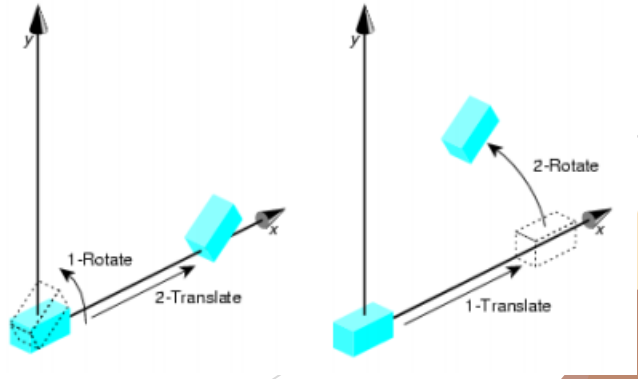
# EPL 426

## Lab 4 – OpenGL - Μετασχηματισμοί

Andreas Andreou

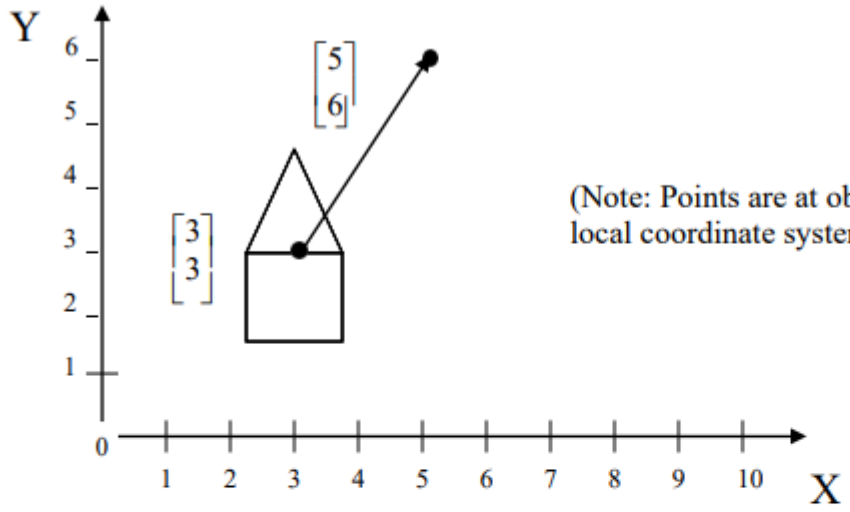
# Transformations in OpenGL

- Όπως ισχύει και στη θεωρία, έτσι και στην OpenGL εφαρμόζουμε τους μετασχηματισμούς για να αλλάξουμε τη θέση και το σχήμα των αντικειμένων στο χώρο
- Βασικοί μετασχηματισμοί
  - glTranslated ή glTrasnslatef
  - glRotated ή glRotatef
  - glScaled ή glScalef
- Επίσης, μπορούμε με τη χρήση του glLoadMatrixd ή glLoadMatrixf να εισάξουμε χειροκίνητα ένα μετασχηματισμό



# 2D Translation

dx = 2
dy = 3




(Note: Points are at object's local coordinate system origin)

$v' = v + t$ where $v = \begin{bmatrix} x \\ y \end{bmatrix}$, $v' = \begin{bmatrix} x' \\ y' \end{bmatrix}$, $t = \begin{bmatrix} dx \\ dy \end{bmatrix}$

