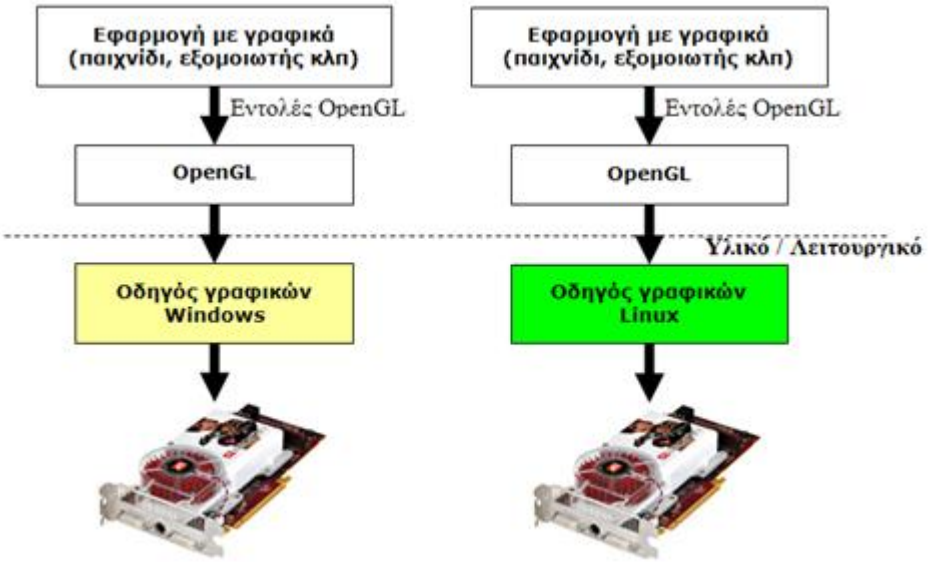
Εισαγωγή στην OpenGL: μέρος 1ο

Τι είναι η OpenGL

Η OpenGL είναι ένα σύνολο εντολών (**A**pplication **P**rogramming **I**nterface – API) που μας επιτρέπει την δημιουργία τριδιάστατων γραφικών. Δεν είναι γλώσσα προγραμματισμού αλλά μπορεί να χρησιμοποιηθεί με μια πληθώρα γλωσσών προγραμματισμού (C, C++, Java και άλλες) και σε μια πληθώρα λειτουργικών συστημάτων (Windows, Unix, Linux, Mac OS).

Προσφέρει πάνω από 300 εντολές για δημιουργία γραφικών και δρα σαν ένα ενδιάμεσο στρώμα ανάμεσα στην εφαρμογή και στη κάρτα γραφικών που θα αναλάβει τα απεικονίσει τα γραφικά στην οθόνη, κρύβοντας λεπτομέρειες υλοποίησης του υλικού και των οδηγών του.

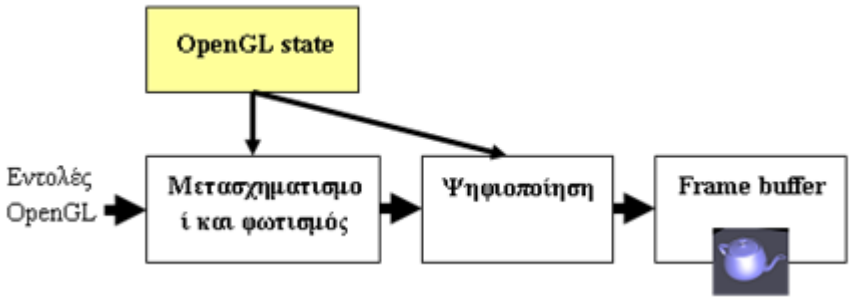


Προγράμματα που χρησιμοποιούν OpenGL μπορούν να τρέξουν και χωρίς επιταχυντή γραφικών (3D graphics card), θα είναι όμως πολύ πιο αργά και πιθανώς να μην υποστηρίζουν όλες τις λειτουργίες της OpenGL.

Η OpenGL είναι το πρώτο ευρέως διαδεδομένο API για γραφικά και υπάρχει σαν στάνταρντ από το 1992. Άλλο ευρέως διαδεδομένο API γραφικών είναι το DirectX της Microsoft το οποίο είναι γραμμένο αποκλειστικά για πλατφόρμες Windows και Xbox/Xbox360.

OpenGL graphics pipeline

Ας δούμε σχηματικά τα βήματα που κάνει η OpenGL από την στιγμή που πάρει τις εντολές γραφικών από την εφαρμογή μας μέχρι την στιγμή δημιουργίας της τελικής εικόνας (rendered image).



