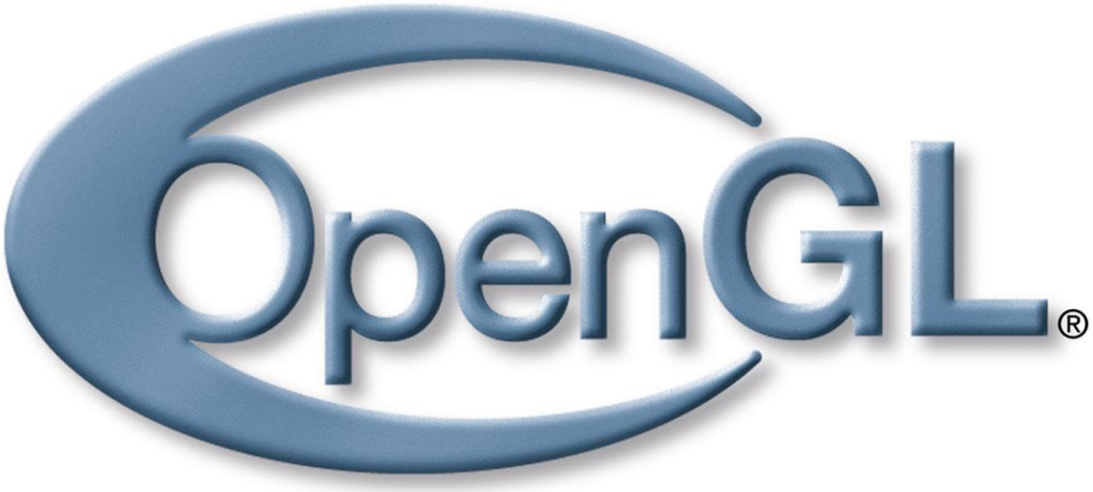
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Δημιουργία 3D εικόνων-σκηνών με C/C++ και OpenGL

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Αρναούτης Ιωάννης

Οκτώβριος 2014



Επιβλέπων καθηγητής: Πασχάλης Ράπτης

Πρόλογος

Η πτυχιακή αυτή εργασία έχει ως σκοπό την εξοικείωση με τον τρόπο λειτουργίας της μηχανής της OpenGL, με απώτερο στόχο τη δημιουργία 3D εικόνων-σκηνών. Η υλοποίηση των εφαρμογών αυτών γίνεται με τη βοήθεια της γλώσσας προγραμματισμού C/C++ και το σύνολο των βιβλιοθηκών της OpenGL που ενσωματώνονται στην γλώσσα αυτή. Αρχικά παραθέτονται κάποια βασικά στοιχεία για της μηχανής της OpenGL, ούτως ώστε να γίνουν κατανοητά και να χρησιμοποιηθούν ως υπόβαθρο κάθε εφαρμογής. Έπειτα στα κεφάλαια που ακολουθούν γίνεται προσπάθεια ανάλυσης διαφόρων εντολών σχεδίασης και τρόπους εμφάνισης της σκηνής, δηλαδή τι και πως θα προβάλλεται στο παράθυρο της εφαρμογής. Η πορεία των κεφαλαίων τείνει να είναι εξελικτική με την έννοια ότι ξεκινούν με απλές υλοποιήσεις της OpenGL πηγαίνοντας σε πιο σύνθετες και ενδιαφέρουσες. Για καλύτερη κατανόηση προστίθενται κάποια απλά παραδείγματα με τον κώδικα και εικόνες με το αποτέλεσμα τους. Ως αποκορύφωση των παραδειγμάτων κρίνεται η τελική εφαρμογή στο τελευταίο κεφάλαιο, στην οποία συνοψίζονται διάφορα στοιχεία σχεδίασης των προηγούμενων κεφαλαίων.

Περίληψη

Η πτυχιακή εργασία εστιάζεται στην δημιουργία 3D εικόνων-σκηνών με C/C++ και της βασικές βιβλιοθήκες της OpenGL. Για να μπορούν να δημιουργηθούν τέτοιες σκηνές πρώτα παρουσιάζονται βασικά στοιχεία και κανόνες της OpenGL, οι οποίοι ισχύουν σε κάθε υλοποίηση της. Αρχικά η εργασία εστιάζεται στην απόδοση απλών δισδιάστατων και τρισδιάστατων γεωμετρικών σχημάτων. Έπειτα γίνεται αναφορά στο πως ο προγραμματιστής μπορεί να χειριστεί διάφορα γεγονότα με κατάλληλες εντολές της OpenGL. Ένα ενδιαφέρον κεφάλαιο είναι οι μετασχηματισμοί οι οποίοι είναι υπεύθυνοι για την αλλαγή της αρχικής κατάστασης των αντικειμένων και την προβολή της σκηνής, όπου ξεχωρίζουν η ορθογραφική προβολή και η προβολή με προοπτική. Για να γίνει πιο ρεαλιστική η απόδοση μιας σκηνής η μηχανή της OpenGL, προσφέρει μια μεγάλη γκάμα εντολών που χειρίζονται το φωτισμό. Οι πιο βασικές περιγράφονται στο κεφάλαιο 6. Τέλος στο κεφάλαιο 7 αναλύεται η υφή (texture) που μπορεί να αποδοθεί σε αντικείμενα διευκολύνοντας πολύ τους προγραμματιστές στην σχεδίαση αντικειμένων του πραγματικού κόσμου και όχι μόνο. Απόρροια όλων αυτών των αναλύσεων είναι η υλοποίηση της 3D εφαρμογής προσέγγισης του ηλιακού συστήματος, συνδυάζοντας αρκετά στοιχεία των κεφαλαίων που προηγήθηκαν.

Abstract

This project focuses on the creation of 3D pictures with C/C++ languages and the basic libraries of OpenGL. To be able to create such scenes firstly presented basics elements and rules of OpenGL, which apply to any implementing. Initially, the study is referred on the performance of simple two-dimensional and three-dimensional geometric shapes. Subsequently, the programmer can handle various events with appropriate commands of OpenGL. An interesting chapter is the transformations that are responsible for changing the initial state of the objects and the projection of the stage, where stand out the orthographic projection and perspective projection. To be more realistic performance of a scene the camera's OpenGL offers a wide range of commands that manipulate the lighting. The most basic are described in Chapter 6. Finally, the Chapter 7 analyzes the texture, which can be assigned to objects facilitating many a programmers to design objects of the real world and beyond. The result of these analyzes is the realization of 3D implementation approach of the solar system, combining several elements of preceding chapters.

Περιεχόμενα

Πρόλογος 2

Περίληψη 3

Abstract 4

1 Εισαγωγή 9

1.1 Περιγραφή της OpenGL 9

1.2 Ιστορικά στοιχεία 9

1.3 OpenGL και ARB Group 10

1.4 Πρώτες εκδόσεις 10

1.5 Επεκτάσεις της OpenGL 11

1.6 Κάποια επιπλέων στοιχεία 11

2 Οι κατηγορίες βιβλιοθηκών της OpenGL 13

2.1 OpenGL core library 13

2.2 OpenGL Utility Library (GLU) 13

2.3 OpenGL Utility Toolkit (GLUT) 13

2.4 freeglut 14

3 Βασικές αρχές και στοιχεία της OpenGL 15

3.1 Τύποι δεδομένων της OpenGL 15

3.2 Μορφή εντολών της OpenGL 16

3.3 Μορφή εντολών της GLUT 16

3.4 Σταθερές της OpenGL 17

3.5 Η μηχανή καταστάσεων (state machine) της OpenGL 17

3.6 Ένα απλό παράδειγμα εφαρμογής της OpenGL 18

3.6.1 Ανάλυση του κώδικα που περιέχεται στη συνάρτηση main. 19

3.6.2 Ανάλυση του κώδικα της συνάρτησης display. 21

3.7 Γεωμετρικά σχήματα στην OpenGL 22

3.7.1 Απλά σχήματα δύο διαστάσεων 23

3.7.2 Σύνθετα σχήματα τριών διαστάσεων 23
3.8 Λίστες απεικόνισης 26
4 Γεγονότα 27
4.1 Κατηγορίες γεγονότων στην OpenGL 27
4.1.1 Γεγονότα σχεδιασμού και επανασχεδιασμού σκηνής 27
4.1.2 Γεγονός μεταβολής διαστάσεων παραθύρου 28
4.1.3 Γεγονότα πληκτρολογίου 28
4.1.4 Γεγονότα ποντικιού 29
4.1.5 Γεγονότα σχεδίασης 29
4.1.6 Γεγονότα χρονισμού 30
4.1.7 “Γεγονότα αδρανείας” 30
4.2 Παράδειγμα με γεγονότα 31
5 Μετασχηματισμοί - Transformations 35
5.1 Μετασχηματισμοί στην OpenGL 35
5.2 Μετασχηματισμοί αντικειμένων 36
5.2.1 Μετατόπιση 36
5.2.2 Περιστροφή 36
5.2.3 Κλιμάκωση 37
5.2.4 Συνδυασμός μετασχηματισμών μοντέλου 37
5.3 Τοποθέτηση κάμερας ή μετασχηματισμοί οπτικής γωνίας 38
5.4 Μετασχηματισμοί προβολής 38
5.4.1 ορθογραφική ή παράλληλη προβολή 39
5.4.2 Προβολή με προοπτική 40
5.5 Στοίβα πινάκων μετασχηματισμού 41
5.6 Παράδειγμα εφαρμογής μετασχηματισμών 42
6 Φωτισμός - Lighting 47
6.1 Πηγές φωτισμού 47

6.2 Κατηγορίες πηγών φωτισμού 48
6.2.1 Σημειακές πηγές 48
6.2.2 Εξασθένηση φωτεινότητας 49
6.2.3 Πηγές σε άπειρη απόσταση 49
6.2.4 Πηγές με κατεύθυνση 50
6.3 Καθολικές παράμετροι φωτισμού 50
6.4 Ιδιότητες υλικού 51
6.5 Παράδειγμα φωτισμού 52
7 Υφή - Texture 55
7.1 Συντεταγμένες υφής 56
7.1.1 Πίνακες συντεταγμένων υφής 56
7.2 Εικόνες υφής 57
7.3 Ρυθμίσεις απόδοσης υφής 59
7.4 Αυτόματη απόδοση υφής σε τετραγωνικές επιφάνειες 59
7.5 Διαχείριση πολλαπλών πινάκων υφής 60
7.6 Συνδυασμός μορφοποιήσεων 60
8 Ολοκληρωμένη εφαρμογή προσομοίωσης ηλιακού συστήματος 62
8.1 Παρουσίαση κώδικα 62
8.1.1 camera.h 62
8.1.2 camera.cpp 63
8.1.3 ImageLoad.h 65
8.1.4 ImageLoad.cpp 66
8.1.5 mainSolarSystem.cpp 68
8.2 Τρέξιμο του κώδικα της εφαρμογής 77
8.2.1 Χειρισμός της κάμερας 79
Συμπεράσματα 81
Βιβλιογραφία 82

1 Εισαγωγή

Στην εισαγωγή παρατίθενται ο ορισμός και διάφορες εγκυκλοπαιδικές πληροφορίες που αφορούν την OpenGL και την πορεία της μέχρι τώρα.

1.1 Περιγραφή της OpenGL

Η OpenGL (Open Graphics Library) είναι ένα πρότυπο υλοποίησης βιβλιοθηκών σχεδίασης 2D/3D γραφικών. Εμπεριέχει δηλαδή ένα σύνολο συναρτήσεων που πρέπει να υλοποιεί μία βιβλιοθήκη γραφικών προκειμένου να είναι συμβατή με αυτό. Μπορεί να χρησιμοποιηθεί από πολλές και διάφορες συσκευές (desktop, laptop, mobile phone) και είναι συμβατό με τα περισσότερα λειτουργικά συστήματα (Windows, Linux/Unix, Mac OS X, Android). Το πρότυπο αυτό λοιπόν, ορίζει μια προγραμματιστική διεπιφάνεια API (Application Programming Interface) για την σχεδίαση γραφικών.

Το API ορίζεται ως μια σειρά συναρτήσεων που μπορούν να καλούνται από το πρόγραμμα πελάτη και παράλληλα από έναν αριθμό που ονομάζεται integer constant (για παράδειγμα, η σταθερά GL_TEXTURE_2D, που αντιστοιχεί στο δεκαδικό αριθμό 3553).

Η OpenGL δεν είναι γλώσσα προγραμματισμού και έτσι υλοποιείται από μια πληθώρα γλωσσών προγραμματισμού όπως η C, C++, Java, Python, Fortran και άλλες.

1.2 Ιστορικά στοιχεία

Η OpenGL αναπτύχθηκε από την Silicon Graphics Inc. (SGI) το 1991 και κυκλοφόρησε τον Ιανουάριο του 1992. Τα πρώτα χρόνια χρησιμοποιούταν για εφαρμογές CAD, εικονικής πραγματικότητας, επιστημονικές εφαρμογές και εξομοιωτές πτήσης, και υποστηριζόταν από επαγγελματικές κάρτες γραφικών η οποίες είχαν πολύ υψηλό κόστος. Το 1997 η OpenGL άρχισε να χρησιμοποιείτε και από παιχνίδια με αποτέλεσμα την κυκλοφορία φθηνών καρτών γραφικών υποστηριζόμενες από την OpenGL.

1.3 OpenGL και ARB Group

Από το 1992 η ανάπτυξη της OpenGL γίνεται αποκλειστικά από το ARB Group (OpenGL Architecture Review Board). Το ARB Group αποτελείται από μεγάλες εταιρίες πληροφορικής όπως η Apple, ATI, Dell, IBM, Intel, nVidia, Sun Microsystems και άλλες. Το ARB Group είναι υπεύθυνο για διάφορες αρμοδιότητες και αποφάσεις που πρέπει να πάρει όπως για την εξέλιξη του API, την έγκριση των προδιαγραφών του, την προώθηση και άλλα.

1.4 Πρώτες εκδόσεις

Η OpenGL αρχικά δημιουργήθηκε ως μία ανοικτή και επαναπροσδιορισμένη εναλλακτική λύση για Iris GL που ήταν αποκλειστικά API γραφικών για τους σταθμούς εργασίας της Silicon Graphics. Αν και η OpenGL ήταν αρχικά παρόμοια σε κάποιες απόψεις με την Iris GL, η έλλειψη μιας επίσημης προδιαγραφής και οι δοκιμές που έγιναν για τη βελτιστοποίηση της Iris GL, καθιστούσαν ακατάλληλη την ευρύτερη υιοθέτηση της. Ο Mark Segal και ο Kurt Akeley συνέγραψαν την OpenGL 1.0 όπου προσπάθησαν να επισημοποιήσουν τη δημιουργία ενός χρήσιμου API γραφικών και την κατασκευή μιας βιώσιμης cross platform. Μία σημαντική παράλειψη από την έκδοση 1.0 του API ήταν η υφή στα αντικείμενα. Η Iris GL είχε ορισμούς και δέσμευε τα στάδια για όλα τα είδη των αντικειμένων, συμπεριλαμβανομένων των υλικών, του φωτισμού, τις υφές και τα περιβάλλοντα της υφής. Η OpenGL αποφύγει αυτά τα αντικείμενα και εντάσσεται υπέρ των σταδιακών αλλαγών της κατάστασης με την ιδέα ότι οι συλλογικές αλλαγές θα μπορούσαν να ενταχθούν στις λίστες της οθόνης. Αυτή έχει παραμείνει η φιλοσοφία με την εξαίρεση ότι τα αντικείμενα υφής (glBindTexture) δεν είναι ξεχωριστό στάδιο ορισμού του βασικού μέρους του API.

Η OpenGL έχει περάσει από μια σειρά αναθεωρήσεων που έχουν κατά κύριο λόγο στοιχειώδης προσθήκες για τις επεκτάσεις του πυρήνα του API και σταδιακά την ενσωμάτωσει στο κύριο σώμα του API. Για παράδειγμα, στην έκδοση OpenGL 1.1, προστίθεται η επέκταση glBindTexture στον πυρήνα του API.

Η OpenGL 2.0 ενσωματώνει τη σημαντική προσθήκη του OpenGL Shading Language (καλείτε και GLSL), μία γλώσσα όπως η C με την οποία η μετατροπή

και τα τμήματα σκίασης των σταδίων της pipeline (αγωγού) μπορεί να προγραμματιστεί.

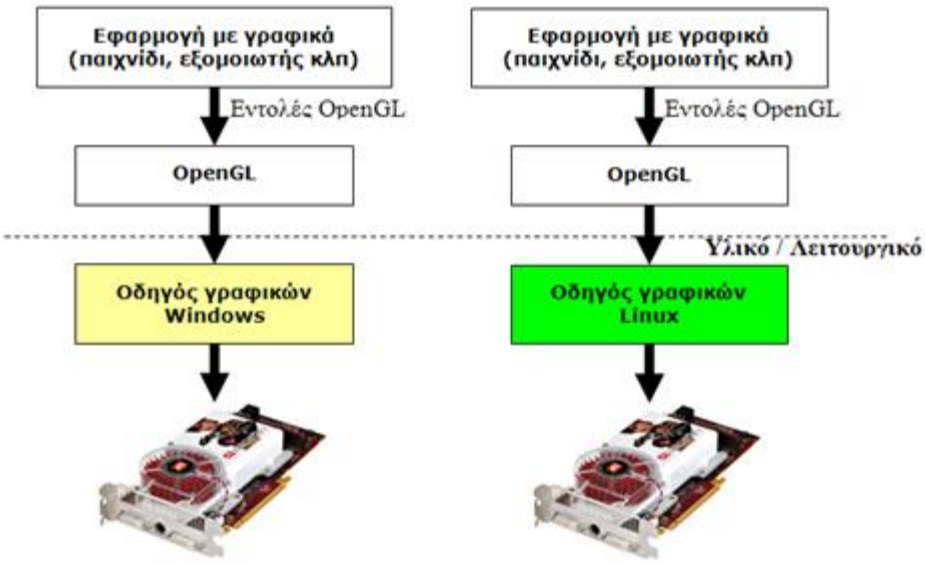
Οι επίσημες εκδόσεις της OpenGL που έχων κυκλοφορήσει μέχρι σήμερα είναι οι: 1.0, 1.1, 1.2, 1.2.1, 1.3, 1.4, 1.5, 2.0, 2.1, 3.0, 3.1, 3.2, 3.3, 4.0, 4.1, 4.2, 4.3, 4.4, 4.5.

1.5 Επεκτάσεις της OpenGL

Ο σχεδιασμός της OpenGL επιτρέπει στους προγραμματιστές να παρέχουν επιπλέον δυνατότητες στο API μέσω των επεκτάσεων (extensions) όταν εμφανίζεται κάποια νέα τεχνολογία. Οι νέες αυτές επεκτάσεις μπορούν να εισάγουν νέες και σταθερές συναρτήσεις, και έχουν τη δυνατότητα να βελτιστοποιήσουν ή να εξαλείψουν περιορισμούς στις ήδη υπάρχουσες συναρτήσεις της OpenGL.

1.6 Κάποια επιπλέων στοιχεία

Η OpenGL προσφέρει πάνω από 500 εντολές για την δημιουργία γραφικών η οποίες συνεχώς αυξάνονται. Επίσης δρα σαν ένα ενδιάμεσο στρώμα ανάμεσα στην εφαρμογή και στη κάρτα γραφικών που αναλαμβάνει να απεικονίσει τα γραφικά στην οθόνη, κρύβοντας λεπτομέρειες υλοποίησης του υλικού και των οδηγών του, όπως φαίνεται και στο επόμενο σχήμα.



Σχήμα 1: Η OpenGL ως ενδιάμεσο στρώμα

Η OpenGL είναι το πρώτο ευρέως διαδεδομένο API γραφικών και υπάρχει ως στερεότυπο από το 1992. Άλλο ένα πασίγνωστο API γραφικών είναι το DirectX της Microsoft το οποίο είναι γραμμένο αποκλειστικά για πλατφόρμες Windows και Xbox/Xbox360.

2 Οι κατηγορίες βιβλιοθηκών της OpenGL

Το κεφάλαιο αυτό αναφέρεται στις κατηγορίες βιβλιοθηκών που χρησιμοποιούνται περισσότερο σε υλοποιήσεις της OpenGL. Με τις βιβλιοθήκες αυτές ο προγραμματιστής έχει τη δυνατότητα να αναπτύξει από απλές έως και αρκετά σύνθετες εφαρμογές γραφικών έχοντας στο δυναμικό του μια πολύ μεγάλη γκάμα εντολών σχεδίασης και απεικόνισης.

2.1 OpenGL core library

Η βασική βιβλιοθήκη της OpenGL περιλαμβάνει της βασικές εντολές σχεδίασης. Οι εντολές της OpenGL core library ξεκινούν με το πρόθεμα gl . Αρκετές από της συναρτήσεις της δέχονται προκαθορισμένα ορίσματα (συμβολικές σταθερές) τα οποία είναι ορισμένα στη βιβλιοθήκη και αντιστοιχούν σε διάφορες παραμέτρους ή καταστάσεις λειτουργίας. Κατά κανόνα, οι σταθερές αυτές ξεκινούν με το πρόθεμα GL_ .