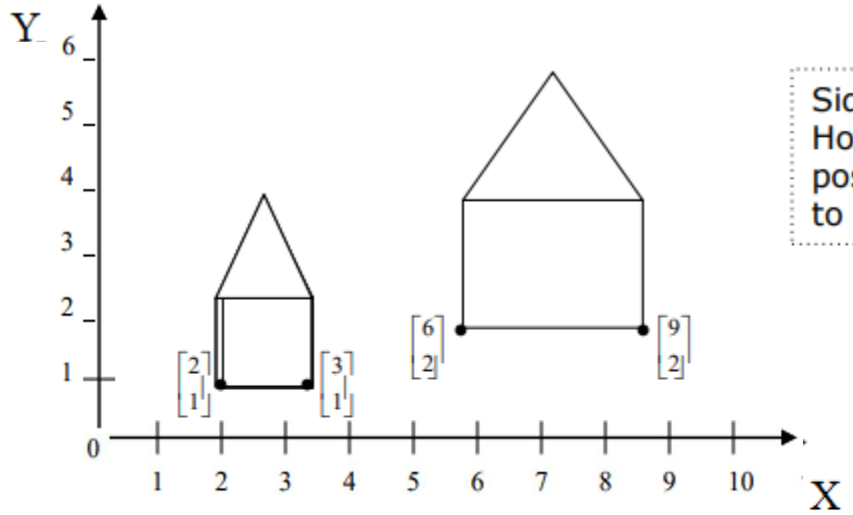
and $x' = x + dx$

$y' = y + dy$

# 2D Scaling

$s_x = 3$

$s_y = 2$

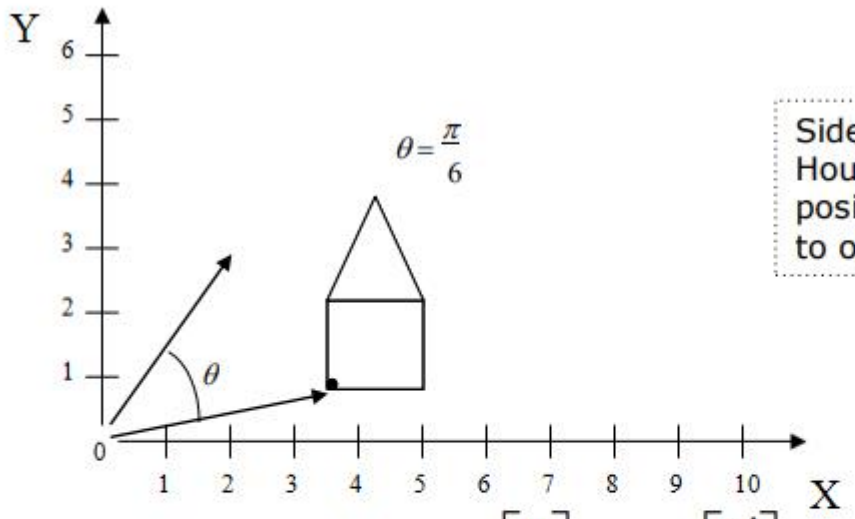



Side effect: House shifts position relative to origin

$\boldsymbol{v'} = \boldsymbol{Sv}$ where $v = \begin{bmatrix} x \\ y \end{bmatrix}$, $v' = \begin{bmatrix} x' \\ y' \end{bmatrix}$

and $S = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix}$ $\quad x' = s_x x$ $\quad y' = s_y y$

# 2D Rotation




Side effect: House shifts position relative to origin

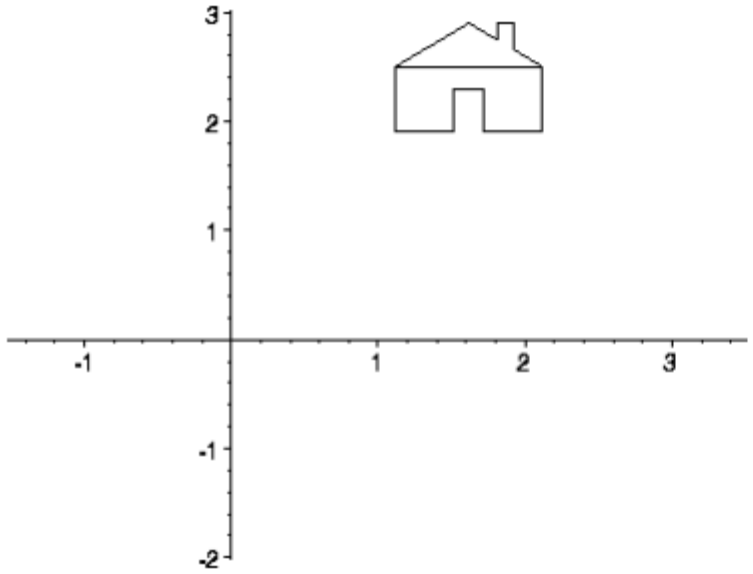
$\boldsymbol{v'} = \boldsymbol{R_\theta}\, \boldsymbol{v}$ where $v = \begin{bmatrix} x \\ y \end{bmatrix}$, $v' = \begin{bmatrix} x' \\ y' \end{bmatrix}$

