ΤΕΧΝΙΚΟΣ ΕΦΑΡΜΟΓΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΜΕ ΠΟΛΥΜΕΣΑ (ΠΟΛΥΜΕΣΑ/WEB DESIGNER – DEVELOPER/VIDEO GAMES)

ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ IV (OPENGL)

ΑΠΑΝΤΗΣΕΙΣ ΣΤΙΣ ΕΡΩΤΗΣΕΙΣ ΠΙΣΤΟΠΟΙΗΣΗΣ

ΟΜΑΔΑ Α. ΓΕΝΙΚΕΣ ΕΡΩΤΗΣΕΙΣ

210. Να γραφεί συνάρτηση σε OpenGL που να σχεδιάζει ένα επίπεδο (plane).

```
void draw_plane()
{
    glPushMatrix();
    glScalef(1, 0, 1);
    glBegin(GL_QUADS);
        glVertex3f(-1, 0, 1);
        glVertex3f(1, 0, 1);
        glVertex3f(1, 0, -1);
        glVertex3f(-1, 0, -1);
    glEnd();
    glPopMatrix();
}
```

211. Σε τι οφείλουμε το γεγονός ότι η OpenGL μπορεί να δείξει από 256 έως δισεκατομμύρια χρώματα χωρίς να χρειαστεί να ξαναγράψουμε ή να ξανακάνουμε compile τον κώδικα;

Στο γεγονός ότι η OpenGL χρησιμοποιεί ενταμιευτές (buffers) για την αναπαράσταση των χρωμάτων.

212. Η OpenGL χρησιμοποιεί μια απλή, βασική, μορφή ονοματολογίας για τις εντολές της. Αναφέρατε ποια είναι αυτή και παραθέστε παράδειγμα.

Στην OpenGL, για ορισμένες εντολές ορίζονται πολλαπλές παραλλαγές, ανάλογα με:

- τον τύπο των ορισμάτων που δέχονται (π.χ. ακέραιοι ή πραγμάτικοί),
- τις διαστάσεις του χώρου (π.χ. σχεδίαση σε δύο ή τρείς διαστάσεις)
- τον αριθμό των συνιστωσών των χρωματικών τιμών (π.χ. τρεις στο μοντέλο RGB, τέσσερις στο μοντέλο RGB με μίξη χρωμάτων)
- τον τρόπο με τον οποίο επιλέγουμε να περάσουμε τις παραμέτρους στην εντολή (πέρασμα αριθμητικών τιμών (call by value) ή πέρασμα διανυσμάτων υπό τη μορφή μητρώων (call by reference)).

Ανάλογα λοιπόν με το συνδυασμό των παραπάνω παραμέτρων, ορισμένες εντολές της OpenGL παρουσιάζουν παράλλαγές. Τα ονόματα των παραλλαγών μιας εντολής καθορίζονται βάσει της εξής σύμβασης:

Όνομα εντολής + Διάσταση χώρου/Πλήθος χρωματικών συνιστωσών + Πρωτογενής τύπος δεδομένων ορισμάτων + Τρόπος κλήσεως ορισμάτων

Στο τέλος δηλαδή του ονόματος της εντολής προσθέτονται επιθήματα που καθορίζουν πλήρως τις παραπάνω παραμέτρους.

Ως παράδειγμα θα επιλέξουμε την εντολη glVertex, με την οποία ορίζουμε τις συντεταγμένες ενός σημείου στο επίπεδο ή στον τρισδιάστατο χώρο. Το επίθημα καθορίζεται από τον τύπο δεδομένων, σύμφωνα με τον Πίνακα 1. Έτσι λχ για τον καθορισμό ενός σημείου στο διδιάστατο χώρο που οι συντεταγμένες του δίνονται υπό τη

μορφή πραγματικών αριθμών απλής ακρίβειας (GLfloat) με κλήση τιμής (call by value) η αντίστοιχη εντολή έχει τη μορφή:

glVertex2f (GLfloat x, GLfloat y);

ενώ η αντίστοιχη συνάρτηση στον τρισδιάστατο χώρο με ορίσματα ακεραίων θα έχει τη μορφή

glVertex3i(GLint x, GLint y, GLint z);

213. Αναφέρατε τι ονομάζουμε GLUT στην OpenGL. Για ποιο λόγο δημιουργήθηκε και ποιες βασικές λειτουργίες προσφέρει;

Η OpenGL Utility Toolkit (GLUT) είναι μία βιβλιοθήκη που αποτελείται από «εργαλεία» για την ανάπτυξη προγραμμάτων σε OpenGL η οποία έχει ως βασικό στόχο να διευκολύνει την αλληλεπίδραση του προγράμματος με Λειτουργικό Σύστημα. Η αλληλεπίδραση αυτή περιλαμβάνει τον έλεγχο των παραθύρων, του πληκτρολογίου, του ποντικιού, χρονισμούς κ.α. Επιπλέον, παρέχει ορισμένα έτοιμα 3Δ σχήματα (π.χ. κύβος, σφαίρα, οκτάεδρο, δωδεκάεδρο, τσαγιέρα της Utah) όπως επίσης δίνει τη δυνατότητα για τη δημιουργία απλών μενού διεπαφής. Ο κώδικας που δημιουργείται είναι μεταφέρσιμος (portable) και ο προγραμματιστής δεν χρειάζεται να γράψει επιπλέον κώδικα για το εκάστοτε Λειτουργικό Σύστημα προκειμένου να επιτύχει όλα τα παραπάνω. Οι εντολές της GLUT ξεκινούν με το πρόθεμα glut, όπως για παράδειγμα η glutPostRedisplay().

214. Με ποια εντολή δημιουργούμε ένα τρισδιάστατο ορθογραφικό παράθυρο με διαστάσεις: αριστερά -3, δεξιά 3, πάνω 3, κάτω -3 και clipping planes στα 1 και 100 για το near και far αντίστοιχα (στην OpenGL);

glOrtho

Σύνταξη: void glOrtho(GLdouble left, GLdouble right, GLdouble bottom, GLdouble top, GLdouble near, GLdouble far);

Παράδειγμα: glOrtho(-3, 3, -3, 3, 1, 100);

215. Εξετάστε την ισχύ των παρακάτω προτάσεων δηλώνοντας ΣΩΣΤΟ ή ΛΑΘΟΣ και εξηγώντας την επιλογή σας (Όπου κρίνετε απαραίτητο, παραθέστε παράδειγμα).