Η OpenGL χρησιμοποιεί μια **μηχανή καταστάσεων** (state machine) για να επικοινωνεί με την εφαρμογή. Σε αυτή την μηχανή καταστάσεων η OpenGL παραμένει διαρκώς σε μια κατάσταση μέχρι να αλλάξει η εφαρμογή την κατάσταση. Παράδειγμα αν θέσουμε το χρώμα που θα χρησιμοποιεί η OpenGL για να ζωγραφίσει ένα μοντέλο, το χρώμα θα παραμείνει στη μνήμη και θα χρησιμοποιείται μέχρι να το αλλάξουμε ή να κλείσουμε την εφαρμογή.

Εφόσον λοιπόν η εφαρμογή καθορίσει το περιβάλλον που θα χρησιμοποιήσει η OpenGL για να απεικονίσει το μοντέλο μας (χρώματα, υφές, πηγές φωτός, κάμερα κλπ), περνάει στο πρώτο στάδιο κατά το οποίο θα μετασχηματίσει και θα φωτίσει (transform and lighting) τα σημεία(vertices) του μοντέλου.

Στην συνέχεια η OpenGL περνά στο στάδιο της ψηφιοποίησης το οποίο λαμβάνει όλες τις πληροφορίες και την γεωμετρία από το προηγούμενο στάδιο (του μετασχηματισμού) και παράγει την τελική, ψηφιακή, εικόνα. Η εικόνα αντιγράφεται στην μνήμη του frame buffer και φτάνει έτσι στην οθόνη του υπολογιστή.

> **Αυτή η ακολουθία βημάτων που ακολουθεί η OpenGL για να απεικονίσει το μοντέλο λέγεται graphics pipeline (διασωλήνωση)**

Τι είναι το GLUT

Η OpenGL όπως είπαμε είναι ένας γρήγορος και ευέλικτος τρόπος να επικοινωνούμε με το hardware γραφικών του υπολογιστή χωρίς να ενδιαφερόμαστε για τις λεπτομέρειες υλοποίησης του. Όμως δεν προσφέρει καθόλου λειτουργίες GUI (**G**raphical **U**ser **I**nterface), δηλαδή δεν έχει την δυνατότητα να ανοίξει και να κλείσει παράθυρα στο λειτουργικό σύστημα, να ζωγραφίσει σε αυτά, ούτε να καταλάβει το πάτημα ενός πλήκτρου ή την κίνηση του ποντικιού, ούτε μπορεί να διαβάσει ένα αρχείο από το δίσκο.

Αυτό έγινε επί σκοπού, μιας και η OpenGL σχεδιάστηκε να τρέχει σε πολλά λειτουργικά συστήματα τα οποία έχουν το δικό τους τρόπους επικοινωνίας με την οθόνη, το ποντίκι, το δίσκο, το πληκτρολόγιο.

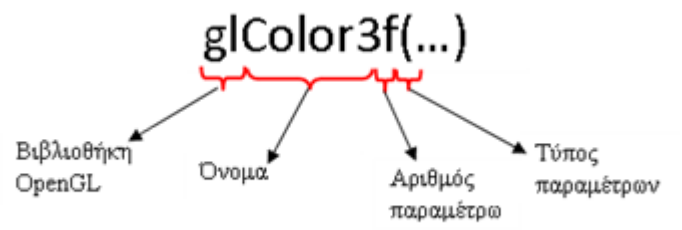
Για να γίνει αυτό θα πρέπει να χρησιμοποιήσουμε απευθείας τις εντολές του λειτουργικού μας συστήματος (Win32 εντολές στην περίπτωση των Windows). Εναλλακτικά μπορούμε να χρησιμοποιήσουμε μια από τις έτοιμες βιβλιοθήκες εντολών που υπάρχουν και που θα κάνουν αυτές τις λειτουργίες για εμάς εύκολα.

Μια από τις πιο διαδεδομένες βιβλιοθήκες για αυτό το σκοπό είναι το GLUT (Open**GL** **U**tility **T**oolkit) το οποίο είναι και αυτό σχεδιασμένο να τρέχει σε πολλά λειτουργικά συστήματα.

Το GLUT προσφέρει ένα σύνολο εντολών που αναλαμβάνουν να ανοίξουν και να κλείσουν εύκολα παράθυρα, να καταγράψουν το πάτημα ενός πλήκτρου ή την κίνηση του ποντικιού. Δεν είναι κατάλληλο να γράψουμε μια κανονική εφαρμογή με αυτό (πχ ένα παιχνίδι) όμως είναι πολύ κατάλληλο για εκπαιδευτικές και πρότυπες εφαρμογές.

