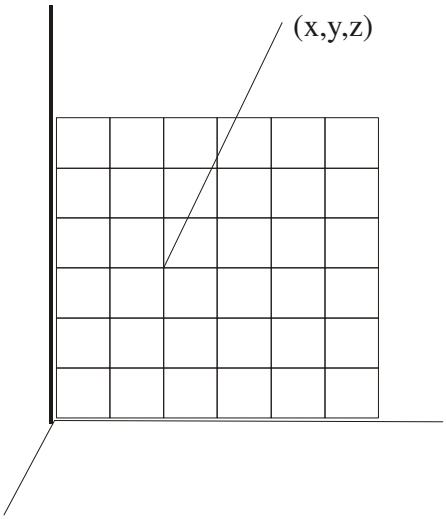
Α. ΓΕΩΜΕΤΡΙΚΟΙ ΜΕΤΑΣΧΗΜΑΤΙΣΜΟΙ ΑΝΤΙΚΕΙΜΕΝΩΝ

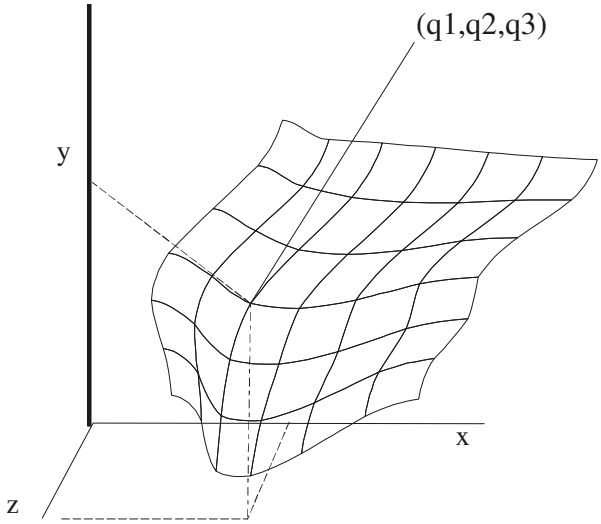
Α1. Καμπυλόγραμμα συστήματα

Ένας γεωμετρικός μετασχηματισμός αντικειμένου μπορεί να περιγραφεί με έναν μετασχηματισμό συστήματος συντεταγμένων και προβολής του στο αρχικό καρτεσιανό σύστημα. Θεωρούμε δηλαδή, τις τιμές των καρτεσιανών συντεταγμένων ως τιμές ενός συστήματος καμπυλόγραμμων συντεταγμένων.

Καρτεσιανό σύστημα $(x, y, z) \rightarrow (q_1, q_2, q_3)$ Καμπυλόγραμμο σύστημα

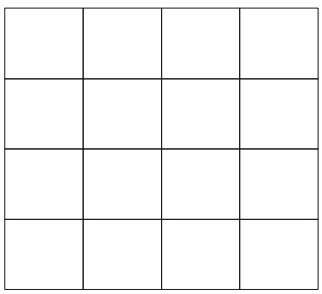
$$x = f_1(q_1, q_2, q_3), \quad y = f_2(q_1, q_2, q_3), \quad z = f_3(q_1, q_2, q_3),$$



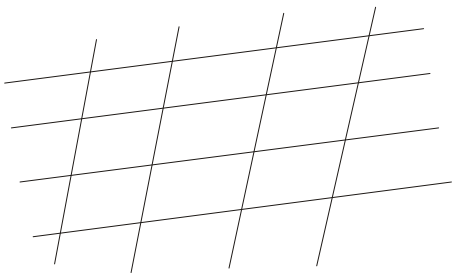


Κάθε ομαλό καμπυλόγραμμο σύστημα μπορεί τοπικά να προσεγγιστεί με ένα «πλαγιογώνιο» σύστημα που ορίζεται με γραμμικές συναρτήσεις f_i, i=1,2,3 (**Γραμμικός Μετασχ**.)

$$\begin{aligned} x' &= a_{11}x + a_{12}y + a_{13}z \\ y' &= a_{21}x + a_{22}y + a_{23}z \qquad a_{ij} \neq \text{const.} \\ z' &= a_{31}x + a_{32}y + a_{33}z \end{aligned}$$



(x,y,z)



(x’,y’,z’)

A2. Αφινικοί μετασχηματισμοί

Ένας **αφινικός μετασχηματισμός** είναι ένας **γραμμικός μετασχηματισμός** συνοδευόμενος και από μια **μετατόπιση**

$$x' = a_{11}x + a_{12}y + a_{13}z + \delta x$$
$$y' = a_{21}x + a_{22}y + a_{23}z + \delta y$$
$$z' = a_{31}x + a_{32}y + a_{33}z + \delta z$$