and $x' = x\cos\Theta - y\sin\Theta$

$y' = x\sin\Theta + y\cos\Theta$

$R_\theta = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$

# 3D Translation OpenGL



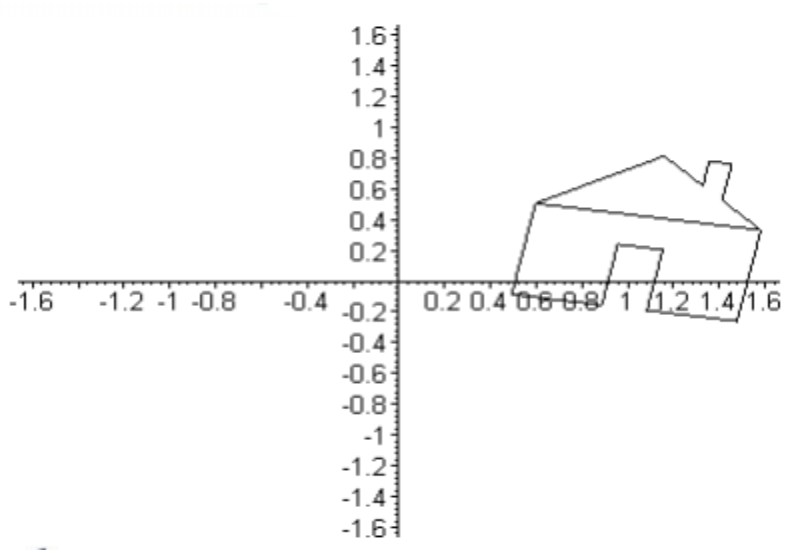

Translation $\begin{bmatrix} 1 & 0 & 0 & dx \\ 0 & 1 & 0 & dy \\ 0 & 0 & 1 & dz \\ 0 & 0 & 0 & 1 \end{bmatrix}$

- void **glTranslated**(GLdouble *x*, GLdouble *y*, GLdouble *z*);
- void **glTranslatef**(GLfloat *x*, GLfloat *y*, GLfloat *z*);

# 3D Translation OpenGL

- Δοκιμάστε στο παράδειγμα του προηγούμενου εργαστηρίου να κάνετε translate τα τρίγωνα
- Στην αρχή δοκιμάστε να τα κάνετε translate όλα μαζί
- Δοκιμάστε να εφαρμόσετε το translation σε διαφορετικούς άξονες
- Για να μπορέσετε να κάνετε ξεχωριστά translate τα τρίγωνα πρέπει να δημιουργήσετε ξεχωριστά glBegin/glEnd structures που θα περιέχει το κάθε ένα, ένα τρίγωνο
- Έτσι μπορείτε να εφαρμόσετε την glTranslatef σε κάθε τρίγωνο ξεχωριστά
- Δοκιμάστε να εφαρμόσετε δύο glTraslatef, μια για κάθε τρίγωνο, να δείτε τι συμβαίνει

