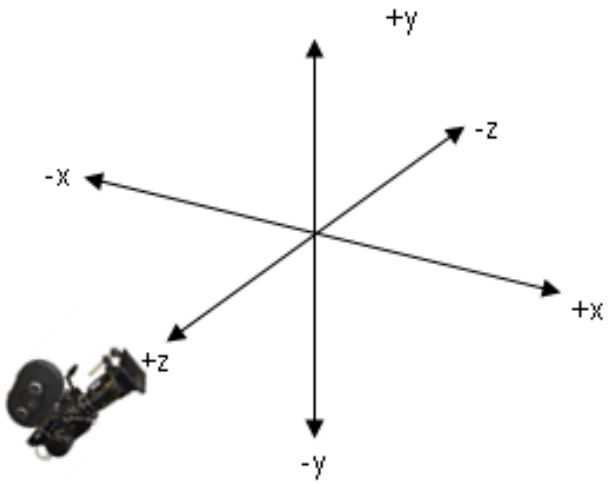
Μετασχηματισμοί στην OpenGL

Η OpenGL υποστηρίζει μια σειρά μετασχηματισμών τους οποίους μπορούμε να χρησιμοποιήσουμε για να τοποθετήσουμε τα αντικείμενα μας στην οθόνη, να τα περιστρέψουμε, να τα μεγεθύνουμε ή να τα μικρύνουμε.

Αναφέραμε στο προηγούμενο εργαστήριο ότι η OpenGL χρησιμοποιεί ένα δεξιόστροφο σύστημα συντεταγμένων κάμερας (view space ή camera space) :



Αυτό σημαίνει ότι αν φανταστούμε την οθόνη του υπολογιστή, ο θετικός άξονας Y δείχνει προς τα πάνω, ο θετικός άξονας X προς τα δεξιά και ο αρνητικός άξονας Z προς τα μέσα της οθόνης.

Ο μετασχηματισμός που τοποθετεί την κάμερα στην κατάλληλη θέση ώστε να «φωτογραφήσουμε» την σκηνή λέγεται **μετασχηματισμός κάμερας** (viewing transformation).

Οι μετασχηματισμοί που τοποθετούν τα αντικείμενα της σκηνής στην επιθυμητή τους θέση, τα μεγεθύνουνε/σμικρύνουνε και τα περιστρέφουνε ονομάζονται **μετασχηματισμοί μοντέλου** (modeling transformations).

> **Στην ορολογία της OpenGL, ο μετασχηματισμός μοντέλου είναι ο μετασχηματισμός που τοποθετεί ένα αντικείμενο στο σύστημα συντεταγμένων της σκηνής (κόσμου). Στην γενική ορολογία γραφικών αυτό το μετασχηματισμό το ονομάζουμε *μετασχηματισμό κόσμου*.**

Ο μετασχηματισμός που καθορίζει τι είναι ορατό στο σύστημα συντεταγμένων κάμερας (view space) και τι είδους προβολή επιθυμούμε (με προοπτική ή όχι) ονομάζεται **μετασχηματισμός προβολής** (projection transformation)

Η OpenGL συνδυάζει τους πίνακες μετασχηματισμού μοντέλου και μετασχηματισμού κάμερας σε έναν μόνο πίνακα, τον μετασχηματισμό **Μοντέλου-Κάμερας** (**ModelView matrix**). Με αυτό το μετασχηματισμό μπορούμε να τοποθετήσουμε ένα αντικείμενο κατευθείαν στο χώρο της κάμερας (χωρίς να περάσουμε από το σύστημα συντεταγμένων κόσμου).

Μπορούμε να κατασκευάσουμε τον μετασχηματισμό Μοντέλου-Κάμερας με τέτοιο τρόπο ώστε να περιέχει κάθε περιστροφή, μετακίνηση και κλίμακα του αντικειμένου.

Μετακίνηση

Η μετακίνηση (translation) ενός αντικειμένου γίνεται με την εντολή

glTranslatef(GLfloat X, GLfloat Y, GLfloat Z);

Η εντολή αυτή παίρνει ως παραμέτρους την μετακίνηση που επιθυμούμε κατά άξονα, κατασκευάζει ένα μετασχηματισμό μετακίνησης T (όπως είδαμε στην θεωρία) και τον εφαρμόζει (πολ/ζει) με τον μετασχηματισμό Μοντέλου-Κάμερας.