ή σε μορφή πινάκων

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{pmatrix} \delta x \\ \delta y \\ \delta z \end{pmatrix}$$

$$\mathbf{r}' = \mathbf{A}\mathbf{r} + \boldsymbol{\delta}$$

- Αν $\boldsymbol{\delta}=(0,0,0)^T$ και $\mathbf{A}=\mathbf{I}$, τότε παίρνουμε τον ταυτοτικό μετασχηματισμό **(identity)**

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \Rightarrow \begin{matrix} x' = x \\ y' = y \\ z' = z \end{matrix}$$

- Αν $\boldsymbol{\delta}=(\delta x,\delta y,\delta z)^T$ και $\mathbf{A}=\mathbf{I}$, τότε παίρνουμε μόνο μετατόπιση **(Translation)**

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{pmatrix} \delta x \\ \delta y \\ \delta z \end{pmatrix} \Rightarrow \begin{matrix} x' = x + \delta x \\ y' = y + \delta y \\ z' = z + \delta z \end{matrix}$$

- Αν $\boldsymbol{\delta}=(0,0,0)^T$ και $\mathbf{A}=\mathrm{diag}\{a,b,c\}$, τότε παίρνουμε τον μετασχηματισμό κλίμακας **(scaling)**

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & c \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \Rightarrow \begin{matrix} x' = ax \\ y' = by \\ z' = cz \end{matrix}$$

Α3. Ορθογώνιοι μετασχηματισμοί

Ορθογώνιος μετασχηματισμός ή **περιστροφή (rotation)** είναι ο αφινικός μετασχηματισμός με $\boldsymbol{\delta}=(0,0,0)^T$ και **A** ένας ορθογώνιος πίνακας

$$\mathbf{A}^T\mathbf{A} = \mathbf{A}\mathbf{A}^T = \mathbf{I} \qquad (\mathbf{A}^T \equiv \mathbf{A}^{-1})$$

Ένας ορθογώνιος μετασχηματισμός συστήματος συντεταγμένων διατηρεί
α) την αρχή των αξόνων ((0,0,0) fixed point)
β) την καθετότητα των αξόνων
γ) την κλίμακα

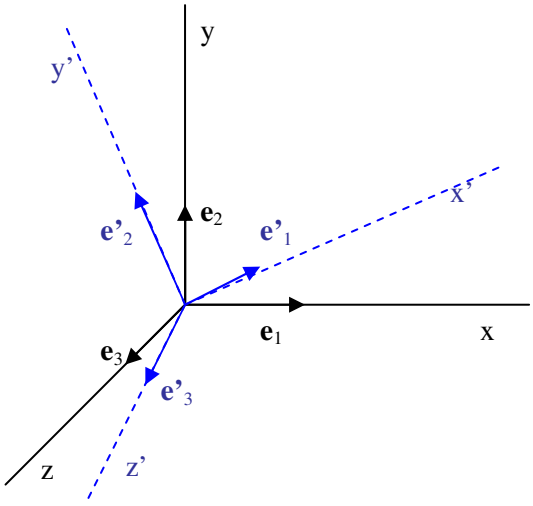
$$|\mathbf{e}'_i| = |\mathbf{e}_i| = 1, \quad \forall i = 1,2,3$$

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

$$a_{ij} = \mathbf{e}'_i \cdot \mathbf{e}_j = \cos(\varphi)$$

ή

$$a_{12} = \mathbf{e}'_1 \cdot \mathbf{e}_2 = \cos(x'\hat{O}y),$$
$$a_{23} = \mathbf{e}'_2 \cdot \mathbf{e}_3 = \cos(y'\hat{O}z),$$
$$\kappa\lambda\pi$$



Ο πίνακας **A**, στην περίπτωση περιστροφής συμβολίζεται με **R**.

