.00     0.00    0.0    0.0
0.000540    3.340    0.0300      0.00     0.00    0.0    0.0


Period of 1st Planet in the list (Physical units) TP1=1
Number of DT1 iterations  NN=20000
```

δ) το αστέρι το δικό μας, το οποίο περιβάλλουν οκτώ πλανήτες.

Sun.txt - Notepad

File Edit Format View Help

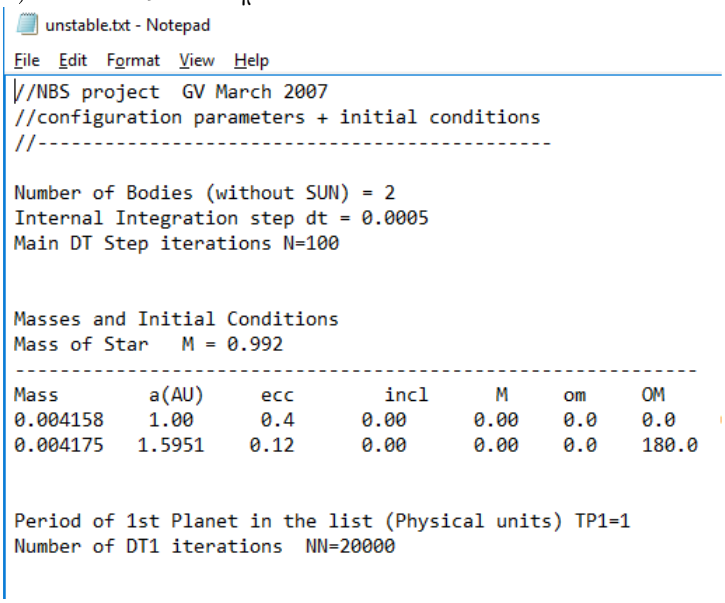
```
//NBS project  GV March 2007
//configuration parameters + initial conditions
//-----------------------------------------------

Number of Bodies (without SUN) = 8
Internal Integration step dt = 0.00001
Main DT Step iterations N=100


Masses and Initial Conditions
Mass of Star   M = 1000.0
-------------------------------------------------------------
Mass        a(AU)     ecc        incl      M     om      OM
0.0026      0.723    0.0068      0.00     0.00    0.0    0.0
1.00000     5.20     0.0485      0.00     0.00    0.0    0.0
0.000338    1.524    0.0934      0.00     0.00    0.0    0.0
0.299000    9.54     0.0555      0.00     0.00    0.0    0.0
0.000174    0.387    0.2060      0.00     0.00    0.0    0.0
0.045700    19.19    0.0469      0.00     0.00    0.0    0.0
0.054000    30.1     0.0090      0.00     0.00    0.0    0.0
0.003100    1.000    0.0167      0.00     0.00    0.0    0.0

Period of 1st Planet in the list (Physical units) TP1=1
Number of DT1 iterations  NN=20000
```

ε) Ασταθές σύστημα.


unstable.txt - Notepad

File Edit Format View Help

```
//NBS project  GV March 2007
//configuration parameters + initial conditions
//-----------------------------------------------

Number of Bodies (without SUN) = 2
Internal Integration step dt = 0.0005
Main DT Step iterations N=100


Masses and Initial Conditions
Mass of Star   M = 0.992
------------------------------------------------------------
Mass        a(AU)     ecc        incl      M     om     OM
0.004158    1.00      0.4      0.00      0.00    0.0    0.0
0.004175   1.5951    0.12      0.00      0.00    0.0    180.0


Period of 1st Planet in the list (Physical units) TP1=1
Number of DT1 iterations  NN=20000
```

στ) Σύστημα με το οποίο μπορεί να πειραματιστεί ο χρήστης αλλάζοντας τις αρχικές συνθήκες.

Τα στοιχεία που χρειάζονται και χρησιμοποιούνται από το πρόγραμμα από αυτά τα συστήματα για να κάνουμε σωστή αναπαράσταση της συμπεριφοράς τους υπάρχουν σε txt αρχεία μαζί με τον κώδικα.

3.3 Η βασική συνάρτηση.