Περιστροφή

Η περιστροφή (rotation) ενός αντικειμένου γίνεται με την εντολή

glRotatef(GLfloat angle, GLfloat x, GLfloat y, GLfloat z);

Η εντολή αυτή κατασκευάζει ένα μετασχηματισμό περιστροφής γύρω από το διάνυσμα που ορίζεται από την τριάδα [x, y, z], κατά γωνία angle (μετριέται σε μοίρες 0-360) με φορά αντίθετη των δεικτών του ρολογιού.

Αν μας ενδιαφέρει απλά να περιστρέψουμε το αντικείμενο γύρω από ένα άξονα (X, Y ή Z) μπορούμε να ορίσουμε το διάνυσμα ως [1, 0, 0] για το περιστροφή γύρω από τον X, ως [0, 1, 0] για το περιστροφή γύρω από τον Y και ως [0, 0, 1] για το περιστροφή γύρω από τον Z.

Κλίμακα

Η μεγέθυνση/σμίκρυνση (scaling) ενός αντικειμένου γίνεται με την εντολή

glScalef(GLfloat x, GLfloat y, GLfloat z);

όπου x, y, z η κλίμακα του αντικειμένου ανά άξονα.

Τοποθέτηση κάμερας

Η OpenGL μας δίνει την δυνατότητα να τοποθετήσουμε την κάμερα στην σκηνή και να θέσουμε κατεύθυνσή της με την χρήση μόνο μιας εντολής:

gluLookAt(GLdouble eyex, GLdouble eyey, GLdouble eyez, GLdouble centerx, GLdouble centery, GLdouble centerz, GLdouble upx, GLdouble upy, GLdouble upz);

Η εντολή αυτή παίρνει ως παραμέτρους 3 διανύσματα το [eyex, eyey, eyez] που ορίζει την **θέση** της κάμερας στην σκηνή, το [centerx, centery, centery] το οποίο καθορίζει

την **κατεύθυνση** της κάμερας (το σημείο στο οποίο δείχνει δηλαδή) και το διάνυσμα [upx, upy, upz] που καθορίζει ποια είναι η**«πάνω» κατεύθυνση** της κάμερας.

Για παράδειγμα αν θέλω να ορίσω μια κάμερα που βρίσκεται στο σημείο [10,10,10], «δείχνει/βλέπει» στην αρχή των αξόνων [0,0,0] και η πάνω κατεύθυνση της είναι ο Y άξονας [0 1 0] θα καλέσω την εντολή ως

gluLookAt(10,10,10,0,0,0,0,1,0);

Εφαρμογή των μετασχηματισμών στην OpenGL

Αναφέραμε ότι η OpenGL χρησιμοποιεί μόνο ένα πίνακα, τον ModelView, για να εφαρμόσει όλους τους μετασχηματισμούς μετακίνησης, περιστροφής και κλίμακας που είδαμε.

Ο πίνακας αυτός αποθηκεύεται στην μνήμη της OpenGL και παραμένει ο ίδιος μέχρι να τον αλλάξουμε με μια άλλη εντολή.

> **Το σύνολο όλων των μεταβλητών/παραμέτρων που διατηρεί η OpenGL για την λειτουργία της ονομάζεται μηχανή καταστάσεων (state machine). Με αυτό τον τρόπο λειτουργίας μια τιμή που θέτουμε σε μια μεταβλητή παραμένει σε ισχύ μέχρι να την αλλάξουμε ξανά.**

Επειδή η OpenGL χρησιμοποιεί πάνω από ένα πίνακες για να ορίσει μετασχηματισμούς πριν αρχίσουμε να κατασκευάσουμε το ModelView μετασχηματισμό πρέπει να τον ενεργοποιήσουμε. Αυτό γίνεται με την εντολή

glMatrixMode(GL_MODELVIEW);

Η παράμετρος GL_MODELVIEW ορίζει ότι θέλουμε να ενεργοποιήσουμε τον πίνακα ModelView (μια άλλη παράμετρος που θα χρησιμοποιήσουμε είναι η GL_PROJECTION που ορίζει ως ενεργό τον πίνακα προβολής)

Από την στιγμή που ενεργοποιήσουμε τον ModelView πίνακα, κάθε μετασχηματισμός που κατασκευάζουμε με τις εντολές glTranslation, glRotation και glScale εφαρμόζονται, με πολλαπλασιασμό, με τον πίνακα που έχει κάθε στιγμή ο ModelView.

