

Δημιουργία γραφικών με χρήση της βιβλιοθήκης OpenGL για γλώσσα προγραμματισμού C

Πίνακας περιεχομένων

1. Τι είναι η OpenGL.....	2
2. Απαιτούμενο λογισμικό για τον προγραμματισμό με OpenGL - GLUT	3
3. Βασικές εντολές εκκίνησης περιβάλλοντος OpenGL.....	5
3.1 Εκκίνηση περιβάλλοντος	5
3.2 Παραμετροποίηση εμφάνισης παράθυρου	6
4. Θεμελιώδη σχήματα της OpenGL.....	8
5. Εντολές δημιουργίας βασικών δισδιάστατων σχημάτων της OpenGL.....	9
5.1 Κουκίδες (points)	10
5.2 Γραμμές (lines).....	11
5.3 Τρίγωνα (triangles).....	13
5.4 Τετράπλευρα (quadrilaterals).....	14
5.5 Πολύγωνα (polygons).....	16
5.6 Κείμενο (text).....	17
6. Τύποι μεταβλητών	19
7. Αλληλεπίδραση με το πληκτρολόγιο	20
8. Πίνακας χαρακτήρων ASCII	22
9. Αλληλεπίδραση με το ποντίκι.....	24
10. Δημιουργία μενού.....	26
Παραπομπές.....	29

1. Τι είναι η OpenGL

Η OpenGL είναι ένα πρότυπο υλοποίησης βιβλιοθηκών σχεδίασης γραφικών. Εμπεριέχει δηλαδή το σύνολο των συναρτήσεων που πρέπει να υλοποιεί μία βιβλιοθήκη γραφικών προκειμένου να είναι συμβατή με αυτό. Εάν βασιστούμε στο πρότυπο της OpenGL, ο κώδικας που συντάσσουμε είναι ανεξάρτητος πλατφόρμας (platform independent) και μπορεί να εκτελεστεί σε ευρεία γκάμα περιβαλλόντων προγραμματισμού χωρίς ριζική τροποποίηση της δομής του. Υπάρχουν βιβλιοθήκες OpenGL γι' αρκετές γλώσσες προγραμματισμού, το παρών κείμενο αναφέρεται στη βιβλιοθήκη για τη γλώσσα προγραμματισμού C.

Η υλοποίηση της OpenGL περιλαμβάνει τις παρακάτω βιβλιοθήκες:

1. Βασική βιβλιοθήκη (OpenGL core library):

Η βασική βιβλιοθήκη της OpenGL περιέχει τις κύριες εντολές σχεδίασης. Όλες οι εντολές της βιβλιοθήκης αυτής διακρίνονται από το πρόθεμα `gl`. Πολλές από τις συναρτήσεις της δέχονται προκαθορισμένα ορίσματα (συμβολικές σταθερές) τα οποία έχουν οριστεί στη βιβλιοθήκη και αντιστοιχούν σε διάφορες παραμέτρους ή καταστάσεις λειτουργίας. Κατά σύμβαση, οι σταθερές αυτές ξεκινούν με το πρόθεμα `GL_`.

2. OpenGL Utility Library (GLU):

Περιλαμβάνει συναρτήσεις που εκτελούν σύνθετους αλγορίθμους όπως π.χ. τον καθορισμό μητρώων προβολής και το σχηματισμό σύνθετων καμπυλών και επιφανειών. Κάθε υλοποίηση της OpenGL εμπεριέχει τη βιβλιοθήκη GLU. Όλες οι εντολές της βιβλιοθήκης GLU ξεκινούν με το πρόθεμα `glu`.