α) Τα επικρατέστερα Λειτουργικά Συστήματα (Windows, Unix, Linux, Mac OS) υποστηρίζουν την OpenGL.	**ΣΩΣΤΟ**
β) Η OpenGL μπορεί αποκλειστικά να κληθεί (is callable) από τις γλώσσες προγραμματισμού C / C++ (δηλαδή υπάρχει μοναδικό language binding).	**ΛΑΘΟΣ**
γ) Το μοναδικό περιβάλλον ανάπτυξης προγραμμάτων OpenGL είναι το DEV C++.	**ΛΑΘΟΣ**
δ) Η OpenGL περιέχει εντολές επιλογής (τύπου If ... else).	**ΛΑΘΟΣ**
ε) Οι εντολές της OpenGL ξεκινούν με το πρόθεμα gl.	**ΣΩΣΤΟ**

216. Ποια η λειτουργία των παρακάτω εντολών στην OpenGL και τι παραμέτρους δέχονται; Να δοθεί παράδειγμα. α) glutInitWindowPosition, β) glutInitWindowSize, γ) glutCreateWindow.

α) glutInitWindowPosition

Σύνταξη: void glutInitWindowPosition(int x, int y);

Περιγραφή: Καθορίζει τη θέση στην οθόνη, στην οποία θα εμφανιστεί το παράθυρο της εφαρμογής (συντεταγμένη της άνω αριστερής κορυφής).. Οι συντεταγμένες x, y είναι σε εικονοστοιχεία (pixels) και οι εξ 'ορισμού τιμές τους είναι -1 και -1 αντίστοιχα. Αν οι συντεταγμένες παραμείνουν αρνητικές ή αν δημιουργούν πρόβλημα τότε η θέση του παραθύρου καθορίζεται από το Λειτουργικό Σύστημα.

β) glutInitWindowSize

Σύνταξη: void glutInitWindowSize(int width, int height);

Περιγραφή: Καθορίζει το πλάτος (width) και το ύψος (height) του παραθύρου σχεδίασης. Οι παράμετροι width και height είναι σε εικονοστοιχεία (pixels) και θα πρέπει να είναι θετικοί αριθμοί. Οι εξ 'ορισμού τιμές τους είναι 300 και 300 αντίστοιχα. Αν για οποιοδήποτε λόγο οι παράμετροι δημιουργούν πρόβλημα τότε οι τιμές τους καθορίζονται από το Λειτουργικό Σύστημα.

γ) glutCreateWindow

Σύνταξη: int glutCreateWindow(char *name);

Περιγραφή: Δημιουργεί το παράθυρο σχεδίασης γραφικών της OpenGL. Το παράθυρο ενεργοποιείται όταν εκτελεστεί η εντολή glutMainLoop. Η παράμετρος name ορίζει το όνομα του παραθύρου.

Παράδειγμα το οποίο περιλαμβάνει και τις 3 παραπάνω εντολές

```
int main(int argc, char** argv)
{
     glutInit(&argc,argv);
     glutInitWindowPosition(100, 100);
     glutInitWindowSize(640, 480);
     glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
     glutCreateWindow("OpenGL");
     glMatrixMode(GL_PROJECTION);
     gluOrtho2D(-240, 240, -180, 180);
     glutDisplayFunc(display);
     glutMainLoop();
     return 0;
}
```

217. Περιγράψτε τρεις βασικές βιβλιοθήκες εντολών της OpenGL που περιέχουν εντολές σχεδίασης, γραφικών και απόδοσης. Σε ποιο σημείο του προγράμματος δηλώνονται;

Τρεις βασικές βιβλιοθήκες εντολών της OpenGL που περιέχουν εντολές σχεδίασης, γραφικών και απόδοσης είναι οι εξής:

1) Core OpenGL (GL): Αποτελεί τον πυρήνα των εντολών της OpenGL. Όλες οι εντολές στην βιβλιοθήκη αυτή ξεκινούν με το πρόθεμα "gl" (π.χ. glColor, glVertex, glTranslate, glRotate) και στοχεύουν στη δημιουργία αντικειμένων μέσω της χρήσης βασικών γεωμετρικών σχημάτων όπως σημεία, γραμμές, τρίγωνα, πολύγωνα.
2) OpenGL Utility Library (GLU): Είναι βασισμένη στις εντολές του πυρήνα της OpenGL. Οι εντολές της βιβλιοθήκης αυτής ξεκινούν με το πρόθεμα "glu" (π.χ. gluLookAt, gluPerpsective, gluOrtho2D) και προσφέρουν επιπλέον δυνατότητες όπως είδη προβολών, χρήση κάμερας αλλά και δημιουργία πιο προχωρημένων μοντέλων-αντικειμένων.
3) OpenGL Utilities Toolkit (GLUT): Επειδή η OpenGL είναι σχεδιασμένη να είναι ανεξάρτητη από το παραθυρικό σύστημα και από το Λειτουργικό Συστήμα, η GLUT προσφέρει ένα σύνολο εντολών που διευκολύνει ενέργειες όπως το να ανοίξουν και να κλείσουν εύκολα παράθυρα, τον έλεγχο για το πάτημα πλήκτρων ή την καταγραφή της κίνησης του ποντικιού. Επιπλέον, μέσω της GLUT υπάρχει η δυνατότητα για την εισαγωγή έτοιμων 3Δ μοντέλων όπως κύβος, σφαίρα, τόρος, τετράεδρο, οκτάεδρο,

δωδεκάεδρο, εικοσάεδρο, τσαγιέρα της Utah. Οι εντολές της GLUT ξεκινούν με το πρόθεμα “glut” (π.χ. glutCreatewindow, glutMouseFunc).

218. Αναφέρετε την λειτουργία της glutMainLoop() στην OpenGL. Σε ποιο σημείο πρέπει να καλείται;

glutMainLoop

Σύνταξη: void glutMainLoop(void);

Περιγραφή: Ξεκινάει τον βρόχο διαχείρισης γεγονότων της GLUT και καλεί διαρκώς μέσα από αυτόν τις δηλωθείσες από πριν συναρτήσεις γεγονότων όπως η glutDisplayFunc, η glutReshapeFunc κλπ. Μετά την εκκίνηση του βρόχου το πρόγραμμα δεν επιστρέφει ποτέ στο σημείο που έγινε η κλήση του. Για τον λόγο αυτό η εντολή θα πρέπει να εκτελείται τελευταία αφού οριστούν πρώτα οι επιθυμητές συναρτήσεις γεγονότων.

