2ο Εργαστήριο Γραφικά & Εικονική Πραγματικότητα OpenGL

- Βασικά Σχήματα
- Χρώμα και Αποχρώσεις (Shading)
- 2Δ Μετασχηματισμοί
- Κίνηση (Animation)

1ο βήμα

Η **glRect** φτιάχνει ορθογώνια. Παίρνει σαν ορίσματα την κάτω αριστερή και πάνω δεξιά γωνία του ορθογωνίου. Για οποιοδήποτε πιο πολύπλοκο σχήμα πρέπει να χρησιμοποιηθούν οι **glBegin** και **glEnd**. Βάλτε σε σχόλια το κομμάτι 00.

Οι **glBegin** και **glEnd** οριοθετούν vertices που καθορίζουν ένα σχήμα. Η **glBegin** δέχεται ένα όρισμα που διευκρινίζει με ποιόν από δέκα τρόπους θα συνδυαστούν τα vertices (Εικόνα 1). Οι τρόποι αυτοί είναι οι εξής:

GL_POINTS

Μεταχειρίζεται κάθε vertex ως ανεξάρτητο σημείο. Η `glPointSize`,πριν τη `glBegin`, μας καθορίζει το μέγεθός τους.

GL_LINES

Χρησιμοποιεί τα vertices κατά ζεύγη για να δημιουργήσει μη συνδεδεμένα ευθύγραμμα τμήματα. Οπότε από Ν σημεία ορίζονται Ν/2 τμήματα.

GL_LINE_STRIP

Ζωγραφίζει μια συνδεδεμένη ομάδα ευθύγραμμων τμημάτων από το πρώτο vertex ως το τελευταίο, ώστε να δημιουργήσει μία ανοικτή πολυγωνική γραμμή. Με Ν σημεία, ζωγραφίζονται Ν-1 τμήματα.

GL_LINE_LOOP

Ζωγραφίζει μια συνδεδεμένη ομάδα ευθύγραμμων τμημάτων από το πρώτο vertex ως το τελευταίο, και έπειτα πίσω στο πρώτο, ώστε να δημιουργήσει μια κλειστή πολυγωνική γραμμή. Με Ν σημεία, ζωγραφίζονται Ν τμήματα

GL_TRIANGLES

Χρησιμοποιεί τα vertices κατά τριάδες ώστε να δημιουργήσει ανεξάρτητα τρίγωνα, χωρίς κοινές ακμές ή κορυφές. Δηλαδή από Ν σημεία ζωγραφίζονται Ν/3 τρίγωνα.

GL_TRIANGLE_STRIP

Χρησιμοποιεί τα vertices κατά τριάδες ώστε να δημιουργήσει μία ταινία από τρίγωνα, με κοινές ακμές. Δηλαδή από Ν σημεία ζωγραφίζονται Ν-2 τρίγωνα.

GL_TRIANGLE_FAN

Χρησιμοποιεί τα vertices κατά τριάδες ώστε να δημιουργήσει διαδοχικά τρίγωνα, με κοινή κορυφή. Το πρώτο vertex χρησιμοποιείται ως η κοινή κορυφή και τα υπόλοιπα ανά δύο ορίζουν τις βάσεις τσν τριγώνων. Από Ν σημεία ζωγραφίζονται Ν-2 τρίγωνα.