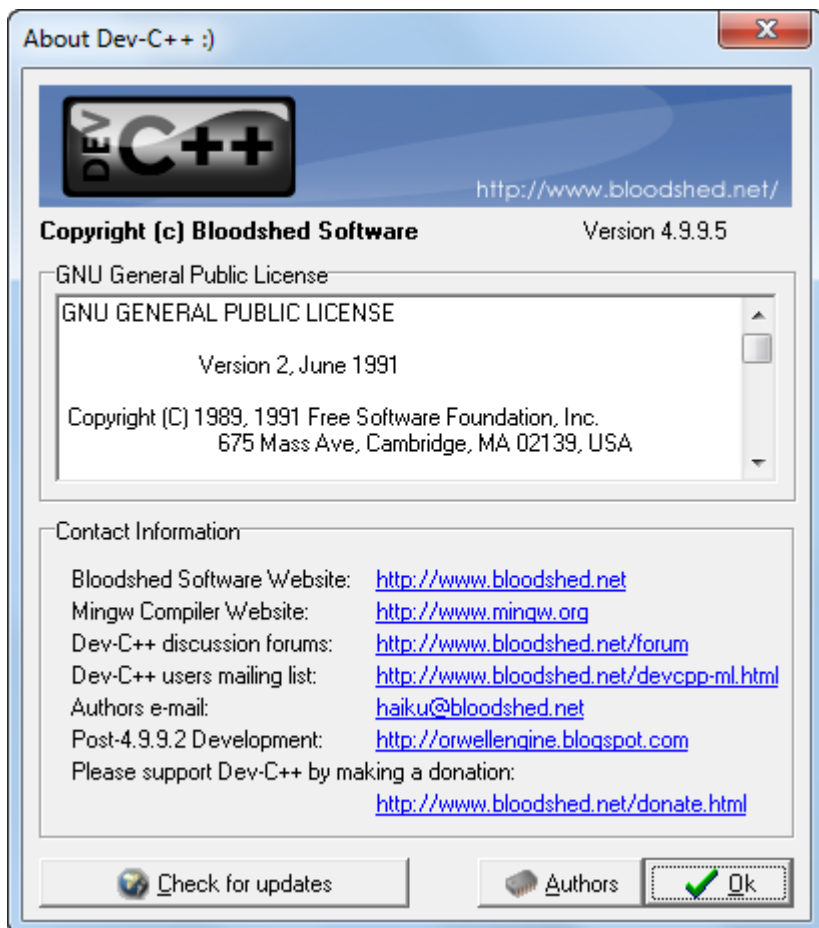
3. OpenGL Utility Toolkit (GLUT):

Όπως αναφέραμε το πρότυπο της OpenGL είναι ανεξάρτητο πλατφόρμας. Ωστόσο μια αυτονόητη απαίτηση ενός προγραμματιστή είναι να έχει τη δυνατότητα να δει το αποτέλεσμα των προγραμμάτων του αλλά και να έχει δηλαδή τη δυνατότητα αλληλεπίδρασης με αυτό. Χρειάζεται δηλαδή εντολές εισόδου-εξόδου. Ωστόσο οι εντολές αλληλεπίδρασης δεν είναι ανεξάρτητες πλατφόρμας και για το λόγο αυτό οι προγραμματιστές σε OpenGL χρησιμοποιούν μία ακόμη βιβλιοθήκη που τους προσφέρει εντολές εισόδου-εξόδου. Μία από τις βιβλιοθήκες που προσφέρει τη λειτουργικότητα αυτή είναι το OpenGL Utility Toolkit (GLUT). Η βιβλιοθήκη αυτή περιλαμβάνει εντολές απεικόνισης παραθύρων στην οθόνη, δημιουργίας menus, διαχείρισης γεγονότων κλπ. Όλες οι εντολές της ξεκινούν με το πρόθεμα `glut`.