54. Περιγράψτε τη βασική δομή μιας εφαρμογής φτιαγμένης σε OpenGL.

```
#include <GL/glut.h>

void display()
{
      glClearColor(1, 1, 1, 1);
      glClear(GL_COLOR_BUFFER_BIT);
      glColor3f(1.0f, 0.0f, 0.0f);
      glBegin(GL_LINES);
            glVertex2f(-80.0f, -60.0f);
            glVertex2f( 40.0f,  40.0f);
      glEnd();
      glFlush();
}

int main(int argc, char** argv)
{
      glutInit(&argc,argv);
      glutInitWindowPosition(100, 100);
      glutInitWindowSize(640, 480);
      glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
      glutCreateWindow("OpenGL");
      glMatrixMode(GL_PROJECTION);
      gluOrtho2D(-240, 240, -180, 180);
      glutDisplayFunc(display);
      glutMainLoop();
      return 0;
}
```

- Αρχικά προσθέτουμε την κεφαλίδα του **GLUT/glut.h**. Γενικά, για την ανάπτυξη ενός προγράμματος που χρησιμοποιεί ρουτίνες της OpenGL, απαιτείται η προσθήκη των κεφαλίδων gl.h και glu.h, ενώ για χρήση του GLUT, απαιτείται η προσθήκη της glut.h. Ωστόσο, με τη δήλωση της κεφαλίδας glut.h δηλώνονται αυτομάτως και οι δύο προηγούμενες.
- Η συνάρτηση **glutInit** ενεργοποιεί τη βιβλιοθήκη GLUT και μέσω αυτής μπορούμε να περάσουμε παραμέτρους στην εφαρμογή μας από τη γραμμή εντολών του DOS.
- Η ρουτίνα **glutInitWindowPosition** καθορίζει τη θέση στην οθόνη, στην οποία θα εμφανιστεί το παράθυρο της εφαρμογής (συντεταγμένη της άνω αριστερής κορυφής).
- Η εντολή **glutInitWindowSize** καθορίζει το πλάτος και ύψος του παραθύρου της εφαρμογής σε pixels.
- Η εντολή **glutInitDisplayMode** καθορίζει παραμέτρους σχετικές με τους ενταμιευτές και το χρωματικό μοντέλο που χρησιμοποιούνται κατά τη σχεδίαση. Στο παράδειγμά μας χρησιμοποιούμε το χρωματικό μοντέλο RGB (**GLUT_RGB**) και την εφαρμογή απλής ενταμίευσης (**GLUT_SINGLE**).
- Η εντολή **glutCreateWindow** εμφανίζει το παράθυρο της εφαρμογής στην οθόνη και του αποδίδει έναν τίτλο.
- Με την εντολή **glMatrixMode** επιλέγουμε το μητρώο το οποίο επιθυμούμε να τροποποιήσουμε. Στο παράδειγμα, δίνοντας ως όρισμα τη σταθερά **GL_PROJECTION** επιλέγουμε το μητρώο προβολής, το οποίο καθορίζει τον τρόπο με τον οποίο προβάλλεται η σκηνή στο επίπεδο του θεατή. Σε συνδυασμό με την εντολή **gluOrtho2D(xMin, xMax, yMin, yMax)** διευκρινίζουμε ότι θα απεικονιστεί η παράλληλη προβολή της σκηνής στο επίπεδο XY (το οποίο, στο συγκεκριμένο παράδειγμα, ταυτίζεται με το παράθυρο της εφαρμογής). Το περιεχόμενο του παραθύρου περιλαμβάνει όλα τα στοιχεία του σκηνικού που εκτείνονται μεταξύ των συντεταγμένων X=[-50,50] και Y=[-50,50]. Ο θετικός άξονας X έχει φορά προς τα δεξιά και το θετικό τμήμα του άξονα Y έχει φορά προς τα πάνω.
- Η εντολή **glutDisplayFunc** εντάσσεται σε μια ειδική κατηγορία συναρτήσεων του GLUT, οι οποίες αποκαλούνται συναρτήσεις κλήσης (callback functions). Η συγκεκριμένη συνάρτηση δέχεται ως όρισμα μια συνάρτηση στην οποία εμπεριέχεται ο κώδικας σχεδίασης γραφικών. Η συνάρτηση αυτή εκτελείται κάθε

φορά που η εφαρμογή διαπιστώσει ότι απαιτείται επανασχεδιασμός της σκηνής. Η συνάρτηση δεν επιστρέφει τιμή και δεν έχει ορίσματα. Στο συγκεκριμένο παράδειγμα δίνουμε ως όρισμα τη συνάρτηση display().

- Η εντολή **glutMainLoop** ενεργοποιεί τον βρόχο διαχείρισης γεγονότων (event processing loop). Στον κύκλο αυτό, η εφαρμογή αναμένει επ' άπειρον και ανταποκρίνεται σε γεγονότα, όπως λ.χ. στο πάτημα ενός κουμπιού, στην αλλαγή του σκηνικού ή στην κίνηση του ποντικιού.
- Η εντολή **glClearColor** καθορίζει το χρώμα που χρησιμοποιείται κάθε φορά που εκτελείται εντολή καθαρισμού της οθόνης. Στο συγκεκριμένο παράδειγμα, χρησιμοποιείται το λευκό χρώμα.
- Η εντολή **glClear** καθαρίζει ενταμιευτές (buffers), συγκεκριμένες περιοχές μνήμης του συστήματος γραφικών (frame buffer). Η μηχανή γραφικών της OpenGL ορίζει ορισμένες κατηγορίες ενταμιευτών. Με την σταθερά **GL_COLOR_BUFFER_BIT** δίνουμε εντολή καθαρισμού του ενταμιευτή χρωματικών τιμών (colour buffer).
- Με την εντολή **glColor3f** ορίζουμε το τρέχον χρώμα σχεδιάσης. Στο παράδειγμα επιλέγουμε ως χρώμα σχεδιάσης το κόκκινο.
- Η εντολή **glBegin** δηλώνει την έναρξη ορισμού ενός ή περισσοτέρων γεωμετρικών σχημάτων. Αναλόγως του ορίσματος, μπορεί να προσδιοριστεί μια ποικιλία σχημάτων. Στο παράδειγμά μας, ορίζουμε ευθύγράμμα τμήματα. Η εντολή **glBegin** εκτελείται πάντα σε συνδυασμό με την εντολή **glEnd** και η δεύτερη ορίζει τη λήξη της επιλεγόμενης ρύθμισης σχεδίασης.
- Η εντολή **glVertex2i** ορίζει σημεία στο διδιάστατο χώρο. Εφόσον έχει προεπιλεγεί η κατάσταση σχεδίασης ευθυγράμμων τμημάτων, τα σημεία ορίζουν ανά ζεύγη τα ευθύγραμμα τμήματα Το παραπάνω παράδειγμα σχεδιάζει ένα ευθύγραμμο τμήμα κόκκινου χρώματος με συνταταγμένες αρχής και τέλους (20,20) (40,40).
- Η εντολή **glFlush** εξαναγκάζει την εκτέλεση των εντολών που εκκρεμούν.