```
int main(int argc, char** argv)
{
        file_input();

        X0 = new double[NEQ];

        GetSystemPosV(oeb, X0);

        NcurData = 0;

        X = new double[NEQ];
        Y = new double[NEQ];

        for (int i = 0; i < NEQ; i++)
        {
                X[i] = X0[i];
        }
        double t = X[0] * t_metric;


        glutInit(&argc, argv);
        glutInitDisplayMode(GLUT_DOUBLE
        |GLUT_RGB |GLUT_DEPTH);
        glutInitWindowSize(1200, 700);
        glutInitWindowPosition(100, 100);
        glutCreateWindow(argv[0]);
    init();
        glutDisplayFunc(display);
        glutReshapeFunc(reshape);
        glutKeyboardFunc(keyDown);
    glutKeyboardUpFunc(keyUp);
        SetupMenu();

        glutMainLoop();
        return 0;
}
```

Η συνάρτηση file_input() δέχεται το όνομα ενός αρχείου που περιέχει τις πληροφορίες για τα πλανητικά συστήματα που μελετώνται και τις διαβάζει. Η GetSystemPosV(oeb, X0) παίρνει τις πληροφορίες για τα τροχιακά στοιχεία για να δώσει τις αρχικές θέσεις (και ταχύτητες) των πλανητών. Όπου η oeb είναι τύπου OEDEF και ορίζεται ως:

```
typedef struct OrbitalElementsDefined
{
        double a;
        double e;
        double i;
        double elpos;
        double omega;
        double Omega;
        int inputangle;
        int modedisplay;
} OEDEF;
```

Έπειτα ορίζονται 2 πίνακες X,Y με αριθμό στοιχείων NEQ (number of equations). Η πρώτη θέση είναι για το χρόνο, μετά είναι σε τριάδες οι συντεταγμένες του κάθε πλανήτη και μετά σε τριάδες οι ταχύτητες του κάθε πλανήτη (στους άξονες x,y,z αντίστοιχα). Δίνονται οι πρώτες τιμές αυτών στον πίνακα X όπως φαίνεται από εδώ:

```
for (int i = 0; i < NEQ; i++)
        {
                X[i] = X0[i];
        }
        double t = X[0] * t_metric;
```

Εδώ φτιάχνονται το παράθυρο οι διαστάσεις, ο φωτισμός και ο timer για να μπορούν να καλούνται οι επόμενες συναρτήσεις κάθε κάποιο χρονικό διάστημα (συγκεκριμένα μερικά ms)

```
glutInit(&argc, argv);
glutInitDisplayMode(GLUT_DOUBLE
|GLUT_RGB |GLUT_DEPTH);
glutInitWindowSize(1200, 700);
glutInitWindowPosition(100, 100);
glutCreateWindow(argv[0]);
init();
```

Στη συνάρτηση display ξεκινάει η λειτουργία της κάμερας η οποία στην αρχή κοιτάει στο (0,0,0) όπου βρίσκεται το αστέρι του συστήματος. Σχεδιάζεται το skybox το οποίο είναι μια μέθοδος που χρησιμοποιείται για να δοθεί η αίσθηση "βάθους" και αποτελείται απο έναν κύβο στον οποίο εφαρμόζονται εικόνες (textures) που θέλουμε να φαίνονται και συγκεκριμένα στο πρόγραμμα αυτό εικόνες με μακρινά αστέρια για να προσομοιωθεί το διάστημα.

Οι πλανήτες φτιάχνονται από σφαίρες που ορίζονται με τις εντολές:

```
GLUquadricObj* quadric=gluNewQuadric();
gluQuadricTexture(quadric, true);
gluQuadricNormals(quadric, GLU_SMOOTH);
```

και τοποθετούνται ως:

```
glBindTexture(GL_TEXTURE_2D,
(planet)->getTextureHandle());
glPushMatrix();
glTranslatef(x, y, z);
glRotatef(angle, x, y, z);
gluSphere(quadric, radius, slices, stacks);
glPopMatrix();
```

Τις συντεταγμένες (x,y,z) τις παίρνουν από τη συνάρτηση:

```
void Symplectic_translate()
{
        Symplectic6thRUN(X, Nsteps, dt, Y);
        t = Y[0];
        for (int i = 0; i < NEQ; i++)
        {
                Data[NcurData][i] = X[i];
                X[i] = Y[i];
        }
        NcurData++;

        return;
}
```

η οποία χρησιμοποιεί τον συμπλεκτικό ολοκληρωτή 6ης τάξης για να βρει τις συντεταγμένες και τις ταχύτητες του επόμενου βήματος.

Συγκεκριμένα η Symplectic6thRUN:

```
void Symplectic6thRUN(double yin[],
int n, double dt, double yout[])
{
        double *y = TM1;
        for (int i = 0; i < neq; i++) y[i] = yin[i];
        double t0 = y[0];
        for (int i = 0; i < n; i++)
        {
                Symplectic6thSTEP(y, dt);
        }
        for (int i = 0; i<neq; i++) yout[i] = y[i];
        return;
}
```

Παίρνει δηλαδή τον πίνακα X που περιέχει όλες τις πληροφορίες (χρόνος, θέσεις, ταχύτητες) και χρησιμοποιώντας το βήμα που του έχει δωθεί δίνει τις επόμενες θέσεις και ταχύτητες (πίνακας Υ). Αυτές μετά γράφονται πάνω στον X και την επόμενη φορά που θα καλεστεί αυτή συνάρτηση χρησιμοποιεί αυτές για να δώσει τις επόμενες. Η μεταβλητή NcuData η οποία στην αρχή έχει την τιμή 0 είναι ο αριθμός των στιγμιοτύπων για αυτό και ανεβαίνει κατά μονάδα κάθε φορά που τρέχει η Symplectic_translate().

Οι τροχιές σχεδιάζονται ως:

```
glDisable(GL_LIGHTING);
int o = 0;
for (int i = 0; i < NB; i++)
        glBegin(GL_POINTS);
for (int j = NcurData - 1; j > 0; j--)
{
glVertex3f(Data[j][o+1],Data[j][o+2],Data[j][o+3]);
}
o += 3;
glEnd();
}
glEnable(GL_LIGHTING);
```

Όπου σε κάθε καρέ ζωγραφίζονται τελείες για κάθε προηγούμενη θέση των πλανητών. Ο Data[a][b] είναι ένας δισδιάστατος πίνακας στον οποίο το "a" συμβολίζει τον αριθμό των στιγμιοτύπων και το "b" περιέχει τις πληροφορίες που χρειάζονται για κάθε στιγμιότυπο (NEQ).

Οι επόμενες εντολές χρησιμοποιούνται για την επανασχεδίαση του παραθύρου όταν αλλάζει μέγεθος και την ενεργοποίηση του πληκτρολογίου και ποντικιού.