2. Απαιτούμενο λογισμικό για τον προγραμματισμό με OpenGL - GLUT

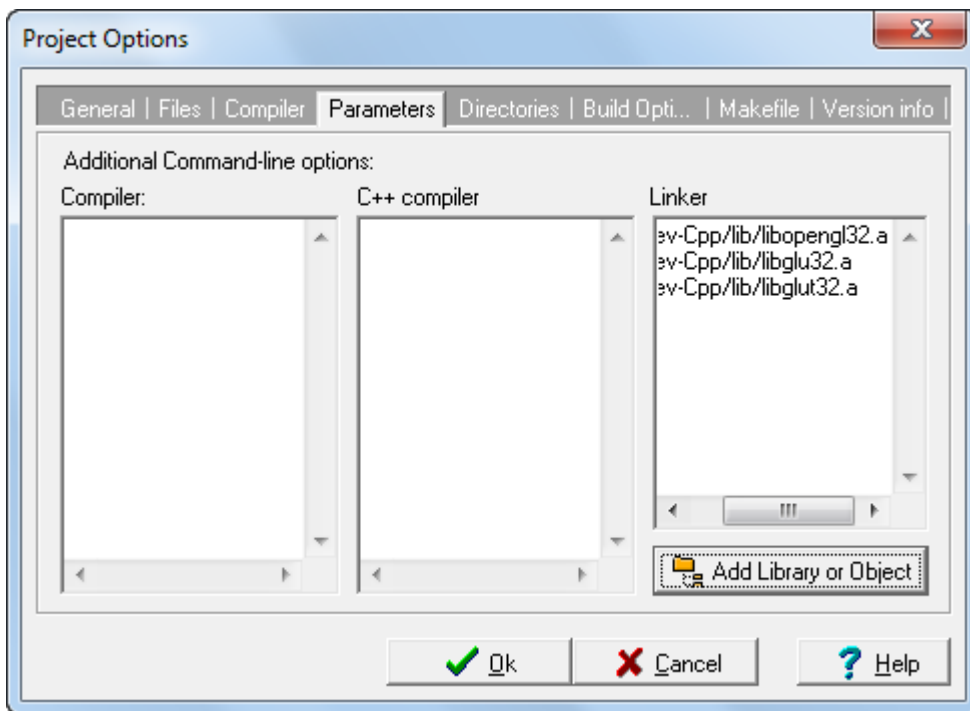
Για να μπορέσουμε να δημιουργήσουμε προγράμματα γραφικών με χρήση OpenGL - GLUT σε περιβάλλον γλώσσας προγραμματισμού C χρειαζόμαστε τα παρακάτω:

1. Περιβάλλον ανάπτυξης εφαρμογών σε γλώσσα προγραμματισμού C. Στο παρόν σύγγραμμα χρησιμοποιείται το εργαλείο Dev C++



2. Βιβλιοθήκες OpenGL - GLUT

- Αρχείο **glut.h**. Είναι C header file και πρέπει να βρίσκεται στο φάκελο include\GL του Dev C++
- Αρχείο **glut32.dll**. Είναι η εκτελέσιμη βιβλιοθήκη της GLUT και πρέπει να βρίσκεται στο φάκελο συστήματος του λειτουργικού συστήματος. Χωρίς αυτό το αρχείο δεν μπορούμε να κάνουμε εκτέλεση των προγραμμάτων
- Αρχεία **libopengl32.a**, **libglu32.a**, **libglut32.a**. Είναι archived βιβλιοθήκες που βρίσκονται στο φάκελο lib του Dev C++ και πρέπει να συνδεθούν στο C project, ΠΡΟΣΟΧΗ με την σειρά που αναφέρονται:
Επιλέγουμε Project→Project Options→Add library or object