GL_QUADS

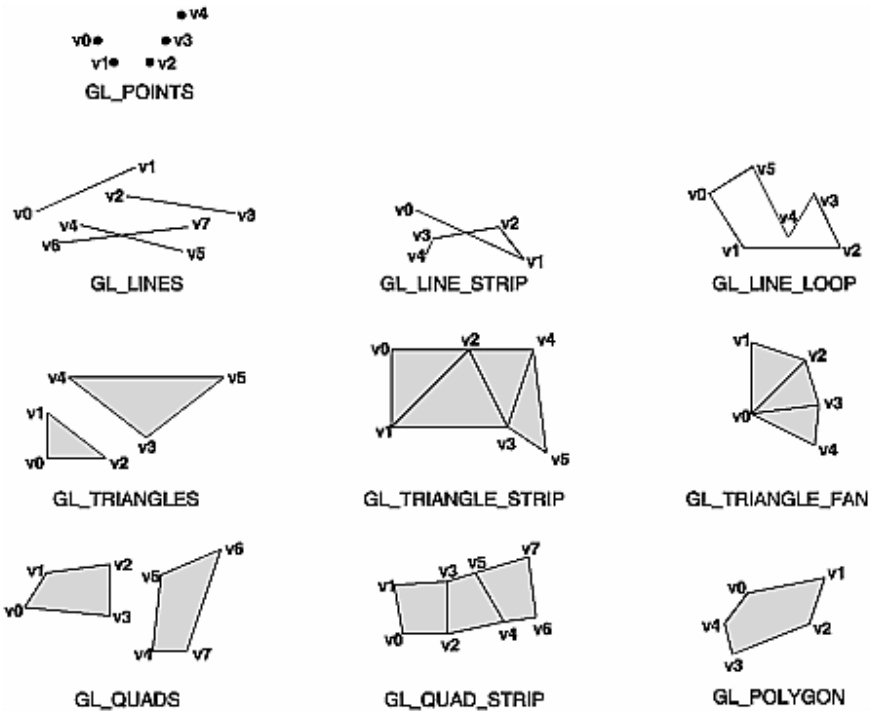
Χρησιμοποιεί τα vertices κατά τετράδες ώστε να δημιουργήσει ανεξάρτητα τετράπλευρα, χωρίς κοινές ακμές ή κορυφές. Δηλαδή από Ν σημεία ζωγραφίζονται Ν/4 τετράπλευρα.

GL_QUAD_STRIP

Χρησιμοποιεί τα vertices κατά τετράδες ώστε να δημιουργήσει μία ταινία από τετράπλευρα, με κοινές ακμές. Από Ν σημεία ζωγραφίζονται (Ν-2)/2 τετράπλευρα.

GL_POLYGON

Χρησιμοποιεί τα vertices ώστε να δημιουργήσει ένα ενιαίο πολύγωνο.



Εικόνα 1: Οι 10 τρόποι σύνδεσης των vertices για την δημιουργία βασικών σχημάτων

Αφαιρέστε τα σχόλια από το κομμάτι (01a) και πειραματισθείτε με τις διάφορες παραμέτρους, αφήνοντας μόνο ένα **glBegin** σε κάθε χρήση, για να δείτε τα `GL_POINTS`, `GL_LINES`, `GL_LINE_STRIP`, `GL_LINE_LOOP`, `GL_TRIANGLES`, και `GL_TRIANGLE_FAN`.

Αφαιρέστε τα σχόλια από το κομμάτι (01b) και πειραματισθείτε με τις διάφορες παραμέτρους, αφήνοντας μόνο ένα **glBegin** σε κάθε χρήση, για να δείτε χρωματισμένα τα `GL_TRIANGLES`, και `GL_TRIANGLE_STRIP`.

Παρατηρήστε τη θέση καθορισμού του χρώματος `glColor3f`.

Ο καθορισμός του χρώματος ενός κλειστού σχήματος, καθορίζεται από την τιμή του χρώματος της τελευταίας κορυφής.

Η OpenGL σας παρέχει δύο τρόπους καθορισμού του χρώματος, μέσω της συνάρτησης `glShadeModel`, η οποία δέχεται δύο τιμές, `GL_FLAT`, για ενιαίο χρώμα, και `GL_SMOOTH`, για ανάμειξη των χρωμάτων των κορυφών (shading) μέσω interpolation, σύμφωνα με το μοντέλο Gouraud.