```
glutReshapeFunc(reshape);
glutKeyboardFunc(keyDown);
glutKeyboardUpFunc(keyUp);
```

Η SetupMenu() δείχνει τις διαφορετικές επιλογές πλανητικών συστημάτων. Αφού επιλεχθεί διαφορετικό πλανητικό σύστημα διαβάζεται το καινούριο αρχείο μηδενίζεται η NcurData και ορίζονται οι πίνακες X,Y από την αρχή.

3.4 Αλληλεπίδραση με το πρόγραμμα.

Ταυτόχρονα με την κίνηση των πλανητών διαγράφονται και οι τροχιές τους οι οποίες απενεργοποιούνται με το κουμπί "o" .

Η κάμερα στο πρόγραμμα αυτό λειτουργεί ως εξής:

Με τα κουμπιά "w", "s" μετακινείται μπροστά και πίσω αντίστοιχα.

Με τα "a", "d" μετακινείται αριστερά και δεξιά.

Με τα "q", "e" στρέφεται αριστερά και δεξιά.

Με τα "j", "l" στρέφεται αριστερά και δεξιά κοιτώντας το ίδιο επίπεδο.

Με τα "k", "i" στρέφεται προς τα πάνω και κάτω αντίστοιχα.

Με τα "," , "." μικραίνει και μεγαλώνει αντίστοιχα η ταχύτητα με την οποία μετακινείται η κάμερα.

Το μενού ενεργοποιείται με δεξί κλικ και εμφανίζει τα υπάρχοντα συστήματα.

Στο πρόγραμμα υπάρχει και ένα αρχείο με το όνομα user_system για να μπορεί ο χρήστης εύκολα να κάνει αλλαγές.

Για να προστεθεί ή να αλλάξει κάποιο σύστημα πρέπει να δημιουργηθεί ή να τροποποιηθεί ένα αρχείο της μορφής:

HD106315.txt - Notepad

File Edit Format View Help

```
//NBS project  GV March 2007
//configuration parameters + initial conditions
//-----------------------------------------------

Number of Bodies (without SUN) = 2
Internal Integration step dt = 0.0001
Main DT Step iterations N=100


Masses and Initial Conditions
Mass of Star   M = 1.0
-------------------------------------------------------------
Mass       a(AU)    ecc        incl      M     om     OM     r=
0.000025   0.900   0.3000     0.000   0.00    0.0    0.0     0.05
0.000063   1.500   0.2300     1.000   0.00    0.0    0.0     0.09


Period of 1st Planet in the list (Physical units) TP1=1
Number of DT1 iterations  NN=20000
```

Για να προστεθεί και άλλο σώμα πρέπει να προστεθεί μια καινούρια γραμμή που θα έχει τη μάζα του σώματος και τα κεπλέρια στοιχεία με τη σειρά που φαίνεται (στο τέλος της γραμμής υπάρχει και η ακτίνα r με την οποία θα εμφανιστεί στην οθόνη) και να αλλάξει ο αριθμός: Number of Bodies (without SUN) . Το βήμα που χρησιμοποιείται: Internal Integration step dt δεν πρέπει να γίνεται πολύ μεγάλο γιατί θα αρχίσουν να συσσωρεύονται σφάλματα στο σύστημα.

Αφού γίνουν αυτές οι αλλαγές τότε πρέπει να προστεθεί στην συνάρτηση myMenu(int n) μια εντολή:

```
case n+1: file_change("name"); break;
```

και τέλος στην συνάρτηση SetupMenu() να προστεθεί:

```
glutAddMenuEntry("name", n+1);
```

όπου "name" το όνομα του txt αρχείου και n+1 ο αριθμός της εισόδου.

Για διαφορετικά textures για κάποιο πλανήτη αρκεί η αλλαγή της εντολής

```
glBindTexture(GL_TEXTURE_2D,
venus->getTextureHandle());
```

όπου αντί για venus μπορεί να χρησιμοποιηθεί οποιοδήποτε από τα: earth, jupiter, mars, mercury, moon, neptune, pluto, saturn, uranus.

3.5 main.cpp

```
//Published under The MIT License (MIT)

#pragma warning (disable : 4018 )
#pragma warning (disable : 4996) // unsafe functions
#pragma warning (disable : 4305)

#include <GL\glut.h>
#include <solar_system\tga.h>
#include <solar_system\camera.h>
#include <cstdio>
#include <cstdlib>
#include <cmath>
#include <vector>