<u>Περιστροφή γύρω από τους άξονες O*x*, O*y* και O*z* κατα γωνία θ</u>

$$\mathbf{R}_z(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad \mathbf{R}_x(\theta) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{pmatrix}, \quad \mathbf{R}_y(\theta) = \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix}$$

Αντίστροφος Μετασχηματισμός $\mathbf{R}^{-1}(\theta) = \mathbf{R}(-\theta)$ ή $\mathbf{R}(\theta)\mathbf{R}(-\theta) = \mathbf{R}(-\theta)\mathbf{R}(\theta) = \mathbf{I}$
Σύνθεση όμοιων Περιστροφών $\mathbf{R}(\theta)\mathbf{R}(\varphi) = \mathbf{R}(\theta+\varphi)$

Οι ορθογώνιοι μετασχηματισμοί μαζί με τις μετατοπίσεις ονομάζονται ***μετασχηματισμοί στερεού σώματος.***

Α4. Ειδικές περιπτώσεις Περιστροφών

Α) Γενική Περιστροφή γύρω από το (0,0,0): Σύνθεση περιστροφών γύρω από τους τρεις άξονες

$$\mathbf{R} = \mathbf{R}_x\mathbf{R}_y\mathbf{R}_z \quad (\neq \mathbf{R}_y\mathbf{R}_z\mathbf{R}_x \neq \mathbf{R}_y\mathbf{R}_x\mathbf{R}_z \neq \mathbf{R}_z\mathbf{R}_x\mathbf{R}_y)$$

Β) Περιστροφή γωνίας θ γύρω από τυχαίο άξονα (ευθεία) που περνάει από την αρχή των αξόνων (0,0,0)

Αν **v** διάνυσμα επί του άξονα περιστροφής τότε

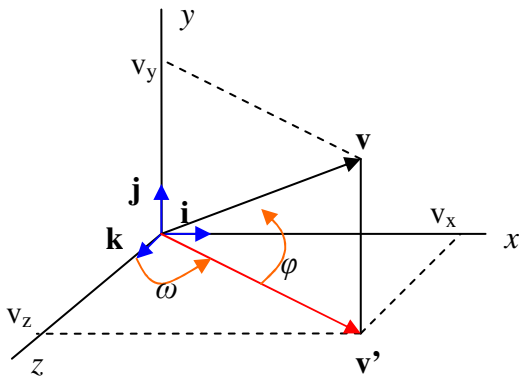
1. Περιστροφή $\mathbf{R}_y(\omega)$
2. Περιστροφή $\mathbf{R}_x(\varphi)$

>> $z' \,//\, \mathbf{v}$

3. Περιστροφή $\mathbf{R}_z(\theta)$

>>Απαλοιφή (αντιστροφή) των πρώτων περιστροφών

4. Περιστροφή $\mathbf{R}_x(-\varphi)$
5. Περιστροφή $\mathbf{R}_y(-\omega)$



Άρα συνολικά έχουμε $\mathbf{R}_\mathbf{v}(\theta) = \mathbf{R}_y(-\omega)\cdot\mathbf{R}_x(-\varphi)\cdot\mathbf{R}_z(\theta)\cdot\mathbf{R}_x(\varphi)\cdot\mathbf{R}_y(\omega)$

Αν $\mathbf{v} = (v_x, v_y, v_z)$ τότε $\mathbf{v}' = (v_x, 0, v_z)$ και

$$\omega = \arccos\left(\frac{v_z}{|\mathbf{v}'|}\right), \qquad \varphi = \arccos\left(\frac{v_x^2 + v_z^2}{|\mathbf{v}||\mathbf{v}'|}\right)$$

Γ) Περιστροφή γωνίας θ γύρω από τυχαίο άξονα (ευθεία) που περνάει από τυχόν σημείο P=(p_1,p_2,p_3)

1. Εφαρμόζουμε μια μετατόπιση **T** ώστε να φέρουμε το σημείο P=(p_1,p_2,p_3) στο (0,0,0).
2. Εφαρμόζουμε την περιστροφή $\mathbf{R}_\mathbf{v}$ γύρω από τον άξονα
3. Απαλείφουμε την αρχική μετατόπιση με την $\mathbf{T}^{-1}$.

Άρα συνολικά έχουμε $\mathbf{R}_\mathbf{P} = \mathbf{T}(p_1, p_2, p_3)\cdot\mathbf{R}_\mathbf{v}(\theta)\cdot\mathbf{T}(-p_1, -p_2, -p_3)$

Α4. Ομοιογενής Φορμαλισμός

Για να εφαρμόσουμε έναν αφινικό μετασχηματισμό απαιτείται ο ορισμός δύο πινάκων, του **A** και του **δ**, και δύο πράξεις, πολλαπλασιασμός και πρόσθεση πινάκων.