Μορφή εντολών OpenGL/GLUT

Η OpenGL χρησιμοποιεί μια απλή, στάνταρντ, μορφή ονοματολογίας για τις εντολές της. Όλες οι εντολές της έχουν μορφή παρόμοια με:



Οι εντολές του GLUT δεν έχουν κάποιο ιδιαίτερο χαρακτηριστικό εκτός του ότι κάθε εντολή ξεκινά με το πρόθεμα glut:

```
glutInitWindowSize(Width, Height);
```

Τύποι δεδομένων OpenGL

Χάριν ομοιομορφίας, και για να μπορέσει να υποστηρίξει διαφορετικές πλατφόρμες, η OpenGL ορίζει μια σειρά από τύπους δεδομένων οι κυριότεροι των οποίων φαίνονται στο παρακάτω πίνακα:

Τύπος δεδομένων OpenGL	**Ορίζεται ως**	**Αντιστοιχία με τύπο δεδομένων στη C**	**Επίθεμα**
GLbyte	Ακέραιος 8-bit	signed char	b
GLshort	Ακέραιος 16-bit	short	s
GLint	Ακέραιος 32-bit	int	i
GLdouble	Κινητής υποδιαστολής 64-bit	double	d
GLubyte	Θετικός ακέραιος 8-bit	unsigned char	ub
GLushort	Θετικός ακέραιος 16-bit	unsigned short	us
GLuint	Θετικός ακέραιος 32-bit	unsigned int	ui
GLchar	Χαρακτήρας 8-bit	char	-
GLfloat	Κινητής υποδιαστολής 32-bit	float	f
GLboolean	Θετικός ακέραιος 8-bit	unsigned char	

Η OpenGL εγγυάται ότι για παράδειγμα ένας GLShort αριθμός θα έχει εύρος 16-bit ανεξάρτητα αν το λειτουργικό σύστημα είναι Windows, Unix ή κάποιο άλλο.

Το επίθεμα είναι το γράμμα που βρίσκουμε στο τέλος μιας OpenGL εντολής και που δείχνει τον τύπο δεδομένων των παραμέτρων.

Γνωριμία με την OpenGL

Στην πρώτη εργαστηριακή άσκηση θα γνωρίσουμε το περιβάλλον ανάπτυξης που θα χρησιμοποιήσουμε για την εισαγωγή στην OpenGL και θα μελετήσουμε τον τρόπο λειτουργίας ενός απλού προγράμματος που χρησιμοποιεί την OpenGL και το GLUT για να ζωγραφίσει ένα αντικείμενο στην οθόνη.

Περιγραφή εφαρμογής

Όλο το πρόγραμμα βρίσκεται μέσα στο αρχείο main.cpp. Ξεκινώντας με την συνάρτηση main που είναι και η είσοδος της εφαρμογής πρέπει να αρχικοποιήσουμε το Glut καλώντας τις συναρτήσεις

```
glutInit(&argc, argv);
glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGBA);
```

Η glutInitDisplayMode() δέχεται μια πληθώρα επιλογών για την αρχικοποίηση της OpenGL που είναι πέρα από τους σκοπούς του πρώτου εργαστηρίου να εξετάσουμε. Στην συγκεκριμένη περίπτωση απλά επιλέγουμε το μοντέλο χρώματος που θα χρησιμοποιήσουμε (Red – Green – Blue και διαφάνεια) και ότι θα χρησιμοποιήσουμε διπλό buffer για την απεικόνιση.

> **Η χρήση διπλού buffer (double buffering) κατά την απεικόνιση μιας σκηνής είναι μια βασική μέθοδος αποφυγής «τρεμοπαίγματος»(flickering). Ο τρόπος με τον οποίο δουλεύει είναι ορίζοντας 2 ενδιάμεσες περιοχές στην μνήμη (buffers) στις οποίες αποθηκεύουμε τις εικόνες που δημιουργούμε. Όσο η οθόνη παίρνει (απεικονίζει) την εικόνα από τον έναν buffer εμείς ζωγραφίζουμε την σκηνή στον άλλον. Έπειτα ανταλλάσουμε τους ρόλους των buffer και συνεχίζουμε. Έτσι η οθόνη δείχνει πάντα μια ολοκληρωμένη εικόνα της σκηνής.**

Στην συνέχεια δημιουργούμε το παράθυρο που θα απεικονίσουμε την τελική εικόνα με το σετ εντολών:

```
glutInitWindowSize(800, 600);
glutInitWindowPosition(150,150);
glutCreateWindow("Computer Graphics - Lab 1");
```

Το παράθυρο θα έχει μέγεθος 800×600 pixels, θα τοποθετηθεί 150,150 pixels από την πάνω αριστερή γωνία της οθόνης και το τίτλο που δηλώσαμε.