55. Να αναφέρετε και να περιγράψετε σύντομα τα callback functions της OpenGL.

glutDisplayFunc()	Ορίζουμε τη συνάρτηση η οποία θα εκτελείται όποτε εγείρεται γεγονός απαίτησης σχεδιασμού/επανασχεδιασμού της σκηνής
glutReshapeFunc()	Ορίζουμε τη συνάρτηση κλήσης για τη διαχείριση γεγονότων κατά την αλλαγή των διαστάσεων του παραθύρου.
glutKeyboardFunc()	Ορίζουμε τη συνάρτηση κλήσης για τη διαχείριση γεγονότων κατά το πάτημα πλήκτρων τα οποία φέρουν ASCII χαρακτήρα.
glutSpecialFunc()	Ορίζουμε τη συνάρτηση κλήσης για τη διαχείριση γεγονότων κατά το πάτημα ειδικών πλήκτρων (function keys, arrow keys, κλπ).
glutMouseFunc()	Ορίζουμε τη συνάρτηση κλήσης για τη διαχείριση γεγονότων κατά το πάτημα των πλήκτρων του ποντικιού.
glutMotionFunc()	Ορίζουμε τη συνάρτηση κλήσης για τη διαχείριση γεγονότων κατά την κίνηση του ποντικιού με πιεσμένο πλήκτρο.
glutPassiveMotionFunc()	Ορίζουμε τη συνάρτηση κλήσης για τη διαχείριση γεγονότων κατά την κίνηση του ποντικιού χωρίς πιεσμένο πλήκτρο.
glutIdleFunc()	Ορίζουμε τη συνάρτηση κλήσης η οποία καλείται όταν ο βρόχος της GLUT βρίσκεται σε αδράνεια. Είναι ιδιαίτερα χρήσιμη σε περιπτώσεις που επιθυμούμε συνεχείς μεταβολές και τακτικό επανασχεδιασμό της σκηνής, όπως τα κινούμενα γραφικά.
glutTimerFunc()	Ορίζουμε μία συνάρτηση κλήσης η οποία θα εκτελείται ανά προγραμματισμένα χρονικά διαστήματα.

56. Ποιες εντολές χρησιμοποιούμε για τις εξής λειτουργίες της OpenGL: α. translate β. rotate γ. scale ή stretch;

α) Μετακίνηση

Η μετακίνηση (translation) ενός αντικειμένου γίνεται με την εντολή:

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

β) Περιστροφή

Η περιστροφή (rotation) ενός αντικειμένου γίνεται με την εντολή:

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

Η 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 περιστροφή γύρω από τον X, ως [0, 1, 0] για περιστροφή γύρω από τον Y και ως [0, 0, 1] για περιστροφή γύρω από τον Z.

γ) Κλιμάκωση

Η κλιμάκωση (μεγέθυνση/σμίκρυνση ή αλλιώς scaling) ενός αντικειμένου γίνεται με την εντολή:

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

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

57. Για ποιο λόγο είναι καλό να χρησιμοποιούμε τους τύπους δεδομένων της OpenGL;

Η OpenGL χρησιμοποιεί έναν αριθμό από προκαθορισμένους τύπους δεδομένων όπως για παράδειγμα GLboolean, GLbyte, GLint, GLfloat, GLdouble και άλλους. Συνίσταται η χρήση των τύπων δεδομένων αυτών έναντι των τύπων δεδομένων που χρησιμοποιεί η εκάστοτε γλώσσα προγραμματισμού διότι στη δεύτερη περίπτωση το εύρος των τύπων δεδομένων ενδέχεται να αλλάζει ανάλογα με το Λειτουργικό Σύστημα και την αρχιτεκτονική του υπολογιστή ενώ αντίθετα στην πρώτη περίπτωση οι τύποι δεδομένων της OpenGL έχουν το ίδιο εύρος ανεξαρτήτου Λειτουργικού Συστήματος και αρχιτεκτονικής.

58. Περιγράψτε τη σύνταξη και τις παραμέτρους της glLookAt() (στην OpenGL).

Η OpenGL μας δίνει την δυνατότητα να τοποθετήσουμε την κάμερα στην σκηνή και να θέσουμε κατεύθυνση 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 τη θέση της κάμερας στην σκηνή, το [centerx, centery, centery] το οποίο καθορίζει την κατεύθυνση της κάμερας (το σημείο στο οποίο «κοιτάει») 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);

59. Ποια είναι η λειτουργία της glClear() και ποια της glClearColor3f() (στην OpenGL);

Η εντολή **glClearColor3f()** καθορίζει το χρώμα που χρησιμοποιείται κάθε φορά που εκτελείται εντολή καθαρισμού της οθόνης **glClear()**.

60. Στην OpenGL όπως και στα περισσότερα API γραφικών χρησιμοποιούμε “normalized values”, τι σημαίνει αυτό;

Στην OpenGL, μετά τη διαδικασία της αποκοπής και πριν την απόδοση συντεταγμένων συσκευής, εκτελείται μια διαδικασία κανονικοποίησης των αποκομμένων συντεταγμένων σε ένα προκαθορισμένο εύρος τιμών. Αυτό γίνεται ούτως ώστε η αναπαράσταση των συντεταγμένων των απομονωμένων σχημάτων να είναι ανεξάρτητη της χρησιμοποιούμενης ανάλυσης της συσκευής εξόδου. Οι συντεταγμένες των σχημάτων που περιέχονται στο παράθυρο αποκοπής κανονικοποιούνται στο εύρος τιμών [-1, 1] (κανονικοποιημένο τετράγωνο).

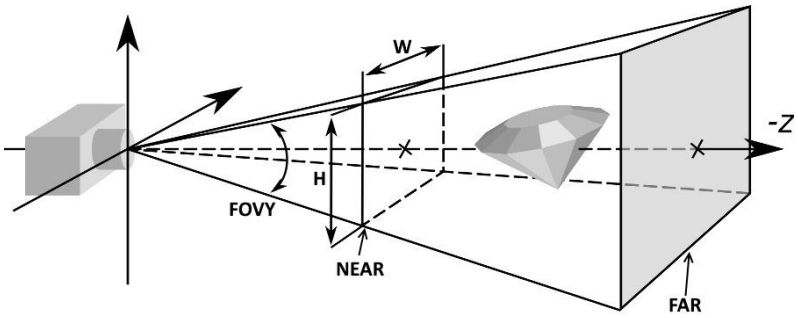
61. Αναφέρατε τις εντολές (μαζί με τις παραμέτρους τους) για προοπτική (perspective) και για ορθογραφική (orthographic) απεικόνιση (στην OpenGL).

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

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.

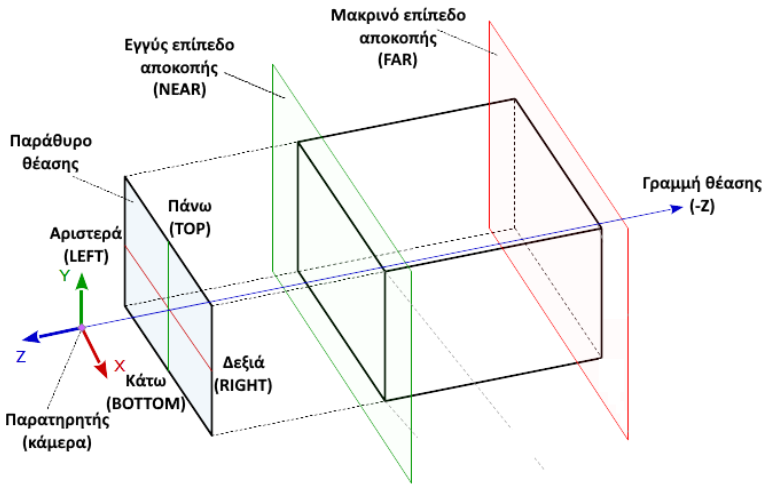


Προοπτική προβολή

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.