Για να αρχικοποιήσουμε τον ModelView πίνακα χρησιμοποιούμε την εντολή

glLoadIdentity();

η οποία του φορτώνει τον μοναδιαίο πίνακα.

> **Εξορισμού (αν δεν έχουμε εφαρμόσει κάποιο άλλο μετασχηματισμό) η κάμερα είναι τοποθετημένη στην αρχή των αξόνων και βλέπει προς τον άξονα -Z**

Παράδειγμα:

glMatrixMode(GL_MODELVIEW);

glLoadIdentity();

gluLookAt(10,10,10, 0,0,0, 0,1,0);

glTranslate(0, 0, -6);

glRotatef(45, 0, 1, 0);

DrawOurModel();

Αρχικά ενεργοποιούμε τον πίνακα ModelView, τον αρχικοποιούμε με τον μοναδιαίο πίνακα I, κατασκευάζουμε και εφαρμόζουμε τον μετασχηματισμό κάμερας View (gluLookAt), κατασκευάζουμε και εφαρμόζουμε μια μετακίνηση T κατά -6 κατά μήκος του άξονα Z και μια περιστροφή R, 45 μοίρες γύρω από τον άξονα Y.

Ο τελικός μετασχηματισμός ModelView θα είναι :

ModelView = R * T * View

Σημαντική παρατήρηση : Η OpenGL εφαρμόζει τους μετασχηματισμούς στο αντικείμενο ***με αντίστροφη σειρά*** από ότι τους δηλώνουμε στο πρόγραμμα. Στο παραπάνω παράδειγμα το αντικείμενο πρώτα θα περιστραφεί, μετά θα μετακινηθεί και μετά θα μεταφερθεί στο χώρο της κάμερας.

Αν θεωρήσουμε ένα σημείο P του αντικειμένου, τότε ο μετασχηματισμός αυτός θα εφαρμοστεί ως

P’ = P * ModelView = P * R * T * View

Στοίβα πινάκων

Είπαμε ότι ό, τι μετασχηματισμό κατασκευάζουμε με τις εντολές glTranslation, glRotation και glScale συσσωρεύεται (με πολ/σμό) στο πίνακα ModelView, ο οποίος με την σειρά του μετασχηματίζει **όλα** τα αντικείμενα της σκηνής. Υπάρχουν περιπτώσεις που πριν «προσθέσουμε» ένα νέο μετασχηματισμό στο ModelView (μια περιστροφή που ίσως αφορά ένα μόνο αντικείμενο και όχι όλη τη σκηνή) θέλουμε να τον αποθηκεύσουμε κάπου προσωρινά ώστε να τον ανακτήσουμε αναλλοίωτο μετά από τον νέο μετασχηματισμό.

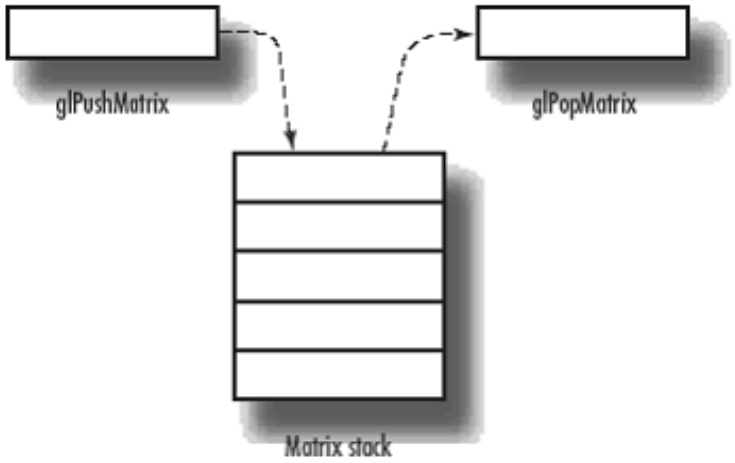
Η OpenGL προσφέρει ένα τέτοιο μηχανισμό και ονομάζεται **στοίβα πινάκων** (**matrix stack**). Η στοίβα είναι μια δομή δεδομένων που μας επιτρέπει μέσω εντολών Push και Pop να προσθέσουμε και να αφαιρέσουμε ένα στοιχείο από την κορυφή της στοίβας (και μόνο).