Ο ομοιογενής φορμαλισμός συνίσταται στο να δηλώσουμε έναν αφινικό μετασχηματισμό «αντικειμένων» με έναν πίνακα M

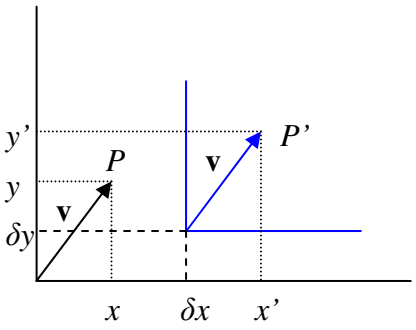
$$\mathbf{p' = Mp}$$

Έστω το διάνυσμα **v** και το σημείο P. Και τα δύο αντικείμενα ορίζονται με τον ίδιο τρόπο

$\mathbf{v} = (x, y)$

$P = (x, y)$

Σε μια μετατόπιση του συστήματος συντεταγμένων το διάνυσμα **v** παραμένει αναλλοίωτο, όμως το *P* μεταφέρεται στη θέση $P'=(x',y')\neq P(x,y)$.



Για να διαχωρίσουμε την παραπάνω ασάφεια ορίζουμε τα διανύσματα και τα σημεία (στον 3D χώρο) με τις τετράδες

$$\mathbf{v} = (x, y, z, 0), \qquad P = (x, y, z, 1)$$

Έτσι στην περίπτωση αυτή διανύσματα ή σημεία μπορούν να μετασχηματίζονται ως

$$\mathbf{p' = Mp} \quad \Rightarrow \quad \begin{pmatrix} x' \\ y' \\ z' \\ i' \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ i \end{pmatrix}, \qquad \begin{matrix} \mathbf{p} \equiv \mathbf{v} \text{ or } P \\ i = 0,1 \end{matrix}$$

Με τον παραπάνω φορμαλισμό μπορούμε να εκτελέσουμε κάθε αφινικό μετασχηματισμό. Πχ

Translation

$$T = \begin{pmatrix} 1 & 0 & 0 & \delta x \\ 0 & 1 & 0 & \delta y \\ 0 & 0 & 1 & \delta z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Scaling

$$S = \begin{pmatrix} a & 0 & 0 & 0 \\ 0 & b & 0 & 0 \\ 0 & 0 & c & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

x-Rotation

$$R_x = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- Ο 4×4 πίνακας **M** έχει 12 βαθμούς ελευθερίας
- Πάντα ένα σημείο μετασχηματίζεται σε σημείο και ένα διάνυσμα σε διάνυσμα.

Α5. Σύνθεση και αντιστροφή μετασχηματισμών

Έστω ότι εκτελούμε δύο μετασχηματισμούς T_1, T_2 με αυτή τη σειρά και με πίνακες **A** και **B**, αντίστοιχα

$$T_1 : (x, y, z) \xrightarrow{\mathbf{A}} (x', y', z')$$
$$T_2 : (x', y', z') \xrightarrow{\mathbf{B}} (x'', y'', z'')$$

Στον συνολικό μετασχηματισμό T αντιστοιχεί ο πίνακας

$$\mathbf{M = B \cdot A}$$

Εν' γένει δεν ισχύει η αντιμεταθετικότητα

$$\mathbf{B \cdot A \neq A \cdot B}$$

Προσοχή : οι πίνακες πολλαπλασιάζονται με την "ανάποδη" σειρά. Στον προγραμματισμό της OpenGL, οι μετασχηματισμοί εισάγονται επίσης με την ανάποδη σειρά.

Σε κάθε μετασχηματισμό με πίνακα **M** ορίζεται και ο αντίστροφος με πίνακα $\mathbf{M}^{-1}$. Έτσι $\mathbf{M}^{-1}\,\mathbf{M = I.}$

Αντίστροφοι Αφινικοί Μετασχηματισμοί

$$T = T(\delta x, \delta y, \delta z) \quad \Rightarrow \quad T^{-1} = T(-\delta x, -\delta y, -\delta z)$$
$$S = S(a, b, c) \quad \Rightarrow \quad S^{-1} = S(1/a, 1/b, 1/c)$$
$$R = R(\theta) \quad \Rightarrow \quad R^{-1} = R(-\theta)$$