3. Βασικές εντολές εκκίνησης περιβάλλοντος OpenGL

3.1 Εκκίνηση περιβάλλοντος

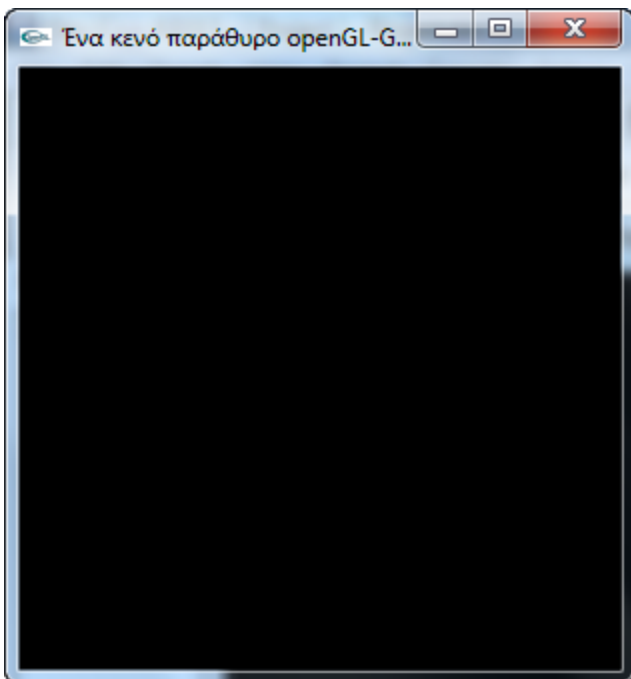
Ο παρακάτω κώδικας είναι ο κατ' ελάχιστα απαραίτητος για την δημιουργία ενός παραθύρου στο οποίο εν συνεχεία μπορούμε να σχεδιάσουμε γραφικά:

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

void display (void) {
    /* Called when OpenGL needs to update the display */
    glClear (GL_COLOR_BUFFER_BIT); /* Clear the window */
    glFlush(); /* Force update of screen */
}