2.2 OpenGL Utility Library (GLU)

Η GLU αποτελείται από μία σειρά λειτουργιών που χρησιμοποιούν τη βασική βιβλιοθήκη OpenGL για την παροχή υψηλότερου επιπέδου σχεδίασης συναρτήσεων, μία από της πιο πρωτόγονες ρουτίνες που παρέχει η OpenGL. Συνήθως παρέχεται μαζί με το πακέτο βάσης OpenGL. Η GLU δεν εμπεριέχεται στην ενσωματωμένη έκδοση του πακέτου OpenGL, OpenGL ES. Η GLU εμπεριέχει συναρτήσεις που εκτελούν σύνθετους αλγορίθμους όπως για παράδειγμα τον καθορισμό πινάκων προβολής και το σχηματισμό σύνθετων καμπυλών και επιφανειών. Κάθε εφαρμογή της OpenGL εμπεριέχει τη βιβλιοθήκη GLU. Οι εντολές της βιβλιοθήκης GLU αρχίζουν με το πρόθεμα glu.

2.3 OpenGL Utility Toolkit (GLUT)

Η OpenGL Utility Toolkit (GLUT) είναι μια βιβλιοθήκη που προσφέρει εντολές εισόδου – εξόδου για να μπορεί ο προγραμματιστής να αλληλεπιδρά με την εφαρμογή. Σε αντίθεση με το πρότυπο OpenGL οι εντολές αυτές δεν είναι ανεξάρτητες της πλατφόρμας και είναι απαραίτητες για τη λειτουργικότητα του

προγράμματος. Η βιβλιοθήκη αυτή περιλαμβάνει βασικές εντολές για απεικόνιση παραθύρων στην οθόνη, δημιουργία μενού, διαχείριση γεγονότων και άλλα. Η GLUT δεν είναι κατάλληλη για την δημιουργία ολοκληρωμένων εφαρμογών ενός παιχνιδιού, είναι ιδανική για εκπαιδευτικές και πρότυπες εφαρμογές. Οι εντολές της GLUT ξεκινούν με το πρόθεμα glut.

2.4 freeglut

Η freeglut είναι μια ανοιχτού κώδικα εναλλακτική λύση της OpenGL Utility Toolkit (GLUT) βιβλιοθήκης. Προορίζεται να είναι μια πλήρης αντικατάσταση της GLUT, και έχει μόνο λίγες διαφορές. Οι εντολές της GLUT καλούνται και από τη freeglut χωρίς καμία αλλαγή. Η freeglut εξελίσσεται συνεχώς βελτιώνοντας την εργαλειοθήκη της, ενώ η GLUT έχει μείνει σε μια στάσιμη κατάσταση. Η freeglut έχει εκδοθεί υπό την άδεια του X Consortium.

3 Βασικές αρχές και στοιχεία της OpenGL

Το κεφάλαιο αυτό είναι μια εισαγωγή στο τρόπο λειτουργίας και της φιλοσοφίας της μηχανής της OpenGL παραθέτοντας κάποιες βασικές αρχές και στοιχεία. Επισημαίνεται ότι η OpenGL λειτουργεί ως μηχανή καταστάσεων και περιγράφονται οι τρόποι με τους οποίους ο προγραμματιστής μπορεί να μεταβάλλει τη λειτουργία του συστήματος γραφικών, ρυθμίζοντας τις παραμέτρους και ενεργοποιώντας ή απενεργοποιώντας προσφερόμενες δυνατότητες.

3.1 Τύποι δεδομένων της OpenGL

Η OpenGL χρησιμοποιεί τους δικούς της τύπους ονοματολογίας δεδομένων, τα οποία είναι συμβατά σε όλα τα συστήματα και το μέγεθος των μεταβλητών είναι ανεξάρτητο του συστήματος. Σε υλοποιήσεις της OpenGL σε C και C++, οι τύποι δεδομένων που υπάρχουν είναι όμοιοι με τους αντίστοιχους που ορίζονται στις γλώσσες C και C++, αλλά και στις περισσότερες γλώσσες προγραμματισμού. Ο Πίνακας 1 παρουσιάζει αντιστοιχίες πρωτογενών τύπων δεδομένων που εμφανίζονται στην OpenGL με τις καθορισμένες γλώσσες C και C++.

Πίνακας 1: Τύποι δεδομένων της OpenGL

Τύπος της OpenGL	Τύπος δεδομένων	Αντίστοιχος τύπος δεδομένων στη C και C++	Επίθεμα
GLbyte	Ακέραιος 8 bits	signed char	**b**
GLshort	Ακέραιος 16 bits	Short	**s**
GLint / GLsizei	Εκτεταμένος ακέραιος 32 bits	int / long	**i**
GLfloat / GLclampf	Κινητής υποδιαστολής	Float	**f**
GLdouble / GLclampd	Κινητής υποδιαστολής διπλής ακρίβειας (64 bits)	double	**d**
GLubyte / GLboolean	Ακέραιος 8 bits χωρίς πρόσημο	unsigned char	**ub**
GLushort	Ακέραιος 16 bits χωρίς πρόσημο	unsigned short	**us**
GLuint / GLenum / GLbitfield	Εκτεταμένος ακέραιος 32 bits χωρίς πρόσημο	unsigned int / unsigned long	**ui**

Όλοι οι τύποι δεδομένων της OpenGL αρχίζουν με το πρόθεμα GL και ακολουθούνται από τον τύπο δεδομένων των παραμέτρων.

3.2 Μορφή εντολών της OpenGL

Οι εντολές-συναρτήσεις γραφικών της OpenGL βιβλιοθήκης αρχίζουν με το πρόθεμα gl και ο αμέσως επόμενος χαρακτήρας ξεκινάει με κεφαλαίο γράμμα.

```
glClear(...), glBegin(...), glEnd(), glFlush()
```

Ορισμένες εντολές γραφικών τελειώνουν με έναν αριθμό n και ένα σύμβολο από τα “i”, ή “f”. Ο αριθμός δηλώνει το πλήθος των ορισμάτων της συνάρτησης, ενώ το σύμβολο τον τύπο δεδομένων (integer και float αντίστοιχα).

Σχήμα 2: Γενική μορφή εντολής

Αν ο χαρακτήρας “i”, ή “f” ακολουθείται από το σύμβολο “v” τότε το όρισμα είναι μια δομημένη μεταβλητή (struct). Ένα χαρακτηριστικό παράδειγμα:

```
struct PointXY {GLfloat x; GLfloat y;}
PointYX p={1.2, 3.4};


glVertex2f(p.x, p.y);
```

ή

```
glVertex2fv( p );
```

3.3 Μορφή εντολών της GLUT

Οι εντολές της GLUT δεν έχουν κάποιο ιδιαίτερο χαρακτηριστικό εκτός του ότι κάθε εντολή ξεκινά με το πρόθεμα glut. Π.χ.

```
glutInit(&argc,argv);
```

3.4 Σταθερές της OpenGL

Οι σταθερές (definitions) που έχουν οριστεί στη βιβλιοθήκη της OpenGL ξεκινούν με το πρόθεμα GL_ με κεφαλαία και στην συνέχεια ακολουθεί η λέξη επίσης στα κεφαλαία. Οι σταθερές χρησιμοποιούνται πολύ συχνά από τις συναρτήσεις ως ορίσματα. Π.χ.

```
GL_PROJECTION, GL_COLOR_BUFFER_BIT, GL_LINES κλπ.
```

3.5 Η μηχανή καταστάσεων (state machine) της OpenGL

Η OpenGL χρησιμοποιεί μια **μηχανή καταστάσεων** (state machine) για να επικοινωνεί με τις εφαρμογές. Ο όρος “μηχανή καταστάσεων” αναφέρεται σε ένα περιβάλλον, το οποίο, σε κάθε χρονική στιγμή, λειτουργεί βάση προκαθορισμένων ιδιοτήτων (attributes) ή αλλιώς μεταβλητών κατάστασης (state variables). Η αρχική τιμή των μεταβλητών κατάστασης είναι προκαθορισμένη, παρόλα αυτά μπορεί να μεταβληθεί κατά την πορεία της εκτέλεσης του κώδικα από τον προγραμματιστή. Επιπλέον, οι αρχικές τιμές τους ή αυτές που τους ανατέθηκαν τελευταία, παραμένουν ενεργές.

Δεδομένου ότι οι αλγόριθμοι στην OpenGL εκτελούνται επαναληπτικά, είναι σημαντικό ο προγραμματιστής να αρχικοποιεί τις μεταβλητές κατάστασης, όποτε αυτό χρειάζεται, καθώς και να παρακολουθεί τις τιμές τους, ούτως ώστε να παράγει το επιθυμητό αποτέλεσμα σε κάθε κύκλο εκτέλεσης.

Μερικά παραδείγματα μεταβλητών κατάστασης που ορίζονται στην OpenGL είναι το χρώμα καθαρισμού της οθόνης, το τρέχον χρώμα σχεδίασης, το πάχος των σχεδιαζόμενων γραμμών, οι τιμές των πινάκων μετασχηματισμού και προβολής και άλλα.

Σε κάθε χρονική στιγμή, η κατάσταση λειτουργίας της OpenGL καθορίζεται από τις τιμές που αναθέτονται σε ένα σύνολο προκαθορισμένων ιδιοτήτων. Οι ιδιότητες μπορούν να διακριθούν σε δύο κατηγορίες, τις ιδιότητες δύο καταστάσεων και τις σύνθετες ιδιότητες κατάστασης.

Οι μεταβλητές της πρώτης κατηγορία μπορούν να πάρουν δύο πιθανές τιμές (TRUE ή FALSE) και καθορίζουν την υποστήριξη ή την μη εξειδικευμένων

λειτουργιών. Οι προκαθορισμένες αυτές τιμές αποδίδονται στην OpenGL με τις GL_TRUE ή GL_FALSE. Η ενεργοποίηση ή απενεργοποίηση αυτών των λειτουργιών γίνεται με τις εντολές glEnable και glDisable.

Προκειμένου να ελεγχθεί αν μια ιδιότητα δύο καταστάσεων είναι ενεργοποιημένη ή απενεργοποιημένη χρησιμοποιείτε η εντολή:

GLboolean glIsEnabled(GLenum property);

η οποία επιστρέφει GL_TRUE ή GL_FALSE αντίστοιχα.

Εκτός από τις μεταβλητές δύο καταστάσεων υπάρχουν και οι μεταβλητές κατάστασης που μπορούν να πάρουν περισσότερες από δύο τιμές. Σε αυτή την περίπτωση δεν μπορούν να χρησιμοποιηθούν οι εντολές glEnable, glDisable και glIsEnabled για να ανατεθεί ή να επιστραφεί η τιμή τους. Αντίθετα η OpenGL χρησιμοποιεί ένα σύνολο εντολών ανάθεσης τιμών. Για τις μεταβλητές πολλαπλών καταστάσεων, έχει καθοριστεί ένα σύνολο από εντολές επισκόπησης (query functions) οι οποίες επιστρέφουν τη τιμή ή τις τιμές που τις χαρακτηρίζουν είναι οι εξής:

void glGetIntegerv(Glenum parameterName, GLint *parameters);

void glGetFloatv(Glenum parameterName, GLfloat *parameters);

void glGetDoublev(Glenum parameterName, GLdouble *parameters);

void glGetBooleanv(Glenum parameterName, GLboolean *parameters);

Το όρισμα parameterName είναι η παράμετρος που εξετάζεται και το όρισμα parameters παραπέμπει στο δείκτη του πίνακα στον οποίο αποθηκεύονται οι τιμές που προσδιορίζουν την εκάστοτε παράμετρο.

3.6 Ένα απλό παράδειγμα εφαρμογής της OpenGL

Το παρακάτω πρόγραμμα είναι ένα απλό παράδειγμα υλοποίησης της OpenGL με την βοήθεια της C++. Οι εντολές που εμφανίζονται υπάρχουν σχεδόν σε κάθε πρόγραμμα OpenGL.

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

```
void display()
{
  glClearColor(1,1,1,1);
  glClear(GL_COLOR_BUFFER_BIT);
  glBegin(GL_LINES);
  glColor3f(1,0,0);
  glVertex2i(20,20);
  glVertex2i(40,40);
  glEnd();
  glFlush();
}
int main(int argc, char** argv)
{
  glutInit(&argc,argv);
  glutInitWindowPosition(50,50);
  glutInitWindowSize(640,480);
  glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
 glutCreateWindow("Sample OpenGL application - Draw a line");
  glMatrixMode(GL_PROJECTION);
  gluOrtho2D(0,50,0,50);
  glutDisplayFunc(display);
  glutMainLoop();
  return 0;
}
```

3.6.1 Ανάλυση του κώδικα που περιέχεται στη συνάρτηση main.

- Πρώτα προστίθεται η κεφαλίδα του GLUT freeglut.h (όπου freeglut αντικαθίσταται με το GLUT, διότι ουσιαστικά είναι το ίδιο). Κάθε πρόγραμμα που χρησιμοποιεί συναρτήσεις της OpenGL, απαιτείται η προσθήκη των κεφαλίδων gl.h και glu.h, ενώ για χρήση της GLUT, απαιτείται η προσθήκη της κεφαλίδας glut.h ή freeglut.h . Ωστόσο, δηλώνοντας την κεφαλίδα glut.h αυτομάτως δηλώνονται και οι δύο προηγούμενες.

- Η glutInit είναι η συνάρτηση που ενεργοποιεί τη βιβλιοθήκη GLUT και μέσω αυτής μπορούν να περαστούν οι παράμετροι στην εφαρμογή από τη γραμμή εντολών του DOS.

- Με την εντολή glutInitWindowPosition καθορίζεται η θέση στην οποία θα εμφανιστεί το παράθυρο της εφαρμογής στην οθόνη (οι συντεταγμένες αφορούν την πάνω αριστερή κορυφή του παραθύρου).

- Η εντολή glutInitWindowSize καθορίζει το μέγεθος του παραθύρου της εφαρμογής σε pixels (default=300x300).

- Η εντολή glutInitDisplayMode καθορίζει τον τρόπο παρουσίασης των γραφικών όπως το χρώμα, το buffering, την απόκρυψη αντικειμένων και άλλα. Οι διάφοροι παράμετροι χωρίζονται μεταξύ τους με το τελεστή or “|” και μπορούν να έχουν ένα ή περισσότερους. Στο παράδειγμά χρησιμοποιείται το χρωματικό μοντέλο RGB (GLUT_RGB) και η εφαρμογή της απλής ενταμίευσης (GLUT_SINGLE).

- Με την εντολή glutCreateWindow δημιουργείται και εμφανίζεται το παράθυρο της εφαρμογής στην οθόνη (αφού προηγηθούν οι εντολές για το μέγεθος, τη θέση κτλ.) και του αποδίδει έναν τίτλο.

- Η εντολή glMatrixMode(GLUint mode) επιλέγει τον πίνακα που θα τροποποιηθεί. Στο παράδειγμα, δίνεται ως όρισμα η σταθερά GL_PROJECTION για προβολή, η οποία καθορίζει τον τρόπο με τον οποίο προβάλλεται η σκηνή στο επίπεδο του θεατή.

- Η εντολή gluOrtho2D(xMin,xMax,yMin,yMax) ενημερώνει ότι θα απεικονιστεί η παράλληλη προβολή της σκηνής στο επίπεδο XY (το οποίο, στο συγκεκριμένο παράδειγμα, είναι ίσο με το παράθυρο της εφαρμογής). Στο περιεχόμενο του παραθύρου αποθηκεύονται όλα τα στοιχεία του σκηνικού που εκτείνονται μεταξύ των συντεταγμένων X=[-50,50] και Y=[-50,50]. Το θετικό τμήμα του άξονα X έχει φορά προς τα δεξιά ενώ ο άξονας Y έχει φορά προς τα πάνω. Θα πρέπει πρώτα να προηγηθεί η εντολή glMatrixMode(…) για να ενεργοποιηθεί αυτή.

- Η εντολή glutDisplayFunc(void function()) ανήκει σε μια ειδική κατηγορία συναρτήσεων της GLUT, οι οποίες αποκαλούνται συναρτήσεις κλήσης (callback functions). Η εντολή αυτή δέχεται ως όρισμα τη συνάρτηση του χρήστη όπου υπάρχει ο κώδικας για την εκτέλεση σχεδίασης των γραφικών. Η συνάρτηση αυτή εκτελείται κάθε φορά που το σύστημα θεωρεί ότι το παράθυρο πρέπει να ανανεωθεί. Είναι μια void συνάρτηση και δεν έχει ορίσματα. Στο παράδειγμα η glutDisplayFunc δίνεται ως όρισμα την συνάρτηση display().

- Με την εντολή glutMainLoop() ενεργοποιείται ο κύκλος διαχείρισης γεγονότων (event processing loop). Στον κύκλο αυτό, η εφαρμογή αναμένει επ' άπειρον και ανταποκρίνεται σε γεγονότα, όπως για παράδειγμα στο πάτημα ενός κουμπιού, στην αλλαγή του σκηνικού ή στην κίνηση του ποντικιού. Η εντολή αυτή ανήκει στη βιβλιοθήκη GLUT όπως όλες οι εντολές που αρχίζουν με glut.

3.6.2 Ανάλυση του κώδικα της συνάρτησης display.

- Ο καθορισμός του χρώματος που χρησιμοποιείται κάθε φορά που εκτελείται εντολή καθαρισμού της οθόνης γίνεται με την εντολή glClearColor(). Το χρώμα του φόντου στην OpenGL είναι μία μεταβλητή κατάσταση, η οποία διατηρεί την τιμή που της ανατέθηκε την τελευταία φορά. Το χρώμα αποδίδεται από τα βάρη του στο χρωματικό μοντέλο RGB. Στο παράδειγμα, χρησιμοποιείται το λευκό χρώμα.

- Με την εντολή glClear() καθαρίζονται οι ενταμιευτές (buffers), που αφορούν συγκεκριμένες περιοχές μνήμης του συστήματος γραφικών (frame buffer). Η μηχανή γραφικών της OpenGL ορίζει συγκεκριμένες κατηγορίες ενταμιευτών. Με τη σταθερά GL_COLOR_BUFFER_BIT δίνεται εντολή να καθαριστεί ο ενταμιευτής χρωματικών τιμών (colour buffer). Η σταθερά αυτή περιέχει τις χρωματικές τιμές των pixels που απεικονίζονται ή πρόκειται να απεικονιστούν στην οθόνη.

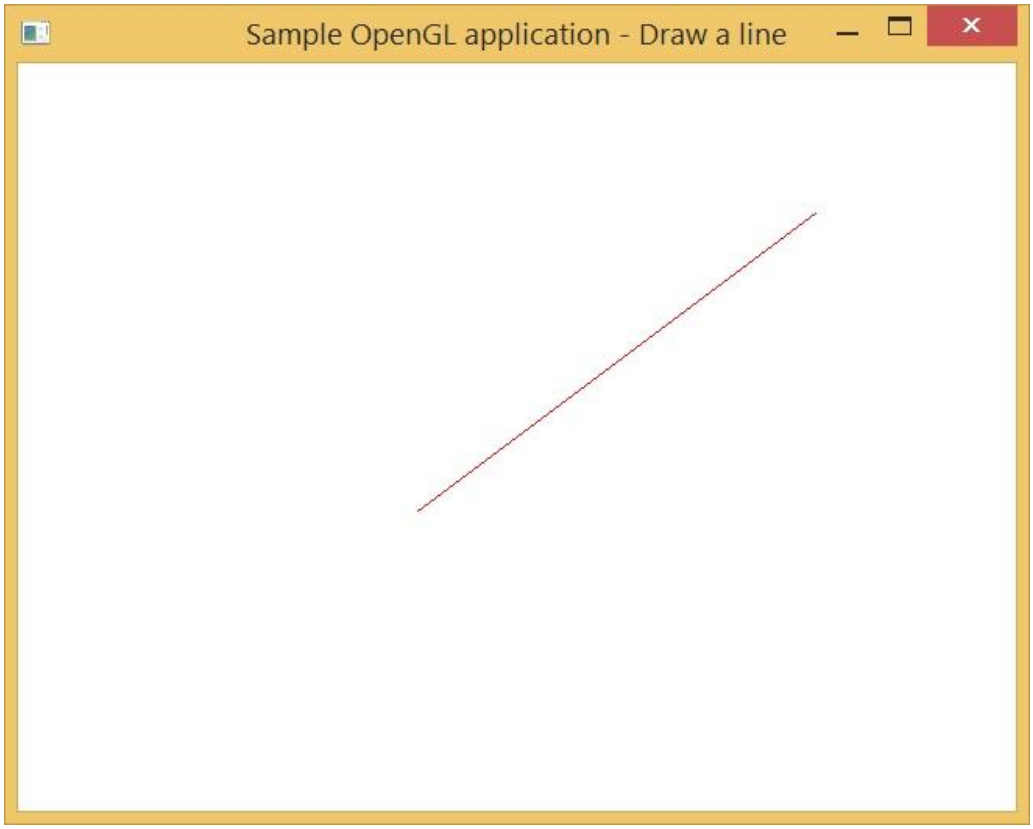
- Η εντολή glColor3f(float r, float g, float b) ορίζει το τρέχον χρώμα σχεδίασης. Η εντολή αυτή είναι μία ακόμη μεταβλητή κατάστασης που καθορίζει το χρώμα που χρησιμοποιείται για τη σχεδίαση γραφικών. Στην εντολή περνάμε ως παραμέτρους τις κανονικοποιημένες ως προς τη μονάδα τιμές των συνιστωσών του κόκκινου, πράσινου και μπλε χρώματος. Στο συγκεκριμένο παράδειγμα επιλέγεται ως χρώμα σχεδίασης της γραμμής το κόκκινο.