Στην OpenGL οι εντολές που αποθηκεύουν και ανακτούν τον ModelView στην κορυφή της στοίβας είναι οι

glPushMatrix();

glPopMatrix();

Σχηματικά αυτό φαίνεται ως εξής:



Παράδειγμα, αν θέλουμε να ζωγραφίσουμε 2 αντικείμενα αφού πρώτα τα μετακινήσουμε το πρώτο κατά -6 κατά μήκος του άξονα Z και το δεύτερο κατά 10 κατά τον άξονα Y τότε

glMatrixMode(GL_MODELVIEW);

glLoadIdentity();

gluLookAt(10,10,10, 0,0,0, 0,1,0);

glPushMatrix();

glTranslate(0, 0, -6);

DrawModel_1();

glPopMatrix();

glTranslate(0, 10, 0);

DrawModel_2();

Στο παράδειγμα αφού ορίσουμε τον μετασχηματισμό κάμερας με την εντολή gluLookAt() αποθηκεύουμε τον ModelView στην στοίβα με την εντολή glPushMatrix(). Έπειτα μετακινούμε το πρώτο αντικείμενο κατά -6 κατά μήκος του άξονα Z και το ζωγραφίζουμε. Στην συνέχεια ανακτούμε από την στοίβα τον ModelView (**που δεν περιέχει την μετακίνηση κατά -6**) με την εντολή glPopMatrix(). Τέλος μετακινούμε το δεύτερο αντικείμενο κατά 10 κατά τον άξονα Y και το ζωγραφίζουμε.

Αν υποθέσουμε ότι δεν χρησιμοποιούμε την στοίβα για να αποθηκεύσουμε τον ModelView:

glMatrixMode(GL_MODELVIEW);

glLoadIdentity();

gluLookAt(10,10,10, 0,0,0, 0,1,0);

glTranslate(0, 0, -6);

DrawModel_1();

glTranslate(0, 10, 0);

DrawModel_2();

τότε το πρώτο αντικείμενο θα μετακινούνταν σωστά κατά -6 κατά μήκος του άξονα Z όμως το δεύτερο αντικείμενο θα μετακινούταν πρώτα κατά -6 κατά μήκος του άξονα Z και έπειτα κατά 10 κατά τον άξονα Y.

Θα εφαρμόζαμε δηλαδή και τις 2 μετακινήσεις στο δεύτερο αντικείμενο αφού όπως είπαμε όλοι οι μετασχηματισμοί που κατασκευάζουμε «συσσωρεύονται» στο πίνακα ModelView.

Η χρήση της στοίβας είναι βασική λοιπόν για την σωστή τοποθέτηση των αντικειμένων στην σκηνή.

Μετασχηματισμός προβολής

Με τον μετασχηματισμό προβολής (projection transform) η OpenGL προβάλλει τα 3διάστατα αντικείμενα στις 2 διαστάσεις ώστε να δημιουργήσει την τελική διδιάστατη εικόνα.

Η OpenGL υποστηρίζει 2 ειδών μετασχηματισμούς προβολών την ορθογραφική προβολή (orthographic projection) και την προβολή με προοπτική (perspective projection).

Μια ορθογραφική προβολή δημιουργείται με την χρήση της εντολής

glOrtho(GLdouble left, GLdouble right, GLdouble bottom,

GLdouble top, GLdouble near, GLdouble far);

Η εντολή αυτή ορίζει ένα κύβο αποκοπής παρέχοντας τις συντεταγμένες 2 διαγώνιων κορυφών.