#include <nbody\dgdsysNBs.h>
#include <nbody\giroeNBs.h>
#include <nbody\gTools.h>
#include <nbody\Symp6s.h>
#include <nbody\restroe3.h>
#include <fstream>
#include <conio.h>
#include <nbody\drun0.h>
#include <iostream>

#define NMAXPOINTS 100000

using namespace std;

double *radius;
extern double dt;
extern OEDEF oeb[NMAXBODIES];
extern int Nsteps;
extern int NB, NDOF, NEQ;
double *X0 , *X ,*Y;

double Data[NMAXPOINTS][1000];
int NcurData;
```

```
double t;

// toggles if orbits are drawn
bool showOrbits = true;

void file_input();
void file_change(char system_name[]);
void Symplectic6thRUN(double yin[], int n,
double dt, double yout[]);
void Symplectic_translate();

// the screen size
int screenWidth, screenHeight;

// The instance of the camera
Camera camera;

// holds the state of the controls for the
camera - when true, the key for that control is being pressed
struct ControlStates
{
        bool forward, backward, left,
        right, yawLeft, yawRight, pitchUp,
        pitchDown, rollLeft, rollRight;
} controls;

// The TGA texture containing the planet textures
and the stars for the skybox
TGA *stars, *sun, *jupiter, *mars,
*mercury, *moon, *neptune, *pluto,
*saturn, *uranus, *venus;

void myMenu(int n)
{
        switch (n)
        {
        case 1: file_change("HD106315");
                break;
```

```
            case 2: file_change("kepler-9");
                    break;
            case 3: file_change("gliese876");
                    break;
            case 4: file_change("Sun");
                    break;
            case5: file_change("unstable");
                    break;
            case 6: file_change("user_program");
                    break;
            case 7: exit(0);
            }
            X0 = new double[NEQ];

            GetSystemPosV(oeb, X0);

            X = new double[NEQ];
            Y = new double[NEQ];

            for (int i = 0; i < NEQ; i++)
            {
                    X[i] = X0[i];
            }
            t = X[0] * t_metric;

            NcurData = 0;

            float vec[3];
            vec[0] = 0;
            vec[1] = 0;
            vec[2] = 0;

            camera.pointAt(vec);

            glutPostRedisplay();
}

void Symplectic_translate()
{
```

```
        Symplectic6thRUN(X, Nsteps, dt, Y);
        t = Y[0];

        for (int i = 0; i < NEQ; i++)
        {
                Data[NcurData][i] = X[i];
                X[i] = Y[i];
        }
        NcurData++;
         //double dE = Ener0 - Energy(X);
        // printf("dE=%d\n", dE);
        // double dL = Lang0 - AngularMomentum(X);

        return;
}

// timer function called every 10ms or more
void timer(int)
{
        glutPostRedisplay(); // post for display func
        glutTimerFunc(10, timer, 0);
// limit frame drawing to 100fps
}

void init(void)
{
glClearColor(0.0,0.0,0.0,0.0);
glShadeModel(GL_SMOOTH);
glEnable(GL_TEXTURE_2D);

// set up lighting
glEnable(GL_LIGHTING);
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();

GLfloat matSpecular[] = { 1.0, 1.0, 1.0, 1.0 };
GLfloat matAmbience[] = { 0.3, 0.3, 0.3, 1.0 };
GLfloat matShininess[] = {20.0};
glClearColor(0.0, 0.0, 0.0, 0.0);
```

```
glShadeModel(GL_SMOOTH);

glMaterialfv(GL_FRONT, GL_SPECULAR, matSpecular);
glMaterialfv(GL_FRONT, GL_SHININESS, matShininess);
glMaterialfv(GL_FRONT, GL_AMBIENT, matAmbience);