Ορθογραφική προβολή

62. Με ποιους τρόπους μπορούμε να μετατοπίσουμε ένα αντικείμενο - σχήμα στην OpenGL;

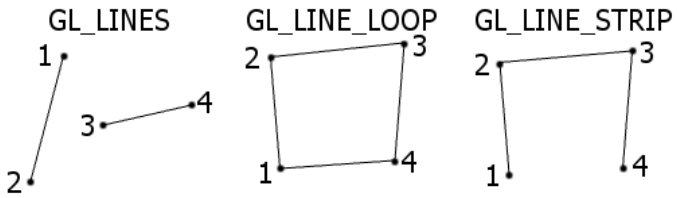
Υπάρχουν 2 βασικοί τρόποι για να μετατοπίσουμε ένα αντικείμενο στην OpenGL:

1) Μεταβολή των συντεταγμένων του αντικειμένου: Αν το αντικείμενο αποτελείται από βασικά σχήματα (σημεία, γραμμές, τρίγωνα, τετράγωνα, πολύγωνα) τότε απλώς αλλάζουμε τις συντεταγμένες x, y, z των βασικών αυτών σχημάτων προσθέτοντας ή αφαιρώντας σε αυτές μια τιμή η οποία εκφράζει τη μετατόπιση που θέλουμε να επιτύχουμε.
2) Μετατόπιση του συστήματος αξόνων: Μπορούμε απλώς να μετατοπίσουμε τους άξονες (X, Y, Z) με τη χρήση της εντολής glTranslate().

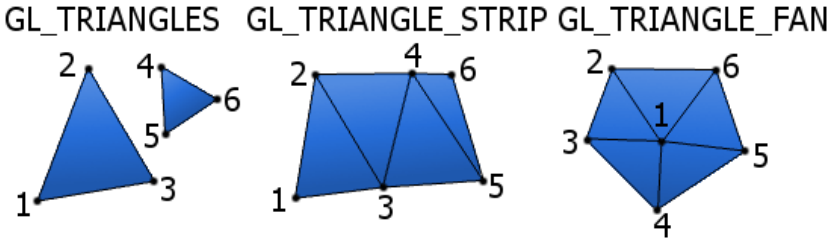
63. Περιγράψτε με εντολές και παραμέτρους της OpenGL, τα βασικά γεωμετρικά σχήματα α) Γραμμές, β) Τρίγωνα, γ) Πολύγωνα. Παραθέστε σχήματα.

α) Γραμμές: Στην OpenGL, γραμμή σημαίνει ευθύγραμμο τμήμα και όχι η μαθηματική έννοια που εκτείνει στο άπειρο και στις δύο κατευθύνσεις. Μπορούμε να καθορίσουμε μία σειρά από συνδεμένα ευθύγραμμα τμήματα ή μία κλειστή σειρά από συνδεμένα ευθύγραμμα τμήματα. Σε όλες τις περιπτώσεις όμως, οι γραμμές που αποτελούν τις συνδεμένες σειρές καθορίζονται με τις συντεταγμένες των άκρων τους.

Για τον σχεδιασμό γραμμών ορίζουμε μεταξύ των εντολών glBegin() και glEnd() τα σημεία που αποτελούν το ευθύγραμμο τμήμα ενώ ως παράμετρο της glBegin() μπορούμε να δώσουμε GL_LINES, GL_LINE_LOOP, GL_LINE_STRIP ανάλογα με το είδος του ευθύγραμμου τμήματος.



β) Τρίγωνα: Για τον σχεδιασμό τριγώνων ορίζουμε μεταξύ των εντολών glBegin() και glEnd() τα σημεία που αποτελούν τα τρίγωνα ενώ ως παράμετρο της glBegin() μπορούμε να δώσουμε GL_TRIANGLES, GL_TRIANGLE_STRIP, GL_TRIANGLE_FAN ανάλογα με το είδος του τριγώνου.



γ) Πολύγωνα: Τα πολύγωνα είναι περιοχές που περικλείονται από απλούς κλειστούς βρόχους ευθύγραμμων τμημάτων, όπου τα ευθύγραμμα τμήματα καθορίζονται από τις κορυφές στα άκρα τους. Τα πολύγωνα σχεδιάζονται συμπαγή (με χρωματισμένα τα pixels στο εσωτερικό τους), ωστόσο έχουμε τη δυνατότητα να σχεδιάσουμε απλώς τα περιγράμματα ή τις κορυφές τους.

Τα πολύγωνα διακρίνονται σε δύο κατηγορίες: τα κυρτά και τα κοίλα. Στα κυρτά πολύγωνα όλες οι εσωτερικές γωνίες είναι μικρότερες των 180^o ενώ ένα πολύγωνο είναι κοίλο όταν περιέχει τουλάχιστον μια εσωτερική γωνία μεγαλύτερη των 180^o.

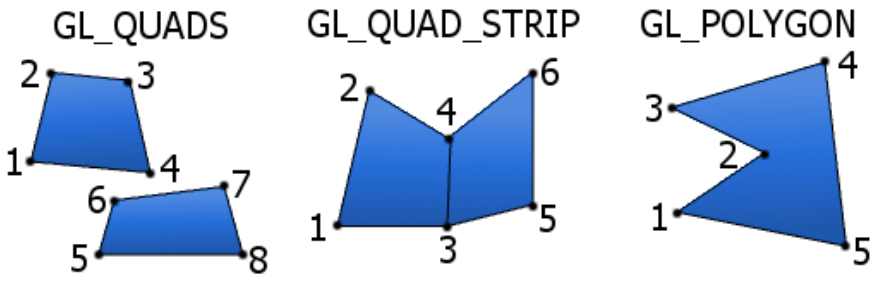
Γενικά τα πολύγωνα μπορεί να είναι πολύπλοκα και για το λόγο αυτό η OpenGL έχει θέσει τους εξής περιορισμούς στις ρουτίνες σχεδίασής τους:

α) Οι ρουτίνες σχεδιάζουν κυρτά πολύγωνα.

β) Οι πλευρές των πολυγώνων δεν μπορούν να τέμνονται .

γ) Οι ρουτίνες δε μπορούν να σχεδιάσουν πολύγωνα με οπές.