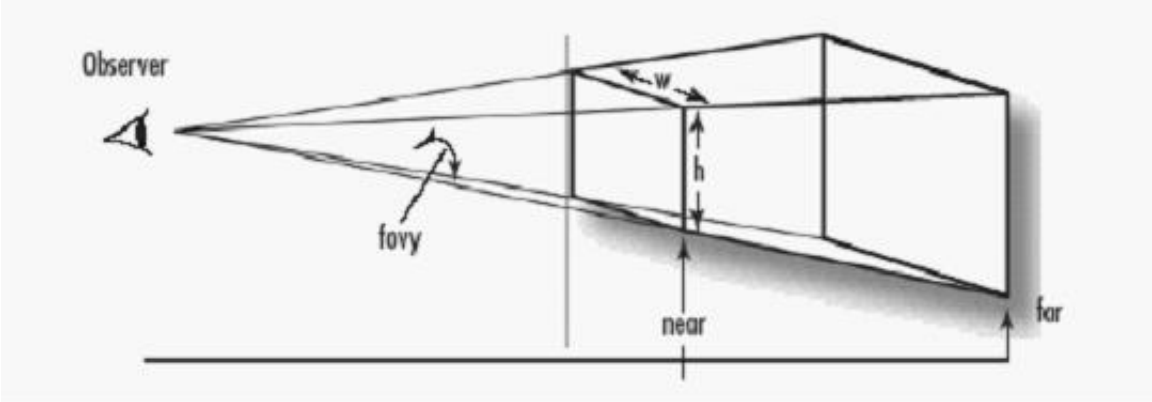
Μια προβολή με προοπτική δημιουργείται με την χρήση της εντολής

gluPerspective(GLdouble fovy, GLdouble aspect,

GLdouble zNear, GLdouble zFar);

Η παράμετρος fovy ορίζει την γωνία οπτικού πεδίου στην κάθετη κατεύθυνση, η aspect καθορίζει τον αναλογία μήκους/πλάτους του κοντινού πεδίου αποκοπής και οι zNear και zFar την απόσταση του κοντινού (near) και μακρινού (far) πεδίου αποκοπής κατά τον άξονα Z.

Σχηματικά η προβολή με προοπτική ορίζεται ως



Η OpenGL αποθηκεύει τον πίνακα προβολής στον πίνακα GL_PROJECTION. Για να τον ενεργοποιήσουμε χρησιμοποιούμε και πάλι την εντολή glMatrixMode()

Παράδειγμα:

glMatrixMode(GL_PROJECTION);

glLoadIdentity();

gluPerspective(45, 4/3, 1, 400);

Εφαρμογή μετασχηματισμών στην OpenGL: Ένα Ηλιακό Σύστημα

Στην σημερινή εργαστηριακή άσκηση θα εφαρμόσουμε τους μετασχηματισμούς της OpenGL για να κατασκευάσουμε ένα (μίνι) ηλιακό σύστημα.

Φορτώστε το project lab2.dev στο περιβάλλον ανάπτυξης εφαρμογών DevC++.

Ο κώδικας της εφαρμογής βρίσκεται στο αρχείο main.cpp. Ο κώδικας περιέχει σχόλια για όλα τα σημεία του προγράμματος που μας ενδιαφέρουν. Κομμάτια του κώδικα που έχουν να κάνουν με φωτισμό σκηνής και άλλες παραμετροποιήσεις δεν μας αφορούν σε αυτή την άσκηση και μένουν ασχολίαστα.

Η εφαρμογή ακολουθεί το format της πρώτης εργαστηριακής άσκησης το οποίο είναι εν συντομία:

main()

Η «συνάρτηση-εισόδου» στο πρόγραμμα είναι η main(). Εκεί αρχικοποιούμε το GLUT, δημιουργούμε το παράθυρο στο οποίο θα δείξουμε την τελική εικόνα, ορίζουμε τις συναρτήσεις που θα χρησιμοποιεί το GLUT για να ζωγραφίζει την σκηνή (renderScene) και θα εξυπηρετεί είσοδο από το πληκτρολόγιο (keyPress), αρχικοποιούμε την εφαρμογή (init) και μπαίνουμε στο κυρίως loop του GLUT (glutMainLoop).

keyPress()

Η συνάρτηση keyPress καλείται όταν πατηθεί κάποιο πλήκτρο. Ελέγχει αν αυτό είναι ‘q’ ή ‘ESC’ και αν ναι, τερματίζει η εφαρμογή.