Το παρακάτω σετ εντολών μας δίνουν μια ιδέα για το πώς δουλεύει ουσιαστικά το GLUT:

```
glutDisplayFunc(renderScene);
glutKeyboardFunc(keyPressFunc);
glutIdleFunc(renderScene);
```

Οι εντολές αυτές καθορίζουν τι θα συμβεί σε περίπτωση όταν λάβει χώρα ένα γεγονός που μας ενδιαφέρει και που θέλουμε να επεξεργαστούμε. Τέτοια συμβάντα είναι για παράδειγμα το πάτημα ενός πλήκτρου, η κίνηση του ποντικιού, η εντολή να επανασχεδιάσουμε το παράθυρο λόγω μετακίνησης του.

Το GLUT μας δίνει την δυνατότητα να ορίσουμε τι θέλουμε να γίνεται όταν λαμβάνει χώρα ένα τέτοιο γεγονός μέσω μια σειράς εντολών της μορφής:

```
Glut_OnomaEntolis_Func(συνάρτηση που θα καλείται);
```

Στην εφαρμογή ενδιαφερόμαστε για 3 γεγονότα, όταν πατηθεί κάποιο πλήκτρο, όταν δοθεί εντολή να επανασχεδιαστεί το παράθυρο και όταν το GLUT δεν κάνει τίποτα (είναι δηλαδή idle). Αυτή η στιγμή είναι κατάλληλη για να σχεδιάσουμε το αντικείμενο στην οθόνη.

> **Το GLUT μας επιτρέπει να ορίσουμε συναρτήσεις που θα εξυπηρετούν διάφορα συμβάντα. Αυτός είναι και ο μόνος τρόπος να προσθέσουμε λειτουργικότητα σε μια GLUT εφαρμογή.**

Στην συνέχεια καλούμε την συνάρτηση

```
init();
```

που στην συνέχεια θα καλέσει εντολές της OpenGL και θα κάνει αρχικοποιήσεις που χρειάζεται να γίνουν μόνο μια φορά. Από αυτές τις εντολές προς το παρόν μας ενδιαφέρουν μόνο οι

```
glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
glColor3f(1.0f, 1.0f, 0.0f);
```

Η πρώτη καθορίζει το χρώμα στο οποίο θα καθαρίζουμε (clear) το παράθυρο (όταν δοθεί η εντολή). Το χρώμα καθαρισμού δεν είναι ανάγκη να είναι μαύρο, μπορεί να είναι οτιδήποτε. Η δεύτερη καθορίζει το χρώμα με το οποίο θα «βάψουμε» το αντικείμενο όταν το θα το ζωγραφίσουμε την οθόνη.

> **Το ότι θέτουμε το χρώμα καθαρισμού και χρωματισμού του αντικειμένου μία φορά κατά την αρχικοποίηση της εφαρμογής είναι ένα καλό παράδειγμα λειτουργίας της OpenGL ως μηχανή καταστάσεων (state machine) – μια τιμή που θέτουμε διατηρείται μέχρι να την ξαναλλάξουμε.**

Τέλος καλούμε την συνάρτηση που θα αρχίσει το κυρίως loop της εφαρμογής.

```
glutMainLoop();
```

Από αυτό το loop η εφαρμογή θα βγει μόνο όταν κλείσουμε το παράθυρο ή την εφαρμογή. Στο μεταξύ το GLUT θα τρέχει αυτό το loop ακούγοντας για διάφορα γεγονότα (όπως είσοδο από πληκτρολόγιο, κίνηση ποντικιού, κίνηση joystick) και ανάλογα με το αν έχουμε δηλώσει κάποια συνάρτηση για κάποιο συμβάν (με την χρήση της glutXXXXXFunc()) την καλεί για να το εξυπηρετήσει.

Μια τέτοια συνάρτηση είναι και η

```
void keyPressFunc(unsigned
char key, int x, int y)
```

Το GLUT θα καλέσει την συνάρτηση αυτή όταν πατηθεί κάποιο πλήκτρο. Στην συνάρτηση ελέγχουμε αν έχει πατηθεί το πλήκτρο ESC και αν ναι τερματίζουμε την εφαρμογή.