Αφαιρέστε τα σχόλια από το κομμάτι (01c), για να δείτε τα αποτελέσματα χρωματισμού ενός τριγώνου. Στην `Setup()`, σχολιάστε την `glShadeModel (GL_FLAT)`, για να απενεργοποιήσετε τον ενιαίο χρωματισμό, και ξεσχολιάστε την `glShadeModel (GL_SMOOTH)`, για να ενεργοποιήσετε τον ανάμεικτο χρωματισμό.

2ο βήμα

Τώρα θα δούμε τους μετασχηματισμούς των αντικειμένων.
Η glTranslatef(x,y,z) μετατοπίζει το αντικείμενο κατά το διάνυσμα x,y,z.
Η glRotatef(θ,x,y,z) περιστρέφει το αντικείμενο κατά θ μοίρες γύρω από τον άξονα που καθορίζεται από διάνυσμα x,y,z.
Η glScalef(sx,sy,sz) κλιμακώνει το αντικείμενο κατά sx,sy,sz.

Να θυμάστε ότι η OpenGL είναι μια μηχανή καταστάσεων. Κάθε μετασχηματισμός θα ισχύει για όλα τα μετέπειτα αντικείμενα, μέχρι να εφαρμοστεί ο αντίστροφός του ή να φορτώσουμε τον μοναδιαίο πίνακα.

Για να δείτε την επίδραση των μετασχηματισμών ξεσχολιάστε τα κομμάτια κώδικα (02a)-(02c) και «παίξτε» με τις τιμές:

```
glTranslatef(30.0, 0.0, 0.0);
glRotatef(30.0, 0.0, 0.0, 1.0);
glScalef(2.0, 1.0, 1.0);
```

Επίσης είναι πολύ σημαντικό να καταλάβετε τη σειρά εκτέλεσης των μετασχηματισμών. Η σειρά εκτέλεσης των μετασχηματισμών στην OpenGL μπορεί να ερμηνευτεί με δύο τρόπους:
Α) Σύμφωνα με την πρώτη ερμηνεία, οι μετασχηματισμοί εκτελούνται σύμφωνα με την σειρά που καλούνται στον κώδικα και *επιδρούν στο σύστημα αναφοράς* (θεώρηση μεταβαλλόμενου συστήματος αναφοράς). Δηλ. πρώτα η `glTranslatef` μετατοπίζει το σύστημα αναφοράς κατά τον άξονα-x, μετά η `glRotatef` το περιστρέφει κατά 30ο γύρω απο τον άξονα-z, και τέλος η `glScalef` μεγεθύνει τον άξονα-x 2 φορές. Οπότε το τετράπλευρο με την κλήση της `glRectf` τοποθετείται

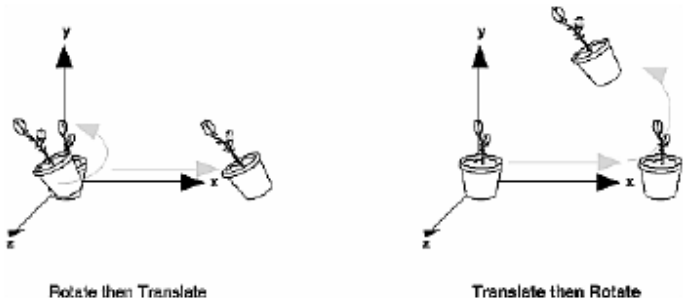
στο νέο σύστημα αναφοράς στην κατάλληλη θέση και με κατάλληλο μέγεθος. Κάθε αντικείμενο απο κεί και πέρα τοποθετείται στο νέο σύστημα αναφοράς.