int main (int argc, char **argv) {
    glutInit (&argc, argv); /* Initialise OpenGL */
    glutCreateWindow ("Ένα κενό παράθυρο OpenGL-GLUT"); /* Create the window */
    glutDisplayFunc (display); /* Register the "display" function */
    glutMainLoop (); /* Enter the OpenGL main loop */
    return 0;
}
```

Το αποτέλεσμα της εκτέλεσης του παραπάνω κώδικα είναι ένα κενό παράθυρο διαστάσεων 300X300 pixels στην πάνω αριστερή πλευρά της οθόνης:



Ας αναλύσουμε τον κώδικα:

- Αρχικά προσθέτουμε την βιβλιοθήκη του GLUT *glut.h*
- Η συνάρτηση *glutInit()* ενεργοποιεί τη βιβλιοθήκη GLUT
- Η εντολή *glutCreateWindow("Τίτλος")* εμφανίζει το παράθυρο της εφαρμογής στην οθόνη και του αποδίδει έναν τίτλο

- Η εντολή ***glutDisplayFunc(void func())*** εντάσσεται σε μια ειδική κατηγορία συναρτήσεων του GLUT, οι οποίες αποκαλούνται συναρτήσεις κλήσης (callback functions). Η συγκεκριμένη συνάρτηση δέχεται ως όρισμα μια συνάρτηση στην οποία εμπεριέχεται ο κώδικας σχεδίασης γραφικών. Η συνάρτηση αυτή εκτελείται κάθε φορά που η εφαρμογή διαπιστώσει ότι απαιτείται επανασχεδιασμός της σκηνής. Η συνάρτηση δεν επιστρέφει τιμή και δεν έχει ορίσματα. Στο συγκεκριμένο παράδειγμα δίνουμε ως όρισμα τη συνάρτηση `display()`