- Με την εντολή glBegin(GLEnum MODE) δηλώνεται η έναρξη ορισμού ενός ή περισσοτέρων γεωμετρικών σχημάτων. Ανάλογα με την παράμετρο, μπορεί να προσδιοριστεί μια ποικιλία σχημάτων. Στο παράδειγμά, ορίζονται ευθύγραμμα τμήματα βάζοντας ως παράμετρο τη σταθερά GL_LINES. Η εντολή glBegin εκτελείται πάντα σε συνδυασμό με την εντολή glEnd() και η δεύτερη ορίζει τη λήξη της επιλεγμένης ρύθμισης που αφορά τη σχεδίαση.

- Η εντολή glVertex2i ορίζει σημεία του δισδιάστατου χώρου. Εφόσον έχει προεπιλεγεί η κατάσταση σχεδίασης ευθυγράμμων τμημάτων όπως στο παράδειγμα, τα σημεία ορίζουν ανά ζεύγη τα ευθύγραμμα τμήματα. Στο παράδειγμα σχεδιάζεται ένα ευθύγραμμο τμήμα κόκκινου χρώματος με συντεταγμένες αρχής και τέλους (20,20) (40,40).

- Τέλος η εντολή glFlush() εξαναγκάζει την εκτέλεση των εντολών που εκκρεμούν.

Το παρακάτω σχήμα εμφανίζει την εκτέλεση του κώδικα.



Σχήμα 3: Παράθυρο του παραδείγματος

3.7 Γεωμετρικά σχήματα στην OpenGL

Η OpenGL διαθέτει στην βιβλιοθήκη γραφικών της μια μεγάλη γκάμα γεωμετρικών σχημάτων που μπορεί να υλοποιήσει, καθώς επίσης και διάφορες ιδιότητες για κάθε γεωμετρικό σχήμα. Τα σχήματα αυτά μπορούν να είναι δισδιάστατα και τρισδιάστατα.

3.7.1 Απλά σχήματα δύο διαστάσεων

Ορισμένα πρωταρχικά είδη των γεωμετρικών σχημάτων που μπορούν να σχεδιαστούν στην OpenGL εξαρτώνται από την παράμετρο που δίνεται στο όρισμα της εντολής glBegin. Τα σχήματα αυτά ανήκουν στο άξονα των δύο διαστάσεων. Ο παρακάτω πίνακας δείχνει τις παραμέτρους που δέχεται ως ορίσματα η glBegin( ).

Πίνακας 2: Παράμετροι της glBegin()

Παράμετρος	Σημασία
GL_POINTS	Σημεία ανεξάρτητα μεταξύ τους.
GL_LINES	Ζεύγη κορυφών που αποδίδουν ξεχωριστά ευθύγραμμα τμήματα.
GL_POLYGON	Ορισμός ενός απλού κυρτού πολυγώνου.
GL_TRIANGLES	Τριάδες κορυφών που αποδίδονται ως τρίγωνα.
GL_QUADS	Τετράδες κορυφών που αποδίδονται ως πολύγωνα τεσσάρων πλευρών.
GL_LINE_STRIP	Ορίζονται ως σειρές από συνδεδεμένα ευθύγραμμα τμήματα.
GL_LINE_LOOP	Το ίδιο με το GL_LINE_STRIP με τη διαφορά ότι προστίθεται ένα ευθύγραμμο τμήμα μεταξύ της τελευταίας και της πρώτης κορυφής.
GL_TRIANGLE_STRIP	Ενωμένα τρίγωνα σε σειρά
GL_TRIANGLE_FAN	Ενωμένα τρίγωνα σε έλικα
GL_QUAD_STRIP	Ενωμένα πολύγωνα τεσσάρων πλευρών σε σειρά.

Από την άλλη μεριά υπάρχουν σχήματα που ορίζονται από έτοιμες συναρτήσεις όπως τα ορθογώνια και άλλα πιο περίπλοκα όπως ο κύκλος που ορίζονται από τον προγραμματιστή.

3.7.2 Σύνθετα σχήματα τριών διαστάσεων

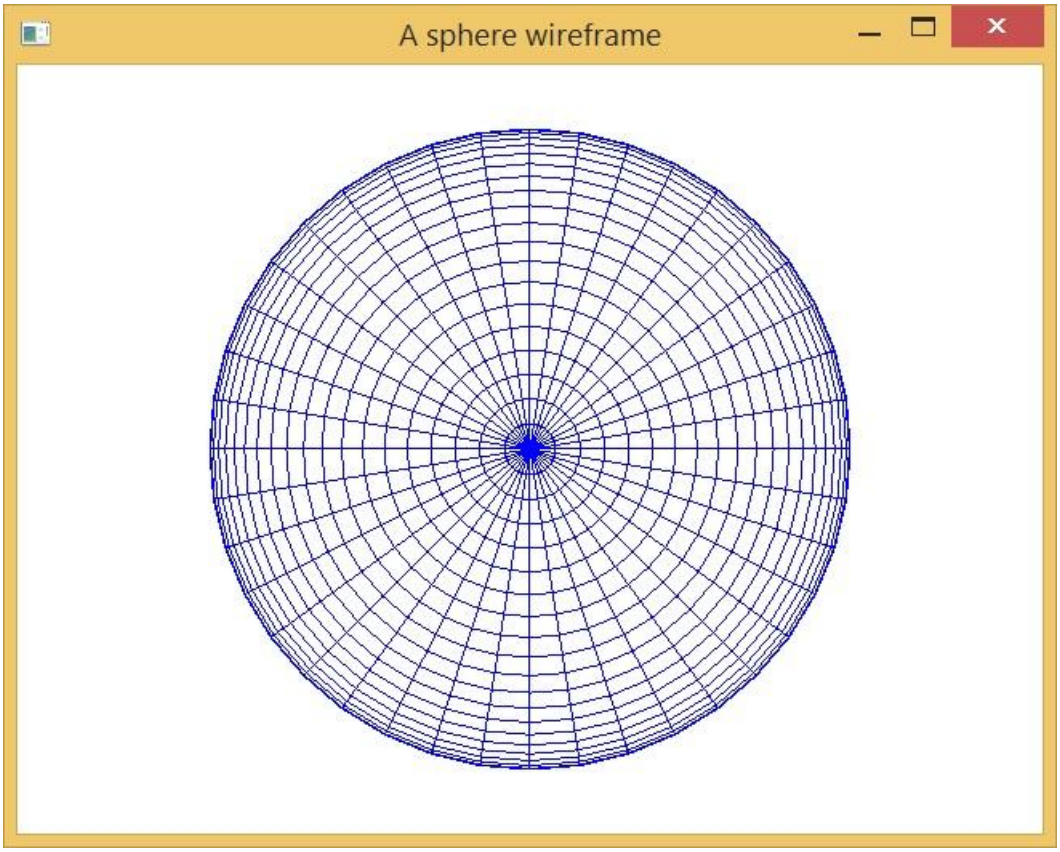
Εκτός από τα δισδιάστατα σχήματα η βιβλιοθήκης της OpenGL συμπεριλαμβάνει στην συλλογή της εντολές για άμεση σχεδίαση τρισδιάστατων, πιο πολύπλοκων αντικειμένων όπως κύβους, σφαίρες, κώνους, κυλίνδρους και άλλα. Το κάθε σχήμα μπορεί να σχεδιαστεί ως περίγραμμα ή ως συμπαγή επιφάνεια, επίσης ο προγραμματιστής μπορεί να ορίσει το δικό του αντικείμενο για την απόδοση επιφάνειας σε κάποιο σχήμα. Τα τρισδιάστατα αυτά αντικείμενα δε χρειάζεται να περικλείονται από εντολές όπως η glBegin ούτε να οριστούν οι κορυφές τους, απλά καλούνται παίρνοντας ορίσματα σχετικά με το μέγεθος, τη

λεπτομέρεια και άλλα. Ένα ενδεικτικό παράδειγμα σχεδίασης μιας σφαίρας είναι το παρακάτω.

```
#include <GL/freeglut.h>
void display()
{
  //ορίζει το τρέχον χρώμα σχεδίασης
  glColor3f(0,0,1);

  /*καθορίζει το χρώμα κάθε φορά που εκτελείτε η εντολή
  καθαρισμού οθόνης*/
  glClearColor(1,1,1,0);

  //καθορίζει τον ενταμιευτή χρωματικών τιμών
  glClear(GL_COLOR_BUFFER_BIT);

  /*σχεδίαση σφαιρικού πλέγματος το πρώτο όρισμα αντιστοιχεί
στην ακτίνα της σφαίρας. Τα υπόλοιπα αντιστοιχούν στα slices
και stacks αντίστοιχα. Η γενική μορφή της εντολής είναι:
  void glutWireSphere( GLdoule radius, GLint slices, GLint
  stacks ); */
  glutWireSphere(40,40,40);

  /*η εντολή αυτή εξαναγκάζει την εκτέλεση
των εντολών που εκκρεμούν*/
  glFlush();

}
  int main(int argc, char **argv)
{
  //ενεργοποίηση της βιβλιοθήκης GLUT
  glutInit(&argc,argv);

  /*καθορισμός της θέσης που θα εμφανιστεί το παράθυρο
στην οθόνη*/
  glutInitWindowPosition(50,50);

  //καθορισμός μεγέθους του παραθύρου
  glutInitWindowSize(640,480);

  /*καθορισμός διπλού ενταμιευτή έναν για την εφαρμογή
απλής ενταμίευσης και ένα για το καθορισμό του
χρωματικού μοντέλου RGB */
  glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);

  /*εντολή εμφάνισης του παραθύρου με τον αναγραφόμενο
  τίτλο*/
  glutCreateWindow("A sphere wireframe");

  /*επιλογή του GL_PROJECTION μητρώου για τροποποίηση
```

```
το οποίο αντιστοιχεί τον τρόπο με τον οποίο θα προβάλλεται η
σκηνή στο επίπεδο του θεατή */
  glMatrixMode(GL_PROJECTION);

  /*διευκρινίζει ότι θα εφαρμοστεί η παράλληλη προβολή
της σκηνής */
  glOrtho(-64,64,-48,48,0,100);

  /*Η συνάρτηση δέχεται ως όρισμα τη συνάρτηση που περιέχει
  τον κώδικα σχεδίασης γραφικών και καλείται κάθε φορά που
  απαιτείται επανασχεδιασμός της σκηνής */
  glutDisplayFunc(display);

  /*Ενεργοποιεί τον κύκλο διαχείρισης γεγονότων. Στον κύκλο
αυτό η εφαρμογή αναμένει επ' άπειρον και
ανταποκρίνεται στα γεγονότα*/
  glutMainLoop();
  return 0;
}
```

Το αποτέλεσμα της εκτέλεσης του κώδικα φαίνεται στο παρακάτω σχήμα.



Σχήμα 4: Παράθυρο που εμφανίζει η εκτέλεση του παραπάνω κώδικα

3.8 Λίστες απεικόνισης

Οι λίστες απεικόνισης είναι χρήσιμες για την περιγραφή σύνθετων σχημάτων και εκτελούνται για την σχεδίαση αυτών. Μια λίστα απεικόνισης περικλείεται μεταξύ δύο εντολών της glNewList και της glEndList. Κάθε λίστα απεικόνισης έχει το μοναδικό “αναγνωριστικού αριθμού” της (list identifier). Προκειμένου να αποδοθούν οι αναγνωριστικοί αυτοί αριθμοί σε λίστες απεικόνισης, πρέπει να δεσμευτεί το απαιτούμενο εύρος τιμών. Η δέσμευση αυτή γίνεται με την εντολή GLuint glGenLists(GLint range);

όπου η τιμή του ορίσματος range αντιστοιχεί στο πλήθος των αναγνωριστικών που χρησιμοποιούνται. Η glGenLists επιστρέφει μια ακέραιη τιμή που αντιστοιχεί στον πρώτο αναγνωριστικό αριθμό.

Ο ορισμός μια νέας λίστα απεικόνισης γίνεται με την εντολή glNewList:

void glNewList(GLuint listID, GLenum listMode);

όπου η τιμή του όρισματος listID το αναγνωριστικό που θα αποδοθεί στη λίστα απεικόνισης. Η παράμετρος listMode έχει δύο πιθανές τιμές. Η τιμή GL_COMPILE όπου δηλώνεται ο κώδικας σχεδιασμού του σύνθετου αντικειμένου που περιγράφεται στη λίστα απεικόνισης και η GL_COMPILE_AND_EXECUTE όπου δηλώνει και εκτελεί ταυτόχρονα τον κώδικα σχεδιασμού που περιέχεται στη λίστα απεικόνισης.

4 Γεγονότα

Ο όρος γεγονός στον κόσμο της πληροφορική σημαίνει την καταγραφή κάποιας δραστηριότητας του συστήματος. Υπάρχουν γεγονότα που αφυπνίζονται από κάποια συσκευή εισόδου όπως ένα πληκτρολόγιο ή ένα ποντίκι, καθώς επίσης και γεγονότα που εγείρονται από το λειτουργικό σύστημα υπό ορισμένες συνθήκες. Μία καταγραφή γεγονότος περιέχει της απαραίτητες πληροφορίες που χρειάζονται για να είναι επαρκώς προσδιορισμένο. Όπως για παράδειγμα μια καταγραφή γεγονότος από το ποντίκι περιέχει πληροφορίες για το πλήκτρο του ποντικιού που πιέστηκε και την τοποθεσία του δείκτη στην οθόνη όταν πιέστηκε.

4.1 Κατηγορίες γεγονότων στην OpenGL

Η OpenGL δεν ήταν δυνατόν να μην συμπεριλάβει στην βιβλιοθήκη της εντολές διαχείρισης γεγονότων, όπου καθιστούν δυνατή την αλληλεπιδράση χρήστη - εφαρμογής στην οποία οφείλεται η μεγάλη αύξηση των δυνατοτήτων της καθώς και της λειτουργικότητας της. Οι κατηγορίες γεγονότων που περιλαμβάνονται στην OpenGL είναι τα γεγονότα σχεδιασμού και επανασχεδιασμού της σκηνής, γεγονότα μεταβολής διαστάσεων παραθύρου, πληκτρολογίου, ποντικού, που αφορούν την σχεδίαση, γεγονότα χρονισμού, “γεγονότα αδράνειας”.

4.1.1 Γεγονότα σχεδιασμού και επανασχεδιασμού σκηνής

Όλες οι συναρτήσεις σχεδιασμού σκηνής μπαίνουνε ως όρισμα στην συνάρτηση glutDisplayFunc. Οι συναρτήσεις σχεδιασμού δεν επιστρέφουν τιμή και δεν έχουν όρισμα. Για τον επανασχεδιασμό της σκηνής χρησιμοποιείται η εντολή glutPostRedisplay η οποία δεν παίρνει ορίσματα και δεν επιστρέφει κάτι επίσης.

Γενική μορφή εντολής:

void glutDisplayFunc(void display());

όπου display η συνάρτηση σχεδιασμού της σκηνής.

4.1.2 Γεγονός μεταβολής διαστάσεων παραθύρου

Το γεγονός μεταβολής διαστάσεων παραθύρου (reshape) ενεργοποιείται την πρώτη φορά που σχεδιάζεται το παράθυρο της εφαρμογής καθώς και κάθε φορά που μεταβάλλονται οι διαστάσεις του παραθύρου από τον χρήστη.

Γενική μορφή εντολής:

void glutReshapeFunc(void reshape(int width, int height));

Η συνάρτηση glutReshapeFunc δέχεται ως όρισμα μια συνάρτηση μεταβολής διαστάσεων παραθύρου π.χ. την reshape η οποία έχει ως ορίσματα το πλάτος και το ύψος του παραθύρου της εφαρμογής.

4.1.3 Γεγονότα πληκτρολογίου

Τα γεγονότα πληκτρολογίου (keyboard events) εμφανίζονται κάθε φορά που πιέζεται ένα πλήκτρο του πληκτρολογίου. Τα πλήκτρα διαχωρίζονται σε δύο κατηγορίες, τα πλήκτρα χαρακτήρων και τα ειδικά πλήκτρα (Function keys, Page Up, Page Down, Insert κλπ.), αντίστοιχα ορίζονται και δύο κατηγορίες γεγονότων πληκτρολογίου.

Γενική μορφή εντολής πλήκτρων χαρακτήρων:

void glutKeyboardFunc(void keyboard(unsigned char key, int x, int y));

όπου το όρισμα key (τύπου unsigned char) περιέχει τον χαρακτήρα που αντιστοιχεί στο πλήκτρο που πιέστηκε και τα ορίσματα x, y προσδιορίζουν τη θέση στην επιφάνεια σχεδίασης, όπου βρισκόταν ο δείκτης του ποντικιού, όταν πιέστηκε το πλήκτρο.

Γενική μορφή εντολής ειδικών πλήκτρων:

void glutSpecialFunc(void specialKey(int key, int x, int y));

ισχύουν τα ίδια με την παραπάνω συνάρτηση απλά στο key αντί για το χαρακτήρα αντιστοιχεί κάποιος ακέραιος αριθμός για κάθε ειδικό χαρακτήρα.

4.1.4 Γεγονότα ποντικιού

Τα γεγονότα ποντικού (mouse events) εγείρονται κατά τη χρήση του ποντικιού. Στην κατηγορία αυτή ανήκει η ενεργή κίνηση του δείκτη (κίνηση του δείκτη του ποντικιού με ένα από τα πλήκτρα του πιεσμένο), η παθητική κίνηση (κίνηση του δείκτη του ποντικιού χωρίς πιεσμένο πλήκτρο) και η πίεση των πλήκτρων του ποντικιού.

Γενική μορφή εντολής πίεσης των πλήκτρων του ποντικιού:

void glutMouseFunc(void mouseClicked(int button, int state, int x, int y));

Όπου η τιμή του ορίσματος button που επιστρέφεται από το λειτουργικό σύστημα εξαρτάται από το κουμπί του ποντικιού που πιέστηκε. Το αριστερό, το μεσαίο και το δεξί αντιστοιχούν στις σταθερές GL_LEFT_BUTTON, GL_MIDDLE_BUTTON και GL_RIGHT_BUTTON. Η τιμή του ορίσματος state εξαρτάται από το αν το κουμπί του ποντικιού πιέστηκε ή απελευθερώθηκε GLUT_DOWN ή GLUT_UP. Ενώ τα ορίσματα x, y εκφράζουν τη θέση του δείκτη του ποντικιού όταν πιέστηκε το κουμπί.

Γενική μορφή εντολής ενεργής μετακίνησης του δείκτη του ποντικιού:

void glutMotionFunc(void activeMouseMotion(int x, int y));

Τα ορίσματα x, y ορίζουν τις συντεταγμένες του σημείου όπου σύρεται ο δείκτης.

Γενική μορφή εντολής ενεργής μετακίνησης του δείκτη του ποντικιού:

void glutPassiveMotionFunc(void passiveMouseMotion(int x, int y));

Τα ορίσματα x, y ορίζουν τις συντεταγμένες του σημείου όπου σύρεται ο δείκτης.

4.1.5 Γεγονότα σχεδίασης

Τα γεγονότα σχεδίασης εμφανίζονται κάθε φορά που το σύστημα διαπιστώνει ότι απαιτείται σχεδίαση ή επανασχεδιασμός της σκηνής. Τέτοια γεγονότα σχεδίασης αφυπνίζονται:

i. όταν μετακινείται το παράθυρο της εφαρμογής στην οθόνη
ii. όταν επαναφέρεται το παράθυρο στο προσκήνιο (restore)
iii. όταν ο προγραμματιστής δώσει εντολή επανασχεδιασμού της σκηνής

Γενική μορφή εντολής:

void glutIdleFunc(void idle);

Η εντολή δέχεται ως όρισμα τη συνάρτηση idle που είναι δηλωμένη ως “callback”. Η συνάρτηση αυτή δεν έχει ορίσματα και εκτελείται σε κάθε κύκλο της CPU όταν δεν υπάρχουν γεγονότα προς επεξεργασία.

4.1.6 Γεγονότα χρονισμού

Στην κατηγορία αυτή ανήκουν γεγονότα που εμφανίζονται σε συγκεκριμένη χρονική στιγμή. Η εκτέλεση κάποιου κώδικα ενεργοποιεί τα γεγονότα αυτά κάποια χρονική στιγμή. Ουσιαστικά, τα γεγονότα χρονισμού λειτουργούν ως μετρητές (timers) για τον προγραμματισμό εργασιών (scheduling).

Γενική μορφή εντολής:

void glutTimerFunc(unsigned int time, void timerFun, int check);

οπου η τιμή του ορίσματος time αντιστοιχεί στο χρόνο καθυστέρησης (σε milliseconds) πριν κληθεί η συνάρτηση timerFun. Το όρισμα timerFunction είναι η συνάρτηση χρονισμού που καλείται από την εντολή και δέχεται ως όρισμα τον ακέραιο check. Το check δηλώνεται και ως τρίτο όρισμα της glutTimerFunc, συμπεριφέρεται ως static και χρησιμοποιείται για τον έλεγχο της διαδικασίας.

4.1.7 “Γεγονότα αδρανείας”

Ορισμένες φορές είναι απαραίτητη η ακατάπαυστη εκτέλεση κάποιου τμήματος κώδικα, ανεξαρτήτως της εμφάνισης γεγονότων, όπως για παράδειγμα σε εφαρμογές κινούμενων γραφικών. Στην αυτή την περίπτωση, θεωρείται ότι αυτό το τμήμα κώδικα θα εκτελείται κατά την εμφάνιση ενός “γεγονότος” αδράνειας του συστήματος. Όταν δεν εγείρονται γεγονότα το σύστημα θεωρείται αδρανές.