Β) Σύμφωνα με την δεύτερη ερμηνεία, οι μετασχηματισμοί εκτελούνται σύμφωνα με την αντίστροφη σειρά και επιδρούν στα αντικείμενα που έπονται, *χωρίς να μεταβάλεται το σύστημα αναφορας* (θεώρηση αναλλοίωτου συστήματος αναφοράς). Δηλ. το τετράπλευρο τοποθετείται με την κλήση της `glRectf` στο παγκόσμιο σύστημα αναφοράς, πρώτα η `glScalef` διπλασιάζει τη x-διάσταση του αντικειμένου, μετά η `glRotatef` περιστρέφει το αντικείμενο κατά 30^{o} γύρω απο τον άξονα-z και τέλος η `glTranslatef` το μετατοπίζει κατά τον άξονα-x. Η δεύτερη ερμηνεία στηρίζεται στο γεγονός οτι η OpenGL πολλαπλασιάζει τους πίνακες από δεξιά (post-multiplication) για να βρεί το αποτέλεσμα του μετασχηματισμού, και μετά τον εφαρμόζει σε κάθε vertex. Επομένως ισχύει:

$$(T * R * S) * v = T(R(S(v)))$$

και επιδρά πρώτα ο μετασχηματισμός που είναι πλησιέστερα στην κλήση του αντικειμένου, και μετά οι υπόλοιποι με αντίστροφη σειρά.

Οι δύο θεωρήσεις-ερμηνείες *είναι ισοδύναμες* και μπορείτε να υιοθετήσετε όποια σας βολεύει, αρκεί να ξέρετε τι κάνετε.



Εικόνα 2: Περιστροφή και μεταφορά (κλήση `glTranslatef` και μετά `glRotatef`) vs. μεταφοράς και περιστροφής (κλήση `glRotatef` και μετά `glTranslatef`)

Πολύ σημαντικό επίσης είναι να ξέρετε οτι δεν ισχύει η μεταθετικότητα στους μετασχηματισμούς (εκτός αν είναι μόνο μεταφοράς). Αυτό συμβαίνει γιατί γενικά δεν ισχύει η μεταθετική ιδιότητα στον πολλαπλασιασμό πινάκων, ενώ ισχύει η μεταθετική ιδιότητα στην πρόσθεση διανυσμάτων μετατόπισης. Στα κομμάτια

κώδικα (02d-*) φαίνεται η διαφορά στην εφαρμογή πρώτα ενός μετασχηματισμού περιστροφής και μιας μεταφοράς και στην εφαρμογή τους με την αντίστροφη σειρά. Μπορείτε να το δείτε και στην Εικόνα 2.

Τέλος στα κομμάτια κώδικα (02e-*) μπορείτε να δείτε ότι δύο μετασχηματισμοί μεταφοράς, είναι ισοδύναμοι με έναν όπου η μεταφορά σε κάθε άξονα είναι το άθροισμα των δύο άλλων.

3^{ο} βήμα

Παρατηρούμε μια νέα συνάρτηση (που είναι δηλωμένη στο .h αρχείο), την idle.

Στο αρχείο main.cpp , στη main χρησιμοποιούμε την εντολή:

```
glutIdleFunc(Idle);
```

Με την εντολή αυτή υποδεικνύουμε στο glut ποια συνάρτηση (που δεν παίρνει ορίσματα και δεν επιστρέφει τιμή) θα χρησιμοποιείται όταν το glut δεν «ασχολείται» με κάτι άλλο (κατάσταση «ηρεμίας» - idle), δηλαδή όταν δεν κάνει render τα γραφικά μας και δεν δέχεται είσοδο από το πληκτρολόγιο ή το ποντίκι. Η κατάσταση ηρεμίας έρχεται όταν τελειώνει η επεξεργασία ενός καρέ, οπότε η `Idle()` μπορεί να χρησιμοποιηθεί για να δημιουργήσουμε κίνηση. Εδώ ορίζουμε τη συνάρτηση αυτή ‘Idle’, αλλά μπορείτε να την ονομάσετε όπως θέλετε. Ξεσχολιάστε τα τμήματα κώδικα (03) στην `main()` και `render()` και τρέξτε το πρόγραμμα.

Ουσιαστικά χρησιμοποιούμε την Idle για να δώσουμε κίνηση (animation) στην σκηνή μας. Ανανεώνουμε τις μεταβλητές που επηρεάζουν την κίνηση στην Idle και τις χρησιμοποιούμε μέσα σε μετασχηματισμούς στην Render.