- Η εντολή ***glutMainLoop()*** ενεργοποιεί τον κύκλο διαχείρισης γεγονότων (event processing loop). Στον κύκλο αυτό, η εφαρμογή αναμένει επ' άπειρον και ανταποκρίνεται σε γεγονότα, όπως λ.χ. στο πάτημα ενός κουμπιού, στην αλλαγή του σκηνικού ή στην κίνηση του ποντικιού. Από το πρόθεμά της, παρατηρούμε ότι η συγκεκριμένη εντολή εμπεριέχεται στη βιβλιοθήκη GLUT, εφόσον το πρότυπο της OpenGL, ως πρότυπο ανεξάρτητο πλατφόρμας, δεν ορίζει διαδικασίες εισόδου-εξόδου
- Η εντολή ***glClearColor()*** καθαρίζει ενταμιευτές (buffers), συγκεκριμένες περιοχές μνήμης του συστήματος γραφικών (frame buffer). Η μηχανή γραφικών της OpenGL ορίζει ορισμένες κατηγορίες ενταμιευτών. Με την σταθερά `GL_COLOR_BUFFER_BIT` δίνουμε εντολή καθαρισμού του ενταμιευτή χρωματικών τιμών (colour buffer). Αυτή περιέχει τις χρωματικές τιμές των pixels που απεικονίζονται (ή πρόκειται να απεικονιστούν) στην οθόνη
- Η εντολή ***glFlush()*** εξαναγκάζει την εκτέλεση των εντολών που εκκρεμούν

3.2 Παραμετροποίηση εμφάνισης παράθυρου

Οι παρακάτω εντολές επηρεάζουν την εμφάνιση του OpenGL παράθυρου:

- Η εντολή ***glutInitWindowSize(Width,Height)*** καθορίζει το πλάτος και ύψος του παραθύρου της εφαρμογής σε pixels