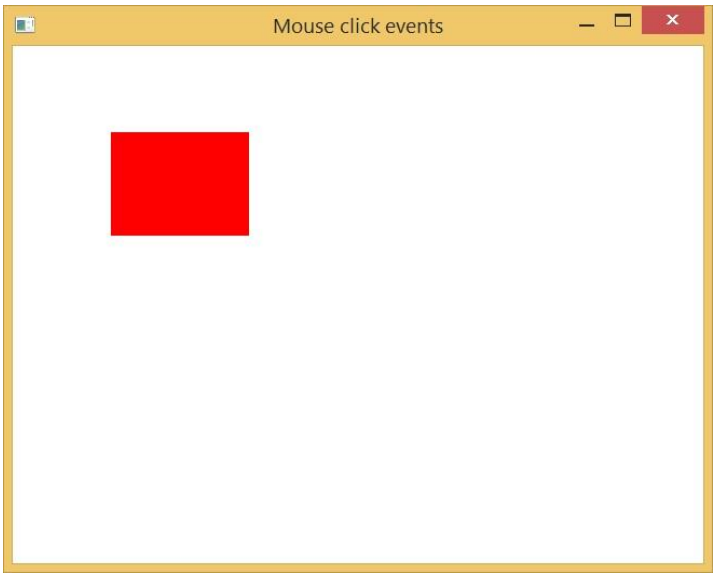
4.2 Παράδειγμα με γεγονότα

```
#include <GL/freeglut.h>
  //όρια παραθύρου
  float xwmin=-50, xwmax=50, ywmin=-50, ywmax=50;
  //διαστάσεις παραθύρου της εφαρμογής
  int windowWidth=640, windowHeight=480;
  //κορυφές ορθογωνίου
  GLfloat xw1=10.0,yw1=10.0,xw2=20.0,yw2=20.0;
void display()
{
  glClearColor(1,1,1,0);
  glClear(GL_COLOR_BUFFER_BIT);
  /*εντολή σχεδίασης ορθογωνίου παίρνει ως όρισμα τις κορυφές
του*/
  glRectf(xw1,yw1,xw2,yw2);
  glFlush();
}
/*συνάρτηση διαχείρισης γεγονότων πίεσης ποντικιού. Οι τιμές
των ορισμάτων επιστρέφονται από το λειτουργικό σύστημα κάθε
φορά που εκτελείτε η συνάρτηση. Η τιμή του ορίσματος "button"
εξαρτάται από το κουμπί του ποντικιού που πιέστηκε. Η τιμή
του ορίσματος "state" εξαρτάται από το αν το κουμπί του
ποντικιού πιέστηκε ή απελευθερώθηκε. Τα ορίσματα x, y
προσδιορίζουν τη θέση του δείκτη ποντικιού όταν πιέστηκε το
κουμπί */
void mouseButtonClicked(int button,int state,int x, int y)
{
  //Έλεγχος του ορίσματος state αν το κουμπί πιέστηκε.
  if(state==GLUT_DOWN)
  {
     /*Μετατροπή συντεταγμένων που λήφθηκαν από το συμβάν σε
     μεταβλητές του πραγματικού κόσμου*/
     xw1=(float)(xwmin+(float)(x)*
     (xwmax-xwmin)/(float)(windowWidth));
     yw1=(float)(ywmin+(float)(windowHeight-y)*
     (ywmax-ywmin)/(float)(windowHeight));
     xw2=xw1+20;
     yw2=yw1+20;
     /*Έλεγχος για πιο από τα 3 κουμπιά του ποντικιού
πατήθηκε αποδίδοντας το ανάλογο χρώμα.*/
     if (button==GLUT_LEFT_BUTTON)
          glColor3f(1,0,0);//κόκκινο
     else if (button==GLUT_MIDDLE_BUTTON)
          glColor3f(0,1,0);//πράσινο
     else if (button==GLUT_RIGHT_BUTTON)
          glColor3f(0,0,1);//μπλε
     //επανεκτέλεση της display()
     glutPostRedisplay();
  }
}
```

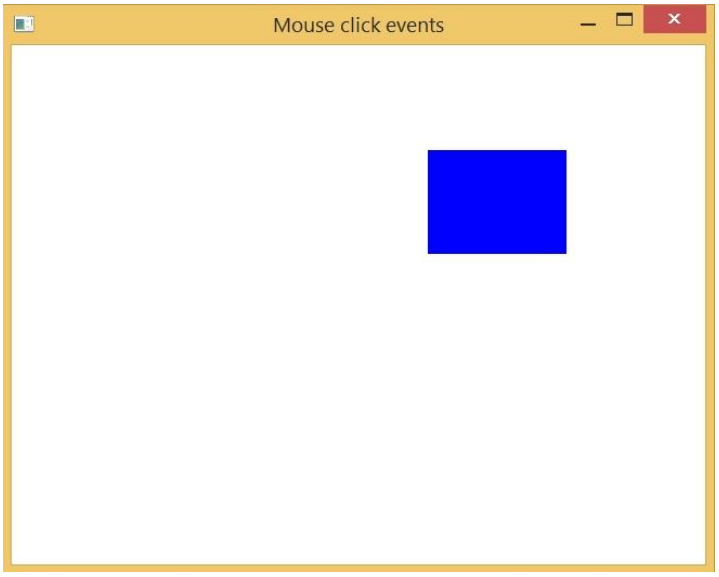
```
/*Συνάρτηση διαχείρισης γεγονότων πληκτρολόγησης ειδικών
πλήκτρων (special keys). Το όρισμα key περιέχει την
αριθμητική τιμή (τύπου int) που καθορίζει το ειδικό πλήκτρο
που πατήθηκε. Τα ορίσματα x, y εκφράζουν τη θέση στην
επιφάνεια σχεδίασης όπου βρισκόταν ο δείκτης του ποντικιού
όταν πιέστηκε το πλήκτρο.*/
void specialKeys(int key,int x, int y)
{
/*Τα παρακάτω if ελέγχουν πιο ειδικό πλήκτρο πατήθηκε και
ανάλογα μετακινούν το ορθογώνιο πάνω, κάτω, αριστερά, δεξιά.
*/
  if (key==GLUT_KEY_UP)
     {
     yw1++;
     yw2++;
  }
  if (key==GLUT_KEY_DOWN)
  {
     yw1--;
     yw2--;
  }
  if (key==GLUT_KEY_LEFT)
  {
     xw1--;
     xw2--;
  }
  if (key==GLUT_KEY_RIGHT)
  {
     xw1++;
     xw2++;
  }
  //επανεκτέλεση της display()
  glutPostRedisplay();
}

int main(int argc, char** argv)
{
  glutInit(&argc,argv);
  glutInitWindowPosition(50,50);
  glutInitWindowSize(windowWidth,windowHeight);
  glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
  glutCreateWindow("Mouse click events");
  glMatrixMode(GL_PROJECTION);
  /*διευκρινίζει ότι θα εφαρμοστεί η παράλληλη προβολή
της σκηνής */
  gluOrtho2D(xwmin,xwmax,ywmin,ywmax);
  glutDisplayFunc(display);
  //Καταχωρεί συναρτήσεις όπως η mouseButtonClicked
  glutMouseFunc(mouseButtonClicked);
  //καταχωρεί συναρτήσεις όπως η specialKeys
  glutSpecialFunc(specialKeys);
```

```
  glutMainLoop();
  return 0;
}
```

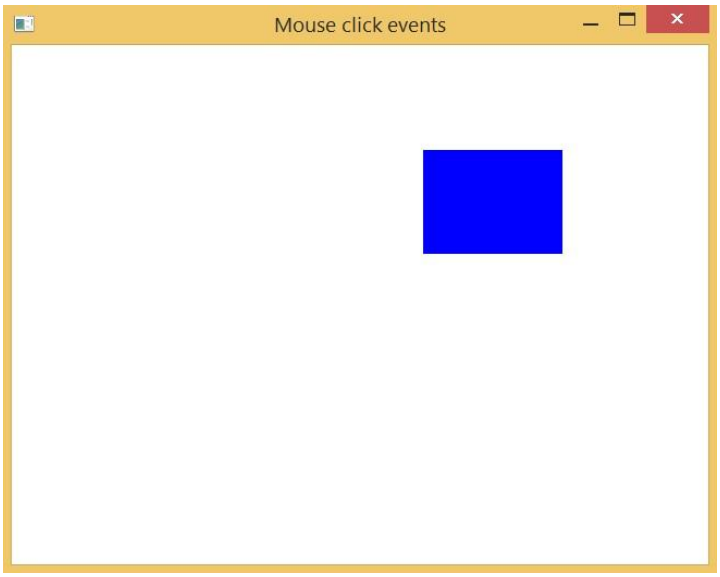
Η εφαρμογή εμφανίζει ένα λευκό παράθυρο. Όταν πατηθεί πάνω στην επιφάνια της ένα από τα τρία κουμπιά του ποντικιού θα εμφανιστεί ένα ορθογώνιο με το ανάλογο χρώμα για κάθε κουμπί με τα “βελάκια” του πληκτρολογίου μπορεί να κινηθεί. Τρείς ενδεικτικές ενέργειες του χρήστη εμφανίζονται στα παρακάτω σχήματα.



Σχήμα 5: Πάτημα αριστερού κουμπιού του ποντικιού και εμφάνιση κόκκινου ορθογωνίου στο σημείο του δέκτη



Σχήμα 6: Πάτημα δεξιού κουμπιού του ποντικιού και εμφάνιση μπλε ορθογωνίου στο σημείο του δέκτη



Σχήμα 7: Πάτημα αριστερού πλήκτρου του πληκτρολογίου και μετατόπιση του ορθογωνίου αριστερά

5 Μετασχηματισμοί - Transformations

Οι μετασχηματισμοί είναι ένα μεγάλο κεφάλαιο τις OpenGL που έχει ως στόχο την καλύτερη δυνατή παρουσίαση της σκηνής στο χώρο. Η OpenGL υποστηρίζει μια σειρά από μετασχηματισμούς οι οποίοι χρησιμοποιούνται για τοποθέτηση αντικειμένων στην οθόνη, περιστροφή, μετατόπιση, μεγέθυνση, κλπ. Επίσης χρησιμοποιούνται μετασχηματισμοί για τοποθέτηση κάμερας στη σκηνή, συστήματος συντεταγμένων, προβολής, κλπ.

5.1 Μετασχηματισμοί στην OpenGL

Ο χώρος του συστήματος συντεταγμένων καθώς και τα αντικείμενα σχεδίασης, μπορούν να μετασχηματιστούν εφαρμόζοντας μια σειρά από μετασχηματισμούς που αντιστοιχούν σε πίνακες μετασχηματισμών.

Στην OpenGL, ο χρήστης καθορίζει ποιον πίνακα επιθυμεί να τροποποιήσει την εκάστοτε χρονική στιγμή με την εντολή glMatrixMode(---).

Υπάρχουν τρείς βασικοί μετασχηματισμοί:

i. ModelView : μετασχηματισμοί αντικειμένων παρατήρησης (πίνακας αντικειμένων παρατήρησης ή πίνακας μοντέλου) οι οποίοι ενεργοποιούνται βάζοντας ως όρισμα την σταθερά *GL_MODELVIEW στην* την εντολή glMatrixMode(*GL_MODELVIEW*).

ii. Projection : μετασχηματισμοί που απεικονίζουν τη σκηνή σε έναν σταθερό τρισδιάστατο χώρο (πίνακας προβολής) οι οποίοι ενεργοποιούνται βάζοντας ως όρισμα την σταθερά *GL_PROJECTION στην* εντολή glMatrixMode(*GL_PROJECTION*).

iii. ViewPort : Προβολή του τρισδιάστατου χώρου στο επίπεδο της οθόνης και ενεργοποιούνται με την εντολή glViewPort: void glViewport(GLint x, GLint y, GLsizei width, GLsizei height); Η εντολή αυτή καθορίζει τη θέση και την έκταση του παραθύρου παρατήρησης στην επιφάνεια σχεδίασης. Τα x και y καθορίζουν το κάτω αριστερό άκρο παρατήρησης (συνήθως είναι x,y=0) και τα width, height καθορίζουν το ύψος και πλάτος του

παραθύρου σε Pixels. Ο μετασχηματισμός αυτός δεν ανήκει στην κατηγορία matrix operations. Εφαρμόζεται ανεξάρτητα και πάντα πριν την απεικόνιση στην οθόνη.

Η αρχικοποίηση των ModelView και Projection γίνεται με την εντολή glLoadIdentity :

void glLoadIdentity ( );

Η οποία αρχικοποιεί το πίνακα μετασχηματισμού που επεξεργάζεται στην εκάστοτε χρονική στιγμή. Π.χ.

```
glMatrixMode(GL_MODELVIEW);
.....
.....
glLoadIdentity();
```

5.2 Μετασχηματισμοί αντικειμένων

Τα αντικείμενα τοποθετούνται στην σκηνή σε κάποια θέση, οι μετασχηματισμοί μοντέλου τροποποιούν την αρχική κατάσταση των αντικειμένων και τα αφήνουν σε μια επιθυμητή κατάσταση, διαφορετική από την αρχική. Τέτοιοι μετασχηματισμοί είναι η μετατόπιση, η περιστροφή, η κλιμάκωση, αλλά και ο συνδυασμός αυτών. Προκειμένου να μπορούν να εφαρμοστούν τέτοιοι μετασχηματισμοί είναι αναγκαία η μετάβαση στην κατάσταση επεξεργασίας του πίνακα προβολής, αυτό γίνεται δίνοντας στην συνάρτηση glMatrixMode το όρισμα GL_MODELVIEW.

5.2.1 Μετατόπιση

Στην OpenGL η μετατόπιση των συντεταγμένων της σκηνής στο χώρο κατά (x,y,z) σταθερές εκτελείται με την εντολή ***glTranslate****:

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

glTranslated (GLdouble x, GLdouble y, GLdouble z);

5.2.2 Περιστροφή

Στην OpenGL μπορούν να εκτελεστούν μετασχηματισμοί περιστροφής ως προς οποιαδήποτε άξονα περιστροφής. Αυτό επιτυγχάνεται ορίζοντας τις συνιστώσες

του διανύσματος που καθορίζει τη διεύθυνση ενός άξονα περιστροφής ο οποίος διέρχεται από την αρχή των αξόνων. Μετασχηματισμοί περιστροφής εκτελούνται με την εντολή ***glRotate****:

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

glRotated (GLdouble angle, GLdouble x, GLdouble y, GLdouble z);

Το όρισμα *angle* αντιστοιχεί στη γωνία περιστροφής σε μοίρες, ενώ τα *x,y,z* αντιστοιχούν στις συνιστώσες του διανύσματος που εκφράζει τη διεύθυνση του άξονα περιστροφής.

Η φορά περιστροφής καθορίζεται από τη φορά του διανύσματος σύμφωνα με τον κανόνα του δεξιού χεριού.

5.2.3 Κλιμάκωση

Στην OpenGL, η κλιμάκωση εκτελείται με την εντολή **glScale***:

glScalef(GLfloat sx, Glfloat sy, Glfloat sz);

glScaled(GLdouble sx, Gldouble sy Gldouble sz);

όπου sx,sy,sz οι συντελεστές κλιμάκωσης κατά τις διευθύνσεις x,y,z αντίστοιχα. Έτσι δηλαδή στην κλιμάκωση, οι συντεταγμένες πολλαπλασιάζονται με ένα σταθερό ανά διεύθυνση συντελεστή.

5.2.4 Συνδυασμός μετασχηματισμών μοντέλου

Η OpenGL προσφέρει τη δυνατότητα συνδυασμού όλων αυτών των μετασχηματισμών για την απόδοση του επιθυμητού αποτελέσματος. Ο προγραμματιστής έχει τη δυνατότητα να συνδυάσει οποιουσδήποτε μετασχηματισμούς θέλει, όσοι και να είναι αυτοί, για να επιτύχει το ανάλογο αποτέλεσμα.

5.3 Τοποθέτηση κάμερας ή μετασχηματισμοί οπτικής γωνίας

Στην OpenGL δίνεται η δυνατότητα τοποθέτησης κάμερας στην σκηνή. Η θέση της κάμερας καθορίζεται από δύο παράγοντες τη θέση του παρατηρητή και την κατεύθυνση παρατήρησης.

Η τοποθέτηση της κάμερας στην σκηνή γίνεται με απλό τρόπο χρησιμοποιώντας την εντολή:

gluLookAt(GLdouble eyeX, GLdouble eyeY, GLdouble eyeZ,

GLdouble centerX, GLdouble centerY, GLdouble centerZ,

GLdouble upX, GLdouble upY, GLdouble upZ);

Οι συντεταγμένες eyeX, eyeY, eyeZ ορίζουν την θέση της κάμερας στην σκηνή. Ο προσανατολισμός της κάμερας καθορίζεται από το διάνυσμα με αρχή το σημείο (eyeX, eyeY, eyeZ) και πέρας το σημείο (centerX, centerY, centerZ). Οι συντεταγμένες (upX, upY , upZ) καθορίζουν τον προσανατολισμό πάνω κατεύθυνσης της κάμερας.

Δεδομένου ότι η gluLookAt επενεργεί στους μετασχηματισμούς αντικειμένων και παρατήρησης, πρέπει να μεταβεί η κατάσταση τροποποίησής τους πριν την εκτέλεση της εντολής, δίνοντας στην glMatrixMode την παράμετρο GL_MODELVIEW:

glMatrixMode(GL_MODELVIEW);

5.4 Μετασχηματισμοί προβολής

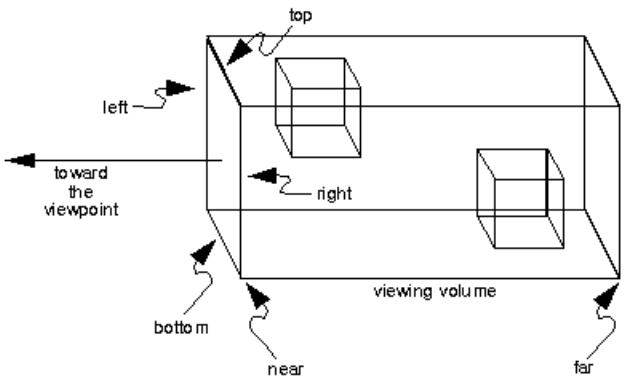
Η διαδικασία της προβολής ουσιαστικά είναι μια διεργασία, κατά την οποία, οι συντεταγμένες της σκηνής αντιστοιχίζονται, σε συντεταγμένες πάνω στο επίπεδο προβολής, βάση ενός καθορισμένου κανόνα. Ο μαθηματικός κανόνας στον οποίο βασίζεται η αντιστοίχιση αυτή, αποθηκεύεται στον πίνακα προβολής. Ο πίνακας προβολής πολλαπλασιάζεται με τις συντεταγμένες των σημείων της σκηνής και παράγει τις συντεταγμένες στις οποίες προβάλλονται στο επίπεδο του παρατηρητή, το λεγόμενο επίπεδο προβολής. Προφανώς, ανάλογα με τον τύπο της προβολής μεταβάλλεται και η δομή του πίνακα προβολής.

Η OpenGL υποστηρίζει δύο τύπος προβολής την ορθογραφική ή παράλληλη προβολή και την προβολή με προοπτική.

Προκειμένου να μπορεί τεθεί σε επεξεργασία ο πίνακας προβολής, δίνουμε στην συνάρτηση glMatrixMode το όρισμα GL_PROJECTION.

5.4.1 ορθογραφική ή παράλληλη προβολή

Σε μια ορθογραφική προβολή, η σκηνή προβολής είναι ένα ορθογώνιο παραλληλεπίπεδο, ή περισσότερα ανεπίσημα, όπως ένα κουτί (βλέπε σχήμα 5). Αντίθετα με την προοπτική προβολή, το μέγεθος του χώρου θέασης δεν μεταβάλλεται από το ένα άκρο στο άλλο, έτσι η απόσταση από την κάμερα δεν επηρεάζει πόσο μεγάλο είναι ένα αντικείμενο που εμφανίζεται. Αυτό το είδος προβολής χρησιμοποιείται από εφαρμογές όπως η δημιουργία αρχιτεκτονικών σχεδίων και computer-aided design, όπου είναι ζωτικής σημασίας να διατηρηθούν οι πραγματικές διαστάσεις των αντικειμένων και οι γωνίες μεταξύ τους, όπως και αν προβάλλονται.