init()

Στην init() καλούμε συναρτήσεις και κάνουμε αρχικοποιήσεις που πρέπει να γίνουν μόνο μια φορά. Στην συγκεκριμένη εφαρμογή την χρησιμοποιούμε για να ορίσουμε τον μετασχηματισμό προβολής GL_PROJECTION και να δημιουργήσουμε τα αστέρια του φόντου. Οι θέσεις των αστεριών είναι τυχαίες, όπως και το μέγεθος τους.

renderScene()

Αυτή είναι η συνάρτηση που ζωγραφίζει όλα τα αντικείμενα της σκηνής μας.

Καλώντας την συνάρτηση

glutGet(GLUT_ELAPSED_TIME);

παίρνουμε το χρόνο που έχει περάσει από την αρχή εκτέλεσης του προγράμματος. Έτσι μπορούμε να υπολογίσουμε την γωνία περιστροφής των πλανητών **a** η οποία θα μεταβάλλεται με τον χρόνο.

Η συνάρτηση renderScene() ενεργοποιεί τον ModelView πίνακα, τον αρχικοποιεί, και καθορίζει την θέση και φορά της κάμερας:

glMatrixMode(GL_MODELVIEW);

glLoadIdentity();

gluLookAt(0,6,13, 0,0,0, 0,1,0);

Στην συνέχεια ζωγραφίζει τα αστέρια του φόντου ως σημεία στο χώρο διαβάζοντας τις θέσεις και τα μεγέθη τους από τον πίνακα Stars.

Σ'αυτό το σημείο βλέπουμε πως μπορούμε να σχεδιάσουμε στην OpenGL χωρίς να χρησιμοποιήσουμε τις έτοιμες συναρτήσεις του GLUT όπως glutSolidSphere(), glutSolidCube();

Ο τρόπος που γίνεται αυτό είναι στέλνοντας στην OpenGL τα σημεία της γεωμετρίας μας με την εντολή

glVertex3f(GLfloat x, GLfloat y, GLfloat z);

Για να ζωγραφίσει τα σημεία αυτά η OpenGL χρειάζεται ακόμα μια πληροφορία: πως θα τα ενώσει μεταξύ τους ώστε να σχηματιστεί το αντικείμενο. Υπάρχουν πολλές επιλογές: GL_POINTS για σημεία, GL_LINES για ευθύγραμμα τμήματα, GL_TRIANGLES για τρίγωνα και άλλες.

Την παράμετρο αυτή την περνάμε στο ζεύγος glBegin()/glEnd() που οριοθετεί την περιοχή που προσδιορίζουμε τα σημεία της γεωμετρίας.

glBegin(GL_POINTS);

glVertex3f(Stars[i].x,Stars[i].y, -20.0f);

glEnd();

Στην συνέχεια ζωγραφίζουμε τους πλανήτες ξεκινώντας με τον ήλιο που βρίσκεται στο κέντρο του συστήματος

//σώσε τον πίνακα ModelView στην στοίβα

glPushMatrix();

//κλιμάκωσε τον ModelView (άλλαξε μέγεθος του ήλιου)

glScalef(1,1,1);

//ζωγράφισε μια σφαίρα (ήλιος)

glutSolidSphere(1,64,64);

//φόρτωσε τον πίνακα ModelView από την στοίβα

glPopMatrix();

Ο λόγος που αποθηκεύουμε τον ModelView στην στοίβα είναι γιατί θέλουμε να εφαρμόσουμε ένα μετασχηματισμό κλίμακας στον ήλιο. Αν δεν αποθηκεύαμε τον πίνακα η κλίμακα θα επιδρούσε σε όλα τα αντικείμενα που θα σχεδιάζαμε μετέπειτα. Αφού ζωγραφίσουμε τον ήλιο με την glutSolidSphere επαναφέρουμε τον ModelView από την στοίβα για να αναιρέσουμε την κλίμακα.