GLfloat lightAmbient[] = { 0.3, 0.3, 0.3, 1.0 };
GLfloat lightDiffuse[] = { 1.0, 1.0, 1.0, 1.0 };
GLfloat lightSpecular[] = { 1.0, 1.0, 1.0, 1.0 };

glLightfv(GL_LIGHT0, GL_AMBIENT, lightAmbient);
glLightfv(GL_LIGHT0, GL_DIFFUSE, lightDiffuse);
glLightfv(GL_LIGHT0, GL_SPECULAR, lightSpecular);

glEnable(GL_LIGHTING);
glEnable(GL_LIGHT0);
glDisable(GL_LIGHTING);

stars = new TGA("images/stars.tga");
sun = new TGA("images/sun.tga");
jupiter = new TGA("images/jupiter.tga");
mars = new TGA("images/mars.tga");
mercury = new TGA("images/mercury.tga");
moon = new TGA("images/moon.tga");
neptune = new TGA("images/neptune.tga");
pluto = new TGA("images/pluto.tga");
saturn = new TGA("images/saturn.tga");
uranus = new TGA("images/uranus.tga");
venus = new TGA("images/venus.tga");

// reset controls
controls.forward = false;
controls.backward = false;
controls.left = false;
controls.right = false;
controls.rollRight = false;
controls.rollLeft = false;
controls.pitchDown = false;
controls.pitchUp = false;
```

```
controls.yawLeft = false;
controls.yawRight = false;

timer(0);
}

void drawCube(void);

void display(void)
{
if (controls.forward) camera.forward();
if (controls.backward) camera.backward();
if (controls.left) camera.left();
if (controls.right) camera.right();
if (controls.yawLeft) camera.yawLeft();
if (controls.yawRight) camera.yawRight();
if (controls.rollLeft) camera.rollLeft();
if (controls.rollRight) camera.rollRight();
if (controls.pitchUp) camera.pitchUp();
if (controls.pitchDown) camera.pitchDown();

// clear the buffers
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
glColor3f(1.0, 1.0, 1.0);

// set up the perspective matrix for rendering the 3d world
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluPerspective(70.0f,
(float)screenWidth/(float)screenHeight, 0.001f, 500.0f);
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();

// perform the camera orientation transform
camera.transformOrientation();

// draw the skybox
glBindTexture(GL_TEXTURE_2D, stars->getTextureHandle());
drawCube();
```

```
// perform the camera translation transform
camera.transformTranslation();

GLfloat lightPosition[] = { 0.0, 0.0, 0.0, 1.0 };
glLightfv(GL_LIGHT0, GL_POSITION, lightPosition);

// render the solar system
glEnable(GL_DEPTH_TEST);
glEnable(GL_LIGHTING);

// render as a GLU sphere quadric object
GLUquadricObj* quadric = gluNewQuadric();
gluQuadricTexture(quadric, true);
gluQuadricNormals(quadric, GLU_SMOOTH);

Symplectic_translate();

glBindTexture(GL_TEXTURE_2D, sun->getTextureHandle());

glPushMatrix();
glTranslatef(0, 0, 0);
glRotatef(Y[0]*10, 0.0f, 0.0f, 1.0f);
glDisable(GL_LIGHTING);
gluSphere(quadric, 0.2, 30, 30);

// sun
glEnable(GL_LIGHTING);
glPopMatrix();

int j;
j = 0;

for (int i = 1; i <= NB; i++)
{
        if (i == 1)
        {
glBindTexture(GL_TEXTURE_2D, venus->getTextureHandle());
        }
```

```
        if (i == 2)
        {
glBindTexture(GL_TEXTURE_2D, pluto->getTextureHandle());
        }
        if (i == 3)
        {
glBindTexture(GL_TEXTURE_2D, jupiter->getTextureHandle());
        }
        if (i == 4)
        {
glBindTexture(GL_TEXTURE_2D, mars->getTextureHandle());
        }
        if (i == 5)
        {
glBindTexture(GL_TEXTURE_2D, saturn->getTextureHandle());
        }
        if (i > 5)
        {
glBindTexture(GL_TEXTURE_2D, moon->getTextureHandle());
        }

        glPushMatrix();
        glTranslatef(Y[1 + j], Y[2 + j], Y[3 + j]);
        gluSphere(quadric, radius[i], 30, 30);
        glPopMatrix();
        j += 3;
}

if (showOrbits)
        {
        glDisable(GL_LIGHTING);
        int o = 0;
        for (int i = 0; i < NB; i++)
                {
        glBegin(GL_POINTS);
        for (int j = NcurData - 1; j > 0; j--)
                        {
        glVertex3f(Data[j][o + 1],
        Data[j][o + 2], Data[j][o + 3]);
```