Σχήμα 8: Ο χώρος ορθογραφικής προβολής

Στην OpenGL, ο ορισμός της ορθογραφικής προβολής στις τρεις διαστάσεις συνδυάζεται με την δήλωση των επιπέδων αποκοπής χρησιμοποιώντας την εντολή glOrtho:

void glOrtho(GLdouble *left*, GLdouble *right*, GLdouble *bottom*,

GLdouble *top*, GLdouble *near*, GLdouble *far*);

όπου left, right, και bottom το αριστερό, το δεξιό, το κάτω και το άνω επίπεδο αποκοπής αντίστοιχα. Το επίπεδο προβολής θεωρείται ο άξονας z. Οι συντεταγμένες του κοντινού και μακρινού επιπέδου αποκοπής δίνονται ως θετικοί αριθμοί near και far αντίστοιχα και ισχύει:

z = znear = -near και z = zfar = -far

Επίσης υπάρχει και η αντίστοιχη εντολή glOrtho2D:

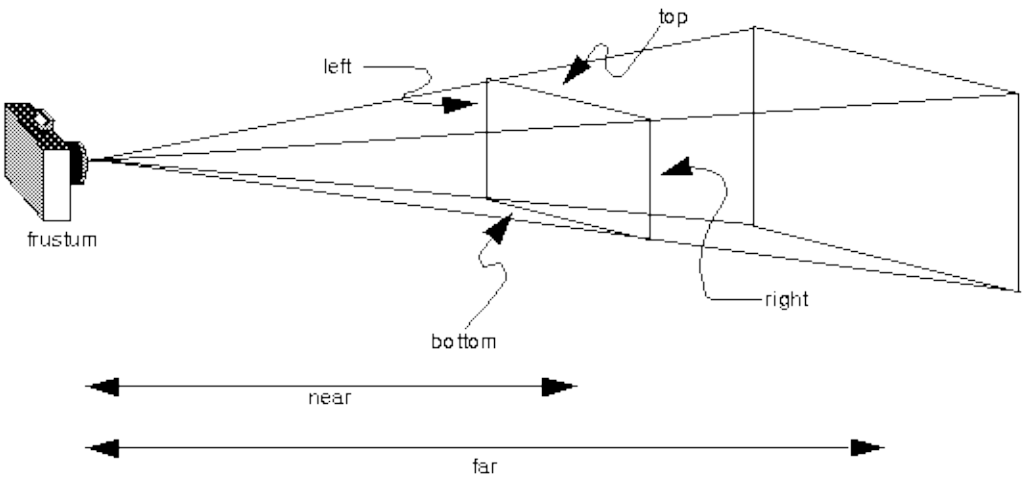
void glOrtho2D(GLdouble left, GLdouble right, GLdouble bottom,

GLdouble top);

Η οποία χρησιμοποιείτε για δισδιάστατα γραφικά και ισχύουν τα ίδια.

5.4.2 Προβολή με προοπτική

Στη προβολή με προοπτική, τα σημεία της σκηνής, αντί να προβάλλονται στο επίπεδο προβολής ακολουθώντας παράλληλες δέσμες, ακολουθούν δέσμες οι οποίες συγκλίνουν σε ένα κοινό σημείο, το κέντρο προβολής, όπως φαίνεται και στο σχήμα 6. Συνεπώς, όσο πιο κοντά στο κέντρο προβολής είναι ένα αντικείμενο τόσο πιο μεγάλο φαίνεται.



Σχήμα 9: Προβολή με προοπτική

Αυτή η μέθοδος προβολής χρησιμοποιείται συνήθως για κινούμενα σχέδια, οπτική προσομοίωση, και οποιασδήποτε άλλες εφαρμογές που θέλουν να

επιτύχουν ρεαλιστικές εικόνες. Η προβολή με προοπτική λειτουργεί με παρόμοιο τρόπο όπως το μάτι και η φωτογραφική μηχανή.

Μια προβολή με προοπτική στην OpenGL μπορεί να δημιουργηθεί με δύο διαφορετικές εντολές την gluPerspective και την glFrustum.

Η εντολή gluPerspective:

void gluPerspective(GLdouble *fovy*, GLdouble *aspect*,

GLdouble *zNear*, GLdouble *zFar*);

Όπου η παράμετρος fovy ορίζει την γωνία θέασης του πάνω και κάτω επιπέδου του χώρου θέασης, η aspect καθορίζει την 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.

Η εντολή glFrustum:

void glFrustum(GLdouble left, GLdouble right, GLdouble bottom, GLdouble top, GLdouble near, GLdouble far);

Τα ορίσματα δηλώνονται όπως και στην glOrtho και οι τιμές near και far είναι πάντα θετικές.

Η μόνη ουσιαστική διαφορά ανάμεσα στις δύο εντολές είναι ότι η *glFrustum μπορεί να ορίσει* μια συμμετρική ή πλάγια προβολή με προοπτική ανάλογα με τις τιμές που θα δοθούν, ενώ η gluPerspective μπορεί να ορίσει μόνο συμμετρική προβολή με προοπτική.

5.5 Στοίβα πινάκων μετασχηματισμού

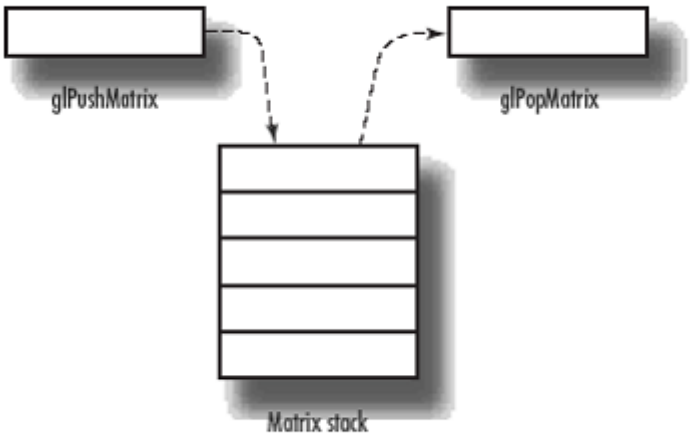
Για κάθε κατηγορία πίνακα μετασχηματισμού (μοντέλου και προβολής) η μηχανή της OpenGL προβλέπει την ύπαρξη μιας στοίβας, η οποία προσφέρει τη δυνατότητα αποθήκευσης πολλαπλών προφίλ για κάθε μητρώο μετασχηματισμού, με σκοπό τη μελλοντική τους χρήση.

Στην OpenGL οι εντολές που αποθηκεύουν και ανακτούν τις κατηγορίες πινάκων μετασχηματισμού στην κορυφή της στοίβας είναι οι:

glPushMatrix();

glPopMatrix();

Όπως φαίνεται και στο σχήμα 7.



Σχήμα 10: Στοίβα πινάκων μετασχηματισμού

Είναι κατανοητό ότι για να ενεργοποιηθεί η στοίβα πρώτα πρέπει να προηγηθεί η εντολή glMatrixMode με οποιοδήποτε όρισμα.

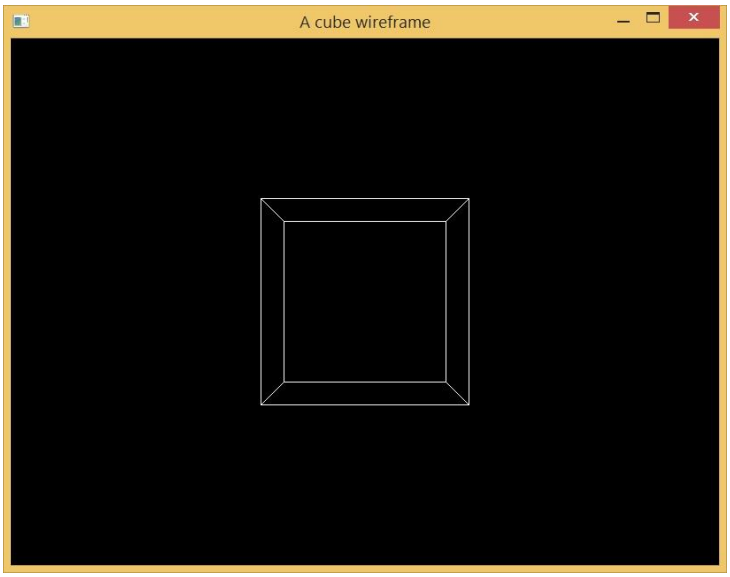
5.6 Παράδειγμα εφαρμογής μετασχηματισμών

```
#include <GL/freeglut.h>
  /*Μεταβλητές που αντιστοιχούν στις συντεταγμένης που θα
χρησιμοποιηθούν για την περιστροφή*/
GLfloat xRotated, yRotated, zRotated;

void Display()
{

  glClearColor(0,0,0,0);
  glClear(GL_COLOR_BUFFER_BIT);
  //αρχικοποιεί τους πίνακες μετασχηματισμού
  glLoadIdentity();
  //μετατόπιση των συντεταγμένων της σκηνής στο χώρο
  glTranslatef(0.0,0.0,-4.0);
  /*Μετασχηματισμός περιστροφής όπου xRotated, yRotated και
zRotated οι μοίρες περιστροφής και x,y,z οι συνιστώσες του
διανύσματος που εκφράζει τη διεύθυνση του άξονα
περιστροφής.*/
  glRotatef(xRotated,1.0,0.0,0.0);//άξονας x
```

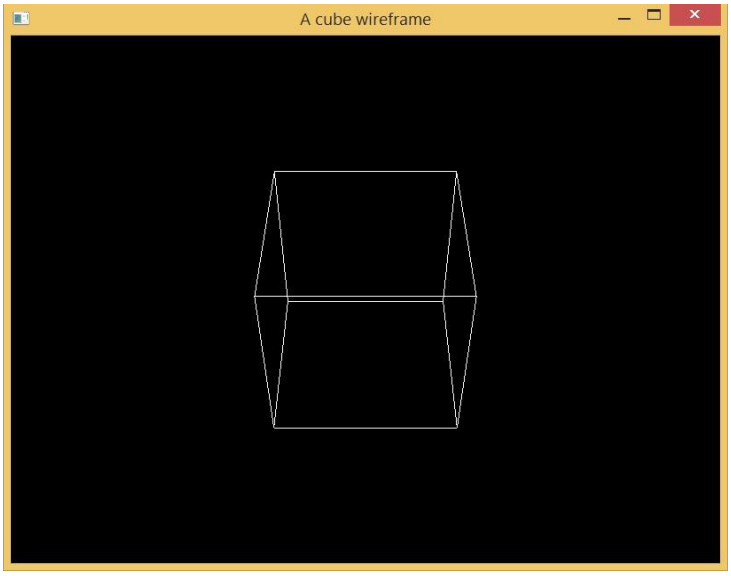
```
  glRotatef(yRotated,0.0,1.0,0.0);//άξονας y
  glRotatef(zRotated,0.0,0.0,1.0);//άξονας z
  //άσπρο χρώμα σχεδίασης του κύβου
  glColor3f(1,1,1);
  //σχεδίαση κύβου μεγέθους 1
  glutWireCube(1);
  //εξαναγκασμός εκτέλεσης των εντολών που εκκρεμούν
  glFlush();
}
/*Συνάρτηση διαχείρισης γεγονότων μεταβολής διαστάσεων
παραθύρου*/
void Reshape(int x, int y)
{
  //έλεγχος μηδενικού μεγέθους παραθύρου εφαρμογής
  if (y == 0 || x == 0) return;
  //ορισμός ενός νέου πίνακα προβολής
  glMatrixMode(GL_PROJECTION);
  //αρχικοποιεί τους πίνακες μετασχηματισμού
  glLoadIdentity();
  /*ορισμός προβολής με προοπτική
  γωνία θέασης:40 μοίρες
  καθορισμός αναλογίας μήκους/πλάτους του κοντινού πεδίου
αποκοπής, δηλαδή x και y οι διαστάσεις του παραθύρου
  απόσταση κοντινού πεδίου αποκοπής: 0.5
  απόσταση μακρινού πεδίου αποκοπής: 20.0*/
  gluPerspective(40.0,(GLdouble)x/(GLdouble)y,0.5,20.0);
  //ορισμός ενός νέου πίνακα μοντέλου
  glMatrixMode(GL_MODELVIEW);
  /*χρησιμοποιεί ολόκληρο το παράθυρο της εφαρμογής για
απόδοση*/
  glViewport(0,0,x,y);
}
/* συνάρτηση η οποία διαχειρίζεται γεγονότα πληκτρολογίου
ειδικών χαρακτήρων*/
void specialKeys(int key,int x, int y)
{
  /*Ανάλογα με το ειδικό πλήκτρο που θα πατηθεί μεταβάλλονται
οι μεταβλητές περιστροφής*/
  if (key==GLUT_KEY_UP)
  {
  xRotated -= 1;
  }
  if (key==GLUT_KEY_DOWN)
  {
  xRotated += 1;
  }
  if (key==GLUT_KEY_LEFT)
  {
  yRotated += 1;
  }
  if (key==GLUT_KEY_RIGHT)
```

```
  {
  zRotated += 1;
  }
  //επανεκτέλεση της display
  glutPostRedisplay();
}

int main(int argc, char **argv)
{
  glutInit(&argc,argv);
  glutInitWindowPosition(50,50);
  glutInitWindowSize(800,600);
  glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
  glutCreateWindow("A cube wireframe");
  //ορισμός ενός νέου πίνακα προβολής
  glMatrixMode(GL_PROJECTION);
  //Ορισμός παράλληλης προβολής
  glOrtho(-80,80,-60,60,0,100);
  //ορισμός ενός νέου πίνακα μοντέλου
  glMatrixMode(GL_MODELVIEW);
  //Η οπτική γωνία του παρατηρητή
  gluLookAt(-30,-30,40,0,0,0,0,1,0);

  glutDisplayFunc(Display);
  /* Καταχώρηση της συνάρτηση κλήσης reshape για τη
διαχείριση γεγονότων γεγονός αλλαγής των διαστάσεων του
παραθύρου.*/
     glutReshapeFunc(Reshape);
  /*εκτελεί τη συνάρτηση specialKeys η οποία διαχειρίζεται
γεγονότα πληκτρολογίου ειδικών χαρακτήρων*/
     glutSpecialFunc(specialKeys);
  glutMainLoop();
  return 0;
}
```

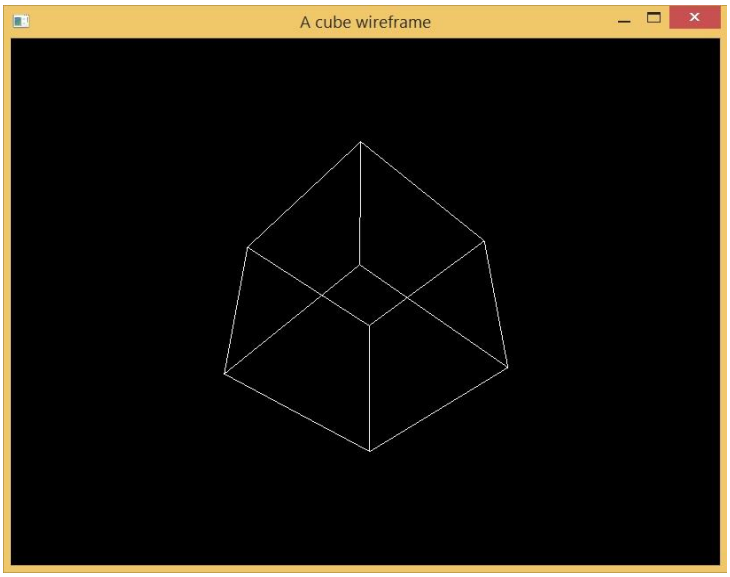
Με την εκτέλεση του παραπάνω κώδικα θα εμφανιστεί ένα κύβος. Πιέζοντας το πάνω και το κάτω βελάκι του πληκτρολογίου θα περιστραφεί ο κύβος γύρο από τον εαυτό του στον άξονα x αναλόγως. Πιέζοντας το αριστερό βελάκι του πληκτρολογίου θα περιστραφεί ο κύβος ως προς τον άξονα y ενώ πιέζοντας το δεξί θα περιστραφεί ο κύβος ως προς των άξονα z (γύρο από τον εαυτό του επίσης). Ένα ενδεικτικό παράδειγμα εκτέλεσης του κώδικα εμφανίζεται στα παρακάτω σχήματα.



Σχήμα 11: Εμφάνιση παραθύρου με την εκτέλεση του κώδικα



Σχήμα 12: Πίεση άνω βέλους του πληκτρολογίου και περιστροφή κύβου αρνητικά ως προς το άξονα x



Σχήμα 13: Πίεση δεξιού βέλους του πληκτρολογίου και περιστροφή κύβου θετικά ως προς τον άξονα y

6 Φωτισμός - Lighting

Αν κοιτάξει κανείς σε μια φυσική επιφάνεια, η αντίληψη των ματιών εξαρτάται από το χρώμα και την κατανομή φωτονίων που φτάνουν και ενεργοποιούν τα κύτταρα των ματιών. Τα φωτόνια προέρχονται από μια οι περισσότερες πηγές φωτός, ορισμένα από τα οποία απορροφώνται και άλλα αντανακλώνται από την επιφάνεια. Επιπλέον, οι διαφορετικές επιφάνειες μπορεί να έχουν πολύ διαφορετικές ιδιότητες όπως οι γυαλιστερές που αντανακλούν το φως σε συγκεκριμένες κατευθύνσεις, ενώ αντίθετα οι τραχείες διασκορπίζουν το εισερχόμενο φως σε πάσα κατεύθυνση. Οι περισσότερες επιφάνειες είναι κάπου στη μέση.

Η OpenGL προσεγγίζει το φως και το φωτισμό, διαιρώντας το σε κόκκινα, πράσινα και μπλε στοιχεία. Έτσι, το χρώμα των πηγών φωτισμού χαρακτηρίζεται από την ποσότητα του κόκκινου, πράσινου και μπλε φως που εκπέμπουν, και το υλικό των επιφανειών χαρακτηρίζεται από το ποσοστό κόκκινου, πράσινου και μπλε που εισέρχεται σε αυτές και ανακλάται σε διάφορες κατευθύνσεις. Οι εξισώσεις φωτισμού της OpenGL είναι σχετικά μια απλή προσέγγιση, παρόλα αυτά λειτουργούν αρκετά καλά και μπορούν να υπολογιστούν σχετικά γρήγορα.

6.1 Πηγές φωτισμού

Στο μοντέλο φωτισμού της OpenGL, ο φωτισμός μιας σκηνής μπορεί να προέρχεται από διάφορες πηγές φωτός οι οποίες μπορούν ξεχωριστά να ενεργοποιηθούν ή να απενεργοποιηθούν. Αυτό γίνεται δίνοντας στην εντολή glEnable το όρισμα GL_LIGHTING:

```
glEnable( GL_LIGHTING );
```

Η γενική μορφή των εντολών glLightf και glLightfv που ορίζουν μια πηγή φωτός είναι:

```
void glLightf (GLenum light, GLenum parameterName, GLfloat parameter);
```

Το όρισμα light είναι μια συμβολική σταθερά που προσδιορίζει την πηγή φωτισμού στην οποία αποδίδεται το χαρακτηριστικό. Οι υλοποιήσεις της OpenGL

υποστηρίζουν οκτώ πηγές φωτισμού. Επομένως, η σταθερά light μπορεί να δεχτεί τις τιμές από GL_LIGHT0 έως GL_LIGHT7. Το όρισμα parameterName είναι μια συμβολική σταθερά που καθορίζει το χαρακτηριστικό που αποδίδεται. Ενώ το όρισμα parameter δηλώνει την αριθμητική τιμή που αποδίδεται στην ιδιότητα parameterName.

void glLightfv (GLenum light, GLenum parameterName, const GLfloat * parameterArray);

Στα πρώτα δυο ορίσματα ισχύουν τα ίδια. Το όρισμα parameterArray είναι δείκτης σε πίνακα. Ο πίνακας αυτός περιέχει το σύνολο των τιμών που προσδιορίζουν το χαρακτηριστικό parameterName.

Αφού δηλωθούν οι παράμετροι των πηγών φωτισμού, απαιτείται επιπλέον η ενεργοποίηση κάθε μίας πηγής ξεχωριστά.

6.2 Κατηγορίες πηγών φωτισμού

Οι πηγές φωτισμού της μηχανής της OpenGL μπορούν να διακριθούν σε σημειακές πηγές, πηγές με εξασθένηση φωτεινότητας, πηγές σε άπειρη απόσταση και πηγές με κατεύθυνση.

6.2.1 Σημειακές πηγές

Οι σημειακές πηγές φωτισμού είναι σημεία (όπως είναι γνωστό τα σημεία καταλαμβάνουν απειροελάχιστο χώρο) στο χώρο και δηλώνονται ορίζοντας τη θέση τους στη σκηνή. Σε μια ομοιόμορφη σημειακή πηγή, οι ακτίνες διαδίδονται σφαιρικά προς όλες τις κατευθύνσεις και με την ίδια ένταση όπως μια λάμπα σε ένα γραφείο.

Για να οριστεί μια σημειακή πηγή στη σκηνή, πρέπει να προσδιοριστεί η θέση της με την εντολή glLightfv:

glLightfv(light, GL_POSITION, positionVector);

Όπου η παράμετρος light θα πρέπει να αντιστοιχεί στη σταθερά GL_LIGHT0. Το GL_POSITION είναι η σταθερά που ορίζει ότι πρόκειται για σημείο. Το όρισμα positionVector είναι δείκτης σε πίνακα με τέσσερα στοιχεία. Τα τρία πρώτα

στοιχεία του ορίζουν τη θέση ενώ το τέταρτο ορίζει εάν η πηγή φωτός είναι σημειακή. Με μη μηδενική τιμή ορίζεται μια σημειακή πηγή.

6.2.2 Εξασθένηση φωτεινότητας

Στον πραγματικό κόσμο, η ένταση του φωτός μειώνεται καθώς απόσταση από την πηγή του φωτός αυξάνει. Στην OpenGL, οι συντελεστές εξασθένησης για την πηγή φωτισμού light (light=GL_LIGHT0,…,GL_LIGHT7) ορίζονται χρησιμοποιώντας τις εντολές:

glLightf (light, GL_CONSTANT_ATTENUATION, term0);

glLightf (light, GL_LINEAR_ATTENUATION, term1);

glLightf(light, GL_QUADRATIC_ATTENUATION, term2);

για τον καθορισμό του τετραγωνικού όρου term0, term1 και term2 αντίστοιχα. Οι συντελεστές term0, term1, term2 αντιστοιχούν σε αριθμούς κινητής υποδιαστολής. Οι default τιμές τους είναι term0=1, term1=0 και term2=0 αλλά δεν αντιστοιχούν στην εξασθένηση φωτεινότητας.