Με παρόμοιο τρόπο ζωγραφίζουμε την γη αφού την μετακινήσουμε σε σχέση με τον ήλιο κατά -6 και την περιστρέψουμε γύρω από αυτόν κατά γωνία **a**.

glRotatef(a,0,1,0);

//μετακίνησε την γη κατά -6 κατά μήκος του άξονα Z

glTranslatef(0,0,-6);

//σώσε τον πίνακα ModelView στην στοίβα

glPushMatrix();

//κλιμάκωσε την γη

glScalef(0.5,0.5,0.5);

//ζωγράφισε την γη σαν σφαίρα

glutSolidSphere(1, 16, 16);

//φόρτωσε τον πίνακα ModelView από την στοίβα

glPopMatrix();

Ξανά, πριν εφαρμόσουμε τον μετασχηματισμό κλίμακας με την glScale() σώνουμε τον πίνακα ModelView στην στοίβα και αφού ζωγραφίσουμε την γη τον ανακτούμε από αυτήν.

Με ακριβώς όμοιο τρόπο ζωγραφίζουμε και την σελήνη να περιστρέφεται γύρω από την γη.

Πράγματα να δοκιμάσετε

Α) Αλλάξτε το χρώμα των ουράνιων αντικειμένων αλλάζοντας τις παραμέτρους της εντολής

glColor3f(GLfloat red, GLfloat green, GLfloat blue)

Θυμίζουμε ότι οι παράμετροι red, green, blue παίρνουν τιμές από 0 μέχρι 1 και ορίζουν το ποσοστό του κάθε χρώματος στο τελικό χρώμα (δηλαδή πόσο κόκκινο, πράσινο και μπλε θα περιέχει το χρώμα);

Πχ η εντολή glColor3f(1,1,0) θα θέσει το χρώμα σε κίτρινο, ενώ η εντολή glColor3f(0.5,0.5,0.5) σε γκρι.

Β) Μεγαλώστε τον ήλιο σε μέγεθος κλιμακώνοντας τον κατά 2 σε κάθε διάσταση

Γ) Μετακινήστε το φεγγάρι σε διπλάσια απόσταση από τη γη από ότι είναι τώρα

Δ) Κάντε την γη να γυρίζει 2 φορές πιο γρήγορα γύρω από τον ήλιο

Ε) Δοκιμάστε να μετακινήσετε την κάμερα, αλλάζοντας τις παραμέτρους της εντολής

*gluLookAt(GLdouble **eyex**, GLdouble **eyey**, GLdouble **eyez**,*

*GLdouble **centerx**, GLdouble **centery**, GLdouble **centerz**,*

*GLdouble **upx**, GLdouble **upy**, GLdouble **upz**);*

Υπενθυμίζουμε ότι η τριάδα [eyex, eyey, eyez] ορίζει την **θέση** της κάμερας στην σκηνή, το [centerx, centery, centery] καθορίζει την **κατεύθυνση** της κάμερας (το σημείο στο οποίο δείχνει δηλαδή) και το διάνυσμα [upx, upy, upz] καθορίζει ποια είναι η **«πάνω» κατεύθυνση** της κάμερας.

Ζ) Προσθέστε ακόμα ένα πλανήτη στο ηλιακό μας σύστημα. Αυτό γίνεται ακριβώς όπως προσθέσαμε την γη με μια διαφορά: πριν από το τμήμα που προσθέτει την γη στη σκηνή πρέπει να αποθηκεύσουμε τον ModelView στην στοίβα και να τον ανακτήσουμε αμέσως μόλις ζωγραφίσουμε και το φεγγάρι. Ο λόγος που πρέπει να γίνει αυτό είναι γιατί όταν έρθει η ώρα να προσθέσουμε τον νέο πλανήτη δεν θέλουμε ο ModelView να περιέχει τους μετασχηματισμούς (μετακινήσεις, περιστροφές και κλίμακες) που προσθέσαμε για τη γη και το φεγγάρι.

1. Τραβήξτε την κάμερα πιο πίσω ώστε να έχετε μεγαλύτερο οπτικό πεδίο πχ gluLookAt(5,8,20, 0,0,0, 0,1,0);

1. βάψτε τον πλανήτη ότι χρώμα θέλετε,

1. μετακινήστε το νέο πλανήτη κατά -9 στον άξονα Z,

1. κλιμακώστε τον κατά 0.8,
2. κάντε τον να περιστρέφετε γύρω από τον άξονα Υ κατά αντίθετη φορά από ότι η Γη.