Για τον σχεδιασμό τριγώνων ορίζουμε μεταξύ των εντολών glBegin() και glEnd() τα σημεία που αποτελούν τα πολύγωνα ενώ ως παράμετρο της glBegin() μπορούμε να δώσουμε GL_QUADS, GL_QUAD_STRIP, GL_POLYGON ανάλογα με το είδος του πολυγώνου.



64. Να γραφεί function (στην OpenGL) με όνομα keyboard που να λειτουργεί έτσι ώστε, όταν ο χρήστης πατήσει το 'Q' (ή 'q') να κλείνει την εφαρμογή και όταν πατήσει το 'C' (ή 'c') να καθαρίζει την οθόνη στο χρώμα του GL_COLOR_BUFFER_BIT. Επίσης να γράψετε την εντολή του mainloop που θα το καλέσετε.

```
#include <GL/glut.h>

void display()
{
      glClearColor(1, 1, 1, 1);
      glClear(GL_COLOR_BUFFER_BIT);
      glColor3f(1.0f, 0.0f, 0.0f);
      glBegin(GL_LINES);
            glVertex2f(-80.0f, -60.0f);
            glVertex2f( 40.0f,  40.0f);
      glEnd();

      glutSwapBuffers();
}

void keyboard(unsigned char key, int x, int y)
{
    if(key == 'c' || key == 'C')
    {
        glClear(GL_COLOR_BUFFER_BIT);
        glutSwapBuffers();
    }
    if(key == 'q' || key == 'Q')
    {
        exit(0);
    }
}

int main(int argc, char** argv)
{
      glutInit(&argc,argv);
      glutInitWindowPosition(100, 100);
      glutInitWindowSize(640, 480);
      glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
      glutCreateWindow("OpenGL");
```

```
	glMatrixMode(GL_PROJECTION);
	gluOrtho2D(-240, 240, -180, 180);
	glutDisplayFunc(display);
	glutKeyboardFunc(keyboard);
	glutMainLoop();
	return 0;
}
```

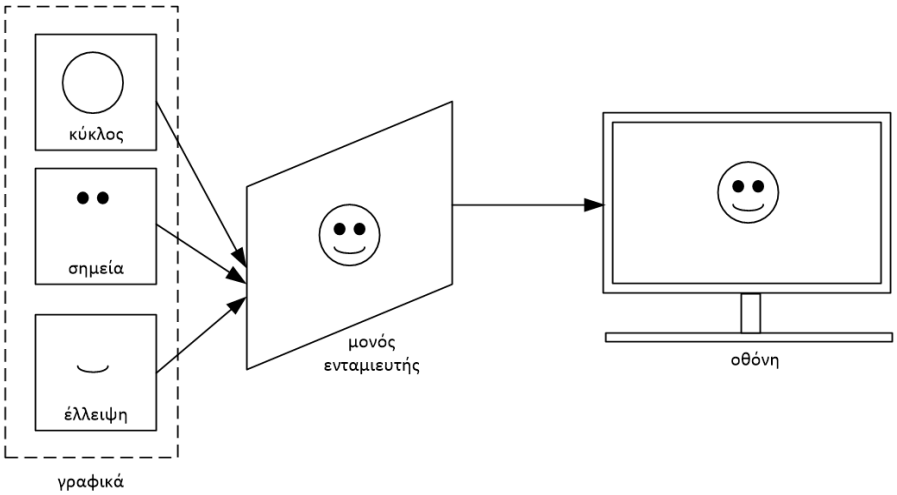
65. Περιγράψτε τα βήματα για τη δημιουργία animation στην OpenGL.

1. Χρήση της εντολής glutDisplayFunc() για τον ορισμό του ονόματος της συνάρτησης η οποία θα εκτελείται κάθε φορά που θέλουμε να σχεδιάσουμε γραφικά στο κυρίως παράθυρο.
2. Αν display() είναι το όνομα της συνάρτησης που καλείται από την περίπτωση του προηγούμενου βήματος τότε εντός της display() θα πρέπει:
   i. Να γίνετε καθαρισμός του παραθύρου (εντολές glClearColor() και glClear()).
   ii. Να σχεδιάζονται τα νέα γραφικά στις νέες θέσεις.
   iii. Να εμφανίζονται τα γραφικά στο παράθυρο (εντολή glFlush() ή glutSwapBuffers()).
3. Η συνάρτηση display() θα πρέπει να εκτελείται πολλές φορές στη μονάδα του χρόνου για να δίνεται η ψευδαίσθηση της κίνησης. Αυτό μπορεί να επιτευχθεί είτε με χρήση της εντολής glutIdleFunc() και με παράμετρο την συνάρτηση display() είτε με χρήση ενός χρονιστή μέσω της εντολής glutTimerFunc() ο οποίος θα καλεί την display() ανά τακτά χρονικά διαστήματα.

67. Περιγράψτε αναλυτικά τι είναι το double buffering. Με ποια εντολή ενεργοποιούμε το double buffering σε RGB mode στην OpenGL; Με ποια εντολή κάνουμε swap τον front και τον back buffer έτσι ώστε να έχουμε animation;

Τεχνική μονού ενταμιευτή (single buffering)

- Οι εντολές με τις οποίες σχεδιάζουμε τα γραφικά μας, δημιουργούν ένα αποτέλεσμα το οποίο δεν εμφανίζεται κατευθείαν στην οθόνη αλλά αποθηκεύεται πρώτα σε έναν **ενταμιευτή** (buffer).
- Όταν όλες οι εντολές για τον σχεδιασμό των γραφικών ολοκληρωθούν τότε ο ενταμιευτής «αδειάζει» (flush) το περιεχόμενό του στην οθόνη και δημιουργείται ένα **καρέ** (frame).
- Ο λόγος που συμβαίνει αυτό είναι η ταχύτητα. Προσπαθούμε να ξεγελάσουμε το μάτι και να δει ένα ολοκληρωμένο καρέ παρά να δει σταδιακά τον σχεδιασμό του.
- Η τεχνική αυτή ονομάζεται **τεχνική μονού ενταμιευτή** (single buffering).