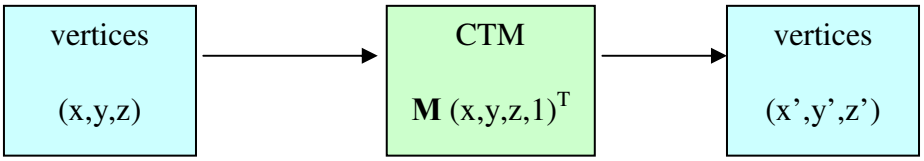
- Με παρόμοιο τρόπο ορίζεται η σύνθεση περισσοτέρων μετασχηματισμών και ισχύει η προσεταιριστικότητα, πχ

$$\mathbf{M = C \cdot (B \cdot A) = (C \cdot B) \cdot A = C \cdot B \cdot A}$$

Β. ΜΕΤΑΣΧΗΜΑΤΙΣΜΟΙ ΣΤΗΝ *OpenGL*

Β1. Ο «τρέχον» πίνακας μετασχηματισμού

- Η OpenGL πάντα εφαρμόζει έναν μετασχηματισμό που αντιστοιχεί σε έναν πίνακα **M** (4×4) και ο οποίος ονομάζεται *Current Transformation Matrix* (CTM).



- Η αρχική τιμή του πίνακα CTM είναι ο μοναδιαίος πίνακας **I**.
- Θέτουμε τον μοναδιαίο πίνακα **I** ως CTM με την συνάρτηση

```
glLoadIdentity( );
```

- Κάθε φορά που εφαρμόζουμε ένα μετασχηματισμό που περιγράφεται από ένα πίνακα **A**(4×4) τότε ο πίνακας **M** παίρνει την τιμή

$$\mathbf{M}' = \mathbf{M}\cdot\mathbf{A}$$

ή

$$M'(i,j) = \sum_{k=1}^{4} M(i,k)A(k,j)$$

δηλαδή ο μετασχηματισμός που εκτελείται πρώτος δηλώνεται τελευταίος.

Β1. Αφινικοί μετασχηματισμοί στην OpenGL

Η μετατόπιση, η κλίμακα και η περιστροφή μπορούν να εφαρμοστούν αυτόματα πάνω στον CTM και να τον τροποποιήσουν με την χρήση των παρακάτω εντολών

Α. Περιστροφή $\mathbf{R_V}$

```
glRotatef(angle, vx,vy,vz);
```

Β. Μετατόπιση **T**

```
glTranslatef( dx, dy, dz );
```

Γ. Κλίμακα **S**

```
glScalef( a, b, c );
```

βλ. κώδικα code60,61.cpp

B3. MATRIX MODES (Πίνακες Μετασχηματισμών)

Δύο από τα συστήματα συντεταγμένων που χρησιμοποιούνται στην OpenGL είναι οι συντεταγμένες των *Αντικειμένων-Μοντέλου* και οι συντεταγμένες *Σκηνικού-Προβολής*.

Model coordinates → World Coordinates

Για το κάθε σύστημα συντεταγμένων ορίζεται και ένας διαφορετικός πίνακας CTM. Η επιλογή του 1ου ή του 2ου συστήματος γίνεται με τις εντολές

```
glMatrixMode(GL_MODELVIEW);
glMatrixMode(GL_PROJECTION);
```

Αρχικά και τα δύο συστήματα έχουν ως CTM τον μοναδιαίο πίνακα.

Ο χώρος για την ορθογραφική προβολή του σκηνικού καθορίζεται με την

```
glOrtho(xmin, xmax, ymin,ymax, near, far)
```

Η παραπάνω εντολή εκτελεί κατάλληλο scaling και Translation στον CTM της κατάστασης PROJECTION.

$$q_w = \frac{q}{q_{\max} - q_{\min}} + \frac{q_{\max} - q_{\min}}{2}, \quad q \to x, y$$

όπου q_w η τιμή σε world coordinates.
* Για τον z-άξονα είναι $q_{max}-q_{min}=-$ (*far-near*)

Η `glOrtho` είναι εύχρηστη και εν γένει απαραίτητη. Όμως συνήθως θέλουμε να εκτελούμε μετασχηματισμούς στα αντικείμενα και όχι στο σκηνικό. Έτσι συνήθως χρησιμοποιείται στην αρχή του κώδικα η παρακάτω σειρά εντολών

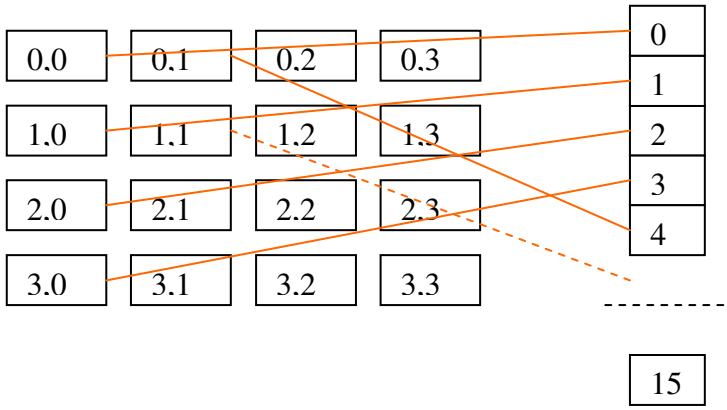
```
glMatrixMode(GL_PROJECTION);
glOrtho(xmin, xmax, ymin,ymax, near,far);
glMatrixMode(GL_MODELVIEW)
```

Στη συνέχεια ο CTM θα αναφέρεται στις συντεταγμένες του μοντέλου.