Η συνάρτηση που θα ζωγραφίσει το αντικείμενο στην οθόνη είναι η

```
renderScene(void);
```

Η συνάρτηση αυτή αρχικά καθαρίζει το τρέχον buffer στο χρώμα το οποίο θέσαμε στην init() με την εντολή glClearColor().

Στην συνέχεια καλεί μια εντολή GLUT για να ζωγραφίσει το αντικείμενο στον buffer.

```
glutSolidTeapot(15.0);
```

με μέγεθος 15. Η εντολή αυτή καλεί πολλές OpenGL εντολές για να δημιουργήσει και να απεικονίσει το μοντέλο της τσαγιέρας. Το GLUT προσφέρει και άλλες εντολές που δημιουργούν μοντέλα όπως< glutSolidCube(), glutSolidCone(), glutSolidTetrahedron().

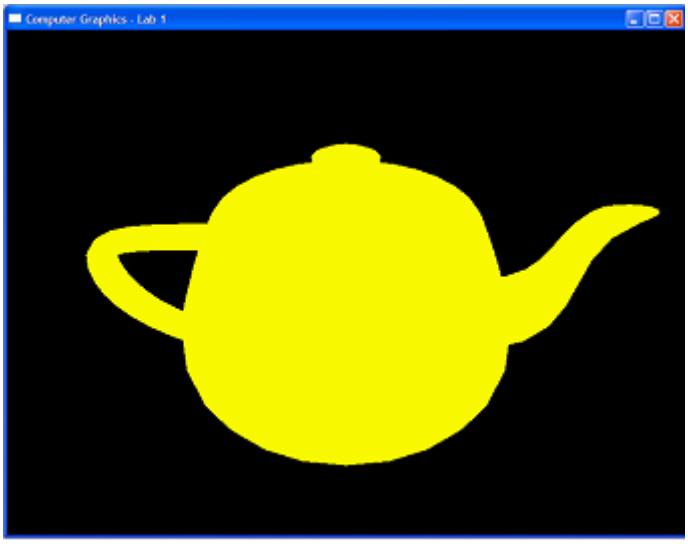
Στο τέλος καλεί την εντολή

```
glutSwapBuffers();
```

η οποία θα αντικαθιστά τον buffer που χρησιμοποιεί η οθόνη με αυτόν που περιέχει το αντικείμενο που μόλις ζωγραφίσαμε.

> **Για να δείτε το double buffering σε λειτουργία δοκιμάστε να σβήσετε αυτή την εντολή και να αντικαταστήστε την παράμετρο GLUT_DOUBLE με GLUT_SINGLE στην εντολή glutInitDisplayMode().**

Η έξοδος του προγράμματος θα πρέπει να είναι η παρακάτω, μια κίτρινη τσαγιέρα:



Κώδικας παραδείγματος

Για να τρέξετε το κώδικα του παραδείγματος πρέπει να εγκαταστήσετε το περιβάλλον ανάπτυξης ***Dev-C++ 5.0 beta 9.2 (4.9.9.2)* (9.0 MB) with Mingw/GCC 3.4.2** και στην συνέχεια να το παραμετροποίησετε ώς εξής:

1. Τρέξτε το Dev-Cpp που μόλις εγκαταστήσατε και μόλις ξεκινήσει, από το μενού Tools, επιλέξτε Check for Updates/Packages.
2. Στο Select devpack server επιλέξτε devpack.org και πατήστε το Check for Updates κάτω αριστερά

3. Σαν Group επιλέξτε OpenGL
4. Από την λίστα που θα εμφανιστεί επιλέξτε να εγκαταστήσετε το πακέτο freeglut

Στην συνέχεια αποσυμπιέστε να αρχεία lab1.zip και shared.zip σε αντίστοιχα folder (με ονόματα dev-cpp/lab1 και dev-cpp/shared δηλαδή), και ανοίξτε το αρχείο lab1.dev του καταλόγου lab1. Αν όλα πήγαν καλά, με F9 ο κώδικας πρέπει να εκτελεστεί και να δείτε το παράθυρο της εφαρμογής με την τσαγιέρα.

Με μικρές αλλαγές ο κώδικας μπορεί να τρέξει σε οποιοδήποτε περιβάλλον προγραμματισμού C/C++ διαθετετε.

Πηγή **Εισαγωγή στην OpenGL: μέρος 1ο**