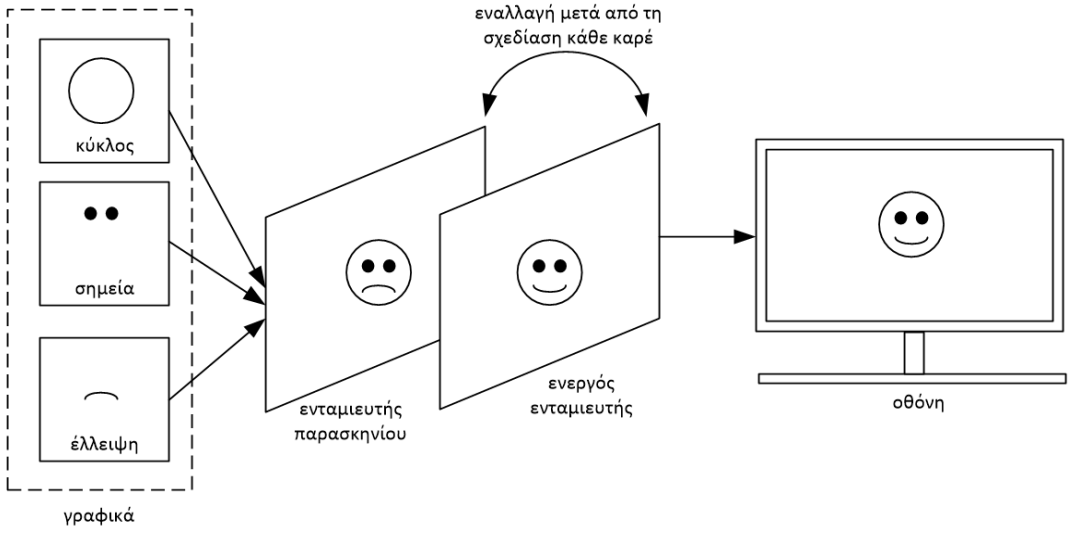


- Όταν θέλουμε να σχεδιάσουμε μόνο ένα καρέ στην οθόνη τότε η τεχνική του μονού ενταμιευτή αρκεί.

- Αν όμως θελήσουμε να δημιουργήσουμε κίνηση τότε θα πρέπει να σχεδιάσουμε πολλά διαδοχικά καρέ σε πολύ μικρό χρονικό διάστημα.
- Όσα περισσότερα καρέ σχεδιαστούν στη μονάδα του χρόνου τόσο πιο φυσιολογικό το αποτέλεσμα.
- Επειδή πριν τον σχεδιασμό κάθε νέου καρέ «καθαρίζουμε» πρώτα τα γραφικά από το προηγούμενο καρέ, οι γρήγορες αυτές εναλλαγές (καθάρισμα-σχεδιασμός) δημιουργούν ένα «τρεμόπαιγμα» στο αποτέλεσμα.
- Το αποτέλεσμα είναι ιδιαίτερα εμφανές σε υπολογιστικά συστήματα με αργές κάρτες γραφικών ή σε οθόνες με χαμηλό ρυθμό ανανέωσης.

Τεχνική διπλού ενταμιευτή (double buffering)

- Τη λύση στο πρόβλημα του τρεμοπαίγματος έρχεται να δώσει η τεχνική που ονομάζεται **τεχνική διπλού ενταμιευτή** (double buffering).
- Οι ενταμιευτές που χρησιμοποιούνται είναι δύο: ένας ενεργός του οποίου το περιεχόμενο εμφανίζεται στην οθόνη κι ένας στο παρασκήνιο στον οποίο σχεδιάζεται το επόμενο καρέ όσο ο ενεργός δείχνει το δικό του.
- Όταν η σχεδίαση ολοκληρωθεί στον ενταμιευτή του παρασκηνίου τότε οι ενταμιευτές εναλλάσσονται.
- Με τον τρόπο αυτό η κίνηση που απεικονίζεται είναι πιο ομαλή στο μάτι, χωρίς το φαινόμενο του τρεμοπαίγματος.



68. Για ποιό σκοπό χρειάζονται οι συναρτήσεις κλήσης στην OpenGl και πώς ορίζονται;

Ως γεγονός ορίζουμε την καταγραφή κάποιας δραστηριότητας του συστήματος, συνήθως μιας δραστηριότητας από κάποια συσκευή εισόδου όπως ένα πληκτρολόγιο ή ένα ποντίκι. Ωστόσο υπάρχουν και γεγονότα που εγείρονται από το λειτουργικό σύστημα υπό ορισμένες συνθήκες.

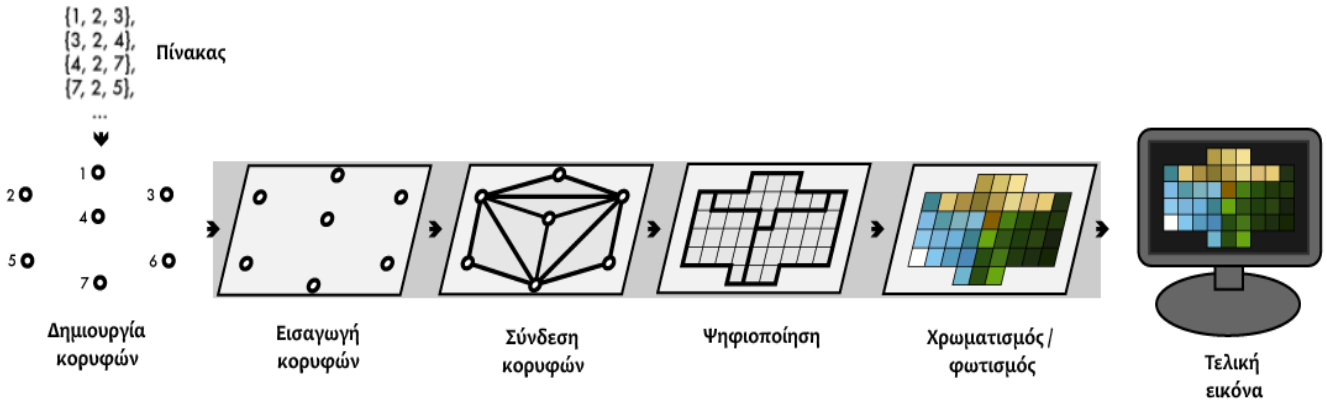
Τα γεγονότα καταγράφονται σε μια σειρά , την οποία διαχειρίζεται το λειτουργικό σύστημα. Αυτό, τηρεί ένα αρχείο με όλα τα γεγονότα, παρακολουθεί τη σειρά με την οποία εγείρονται και αναθέτει τη διαχείριση κάθε γεγονότος σε ένα τμήμα κώδικα το οποίο έχει προκαθοριστεί να εκτελείται κατά την εμφάνιση του γεγονότος.