βλ. κώδικα code62.cpp

Β4. Διαχείριση του Πίνακα Μετασχηματισμού στην OpenGL

Από «τεχνικής» άποψης ως προς τους εσωτερικούς υπολογισμούς, ο πίνακας Μετασχηματισμού που χρησιμοποιεί η OpenGL είναι ένας μονοδιάστατος πίνακας με 16 στοιχεία και που είναι σε 1-1 αντιστοιχία με τον 4x4 πίνακα **M**



Έχοντας έναν δισδιάστατο πίνακα μετασχηματισμού **M**, πχ τον `myarray2D[4][4]`, μπορούμε να δημιουργήσουμε έναν 1D πίνακα, π.χ. τον `myarray[16]`, που μπορεί να χρησιμοποιηθεί ως πίνακας μετασχηματισμού, σύμφωνα με την παρακάτω διαδικασία:

```
void From2Dto1DMatrix()
{
     for(int i=0;i<4;i++) for(int j=0;j<4;j++)
                    myarray[4*j+i]=myarray2D[i][j];
}
```

Μπορούμε να φορτώσουμε τον `myarray[16]` και να τον <u>θέσουμε</u> ως CTM με την εντολή

```
glLoadMatrixf( myarray );
```

- Μπορούμε να περάσουμε τον τρέχοντα CTM στον πίνακα `myarray[16]` με την "`Get`" :

  ```
  glGetFloatv(GL_MODELVIEW_MATRIX,myarray);
  ```

- Αν δεν θέλουμε την εκτέλεση κάποιου μετασχηματισμού στα δοσμένα vertices τότε θέτουμε ως CTM τον ταυτοτικό πίνακα με την εντολή

  ```
  glLoadIdentity();
  ```

- Αν θέλουμε να συνθέσουμε* τον πίνακα `myarray[16]` με τον CTM τότε εκτελούμε την εντολή

```
glMultMatrixf( myarray );
```

$$M' = M \cdot A, \qquad A = myarray$$

- `glPushMatrix()` : Αποθηκεύει τον CTM στη μνήμη (stack)

- `glPopMatrix()` : Επανάκτηση του αποθηκευμένου πίνακα και χρήση του ως CTM.