6.2.3 Πηγές σε άπειρη απόσταση

Πολλές φορές στην σχεδίαση κάποιου τοπίου επιλέγεται φωτισμός σε άπειρη απόσταση για να αποδοθεί ο ήλιος. Τέτοιες πηγές χαρακτηρίζονται στο ότι όλες οι εκπεμπόμενες ακτίνες ακολουθούν παράλληλες τροχιές σε αντίθεση με την περίπτωση σημειακών πηγών που οι ακτίνες διαδίδονται σφαιρικά και με αμείωτη ένταση.

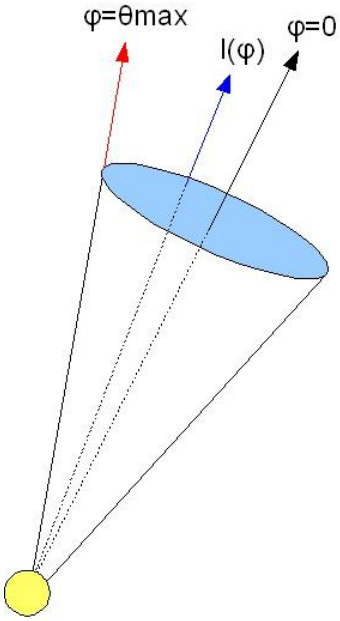
Στην OpenGL, ορίζεται μια πηγή σε άπειρη απόσταση με την εντολή glLightfv:

glLightfv(light, GL_POSITION, positionVector);

Στα δύο πρώτα ισχύουν τα ίδια με τις σημειακές πηγές ενώ στο τέταρτο στοιχείο του πίνακα positionVector ίσο με μηδέν. Στην περίπτωση αυτή, τα τρία πρώτα στοιχεία του πίνακα positionVector δε δηλώνουν τη θέση της πηγής (εφόσον βρίσκεται σε άπειρη απόσταση) αλλά το διάνυσμα κατεύθυνσης των ακτινών που αναχωρούν από την πηγή.

6.2.4 Πηγές με κατεύθυνση

Η πηγές αυτές λειτουργούν όπως οι σημειακές με τη διαφορά ότι έχουν περιορισμένο γωνιακό εύρος. Σε αυτή την περίπτωση, το τμήμα της σκηνής που φωτίζεται από την πηγή με κατεύθυνση περικλείεται σε έναν κώνο. Το άνοιγμα του κώνου διάχυσης καθορίζεται από μια γωνία αποκοπής θmax, όπως φαίνεται στο σχήμα 8. Τα σημεία της σκηνής που βρίσκονται εκτός του κώνου διάχυσης δε φωτίζονται καθόλου από τη πηγή κατεύθυνσης.



Σχήμα 14: πηγή με κατεύθυνση

6.3 Καθολικές παράμετροι φωτισμού

Όπως αναφέρθηκε και παραπάνω, κάθε πηγή φωτός μπορεί να συμβάλει στο φωτισμό του περιβάλλοντος σε μια σκηνή. Επιπλέον, μπορεί να υπάρχει άλλου είδους φωτισμός που δεν εκπέμπεται από κάποια συγκεκριμένη πηγή. Η OpenGL υποστηρίζει τέτοιου είδους φωτισμό οι οποίες ονομάζονται καθολικές παραμέτρους φωτισμού.

Για τη ρύθμιση καθολικών παραμέτρων φωτισμού, χρησιμοποιείται η εντολή glLightModel{if}{v}:

void glLightModel*(parameterName, parameterValue);

όπου parameterName η καθολική παράμετρος που ορίζεται και parameterValue η τιμή ή ο πίνακας τιμών που προσδιορίζει την παράμετρο parameterName.

Ο καθολικός περιβάλλων φωτισμός, είναι μια συχνά ρυθμιζόμενη καθολική παράμετρος είναι μια παράμετρος περιβάλλοντος φωτισμού που δεν εντάσσεται σε κάποια από τις πηγές φωτισμού (GL_LIGHT0, …, GL_LIGHT7) αλλά ρυθμίζεται ανεξάρτητα. Η καθολική παράμετρος περιβάλλοντος φωτισμού ορίζεται με την χρήση της εντολής glLightModelfv ως εξής:

GLfloat globalAmbient[]={r,g,b};

glLightModelfv(GL_LIGHT_MODEL_AMBIENT, globalAmbient);

όπου globalAmbient οι συνιστώσες του καθολικού περιβάλλοντος φωτισμού.

6.4 Ιδιότητες υλικού

Ο προγραμματιστής με τη χρήση της μηχανή της OpenGL μπορεί να ορίσει τις ιδιότητες του υλικού των αντικειμένων στη σκηνή. Τέτοιες ιδιότητες αφορούν τον περιβάλλοντα φωτισμό, τον διάχυτο φωτισμό, τον κατοπτρικό φωτισμό καθώς επίσης χρώματα, η λάμψη, και το χρώμα του κάθε εκπεμπόμενου φωτός. Οι περισσότερες από τις ιδιότητες του υλικού είναι εννοιολογικά παρόμοιες με αυτές που χρησιμοποιούνται για την δημιουργία πηγών φωτισμού. Ο μηχανισμός για τον ορισμό τους είναι παρόμοιος, εκτός από το ότι η εντολή που χρησιμοποιείται ονομάζεται glMaterial *().

Ανάλογα με το επίθεμα, χρησιμοποιείται η εντολή

glMaterialf(GLenum face, GLenum property, GLfloat propertyValue);

για ιδιότητες που προσδιορίζει μία αριθμητική τιμή ή

glMaterialfv(GLenum face, GLEnum property, GLfloat *propertyValues);

για ιδιότητες που περιγράφει ένα σύνολο αριθμητικών τιμών.

Το όρισμα face καθορίζει την όψη στην οποία αποδίδεται το ανάλογο χαρακτηριστικό: GL_FRONT, GL_BACK και GL_FRONT_AND_BACK όπου το χαρακτηριστικό αποδίδεται στη μπροστινή όψη των επιφανειών, στην πίσω και

στις δύο όψεις αντίστοιχα. Το όρισμα propertyValue ή propertyValues είναι η αριθμητική τιμή ή ο δείκτης στον πίνακα των αριθμητικών τιμών που καθορίζουν το χαρακτηριστικό αντίστοιχα.

Οι πιθανές τιμές για το όρισμα property παρουσιάζονται στον Πίνακα 3. Σημειώνεται ότι η παράμετρος GL_AMBIENT_AND_DIFFUSE επιτρέπει τη ρύθμιση χρώματος του περιβάλλοντα και διάχυτου υλικού ταυτόχρονα με την ίδια τιμή RGBA.

Πίνακας 3: Πιθανές τιμές του ορίσματος property

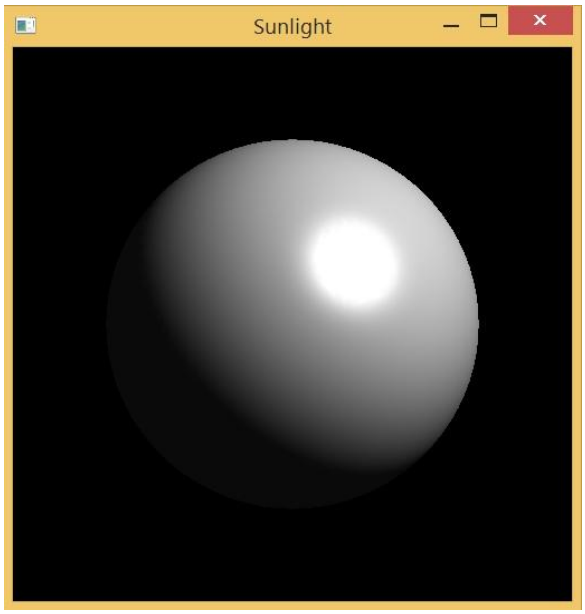
Παράμετρος	Προκαθορισμένη τιμή	Σημασία
GL_AMBIENT	(0.2, 0.2, 0.2, 1.0)	περιβάλλοντας φωτισμός
GL_DIFFUSE	(0.8, 0.8, 0.8, 1.0)	διάχυτος φωτισμός
GL_AMBIENT_AND_DIFFUSE	-	Περιβάλλοντας και διάχυτος φωτισμός
GL_SPECULAR	(0.0, 0.0, 0.0, 1.0)	κατοπτρικός φωτισμός
GL_SHININESS	0.0	Κατοπτρικός εκθέτης
GL_EMISSION	(0.0, 0.0, 0.0, 1.0)	Εκπομπή χρώματος φωτισμού
GL_COLOR_INDEXES	(0,1,1)	Δίκτης χρώματος για τον περιβάλλοντα, διάχυτο και κατοπτρικό φωτισμό

6.5 Παράδειγμα φωτισμού

```
#include <GL/freeglut.h>
#include <stdlib.h>

/*  Initialize material property, light source, lighting
model and depth buffer. */
void init(void)
{
   GLfloat mat_specular[] = { 1.0, 1.0, 1.0, 1.0 };
   GLfloat mat_shininess[] = { 50.0 };
   GLfloat light_position[] = { 1.0, 1.0, 1.0, 0.0 };

   glClearColor (0.0, 0.0, 0.0, 0.0);
   glShadeModel (GL_SMOOTH);
   //εμφάνιση κατοπτρικής ανάκλασης
   glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
   //εμφάνιση λείας επιφάνειας
   glMaterialfv(GL_FRONT, GL_SHININESS, mat_shininess);
   //ορισμός πηγής φωτισμού
   glLightfv(GL_LIGHT0, GL_POSITION, light_position);

   //ενεργοποίηση εντολών
```

```
   glEnable(GL_LIGHTING);
   glEnable(GL_LIGHT0);
   glEnable(GL_DEPTH_TEST);
}

void display(void)
{
   glClear (GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
   //δημιουργία σφαίρας
   glutSolidSphere(1,80,80);
   glFlush ();
}
//καθορισμός εμφάνισης της σκηνής
void reshape (int w, int h)
{
   glViewport (0, 0, (GLsizei) w, (GLsizei) h);
   glMatrixMode (GL_PROJECTION);
   glLoadIdentity();
   if (w <= h)
      glOrtho (-1.5, 1.5, -1.5*(GLfloat)h/(GLfloat)w,
         1.5*(GLfloat)h/(GLfloat)w, -10.0, 10.0);

   else
      glOrtho (-1.5*(GLfloat)w/(GLfloat)h,
         1.5*(GLfloat)w/(GLfloat)h, -1.5, 1.5, -10.0, 10.0);

      glMatrixMode(GL_MODELVIEW);
      glLoadIdentity();
}
int main(int argc, char** argv)
{
   glutInit(&argc, argv);
   glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH);
   glutInitWindowSize (500, 500);
   glutInitWindowPosition (100, 100);
   glutCreateWindow ("Sunlight");
   /*καλεί τη συνάρτηση που περιέχει τη διαμόρφωση του
φωτισμού*/
   init ();
   glutDisplayFunc(display);
   glutReshapeFunc(reshape);
   glutMainLoop();
   return 0;
}
```

Ο παραπάνω κώδικας υλοποιεί μια σφαίρα η οποία φωτίζεται από μια σημειακή πηγή και ρυθμίζονται από κάποιες ιδιότητες φωτισμού όπως οι κατοπτρική ανάκλαση και η τραχύτητα της επιφάνειας. Το αποτέλεσμα της εκτέλεσης του παραπάνω κώδικα εμφανίζεται στο Σχήμα 15.



Σχήμα 15: Σημειακός φωτισμός

7 Υφή - Texture

Μέχρι στιγμής στα παραπάνω κεφάλαια, η σχεδίαση κάθε γεωμετρικού αντικειμένου γινόταν είτε με τον ορισμό μιας σταθεράς χρώματος, είτε με την ομαλή σκίαση ανάμεσα στα χρώματα των κορυφών του, χωρίς την χρήση απόδοσης υφής. Για τη σχεδίαση παραδείγματος χάριν ενός μεγάλου τοίχου από τούβλα χωρίς απόδοση υφής, κάθε τούβλο θα πρέπει να σχεδιάζεται ξεχωριστά ως πολύγωνο, πράγμα που καθιστά πολύ δύσκολη και χρονοβόρα τη διαδικασία σχεδίασης μεγάλων επιφανίων του φυσικού κόσμου.

Η απόδοση υφής επιτρέπει στον προγραμματιστή να “κολλήσει” μια εικόνα ενός τοίχου από τούβλα (λαμβάνοντας την, από μια φωτογραφία από ένα πραγματικό τοίχος) σε ένα πολύγωνο, και να σχεδιάσει ολόκληρο τοίχος ως ένα ενιαίο πολύγωνο. Η απόδοση υφής εξασφαλίζει ότι όλα τα αντικείμενα θα παρουσιαστούν με το σωστό τρόπο λ.χ. Όταν το τοίχωμα - πολύγονο παρατηρείται σε προοπτική προβολή, τα τούβλα μειώνονται σε μέγεθος, μαζί με το τοίχωμα, όσο η κάμερα απομακρύνετε από αυτό. Άλλες χρήσεις για την απόδοσης υφής περιλαμβάνουν βλάστηση (η υφή σε μεγάλα πολύγωνα που αντιπροσωπεύουν το έδαφος), προσομοιωτή πτήσης, μοτίβα ταπετσαρίας και υφές που κάνουν τα πολύγωνα να φαίνονται σαν να είναι κατασκευασμένα από φυσικές ουσίες, όπως μάρμαρο, ξύλο, ύφασμα κλπ.

Για να χρησιμοποιηθεί η απόδοση υφής, πρέπει να εκτελεστούν τα εξής βήματα:

1. Καθορίζεται η υφή.
2. Αναφέρεται πώς θα εφαρμοστεί η υφή του σε κάθε pixel.
3. Ενεργοποιείται η απόδοση υφής.
4. Σχεδίαση της σκηνής, παρέχοντας υφή και γεωμετρικές συντεταγμένες.

7.1 Συντεταγμένες υφής

Στην OpenGL για την απόδοση υφής σε γραμμές και επιφάνειες αναθέτονται σε κάθε μία κορυφή συντεταγμένες υφής. Έτσι η μηχανή της OpenGL αποδίδει τα στοιχεία του πίνακα υφής κατά μήκος της γραμμής ή στην έκταση της επιφάνειας ούτως ώστε να ικανοποιούνται οι οριακές συνθήκες που επιβάλλει ο προγραμματιστής.

Η ανάθεση συντεταγμένων υφής πραγματοποιείται με τη χρήση της εντολής glTexCoord{1,2}{i,s,f,d}. Όπου 1 μονοδιάστατες υφές με μοναδική παράμετρο (GLfloat s) και 2 δισδιάστατες υφές με παραμέτρους (GLfloat s, GLfloat t) . Ενώ τα i,s,f,d καθορίζουν το τύπο δεδομένων.

Με τα ορίσματα s, t της glTexCoord* ορίζονται οι τρέχουσες συντεταγμένες υφής, οι οποίες αποδίδονται σε όλα τα σημεία που θα δηλωθούν στη συνέχεια του κώδικα. Οι τρέχουσες συντεταγμένες υφής λειτουργούν ως μεταβλητές κατάστασης δηλαδή διατηρούν τις τιμές που τους ανατέθηκαν την τελευταία φορά.

7.1.1 Πίνακες συντεταγμένων υφής

Με τη λειτουργία αυτή ο προγραμματιστής μπορεί να ορίσει ένα σύνολο από συντεταγμένες υφής και να της συμπεριλάβει σε ένα πίνακα.

Για να γίνει αυτό απαιτείται η ενεργοποίηση της λειτουργίας αυτής με την εντολή glEnableClientState:

glEnableClientState(GL_TEXTURE_COORD_ARRAY);

Η καταχώρηση του πίνακα συντεταγμένων υφής επιτελείται με την εντολή glTexCoordPointer:

void glTexCoordPointer(GLint size, GLenum type, GLsizei stride, const GLvoid *pointer);

Τα 2 πρώτα ορίσματα είναι ίδια με την απλή εντολή glTexCoord. Η παράμετρος stride καθορίζει την απόσταση σε bytes μεταξύ διαδοχικών συντεταγμένων υφής στον πίνακα. Χρησιμοποιείται μόνο όταν στον πίνακα αποθηκεύονται τιμές πολλαπλών ιδιοτήτων (λ.χ. συντεταγμένες σημείων και συντεταγμένες υφής

εναλλάξ). Η προκαθορισμένη τιμή είναι 0. Τέλος η παράμετρος pointer είναι δείκτης στον πίνακα που περιέχει τις συντεταγμένες υφής

7.2 Εικόνες υφής

Οι εικόνες υφής στην OpenGL διακρίνονται σε μονοδιάστατες και δισδιάστατες.

Για την απόδοση μονοδιάστατης και δισδιάστατης εικόνας υφής, αρχικά απαιτείται η ενεργοποίηση της λειτουργίας με την εντολή glEnable:

glEnable(GL_TEXTURE_1D); και glEnable(GL_TEXTURE_2D);

Η φόρτωση ενός πίνακα στοιχείων μονοδιάστατης εικόνας υφής γίνεται με την εντολή glTexImage1D:

void glTexImage1D(GLenum target, GLint level, GLint internalFormat, GLsizei width, GLint border, GLenum format, GLenum type, const GLvoid *texelArray);

Και για δισδιάστατη με την εντολή glTexImage2D:

void glTexImage2D(GLenum target, GLint level, GLint internalFormat, GLsizei width, GLsizei height, GLint border, GLenum format, GLenum type, const GLvoid *texelArray);

Για την ανάλυση των παραπάνω εντολών ισχύουν τα ίδια.

- Το όρισμα target καθορίζει την υφή και παίρνει ως όρισμα GL_TEXTURE_1D ή GL_PROXY_TEXTURE_1D.
- Η ακέραια παράμετρος *level* καθορίζει αν ο πίνακας υφής χρησιμοποιείται ως μια βαθμίδα σε μια πυραμίδα πινάκων. Αυτή η παράμετρος είναι χρήσιμη μόνο όταν χρησιμοποιούνται πίνακες υφής σε πολλαπλές αναλύσεις. Με τιμή 0 ορίζεται ότι η υφή ανήκει στην κορυφή της πυραμίδας και είναι η συνιστώμενη τιμή όταν δεν χρησιμοποιούνται πολλαπλές αναλύσεις.

- Το όρισμα internalFormat καθορίζει το πλήθος των συνιστωσεών του χρωματικού μοντέλου της υφής. Οι σταθερές που χρησιμοποιούνται συνήθως είναι οι εξής:

  GL_LUMINANCE: αποχρώσεις του γκρίζου.

  GL_RGB: RGB μοντέλο με 3 συνιστώσες.

  GL_RGBA: RGBA μοντέλο με 4 συνιστώσες.

- Το όρισμα width Προσδιορίζει το πλάτος της εικόνας υφής. Όλες οι υλοποιήσεις υποστηρίζουν εικόνες υφής που είναι τουλάχιστον 1024 texels πλάτος. Το ύψος της υφής εικόνας 1D είναι 1.

- Το όρισμα height καθορίζει το ύψος της εικόνας υφής, ή τον αριθμό των στρωμάτων σε μία υφή αντιστοιχία, στην περίπτωση της GL_TEXTURE_1D_ARRAY και GL_PROXY_TEXTURE_1D_ARRAY. Όλες οι υλοποιήσεις υποστηρίζουν εικόνες 2D υφής που είναι τουλάχιστον 1024 texels ύψος, και τουλάχιστον 256 στρώματα βάθος.

- Το όρισμα border καθορίζει αν η υφή θα περιβάλλεται από όριο πάχους ενός pixel. Η τιμή 0 καθορίζει ότι δε χρησιμοποιείται όριο, ενώ η τιμή 1 καθορίζει ότι χρησιμοποιείται όριο.

- Με το όρισμα format καθορίζεται η διαδοχή με την οποία δίνονται οι συνιστώσες του χρωματικού μοντέλου. Οι τιμές που υποστηρίζονται είναι: GL_ALPHA, GL_RGB, GL_RGBA, GL_LUMINANCE, και GL_LUMINANCE_ALPHA.

- Το όρισμα type καθορίζει τον πρωτογενή τύπο δεδομένων με τον οποίο δίνονται τα texels στον πίνακα υφής. Οι συνήθεις τιμές είναι GL_INT, GL_FLOAT και GL_UNSIGNED_BYTE για φόρτωση αρχεία εικόνων.

- Η παράμετρος texelArray είναι δείκτης στον πίνακα που περιέχει τις χρωματικές τιμές των texels. Στην περίπτωση που ο πίνακας είναι μονοδιάστατος, τα στοιχεία του πίνακα προσδιορίζουν τις τιμές των

texels ανά n-άδες όπου n το πλήθος των χρωματικών συνιστωσών που περιγράφουν κάθε texel.

7.3 Ρυθμίσεις απόδοσης υφής

Στους αλγόριθμους απόδοσης υφής ο προγραμματιστής μπορεί να πραγματοποιήσει διάφορες ρυθμίσεις με την χρήση της εντολής glTexParameter*:

glTexParameter{if}{v}(GLenum target, GLenum parameterName, TYPE parameterValue);

Όπου το όρισμα target αναφέρεται στο αν η ρύθμιση επιβάλλεται στην κατηγορία μονοδιάστατων ή δισδιάστατων υφών με την τιμή GL_TEXTURE_1D ή GL_TEXTURE_2D αντίστοιχα. Το όρισμα parameterName δίνεται το όνομα της παραμέτρου που ορίζεται. Ενώ με το όρισμα parameterValue δίνεται την τιμή ή ο πίνακας τιμών που προσδιορίζει την παράμετρο parameterName.