Η ανταπόκριση μιας εφαρμογής σε γεγονότα συνίσταται στην ανάθεση ενός κώδικα προς εκτέλεση ανά είδος γεγονότος. Ο κώδικας που θα εκτελείται κατά την εμφάνιση κάθε γεγονότος ορίζεται στις **συναρτήσεις διαχείρισης γεγονότων** ή αλλιώς **συναρτήσεις κλήσης** (callback functions). Ο ορισμός των συναρτήσεων κλήσης γίνεται πριν την έναρξη του κύριου βρόχου της OpenGL. Προφανώς, γεγονότα για οποία δεν έχει καταχωρηθεί συνάρτηση διαχείρισης δεν έχουν καμία επίδραση στην εκτέλεση του προγράμματος. Ο παρακάτω ψευδοκώδικας περιγράφει τον κύκλο διαχείρισης γεγονότων:

```
while(1) //Εκτέλεση βρόχου επ' άπειρον
{
        if (γεγονός σχεδιασμού/επανασχεδιασμού)
        { εκτέλεση κώδικα διαχείρισης γεγονότος σχεδιασμού/επανασχεδιασμού; }
        if (αλλαγή διαστάσεων παραθύρου)
        {εκτέλεση κώδικα διαχείρισης του γεγονότος αλλαγής διάστασης παραθύρου; }
        if (εμφάνιση γεγονότος πληκτρολογίου/ποντικιού)
        { εκτέλεση κώδικα διαχείρισης γεγονότων πληκτρολογίου/ποντικιού }
        ……..
        εκτέλεση κώδικα διαχείρισης "γεγονότος αδρανείας" ;
}
```

71. Με ποια σειρά γίνονται οι διεργασίες για τον υπολογισμό της τελικής εικόνας (rendering) στην OpenGL;

Η ακολουθία βημάτων που ακολουθεί η OpenGL για να απεικονίσει το μοντέλο λέγεται graphics pipeline (διασωλήνωση). Στο επόμενο σχήμα φαίνονται όλα τα στάδια από ορισμό των κορυφών έως και τη δημιουργία της τελικής εικόνας.



72. Με ποια εντολή και ποιες συναρτήσεις θα πραγματοποιήσετε απόδοση υφής σε γραμμές και επιφάνειες στην OpenGl;

```
// Δημιουργία μιας νέας υφής η οποία είναι φορτωμένη στη μνήμη
GLuint textureID;
glGenTextures(1, &textureID);

// Σύνδεση της υφής
glBindTexture(GL_TEXTURE_2D, textureID);

// Απόδοση της υφής στην OpenGL
glTexImage2D(GL_TEXTURE_2D, 0,GL_RGB, width, height, 0, GL_BGR,
GL_UNSIGNED_BYTE, data);

glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
```

73. Για ποιο λόγο χρησιμοποιούνται οι λίστες απεικόνισης(display lists) στην γλώσσα OpenGl και πώς συντάσονται; Δώστε και από ένα παράδειγμα για κάθε περίπτωση.

Συχνά, είναι βολικό η περιγραφή ενός σύνθετου γεωμετρικού σήματος να περικλείεται σε μια αυτόνομη ενότητα κώδικα, η οποία θα εκτελείται κάθε φορά που θέλουμε να σχεδιάσουμε αυτό το σχήμα. Στην OpenGL τη δυνατότητα αυτή δίνουν οι **λίστες απεικόνισης** (display lists). Οι λίστες απεικόνισης διευκολύνουν την επεναχρησιμοποίηση κώδικα και απαλλάσσουν τον προγραμματιστή από περιττές επαναλαμβανόμενες δηλώσεις του ίδιου σύνθετου σχήματος.

Μια λίστα απεικόνισης περικλείεται μεταξύ δύο εντολών: των **glNewList** και **glEndList**. Όταν σε ένα πρόγραμμα ορίζονται πολλαπλές λίστες, μια αυτόνομη λίστα απεικόνισης διακρίνεται από τις υπόλοιπες βάσει ενός ακεραίου αριθμού που παίζει το ρόλο του "αναγνωριστικού αριθμού" της (list identifier). Προεκιμένου να αποδώσουμε αναγνωριστικούς αριθμούς σε λίστες απεικόνισης, πρέπει να δεσμεύσουμε το απαιτούμενο εύρος τιμών. Η δέσμευση αυτή γίνεται με την εντολή **glGenLists()**.

GLuint glGenLists(GLint range);

όπου *range* το πλήθος των αναγνωριστικών που θέλουμε να χρησιμοποιήσουμε. Η συνάρτηση επιστρέφει μια ακέραιη τιμή που αντιστοιχεί στον πρώτο αναγνωριστικό αριθμό.
Π.χ. με τη σύνταξη

listID = glGenLists(2);

παράγουμε 2 identifiers. Ο πρώτος έχει ακέραιη τιμή *listID*, ο δεύτερος *listID+1* και ούτω καθ' εξής.

Η ανάθεση αναγνωριστικής τιμής σε μια λίστα απεικόνισης γίνεται κατά την έναρξη της δήλωσής της στην εντολή ***glNewList***:

void glNewList(GLuint listID, GLenum listMode);

όπου *listID* το αναγνωριστικό που θέλουμε να αποδώσουμε στη λίστα απεικόνισης. Η παράμετρος *listMode* έχει δύο πιθανές τιμές:

- *GL_COMPILE*: Δηλώνουμε τον κώδικα σχεδιασμού του σύνθετου αντικειμένου που περιγράφεται στη λίστα απεικόνισης
- *GL_COMPILE_AND_EXECUTE*: Δηλώνουμε **και εκτελούμε ταυτόχρονα** τον κώδικα σχεδιασμού που περιέχεται στη λίστα απεικόνισης.

Η δήλωση λοιπόν ενός σύνθετου σχήματος σε λίστα απεικόνισης έχει τη μορφή:

```
glNewList();
// Εντολές δήλωσης σχήματος
glEndList();
```

Μεταξύ των εντολών ***glNewList*** και ***glEndList*** ορίζουμε το σύνθετο γεωμετρικό σχήμα της λίστας απεικόνισης, χρησιμοποιώντας τις εντολές σχεδιάσης σχημάτων που αναφέραμε παραπάνω.
Ο κώδικας που περιέχεται σε μία λίστα απεικόνισης εκτελείται δίνοτας τον αναγνωριστικό της αριθμό ως όρισμα στην εντολή ***glCallList()***:

void glCallList(GLuint listID);

Μία σημαντική λεπτομέρεια που ο προγραμματιστής πρέπει να έχει υπόψη, αφορά τις πιθανές αλλαγές των μεταβλητών κατάστασης κατά την εκτέλεση μιας λίστας απεικόνισης. Πρέπει να έχουμε υπόψη ότι, εάν μέσα σε μια display list μεταβάλλουμε την τιμή μιας μεταβλητής κατάστασης (όπως π.χ. του τρέχοντος χρώματος σχεδίασης), η μεταβολή αυτή θα παραμείνει ενεργή και μετά το πέρας εκτέλεσης της λίστας. Δηλαδή εάν μέσα στη display list έχει οριστεί την τελευταία φορά ως χρώμα σχεδίασης το κόκκινο μετά το πέρας της εκτέλεσης της λίστας η παράμετρος θα διατηρήσει την τελευταία τιμή και θα πρέπει να τη μεταβάλλει ο προγραμματιστής. Είναι χρήσιμο λοιπόν ο προγραμματιστής να αποθηκεύσει τις τιμές των ιδιοτήτων που θα μεταβληθούν στη στοίβα ιδιοτήτων, πριν από την κλήση της λίστας, ούτως ώστε να είναι σε θέση να τις επαναφέρει μετά το πέρας της εκτέλεσης της λίστας.