# 3D Rotation OpenGL




Rotation about X-axis

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotation about Y-axis

$$\begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotation about Z-axis

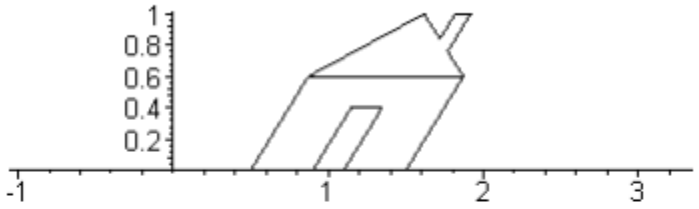
$$\begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- void **glRotated**(GLdouble *angle*, GLdouble *x*, GL double *y*, GLdouble *z*);
- void **glRotatef**(GLfloat *angle*, GLfloat *x*, GLfloat *y*, GLfloat *z*);

# 3D Rotation OpenGL

- Στο παράδειγμα που έχετε δοκιμάστε τις συναρτήσεις περιστροφής
- Δημιουργήστε ένα τετράγωνο στο κέντρο 0,0 και έπειτα περιστρέψτε το
- Δημιουργήστε τώρα ένα άλλο τετράγωνο μετατοπισμένο κατά 5 μονάδες στον άξονα y και περιστρέψτε το με την ίδια γωνιά με το προηγούμενο
- Tip: για να δώσετε συνεχόμενη κίνηση στο πρόγραμμά σας κάντε comment το glLoadIdentity(). Γιατί συμβαίνει αυτό;

# 3D Scaling OpenGL




Scaling $\begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

- void **glScaled**(GLdouble *x*, GLdouble *y*, GLdouble *z*);
- void **glScalef**(GLfloat *x*, GLfloat *y*, GLfloat *z*);

# 3D Scaling OpenGL

- Ομοίως με προηγουμένως, δοκιμάστε να εφαρμόσετε σε κάποιο μοντέλο το transformation του scaling.
- Δοκιμάστε να τα συνδυάσετε τώρα όλα μαζί με κάποια σειρά.
- ΠΡΟΣΟΧΗ! Στην OpenGL η σειρά των μετασχηματισμών είναι ανάποδη από ότι γράφεται. Ξεκινά με τον πιο κάτω μετασχηματισμό και μετά πάει προς τα πάνω

# Matrix Multiplication

$$c_{11} = a_{11}b_{11} + a_{12}b_{21} + a_{13}b_{31} + a_{14}b_{41}$$

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \\ b_{41} & b_{42} & b_{43} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \end{bmatrix}$$

2 x 4 4 x 3 2 x 3

$$c_{22} = a_{21}b_{12} + a_{22}b_{22} + a_{23}b_{32} + a_{24}b_{42}$$

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \\ b_{41} & b_{42} & b_{43} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \times \begin{bmatrix} 10 & 11 \\ 20 & 21 \\ 30 & 31 \end{bmatrix}$$

$$= \begin{bmatrix} 1x10 + 2x20 + 3x30 & 1x11 + 2x21 + 3x31 \\ 4x10 + 5x20 + 6x30 & 4x11 + 5x21 + 6x31 \end{bmatrix}$$

$$= \begin{bmatrix} 10+40+90 & 11+42+93 \\ 40+100+180 & 44+105+186 \end{bmatrix} = \begin{bmatrix} 140 & 146 \\ 320 & 335 \end{bmatrix}$$

# Basic Matrix

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Identity Matrix

$$\begin{pmatrix} 1 & 0 & 0 & tx \\ 0 & 1 & 0 & ty \\ 0 & 0 & 1 & tz \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

glTranslatef(tx,ty,tz)

$$\begin{pmatrix} sx & 0 & 0 & 0 \\ 0 & sy & 0 & 0 \\ 0 & 0 & sz & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

glScalef(sx,sy,sz)

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(d) & -\sin(d) & 0 \\ 0 & \sin(d) & \cos(d) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

glRotatef(d,1,0,0)

$$\begin{pmatrix} \cos(d) & 0 & \sin(d) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(d) & 0 & \cos(d) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

glRotatef(d,0,1,0)

$$\begin{pmatrix} \cos(d) & -\sin(d) & 0 & 0 \\ \sin(d) & \cos(d) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

glRotatef(d,0,0,1)