Οι ρυθμίσεις που μπορούν να γίνουν με τη χρήση της εντολής glTexParameter* αφορούν την σμίκρυνση ή μεγέθυνση υφής και αποκοπή ή επανάληψη υφής με χρήση των κατάλληλων ορισμάτων.

7.4 Αυτόματη απόδοση υφής σε τετραγωνικές επιφάνειες

Σε σύνθετες επιφάνειες η διαδικασία απόδοσης υφής είναι πιο πολύπλοκη διαδικασία. Παρόλα αυτά η OpenGL παρέχει στον προγραμματιστή τη δυνατότητα να αποδώσει αυτόματα συντεταγμένες υφής σε τετραγωνικές επιφάνειες με σχετικά απλό τρόπο, τη χρήση της εντολής gluQuadricTexture :

void gluQuadricTexture (GLUquadric *quadObject, GLboolean textureCoords);

Όπου το όρισμα quadObject καθορίζει το αντικείμενο που αντιστοιχεί στην τετραγωνική επιφάνεια. Ενώ το όρισμα textureCoords καθορίζει αν ενεργοποιεί ή απενεργοποιεί την απόδοση υφής στο εκάστοτε αντικείμενο με τις τιμές GL_TRUE ή GL_FALSE αντίστοιχα.

Με την εντολή gluQuadricTexture αποδίδεται η υφή σε όλο το εύρος της τετραγωνικής επιφανείας. Επίσης οι συντεταγμένες της υφής που αποδίδονται εκτείνονται στο εύρος τιμών [0,1] σε όλες τις διευθύνσεις (s και t).

7.5 Διαχείριση πολλαπλών πινάκων υφής

Αρκετές γραφικές εφαρμογές απαιτούν τη χρήση περισσοτέρων από ένα πίνακα υφής. Π.χ. όταν στη σκηνή υπάρχουν επιφάνειες με διαφορετικά χαρακτηριστικά υφής, απαιτείται η τήρηση πολλαπλών πινάκων υφής.

Η OpenGL επιτρέπει τη διαχείριση πολλαπλών πινάκων υφής αναθέτοντας σε κάθε πίνακα και από ένα αναγνωριστικό αριθμό. Η ανάθεση αναγνωριστικών αριθμών σε κάθε πίνακα επιτρέπει την ανάκλησή του όποτε είναι αναγκαίο. Αυτή η προσέγγιση είναι η πιο αποτελεσματική σε σχέση με την επαναλαμβανόμενη φόρτωση του κάθε πίνακα πριν τη χρήση του.

Αρχικά πρέπει να δημιουργηθεί ένα πλήθος αναγνωριστικών αριθμών υφής με την εντολή glGenTextures:

void glGenTextures(GLsizei n, GLuint *textureID);

όπου textureID ο πίνακας που περιέχει τους αναγνωριστικούς αριθμούς και n το πλήθος των αναγνωριστικών αριθμών.

Κατόπιν η ανάθεση αναγνωριστικού αριθμού σε ένα πίνακα υφής γίνεται με την εντολή glBindTexture:

void glBindTexture(GLenum target, GLuint textureID);

Όπου η παράμετρος target παίρνει την τιμή GL_TEXTURE_1D ή GL_TEXTURE_2D για μονοδιάστατη και δισδιάστατη υφή αντίστοιχα. Το όρισμα textureID αντιστοιχεί στον αναγνωριστικό αριθμό που ανατίθεται στην τρέχουσα υφή.

7.6 Συνδυασμός μορφοποιήσεων

Η μηχανής της OpenGL δίνει στον προγραμματιστή τη δυνατότητα να καθορίσει τον τρόπο με τον οποίο αλληλεπιδρούν οι μεταβλητές κατάστασης χρώματος και της τρέχουσας υφής. Η αλληλεπίδραση των μεταβλητών αυτών καθορίζεται με την εντολή glTexEnv*():

void glTexEnv{if}{v}(GLenum target, GLenum pname, TYPE param);

Όπου ορίζει την τρέχουσα λειτουργία της υφής. Η παράμετρος target ορίζει ένα περιβάλλον υφής το οποίο μπορεί να πάρει τις τιμές GL_TEXTURE_ENV, GL_TEXTURE_FILTER_CONTROL ή GL_POINT_SPRITE.

Η παράμετρος pname καθορίζει το συμβολικό όνομα μια μονό-τιμή παράμετρος περιβάλλοντος υφής. Η παράμετρος αυτή παίρνει διάφορα ορίσματα όπως το GL_TEXTURE_ENV_MODE και το GL_COMBINE_ALPHA.

Η παράμετρος param παίρνει δυο τύπους GLfloat και Glint ανάλογα με το πώς χρησιμοποιείται η εντολή. Η παράμετρος αυτή καθορίζει τον τρόπο με τον οποίο αλληλεπιδρά η υφή με τις παραμέτρους κατάστασης χρώματος και επιδέχεται ένα μεγάλο αριθμό από σταθερές-επιλογές όπως η GL_REPLACE (το χρώμα της υφής αντικαθιστά το χρώμα της επιφανείας) και GL_MODULATE (το χρώμα που ορίζει η υφή πολλαπλασιάζεται με το τρέχον χρώμα της επιφανείας).

8 Ολοκληρωμένη εφαρμογή προσομοίωσης ηλιακού συστήματος

Η εφαρμογή αυτή είναι μια προσπάθεια προσομοίωσης του ηλιακού συστήματος. Στην εφαρμογή αυτή συμπεριλαμβάνονται διάφορα στοιχεία από όλα τα παραπάνω κεφάλαια (που αφορούν την υλοποίηση σε OpenGL) για την εμφάνιση του επιθυμητού αποτελέσματος.

8.1 Παρουσίαση κώδικα

Ο φάκελος του project περιλαμβάνει 5 αρχεία από τα οποία 2 είναι headers με όνομα “camera.h” και “ImageLoad.h” και τα υπόλοιπα 3 είναι αρχεία .cpp με όνομα “ camera.cpp”, “ImageLoad.cpp” και “mainSolarSystem.cpp”.

8.1.1 camera.h

Στο header αυτό ορίζεται το interface της κλάση “CCamera”, καθώς και δομές οι οποίες θα χρησιμοποιηθούν για την απόδοση της κάμερας.

```
// Need to include it here because the GL* types are required
#include <GL\freeglut.h>
#define PI 3.1415265359
#define PIdiv180 3.1415265359/180.0

//Float 3d-vect, normally used
struct SF3dVector
{
     GLfloat x,y,z;
};
struct SF2dVector
{
     GLfloat x,y;
};

class CCamera
{
private:
     SF3dVector Position;
     /*Not used for rendering the camera, but for
"moveforwards". So it is not necessary to "actualize" it
always. It is only actualized when ViewDirChanged is true and
moveforwards is called*/
     SF3dVector ViewDir;
```

```
     bool ViewDirChanged;
     GLfloat RotatedX, RotatedY, RotatedZ;
     void GetViewDir ( void );
public:
     //inits the values (Position: (0|0|0) Target: (0|0|-1))
     CCamera();
     /*executes some glRotates and a glTranslate command
Note: You should call glLoadIdentity before using Render */

     void Render ( void );
     void Move ( SF3dVector Direction );
     void RotateX ( GLfloat Angle );
     void RotateY ( GLfloat Angle );
     void RotateZ ( GLfloat Angle );
     void RotateXYZ ( SF3dVector Angles );
     void MoveForwards ( GLfloat Distance );
     void StrafeRight ( GLfloat Distance );
};

SF3dVector F3dVector(GLfloat x, GLfloat y, GLfloat z );
SF3dVector AddF3dVectors(SF3dVector * u, SF3dVector * v);
void AddF3dVectorToVector(SF3dVector * Dst, SF3dVector * V2);
```

8.1.2 camera.cpp

Το αρχείο αυτό υλοποιεί το περιεχόμενο του header “camera.h”. Εδώ υλοποιούνται όλες οι συναρτήσεις που θα χρειαστούν για το χειρισμό της κάμερας της εφαρμογής.

```
#include "camera.h"
#include "math.h"
#include <iostream>
#include "windows.h"
SF3dVector F3dVector ( GLfloat x, GLfloat y, GLfloat z )
{
     SF3dVector tmp;
     tmp.x = x;
     tmp.y = y;
     tmp.z = z;
     return tmp;
}
SF3dVector AddF3dVectors (SF3dVector* u, SF3dVector* v)
{
     SF3dVector result;
     result.x = u->x + v->x;
     result.y = u->y + v->y;
     result.z = u->z + v->z;
     return result;
```

```
}
//Υπολογίζει νέα θέση στις συντεταγμένες
void AddF3dVectorToVector (SF3dVector * Dst, SF3dVector * V2)
{
     Dst->x += V2->x;
     Dst->y += V2->y;
     Dst->z += V2->z;
}


CCamera::CCamera()
{
     //Init with standard OGL values:
     Position = F3dVector(0.0, 0.0, 0.0);
     ViewDir = F3dVector(0.0, 0.0, -1.0);
     ViewDirChanged = false;
     //Only to be sure:
     RotatedX = RotatedY = RotatedZ = 0.0;
}

void CCamera::GetViewDir( void )
{
     SF3dVector Step1, Step2;
     //Rotate around Y-axis:
     Step1.x = cos( (RotatedY + 90.0) * PIdiv180);
     Step1.z = -sin( (RotatedY + 90.0) * PIdiv180);
     //Rotate around X-axis:
     double cosX = cos (RotatedX * PIdiv180);
     Step2.x = Step1.x * cosX;
     Step2.z = Step1.z * cosX;
     Step2.y = sin(RotatedX * PIdiv180);
     //Rotation around Z-axis not yet implemented, so:
     ViewDir = Step2;
}
//κίνηση κάμερας πάνω και κάτω
void CCamera::Move (SF3dVector Direction)
{
     AddF3dVectorToVector(&Position, &Direction );
}
//περιστροφή της κάμερα ως προς τον άξονα y
void CCamera::RotateY (GLfloat Angle)
{
     RotatedY += Angle;
     ViewDirChanged = true;
}
//περιστροφή της κάμερα ως προς τον άξονα x
void CCamera::RotateX (GLfloat Angle)
{
     RotatedX += Angle;
     ViewDirChanged = true;
}
```

```
//ορισμός θέσης της κάμερας
void CCamera::Render( void )
{
     glRotatef(-RotatedX , 1.0, 0.0, 0.0);
     glRotatef(-RotatedY , 0.0, 1.0, 0.0);
     glRotatef(-RotatedZ , 0.0, 0.0, 1.0);
     glTranslatef( -Position.x, -Position.y, -Position.z );
}
//κίνηση της κάμερας εμπρός πίσω
void CCamera::MoveForwards( GLfloat Distance )
{
     if (ViewDirChanged) GetViewDir();
     SF3dVector MoveVector;
     MoveVector.x = ViewDir.x * -Distance;
     MoveVector.y = ViewDir.y * -Distance;
     MoveVector.z = ViewDir.z * -Distance;
     AddF3dVectorToVector(&Position, &MoveVector );
}
//κίνηση της κάμερας αριστερά και δεξιά
void CCamera::StrafeRight ( GLfloat Distance )
{
     if (ViewDirChanged) GetViewDir();
     SF3dVector MoveVector;
     MoveVector.z = -ViewDir.x * -Distance;
     MoveVector.y = 0.0;
     MoveVector.x = ViewDir.z * -Distance;
     AddF3dVectorToVector(&Position, &MoveVector);
}
```

8.1.3 ImageLoad.h

Το header αυτό χρησιμοποιείται για φόρτωμα εικόνας.

```
#ifndef _IMAGELOAD_
#define _IMAGELOAD_

struct Image {
    unsigned long sizeX;
    unsigned long sizeY;
    char *data;
};
typedef struct Image Image;

int ImageLoad(char *filename, Image *image);
#endif
```

8.1.4 ImageLoad.cpp

Στο αρχείο αυτό υλοποιείται το περιεχόμενο του παραπάνω header. Το πρόγραμμα θα μπορεί να φορτώσει εικόνες σε bitmap αρχεία, έως 24 bit με μόνο ένα επίπεδο.

```
#include <stdio.h>
#include <stdlib.h>
#include "ImageLoad.h"
/* quick and dirty bitmap loader...for 24 bit bitmaps with 1
plane only.*/
int ImageLoad(char *filename, Image *image) {
    FILE *file;
    unsigned long size;                 // size of the image
in bytes.
    unsigned long i;                    // standard counter.
    unsigned short int planes;          // number of planes
in image (must be 1)
    unsigned short int bpp;             // number of bits per
pixel (must be 24)
    char temp;                          // temporary color
storage for bgr-rgb conversion.

    // make sure the file is there.
    if ((file = fopen(filename, "rb"))==NULL)
    {
     printf("File Not Found : %s\n",filename);
     return 0;
    }

    // seek through the bmp header, up to the width/height:
    fseek(file, 18, SEEK_CUR);

    // read the width
    if ((i = fread(&image->sizeX, 4, 1, file)) != 1) {
     printf("Error reading width from %s.\n", filename);
     return 0;
    }
    printf("Width of %s: %lu\n", filename, image->sizeX);

    // read the height
    if ((i = fread(&image->sizeY, 4, 1, file)) != 1) {
     printf("Error reading height from %s.\n", filename);
     return 0;
    }
    printf("Height of %s: %lu\n", filename, image->sizeY);

    /* calculate the size (assuming 24 bits or 3 bytes per
pixel). */
```

```
    size = image->sizeX * image->sizeY * 3;

    // read the planes
    if ((fread(&planes, 2, 1, file)) != 1) {
     printf("Error reading planes from %s.\n", filename);
     return 0;
    }
    if (planes != 1) {
     printf("Planes from %s is not 1: %u\n", filename,
planes);
     return 0;
    }

    // read the bpp
    if ((i = fread(&bpp, 2, 1, file)) != 1) {
     printf("Error reading bpp from %s.\n", filename);
     return 0;
    }
    if (bpp != 24) {
     printf("Bpp from %s is not 24: %u\n", filename, bpp);
     return 0;
    }

    // seek past the rest of the bitmap header.
    fseek(file, 24, SEEK_CUR);

    // read the data.
    image->data = (char *) malloc(size);
    if (image->data == NULL) {
     printf("Error allocating memory for color-corrected
image data");
     return 0;
    }

    if ((i = fread(image->data, size, 1, file)) != 1) {
     printf("Error reading image data from %s.\n", filename);
     return 0;
    }
    // reverse all of the colors. (bgr -> rgb)
    for (i=0;i<size;i+=3) {
     temp = image->data[i];
     image->data[i] = image->data[i+2];
     image->data[i+2] = temp;
    }
    return 1;
}
```

8.1.5 mainSolarSystem.cpp

Στο αρχείο αυτό υπάρχει η συνάρτηση main και όλες οι απαραίτητες συναρτήσεις για την σχεδίαση και απεικόνιση της εφαρμογής.

```
/*****************************************************************
**********
ESC: exit

Κίνηση κάμερας:
w : εμπρός
s : πίσω
a : αριστερά
d : δεξιά
KEY_UP : στροφή πάνω
KEY_DOWN : στροφή κάτω
KEY_RIGHT : στροφή δεξιά
KEY_LEFT : στροφή αριστερά
r : πάνω
f : κάτω

*****************************************************************
**********/
#include <stdio.h> // Header file for standard file i/o.
#include <stdlib.h> // Header file for malloc/free.
#include <GL\freeglut.h>
#include "camera.h"
#include "ImageLoad.h"
#include <iostream>

using namespace std;
//επιστρέφει float τυχαίους αριθμούς ξεκινώντας από το 1.0f
float FloatRand( float MaxVal )
{
     return ( (float)rand( ) / ( (float)RAND_MAX + 1.0f ) ) *
MaxVal;
}

//Δημιουργία αντικειμένου της κλάσης CCamera
CCamera Camera;
//Πίνακας για το σημειακό φωτισμό
GLfloat    lightPos[] = { 0.0f, 0.0f, 0.0f, 1.0f };

/* Συνάρτηση διαχείρισης γεγονότων διαστάσεων του παραθύρου
της εφαρμογής. Περιέχει κώδικα σχετικά με τη ρύθμιση των
διάφορων προβολών που θα χρησιμοποιηθούν */
void reshape(int x, int y)
{
     //Nothing is visible then, so return
     if (y == 0 || x == 0) return;
```

```
     //Set a new projection matrix
     glMatrixMode(GL_PROJECTION);
     glLoadIdentity();

     //Angle of view:40 degrees
     //Near clipping plane distance: 0
     //Far clipping plane distance: 20.0
     gluPerspective(40.0,(GLdouble)x/(GLdouble)y,0,20.0);
     //ορισμός συμμετρικής προοπτικής προβολής
     glMatrixMode(GL_MODELVIEW);
     //Use the whole window for rendering
     glViewport(0,0,x,y);
}//Τέλος συνάρτηση reshape
////////////////////////////////////////////////

//Ορισμός τετραγωνικής επιφάνειας
GLUquadric *sphere;
/*
Η μεταβλητή "texture" μας βοηθάει να δώσουμε ένα unique name
σε ένα texture object με την χρήση της συνάρτησης
glGenTextures, επίσης μας βοηθάει να το κάνουμε bind, δηλαδή
να το δένουμε ώστε  να μπορούμε να εργαστούμε πάνω του με τη
βοήθεια τις συνάρτησης glBindTexture */

//ορισμός πίνακα που θα αποθηκεύονται textures
GLuint texture[5];

//Συνάρτηση φορτώματος εικόνων
void Init(GLfloat x,GLfloat y){

    int i;
    // Δήλωση αντικειμένου τύπου Image
    Image *image1;
    //Και αρχικοποιείτε με την χρήση τις malloc
    image1=(Image*)malloc(5*sizeof(Image));
    //φόρτωση της εικόνας
    if(!ImageLoad("sol.bmp",&image1[0])){
      fprintf(stderr,"Couldn't find the image file\n");
    }
    if(!ImageLoad("world.bmp",&image1[1])){
      fprintf(stderr,"Couldn't find the image file\n");
    }
    if(!ImageLoad("moon.bmp",&image1[2])){
      fprintf(stderr,"Couldn't find the image file\n");
    }
    if(!ImageLoad("stars.bmp",&image1[3])){
      fprintf(stderr,"Couldn't find the image file\n");
    }
    if(!ImageLoad("comets.bmp",&image1[4])){
      fprintf(stderr,"Couldn't find the image file\n");
    }
```

```
    //ενεργοποίηση των textures
    glEnable(GL_TEXTURE_2D);
    /*Δημιουργία ενός texture object με unique name
'texture'*/
    glGenTextures(5,texture);
    /*Κάνουμε bind ώστε να μπορούμε να εργαστούμε πάνω του με
την συνάρτηση glBindTexture*/
    for(i=0;i<5;i++){
     glBindTexture(GL_TEXTURE_2D,texture[i]);
          sphere=gluNewQuadric();
    /*Εδώ θέτουμε magnification και minification texture
filters. Αυτό γίνετε με την χρήση της συνάρτησης
glTexParameter(i,f).*/

     glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_MAG_FILTER,GL_L
INEAR);

     glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_MIN_FILTER,GL_L
INEAR);

    /* Σε αυτό το σημείο με τη χρήση της συνάρτησης
glTexImage μεταφέρουμε τα στοιχεία της εικόνας που έχουμε
φορτώσει ως παράμετρο μέσα στο texture object μας....*/

     glTexImage2D(GL_TEXTURE_2D,0,3,image1[i].sizeX,image1[i]
.sizeY,0,GL_RGB,GL_UNSIGNED_BYTE,image1[i].data);
     gluQuadricTexture(sphere,GL_TRUE);
     }
     free(image1);
}//Τέλος συνάρτησης Init
////////////////////////////////////////////

int el=0;
float x=-400,y=0,z=0,size=0;

//συνάρτηση σχεδίασης γραφικών
void Display(void)
{
    glClear(GL_COLOR_BUFFER_BIT);
    glDisable(GL_LIGHTING);// Sun

    glLoadIdentity();
    //καθορισμός θέσης της κάμερας
    Camera.Render();

#pragma region asteria
    // Store the current matrix
    glPushMatrix();

    // Reset and transform the matrix.
```

```
// Enable/Disable features
glPushAttrib(GL_ENABLE_BIT);
glEnable(GL_TEXTURE_2D);

// Just in case we set all vertices to white.
glColor4f(1,1,1,1);
GLfloat val = 300.0f;

// Render the front quad
glBindTexture(GL_TEXTURE_2D, texture[3]);
glBegin(GL_QUADS);
    glTexCoord2f(0, 0); glVertex3f(val, -val, val);
    glTexCoord2f(1, 0); glVertex3f(-val, -val, val);
    glTexCoord2f(1, 1); glVertex3f(-val, val, val);
    glTexCoord2f(0, 1); glVertex3f(val, val, val);
glEnd();

// Render the left quad
glBindTexture(GL_TEXTURE_2D, texture[3]);
glBegin(GL_QUADS);
    glTexCoord2f(0, 0); glVertex3f(val, -val, -val);
    glTexCoord2f(1, 0); glVertex3f(val, -val, val);
    glTexCoord2f(1, 1); glVertex3f(val, val, val);
    glTexCoord2f(0, 1); glVertex3f(val, val, -val);
glEnd();

// Render the back quad
glBindTexture(GL_TEXTURE_2D, texture[3]);
glBegin(GL_QUADS);
    glTexCoord2f(0, 0); glVertex3f(-val, -val, -val);
    glTexCoord2f(1, 0); glVertex3f(val, -val, -val);
    glTexCoord2f(1, 1); glVertex3f(val, val, -val);
    glTexCoord2f(0, 1); glVertex3f(-val, val, -val);

glEnd();

// Render the right quad
glBindTexture(GL_TEXTURE_2D, texture[3]);
glBegin(GL_QUADS);
    glTexCoord2f(0, 0); glVertex3f(-val, val, -val);
    glTexCoord2f(1, 0); glVertex3f(-val, val, val);
    glTexCoord2f(1, 1); glVertex3f(-val, -val, val);
    glTexCoord2f(0, 1); glVertex3f(-val, -val, -val);
glEnd();

// Render the top quad
glBindTexture(GL_TEXTURE_2D, texture[3]);
glBegin(GL_QUADS);
    glTexCoord2f(0, 0); glVertex3f(val, val, -val);
    glTexCoord2f(1, 0); glVertex3f(val, val, val);
    glTexCoord2f(1, 1); glVertex3f(-val, val, val);
```

```
        glTexCoord2f(0, 1); glVertex3f(-val, val, -val);
    glEnd();

    // Render the bottom quad
    glBindTexture(GL_TEXTURE_2D, texture[3]);
    glBegin(GL_QUADS);
        glTexCoord2f(0, 0); glVertex3f(-val, -val, -val);
        glTexCoord2f(1, 0); glVertex3f(-val, -val, val);
        glTexCoord2f(1, 1); glVertex3f(val, -val, val);
        glTexCoord2f(0, 1); glVertex3f(val, -val, -val);
    glEnd();

    // Restore enable bits and matrix
    glPopAttrib();
    glPopMatrix();
#pragma endregion

#pragma region planites

     //glScalef(3.0,1.0,3.0);//κλιμάκωση

     static int lastMs = glutGet( GLUT_ELAPSED_TIME );
     int curMs = glutGet( GLUT_ELAPSED_TIME );
     double dt = ( curMs - lastMs ) / 1000.0;
     lastMs = curMs;

     static float earth = 0.5;
     static float moon = 0.2;

     earth += 2 * dt;
     moon += 36 * dt;


glPushMatrix();
    {
        glColor3ub( 255, 255, 0 );
        glBindTexture (GL_TEXTURE_2D, texture[0]);
        gluSphere(sphere,13.92,80,80);

        // Position light source at 0,0,0
        glLightfv(GL_LIGHT0,GL_POSITION,lightPos);
        glEnable(GL_LIGHTING);
        glEnable(GL_LIGHT0);

        glPushMatrix();
        {
        /*Εδώ λέμε στην OpenGL να χρησιμοποιήσει το texture
με όνομα 'texture[1]' αυτό γίνετε με τη χρήση της εντολής
glBindTexture πριν το render code.*/
            //Ορισμός της γής
            glBindTexture (GL_TEXTURE_2D, texture[1]);
```

```
            glRotatef( earth, 0, 0, 1 );
            glTranslatef( 60, 0, 0 );

            glColor3ub( 0, 255, 255 );
            gluSphere(sphere,1.2742,80,80);

            //περιστροφή της Γής γύρο από το εαυτό της
               glPushMatrix();
            {
               glRotatef( moon, 0, 0, -1 );
               gluSphere(sphere,1.2742,80,80);
            }
            glPopMatrix();
            //ορισμός φεγγαριού
            glPushMatrix();
            {
               glBindTexture (GL_TEXTURE_2D, texture[2]);
               glRotatef( moon, 0, 0, 1 );
               glTranslatef( 2, 0, 0 );
               glColor3ub( 128, 128, 128 );
               gluSphere(sphere,0.3474,80,80);
            }
            glPopMatrix();
         }
         glPopMatrix();
      }
      glPopMatrix();
#pragma endregion

#pragma region komites
      //ορισμός κομητών
      glPushMatrix();
            /* Μια σειρά από ελέγχους και τυχαίες μεταβλητές
που έχουν να κάνουν με το μέγεθος τη φορά και το χώρο που θα
κινηθεί ο κομήτης */
            if(x<=-500 or x>=500){
                  x=rand() % 60 - 30;
                  y=rand() % 600 - 300;
                  z=rand() % 290 + 10;
                  /*παίρνει τιμή από 0.1 μέχρι 1 που αντιστοιχεί
στο μέγεθος κομήτη */
                  size=FloatRand(1);
                  el=rand() % 60 - 30;

                  if(el<=0)//Αν είναι αρνητικό
                        z=-1*z;//η τιμή του z γίνεται αρνητική
                  if(x>=0){
                        el=1;//έλεγχος προς πια κατεύθυνση θα
                        //κινείται ο κομήτης
                        x=490;//από πού θα ξεκινήσει
                  }
```

```
                else{
                    el=2;
                    x=-490;
                }
            }
            if(el==1){
                x=x-10;
            }
            else
                x=x+10;
            //ορισμός κομήτη
            glBindTexture (GL_TEXTURE_2D, texture[4]);
            glColor3f(1,1,1);
            glTranslatef(x, y, z);

            gluSphere(sphere,size,10,10);

        glPopMatrix();
#pragma endregion

        glFlush();
        glutSwapBuffers();
}//Τέλος συνάρτησης Display
////////////////////////////////////////

/* Συνάρτηση διαχείρισης γεγονότων πληκτρολογίου που αφορούν
την κάμερα */
void specialKeys(int key,int x, int y)
{
        switch (key)
        {
        case GLUT_KEY_UP:
            Camera.RotateX(5.0);
            break;
        case GLUT_KEY_DOWN:
            Camera.RotateX(-5.0);
            break;

        case GLUT_KEY_LEFT:
            Camera.RotateY(5.0);
            break;
        case GLUT_KEY_RIGHT:
            Camera.RotateY(-5.0);
            break;
        }
glutPostRedisplay();
}//Τέλος συνάρτησης specialKeys
////////////////////////////////////////

/* Συνάρτηση διαχείρισης γεγονότων πληκτρολογίου που αφορούν
την κάμερα */
```