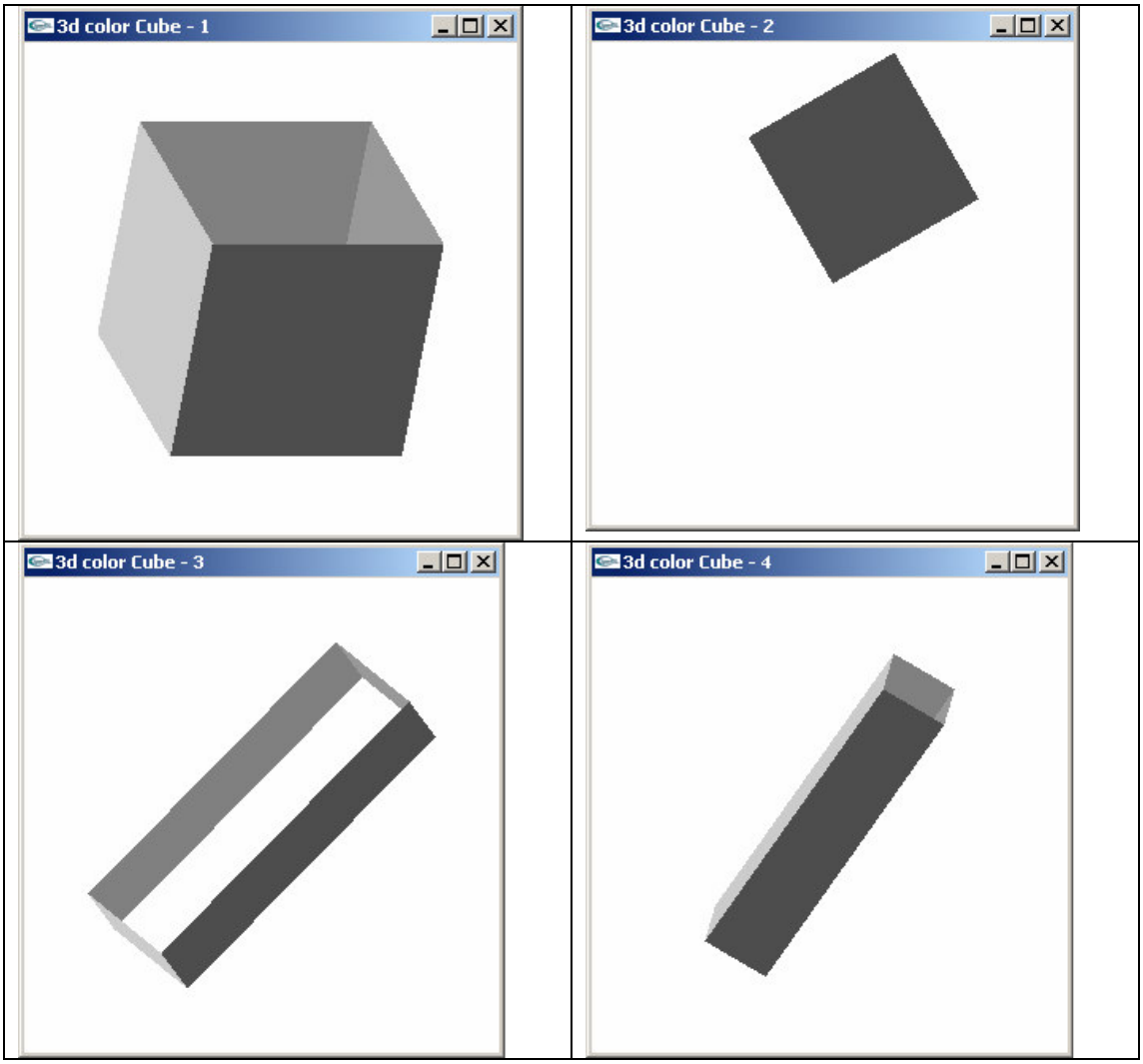
Σημείωση : η `glPushMatrix()` κάθε φορά που χρησιμοποιείται αποθηκεύει ένα αντίγραφο του CTM χωρίς να αντικαθιστά το προηγούμενο αποθηκευμένο αντίγραφο. Όταν η `glPopMatrix()` χρησιμοποιείται περισσότερες από μια φορά τότε ανακτώνται τα προηγούμενα αντίγραφα (το τελευταίο αποθηκευμένο ανακτάται πρώτο, το προτελευταίο αποθηκευμένο ανακτάται μετά από δύο χρήσεις της `glPopMatrix` κ.ο.κ.)

- code63.cpp (push-pop),
- code64.cpp (load, get, mult)
- code65.cpp multilevel push-pop

* και όχι να τον αντικαταστήσουμε όπως με την εντολή `glLoadMatrixf`

Ασκήσεις

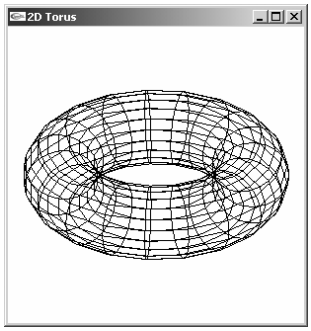
Άσκηση 1.
Χρησιμοποιείστε ως βάση τον κώδικα "Ex1cube.cpp" (αντικείμενο `mybox`) και
α) εκτελέστε κατάλληλους μετασχηματισμούς για να πάρετε τις παρακάτω παραστάσεις του αντικειμένου



β) Εφαρμόστε την συνάρτηση gluLookAt( ) σε ορθοκανονική προβολή για το 4^{o} πλαίσιο έτσι ώστε να βλέπουμε από τη μια ανοιχτή πλευράτου παραλληλεπιπέδου μέχρι την απέναντι επίσης ανοιχτή πλευρά

γ) Εφαρμόστε στο παραπάνω προβολή με προοπτική με τη χρήση της `glFrustum` και της `glPerspective`

Άσκηση 2.