- Η εντολή ***glutInitWindowPosition(X,Y)*** καθορίζει τη θέση στην οθόνη, στην οποία θα εμφανιστεί το παράθυρο της εφαρμογής (συντεταγμένη της άνω αριστερής κορυφής)
- Η εντολή ***glClearColor (Red,Green,Blue,Alpha)*** ορίζει το χρώμα του φόντου του παραθύρου

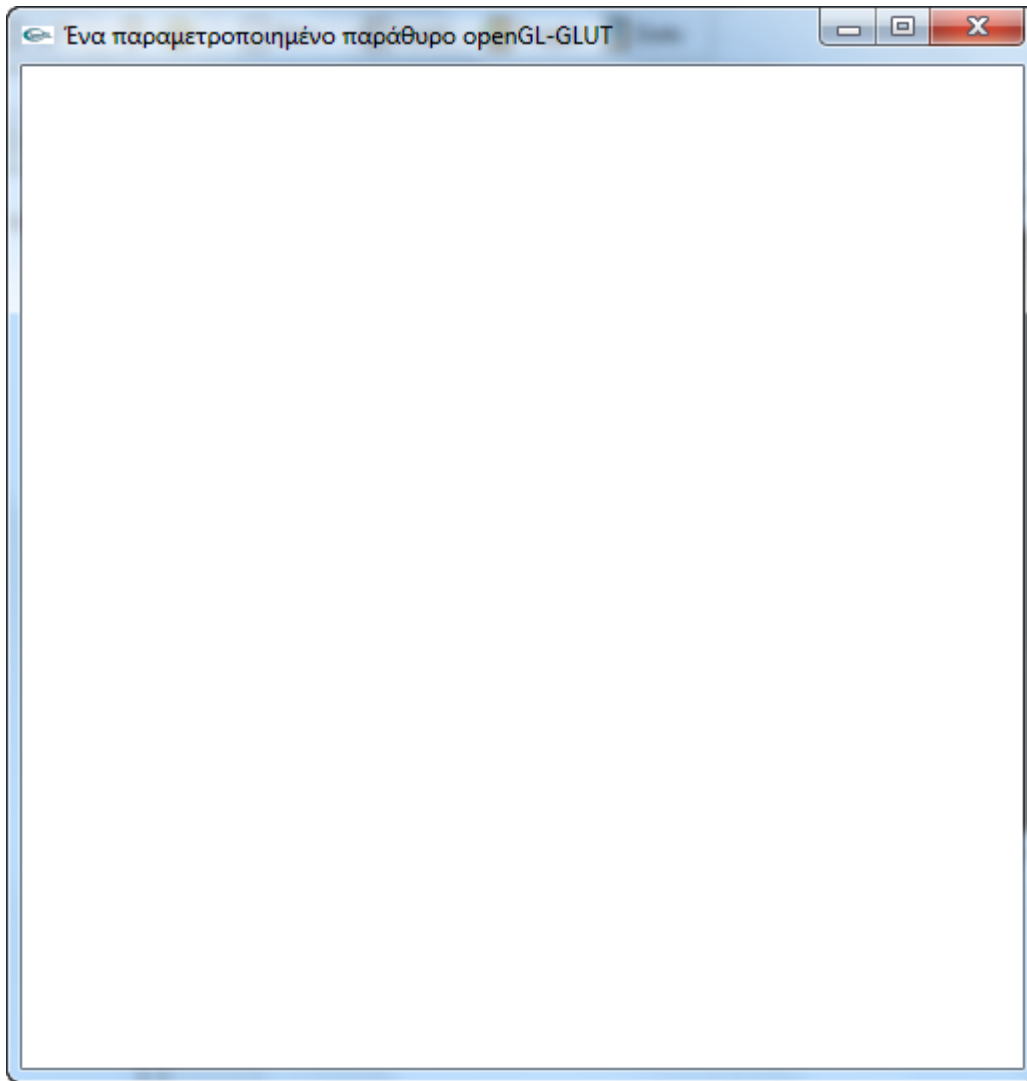
Παράδειγμα κώδικα:

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

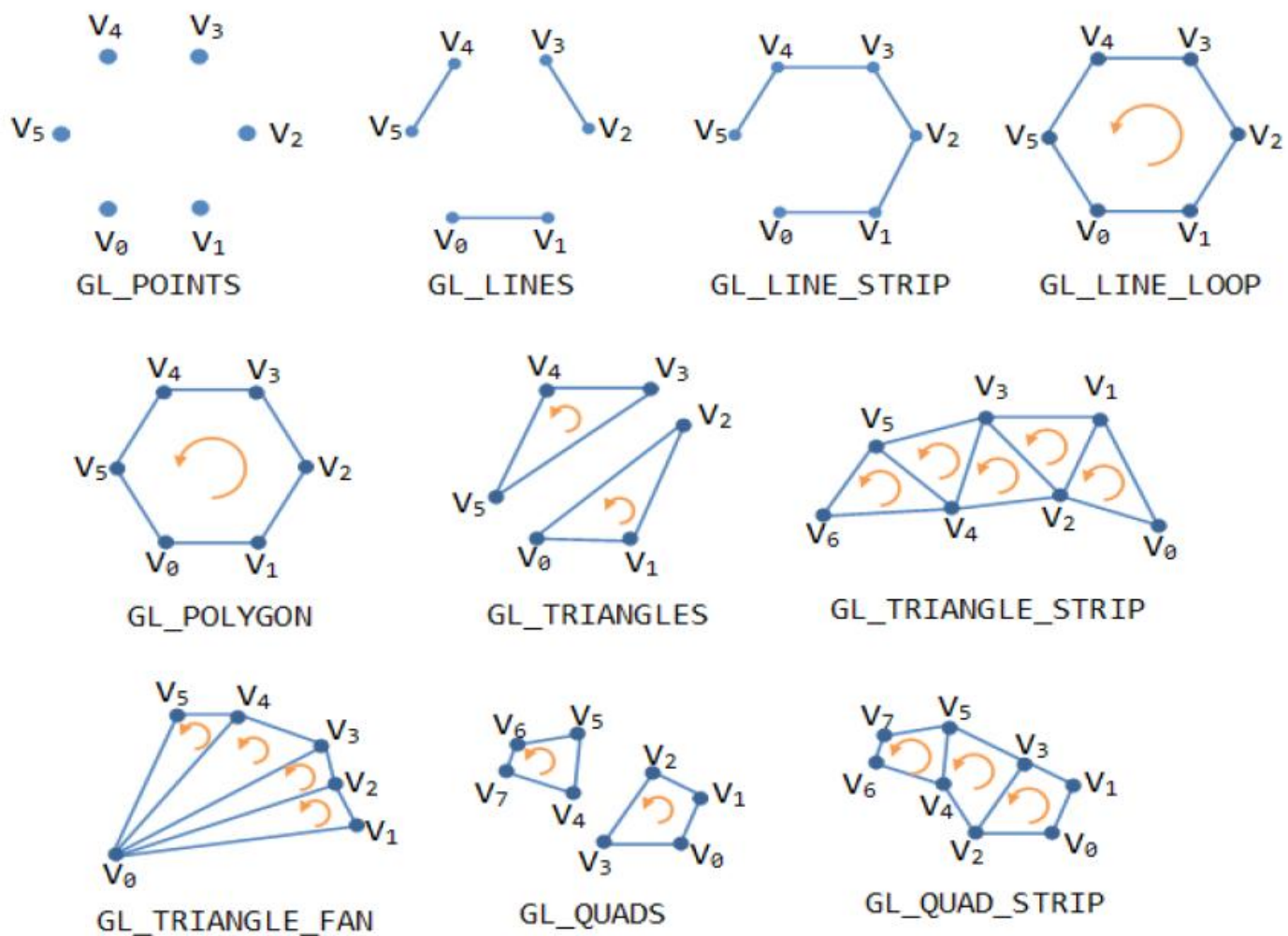
void display (void) {
    /* Called when OpenGL needs to update the display */
    glClearColor (1.0,1.0,1.0,0.0); /* Set the window color*/
    glClear (GL_COLOR_BUFFER_BIT); /* Clear the window */
    glFlush(); /* Force update of screen */
}

int main (int argc, char **argv) {
    glutInit (&argc, argv); /* Initialise OpenGL */
    glutInitWindowSize (500, 500); /* Set the window size */
    glutInitWindowPosition (100, 100); /* Set the window position */
    glutCreateWindow ("Ένα παραμετροποιημένο παράθυρο OpenGL-GLUT"); /* Create the window */
    glutDisplayFunc (display); /* Register the "display" function */
    glutMainLoop (); /* Enter the OpenGL main loop */
    return 0;
}
```