X-Rotation in 3D

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi & 0 \\ 0 & \sin\phi & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Z-Rotation in 3D

$$\begin{bmatrix} \cos\phi & -\sin\phi & 0 & 0 \\ \sin\phi & \cos\phi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Scale in 3D

$$\begin{bmatrix} Sx & 0 & 0 & 0 \\ 0 & Sy & 0 & 0 \\ 0 & 0 & Sz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(4x4)*(4x1) = (4x1)

Y-Rotation in 3D

$$\begin{bmatrix} \cos\phi & 0 & \sin\phi & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\phi & 0 & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Translation in 3D

$$\begin{bmatrix} 1 & 0 & 0 & Tx \\ 0 & 1 & 0 & Ty \\ 0 & 0 & 1 & Tz \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Matrix Multiplication

$$\begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ z' \\ q \end{bmatrix}$$

# glLoadMatrix

- Η glLoadMatrixd ή glLoadMatrixf αντικαθιστά τον τρέχον επιλεγμένο πίνακα με αυτόν που θέτουμε εμείς ως όρισμα
  - void glLoadMatrixd(const GLdouble * m);
  - void glLoadMatrixf(const GLfloat * m);
- Με απλά λόγια μπορούμε να εισάγουμε ένα χειροκίνητο μετασχηματισμό αφού εισάγουμε ένα 4x4 πίνακα

# glLoadMatrix

- Προσοχή! Η OpenGL επεξεργάζεται τους πίνακες διαφορετικά από ότι στη θεωρία

```
0 4 8 c
1 5 9 d
2 6 a e
3 7 b f
```

```
float f[16] =
{
  1,0,0,0,
  0,1,0,0,
  0,0,1,0,
  tx,ty,tz,1
};
```

# glLoadMatrix

- Δηλώστε στην αρχή του προγράμματος ως μια global μεταβλητή έναν πίνακα float16 θέσεων ως ακολούθως:

```
float f[16] =
{
  1,0,0,0,
  0,1,0,0,
  0,0,1,0,
  tx,ty,tz,1
};
```

- Παράξτε ένα translation χρησιμοποιώντας την glTranslated
- Ακολούθως παράξτε το ίδιο translation μόνο με την glLoadMatrix

# Push and pop Matrices

- Επιστρέφοντας και πάλι στο παράδειγμα της προηγούμενης φοράς, αντικαταστήστε τον κώδικα παραγωγής τριγώνων με τον ακόλουθο:

```
glPushMatrix();
glutSolidCube(1);
glTranslatef(2,0,0);
glutSolidCube(1);
glTranslatef(2,0,0);
glutSolidCube(1);
glTranslatef(0,2,0);
glutSolidCube(1);
```

# Push and pop Matrices

- Στη συνέχεια αντικαταστήστε το πιο πάνω με το πιο κάτω:

```
glPushMatrix();
glutSolidCube(1);
glTranslatef(2,0,0);
glutSolidCube(1);
glTranslatef(2,0,0);
glutSolidCube(1);
glPopMatrix();
glTranslatef(0,2,0);
glutSolidCube(1);
```

# Push and pop Matrices

- Τέλος, αντικαταστήστε το πιο πάνω με το πιο κάτω

```
glutSolidCube(1);
glTranslatef(2,0,0);
glutSolidCube(1);
glPushMatrix();
glTranslatef(2,0,0);
glutSolidCube(1);
glPopMatrix();
glTranslatef(0,2,0);
glutSolidCube(1);
```

# Practice

- Δοκιμάστε με αυτά που μάθαμε σήμερα να παράξετε μια μορφή του γνωστού stickman.
- Χρησιμοποιήστε το glutSolidCube και το glutSolidSphere όπως και τους μετασχηματισμούς και το push & pop matrix