```
                        }
        o += 3;

        glEnd();
                }
        glEnable(GL_LIGHTING);
        }

glDisable(GL_LIGHTING);
glDisable(GL_DEPTH_TEST);

glFlush();
glutSwapBuffers();
}

void keyDown(unsigned char key, int x, int y)
{
switch (key)
        {
case 'o':
        showOrbits = !showOrbits;//toggle show orbits
        break;
case ',':
        camera.slowDown(); // slow down camera
        break;
case '.':
        camera.speedUp(); // speed up camera
        break;
                // these are all camera controls
case 'w':
        controls.forward = true;
        break;
case 's':
        controls.backward = true;
        break;
case 'a':
        controls.left = true;
        break;
case 'd':
```

```
        controls.right = true;
        break;
case 'l':
        controls.rollRight = true;
        break;
case 'j':
        controls.rollLeft = true;
        break;
case 'i':
        controls.pitchDown = true;
        break;
case 'k':
        controls.pitchUp = true;
        break;
case 'q':
        controls.yawLeft = true;
        break;
case 'e':
        controls.yawRight = true;
        break;
        }
}

void keyUp(unsigned char key, int x, int y)
{
switch (key)
        {
case 'w':
        controls.forward = false;
        break;
case 's':
        controls.backward = false;
        break;
case 'a':
        controls.left = false;
        break;
case 'd':
        controls.right = false;
        break;
```

```
case 'l':
        controls.rollRight = false;
        break;
case 'j':
        controls.rollLeft = false;
        break;
case 'i':
        controls.pitchDown = false;
        break;
case 'k':
        controls.pitchUp = false;
        break;
case 'q':
        controls.yawLeft = false;
        break;
case 'e':
        controls.yawRight = false;
        break;
        }
}

void reshape(int w, int h)
{
screenWidth = w;
screenHeight = h;
glViewport(0, 0, (GLsizei)w, (GLsizei)h);
}

void SetupMenu()
{
glutCreateMenu(myMenu);
glutAddMenuEntry("HD 106315", 1);
glutAddMenuEntry("kepler-9", 2);
glutAddMenuEntry("gliese876", 3);
glutAddMenuEntry("Sun", 4);
glutAddMenuEntry("unstable", 5);
glutAddMenuEntry("user_system", 6);
glutAddMenuEntry("exit", 7);
glutAttachMenu(GLUT_RIGHT_BUTTON);
```

```
}

int main(int argc, char** argv)
{
radius = new double[100];
file_input();
X0 = new double[NEQ];
GetSystemPosV(oeb, X0);
NcurData = 0;
X = new double[NEQ];
Y = new double[NEQ];
for (int i = 0; i < NEQ; i++)
        {
X[i] = X0[i];
        }
        ouble t = X[0] * t_metric;

glutInit(&argc, argv);
glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
glutInitWindowSize(1200, 700);
glutInitWindowPosition(100, 100);
glutCreateWindow(argv[0]);
init();
glutDisplayFunc(display);
glutReshapeFunc(reshape);
glutKeyboardFunc(keyDown);
        glutKeyboardUpFunc(keyUp);
SetupMenu();

glutMainLoop();

return 0;
}

void drawCube(void)
{
glBegin(GL_QUADS);
// new face
glTexCoord2f(0.0f, 0.0f);glVertex3f(-1.0f, -1.0f, 1.0f);
```