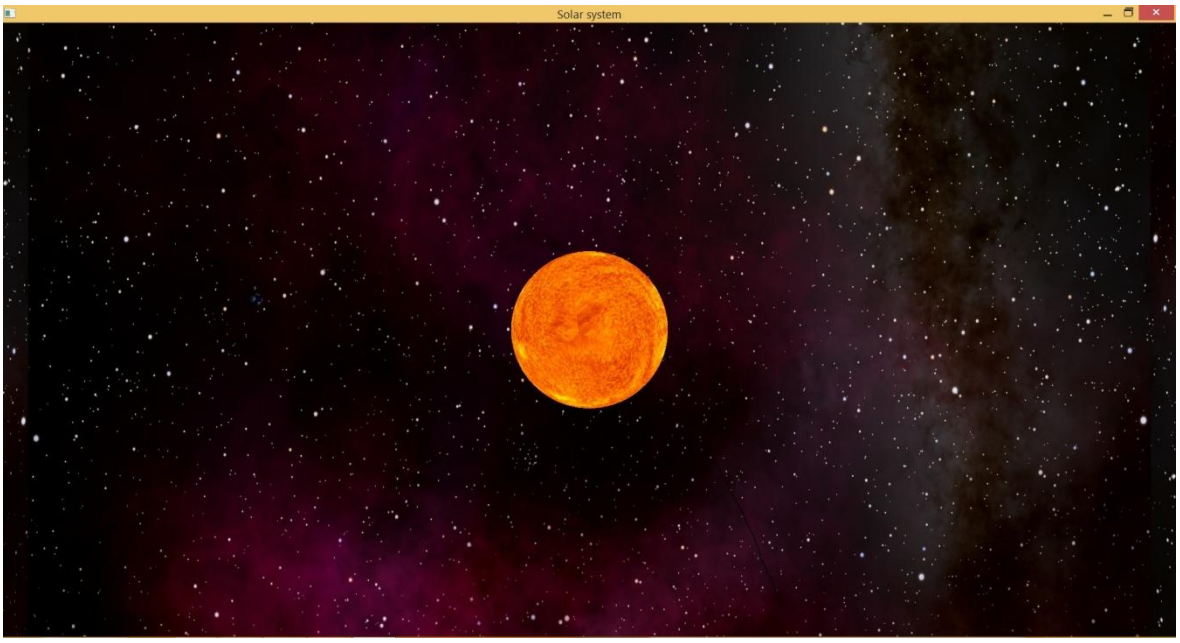
```
void KeyDown(unsigned char key, int x, int y)
{
	switch (key)
	{
	case 27:	//ESC
		PostQuitMessage(0);
		break;
	case 'w':
		Camera.MoveForwards( -1.1 ) ;
		break;
	case 's':
		Camera.MoveForwards( 1.1 ) ;
		break;
	case 'a':
		Camera.StrafeRight(-1.1);
		break;
	case 'd':
		Camera.StrafeRight(1.1);
		break;
	case 'f':
		Camera.Move(F3dVector(0.0,-0.3,0.0));
		break;
	case 'r':
		Camera.Move(F3dVector(0.0,0.3,0.0));
		break;
	}
	glutPostRedisplay();
}//Τέλος συνάρτησης KeyDown
////////////////////////////////////////////////

bool bl=false, br=false;
/* Συνάρτηση διαχείρισης γεγονότων ποντικιού για την
περιστροφή της κάμερας, που συνδυάζεται με την παρακάτω*/
void mouseButtonClicked(int button,int state,int x, int y)
{
	if(state==GLUT_DOWN)
	{
		if (button==GLUT_LEFT_BUTTON){
			bl=true;
			br=false;
		}
		else if(button==GLUT_RIGHT_BUTTON){
			bl=false;
			br=true;
		}
	}
	glutPostRedisplay();
}//Τέλος συνάρτησης mouseButtonClicked
////////////////////////////////////////////////////
int y1=0, x1=0;
```

```
/* Διαχείριση γεγονότων ποντικιού για την περιστροφή της
κάμερας */
void mouseActiveMotion(int x, int y)
{
     if(br==true){
          if(y>y1)
               Camera.RotateX(0.001 *y);
          else
               Camera.RotateX(-0.001 *y);
          y1=y;
     }

     if(bl==true){
          if(x>x1)
               Camera.RotateY(0.001 *x);
          else
               Camera.RotateY(-0.001 *x);
     x1=x;
     }

     glutPostRedisplay();
}//Τέλος συνάρτησης mouseActiveMotion
//////////////////////////////////////////////////

/*Συνάρτηση επανάκλησης με χρονοδιακόπτη που εκτελείται ανά 1
ms καλώντας τη display()*/
void timer(int extra)
{
    glutPostRedisplay();
    glutTimerFunc(1, timer, 0);
}//Τέλος συνάρτησης timer
//////////////////////////////////////////////////

//Η συνάρτηση main
int main(int argc, char **argv)
{
     glutInit(&argc, argv);
     glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
     glutInitWindowSize(640,480);
     glutCreateWindow("Solar system");
     Camera.Move( F3dVector(0.0, 0.0, 0.0 ));
     Camera.MoveForwards( -150.0 );
     Camera.RotateY(180.0);

     Init(640,480);
     glutDisplayFunc(Display);
     glutReshapeFunc(reshape);
     glutKeyboardFunc(KeyDown);
     glutSpecialFunc(specialKeys);

     glutMouseFunc(mouseButtonClicked);
```

```
        glutMotionFunc(mouseActiveMotion);

        glutTimerFunc(0, timer, 0);

        glutMainLoop();
        return 0;
}
```

8.2 Τρέξιμο του κώδικα της εφαρμογής

Με την εκτέλεση του κώδικα της εφαρμογής θα εμφανιστεί το παράθυρο του Σχήματος 16.



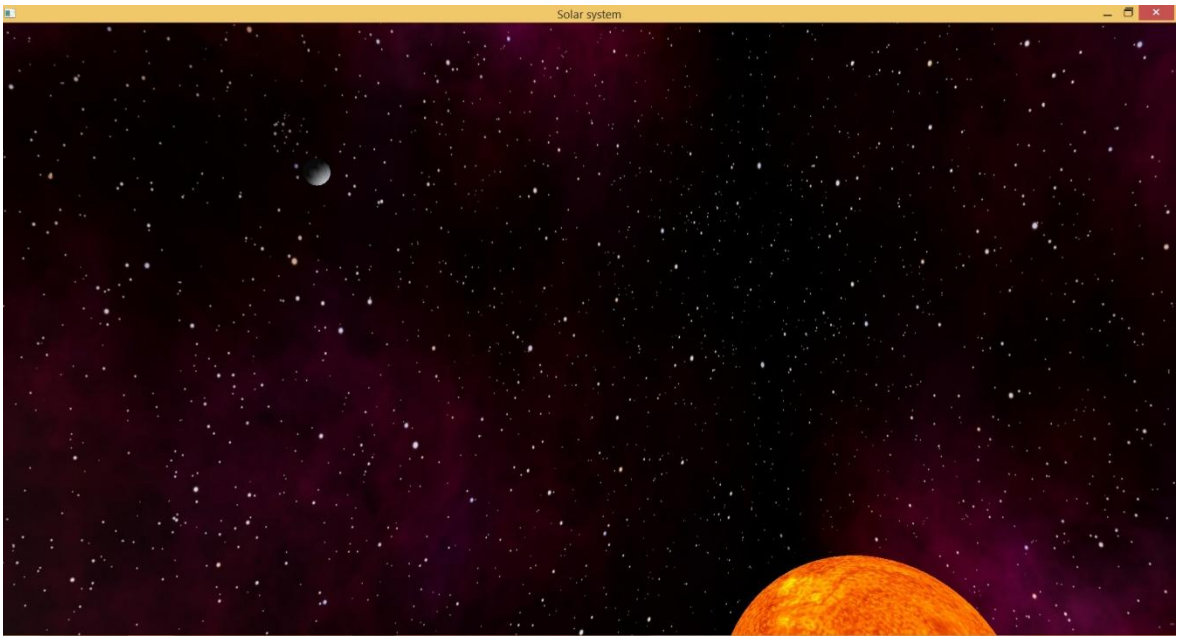
Σχήμα 16: Παράθυρο της εφαρμογής μόλις εκτελεστεί ο κώδικας

Αρχικά εμφανίζεται ο ήλιος στο κέντρο της σκηνής, αριστερά η Γή με την σελήνη και γύρο το σύμπαν. Αν προχωρήσουμε την κάμερα προς το σημείο της Γής (βλέπε Σχήμα 17) θα παρατηρήσουμε καλυτέρα τα εξής:

- Η Γή περιστρέφεται γύρο από τον ήλιο και γύρο από τον εαυτό της.
- Η σελήνη περιστρέφεται γύρο από την Γή.
- Η Γή και η σελήνη φωτίζονται από τον ήλιο, δηλαδή τα σημεία που “βλέπουν” τον ήλιο φωτίζονται ενώ τα υπόλοιπα όχι.

Σχήμα 17: Εστίαση κάμερας στην Γή

Στην εφαρμογή αυτή δημιουργούνται κομήτες που το μέγεθος τους ποικίλει (μέσα σε κάποια λογικά πλαίσια φυσικά) και είναι τυχαίο. Οι κομήτες αυτοί εμφανίζονται τυχαία σε οποιοδήποτε σημείο της σκηνής (εκτός στα σημεία που μπορούν να είναι ο ήλιος, η Γή και η σελήνη) και κινούνται αρκετά γρήγορα από την μία άκρη της σκηνής στην άλλη ώστε να φαίνονται πιο ρεαλιστικοί. Πάντα υπάρχει κάποιος κομήτης που κινείται στη σκηνή, δηλαδή όταν εξαφανίζεται ένας δημιουργείται και εμφανίζεται κάποιος άλλος. Στο Σχήμα 18 εμφανίζεται ένας κομήτης πάνω αριστερά.



Σχήμα 18: κομήτης

8.2.1 Χειρισμός της κάμερας

Με το χειρισμό της κάμερας της εφαρμογής ο χρήστης μπορεί να μπορεί να πλοηγηθεί σε οποιοδήποτε σημείο της σκηνής. Η κάμερα αυτή μπορεί να μετακινηθεί πάνω, κάτω, αριστερά, δεξιά, εμπρός και πίσω καθώς και να περιστραφεί πάνω, κάτω, αριστερά και δεξιά.

Ο χειρισμός τις κάμερα από το πληκτρολόγιο πραγματοποιείται με τα εξής πλήκτρα:

- Βελάκι πάνω: περιστροφή κάμερας προς τα πάνω
- Βελάκι κάτω: περιστροφή κάμερας προς τα κάτω
- Βελάκι αριστερά: περιστροφή κάμερας αριστερά
- Βελάκι δεξιά: περιστροφή κάμερας δεξιά
- w: κίνηση κάμερας προς τα εμπρός
- s: κίνηση κάμερας προς τα πίσω
- a: κίνηση κάμερας προς τα αριστερά

- d: κίνηση κάμερας προς τα δεξιά
- r: κίνηση κάμερας προς τα πάνω
- f: κίνηση κάμερας προς τα κάτω

Ο χειρισμός της κάμερας με το ποντική:

- Έχοντας πατημένο το αριστερό πλήκτρο του ποντικιού και μεταφέροντας το ποντίκι αριστερά ή δεξιά περιστρέφεται η κάμερα δεξιά ή αριστερά αντίστοιχα.
- Έχοντας πατημένο το δεξί πλήκτρο του ποντικιού και μεταφέροντας το ποντίκι πάνω ή κάτω περιστρέφεται η κάμερα πάνω ή κάτω αντίστοιχα.

Συμπεράσματα

Η χρήση της OpenGL είναι ιδανική για τη σχεδίαση απλών 2D/3D εφαρμογών. Προτείνεται κυρίως για εκπαιδευτικούς σκοπούς που έχουν ως στόχο την εξοικείωση των προγραμματιστών με τις εντολές σχεδίασης και τη χρήση διαφόρων ιδιοτήτων σχεδίασης όπως ο φωτισμός και η υφή. Ο προγραμματιστής μπορεί να πειραματιστεί και να ανάπτυξη σχετικά εύκολα τις δικιές του εφαρμογές έχοντας στην διάθεση του πάνω από 500 εντολές, που μπορούν να ανταποκριθούν σε μεγάλες απαιτήσεις.

Βιβλιογραφία

Α. Ελληνική βιβλιογραφία

Αναγνώστου, Κ. (2008), *Εισαγωγή στην OpenGL*, Άρθρο αναρτημένο στην διεύθυνση ιστοσελίδας <http://videogameslab.wordpress.com/2008/11/05/>.

Βογιατζής, Γ. (2014), Σημειώσεις μαθήματος γραφικά υπολογιστών, Θεσσαλονίκη: Α.Π.Θ., διεύθυνση ιστοσελίδας μαθήματος <http://users.auth.gr/~voyatzis/CG/>.

Λούμος, Β., Σημειώσεις μαθήματος Γραφικά με υπολογιστές, Αθήνα: Εθνικό Μετσόβιο πολυτεχνείο, διεύθυνση ιστοσελίδας μαθήματος <http://www.medialab.ntua.gr/education/ComputerGraphics/>.

Μποζάνης, Π. (2010), *Γραφικά υπολογιστών με OpenGL*, Θεσσαλονίκη: Τζιόλας.

Τσομπίκας, Γ., *Εισαγωγή στον προγραμματισμό 3D γραφικών με το OpenGL,* Άρθρα 1, 2, 3 και 4 αναρτημένα στην διεύθυνση ιστοσελίδας <http://nuclear.mutantstargoat.com/articles/gltut/ >

(2009), Σεμινάρια προγραμματισμού 3D γραφικών, Λαμία: Τ.Ε.Ι., διεύθυνση ιστοσελίδας περιεχομένου <http://teilam3dsem.wordpress.com/>

Β. Αγγλική βιβλιογραφία

Buss, S. and Buss, S. R. (2014), *3D Computer Graphics: A Mathematical Introduction with OpenGL*, England: Cambridge University Press (CUP)

Code examples, online: Codecolony, <http://www.codecolony.de/opengl.htm>

Code examples, online: Lmu, <http://cs.lmu.edu/~ray/notes/openglexamples/>

Guha, S. (2010), Computer Graphics Through OpenGL: From Theory to Experiments, London: Chapman & Hall/CRC.

Howard, T. (2004), An Introduction to Graphics Programming with OpenGL, Manchester: Department of Computer Science University.

Movania, M. M. (2013), *OpenGL Development Cookbook*, Birmingham – Mumbai: Packt.

Official site of OpenGL, online: OpenGL, <https://www.opengl.org/>

Richard S., Wright Jr., Lipchak B. and Haemel N. (2013), *OpenGL SuperBible: Comprehensive Tutorial and Reference*, International: Addison-Wesley.

Semester, S. (2014), *OpenGL and GLUT*, Korea: School of Computer Sci. & Eng. Kyungpook National University.

Shreiner, D., Sellers, G., Kessenich J. M. and Licea-Kane B. M. (2013), *OpenGL Programming Guide: The Official Guide to Learning OpenGL*, 8th Edition, International: Addison-Wesley.

(2014), *OpenGL*, Online: Wikipedia, <http://en.wikipedia.org/wiki/OpenGL>