α) Δημιουργήστε τον τόρο του διπλανού σχήματος χρησιμοποιώντας κατάλληλους μετασχηματισμούς περιστροφής για την δημιουργία των απαραίτητων vertices.
β) γεμίστε την επιφάνεια του τόρου με χρώμα

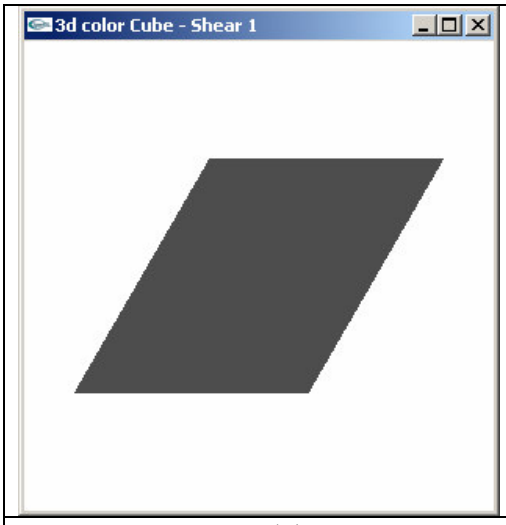
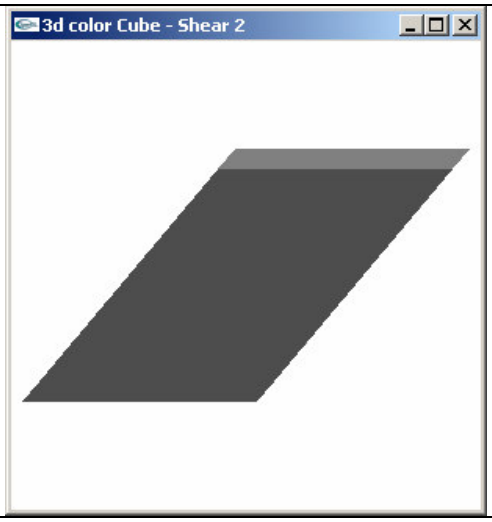
Άσκηση 3.
Εκτελέστε τον κώδικα "Ex3CubeScene.cpp" ορίζοντας `APPLY_SCENE_ROTATION=0` ή `APPLY_SCENE_ROTATION=1`.

α) Ποια η λειτουργία του block `APPLY_SCENE_ROTATION=true` μέσα στην συνάρτηση `myDisplay()`.
β) Όταν προκαλείται "Redisplay" του παραθύρου ο μετασχηματισμός επαναλαμβάνεται και αλλοιώνει το σχήμα. Πως μπορεί να λυθεί το πρόβλημα αυτό?

Άσκηση 4.
Η διατμητική μεταβολή (ή ροή) κατά τον άξονα x και για γωνία θ δίνεται από τον πίνακα μετασχηματισμού **M**[4][4]

$$\mathbf{M} = \begin{pmatrix} 1 & \cot\theta & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Εφαρμόστε τον παραπάνω μετασχηματισμό για να πάρετε το σχήμα (α) καθώς και μια επιπλέον μικρή περιστροφή γύρω από τον άξονα Ox για να πάρετε το σχήμα (β).

	
(α)	(β)

Άσκηση 5.
Θεωρήστε ένα τετράγωνο στο επίπεδο x-y με κέντρο το (0,0). Επίσης θεωρούμε μικρές γωνίες περιστροφής θ και εκφράζουμε τον μετασχηματισμό $\mathbf{R}_z(\theta)$ σε όρους πρώτης τάξης έτσι ώστε

$$\cos\theta \approx 1, \quad \sin\theta \approx \theta$$

και τον συμβολίζουμε ως $\mathbf{R}_{za}(\theta)$.
Έστω θ=1^o. Εφαρμόστε τον μετασχηματισμό $\mathbf{R}_z(90^o)$ και τον μετασχηματισμό $R_{za}^{90} = R_{za} \cdot R_{za} \cdot ..... \cdot R_{za}$ (90 φορες). Παρατηρείτε διαφορά στα δύο αποτελέσματα;

Άσκηση 6.
Σχεδιάστε έναν κώνο χρησιμοποιώντας
α) έναν πίνακα `C[N][2]` που περιέχει τις συντεταγμένες *x,y* από N σημεία της κυκλική βάσης.
β) Έχοντας τη κυκλική βάση στο επίπεδο *x-y*, θεωρείστε ως κορυφή το σημείο (0,0,1) και εφαρμόστε τη σχεδίαση `GL_TRIANGLE_FAN`. Επίσης χρησιμοποιείστε την αλλαγή χρώματος με την ιδιότητα της παρεμβολής
γ) Προβάλλετε τον κώνο κατάλληλα με τη χρήση περιστροφών.