```
glTexCoord2f(1.0f, 0.0f);glVertex3f(1.0f, -1.0f, 1.0f);
glTexCoord2f(1.0f, 1.0f);glVertex3f(1.0f, 1.0f, 1.0f);
glTexCoord2f(0.0f, 1.0f);glVertex3f(-1.0f, 1.0f, 1.0f);
// new face
glTexCoord2f(0.0f, 0.0f);glVertex3f(1.0f, 1.0f, 1.0f);
glTexCoord2f(1.0f, 0.0f);glVertex3f(1.0f, 1.0f, -1.0f);
glTexCoord2f(1.0f, 1.0f);glVertex3f(1.0f, -1.0f, -1.0f);
glTexCoord2f(0.0f, 1.0f);glVertex3f(1.0f, -1.0f, 1.0f);
// new face
glTexCoord2f(0.0f, 0.0f);glVertex3f(1.0f, 1.0f, -1.0f);
glTexCoord2f(1.0f, 0.0f);glVertex3f(-1.0f, 1.0f, -1.0f);
glTexCoord2f(1.0f, 1.0f);glVertex3f(-1.0f, -1.0f, -1.0f);
glTexCoord2f(0.0f, 1.0f);glVertex3f(1.0f, -1.0f, -1.0f);
// new face
glTexCoord2f(0.0f, 0.0f);glVertex3f(-1.0f, -1.0f, -1.0f);
glTexCoord2f(1.0f, 0.0f);glVertex3f(-1.0f, -1.0f, 1.0f);
glTexCoord2f(1.0f, 1.0f);glVertex3f(-1.0f, 1.0f, 1.0f);
glTexCoord2f(0.0f, 1.0f);glVertex3f(-1.0f, 1.0f, -1.0f);
// new face
glTexCoord2f(0.0f, 0.0f);glVertex3f(-1.0f, 1.0f, -1.0f);
glTexCoord2f(1.0f, 0.0f);glVertex3f(1.0f, 1.0f, -1.0f);
glTexCoord2f(1.0f, 1.0f);glVertex3f(1.0f, 1.0f, 1.0f);
glTexCoord2f(0.0f, 1.0f);glVertex3f(-1.0f, 1.0f, 1.0f);
// new face
glTexCoord2f(0.0f, 0.0f);glVertex3f(-1.0f, -1.0f, -1.0f);
glTexCoord2f(1.0f, 0.0f);glVertex3f(1.0f, -1.0f, -1.0f);
glTexCoord2f(1.0f, 1.0f);glVertex3f(1.0f, -1.0f, 1.0f);
glTexCoord2f(0.0f, 1.0f);glVertex3f(-1.0f, -1.0f, 1.0f);

glEnd();
}
```

References

[1] A.E Roy , "Orbital motion" , Adam-Hilger , 1982

[2] Γ. Βουγιατζής , "Μαθήματα -Σημειώσεις OpenGL" , Σημειώσεις ΠΜΣ Υπολογιστικής Φυσικής , ΑΠΘ , 2010

[3] Ο.Κότσας , "Μελέτη της δυναμικής συστημάτων τριών πλανητών : ευστάθεια και παγίδευση σε συντονισμούς , 2015

[4] Δ.Σκουλίδου , "Συμπλεκτική ολοκλήρωση του προβλήματος των Ν σωμάτων και εφαρμογή σε εξωηλιακά συστήματα" , Διπλωματική Εργασία ,Τμήμα Φυσικής , ΑΠΘ , 2017

[5] Κ.Τσιγάνης , "Υπολογιστική Αστροδυναμική : Το πρόβλημα των δύο σωμάτων " , Σημειώσεις ΠΜΣ Υπολογιστικής Φυσικής , ΑΠΘ , 2012

[6] H.Yoshida,Cel. Mech. Dyn. Astron. 1993

[7] Colorado Center for Astrodynamics Research (CCAR) ,2002