Αποτέλεσμα εκτέλεσης κώδικα:



4. Θεμελιώδη σχήματα της OpenGL

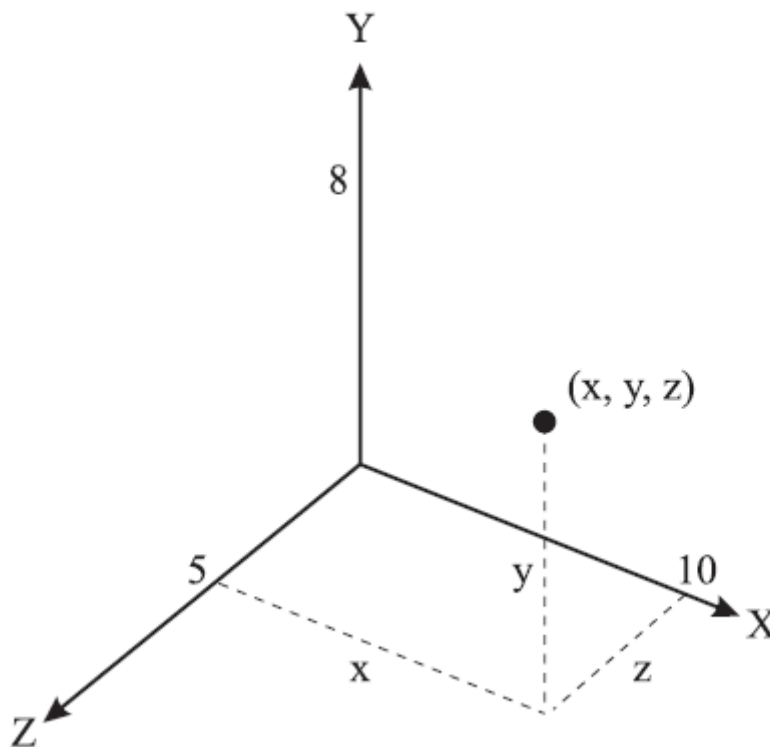


5. Εντολές δημιουργίας βασικών δισδιάστατων σχημάτων της OpenGL

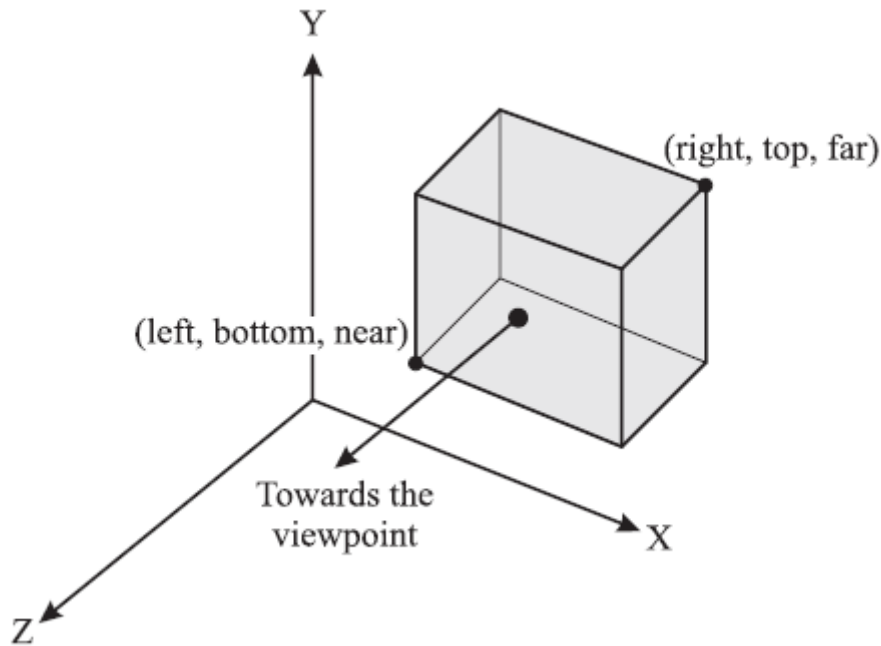
- Η εντολή **glBegin** (*GLenum MODE*) δηλώνει την έναρξη ορισμού ενός ή περισσότερων γεωμετρικών σχημάτων. Αναλόγως του ορίσματος, μπορεί να προσδιοριστεί μια ποικιλία σχημάτων. Δέχεται 10 ορίσματα:

GL_POINTS, GL_LINES, GL_LINE_STRIP, GL_LINE_LOOP, GL_TRIANGLES, GL_TRIANGLE_STRIP, GL_TRIANGLE_FAN, GL_QUADS, GL_QUAD_STRIP, GL_POLYGON

- Η εντολή **glEnd** () εκτελείται πάντα σε συνδυασμό με την εντολή **glBegin**() και ορίζει τη λήξη της επιλεγόμενης ρύθμισης σχεδίασης
- Η εντολή **glVertex** () χρησιμοποιείται εντός των **glBegin**() - **glEnd**() για να ορίσει τις κορυφές των σχημάτων που θέλουμε να εμφανίσουμε. Ανάλογα με το πλήθος των διαστάσεων που θέλουμε να εμφανίσουμε διακρίνεται στις **glVertex2**(), **glVertex3**(), **glVertex4**() που αντιστοιχούν σε δισδιάστατα σχήματα, τρισδιάστατα και 4 διαστάσεων (για προοπτικές όψεις). Κάθε μια από αυτές τις εντολές διακρίνεται σε 4 άλλες ανάλογα με τον τύπο του αριθμού που δέχεται ως όρισμα, έτσι π.χ. η **glVertex2**() διακρίνεται στις **glVertex2s** (*GLshort x, GLshort y*), **glVertex2i** (*GLint x, GLint y*), **glVertex2f** (*GLfloat x, GLfloat y*), **glVertex2d** (*GLdouble x, GLdouble y*)



- Η εντολή **glPointSize** (*GLfloat size*) ορίζει το πάχος των κουκίδων
- Η εντολή **glLineWidth** (*GLfloat width*) ορίζει το πάχος των γραμμών
- Η εντολή **glColor** () ορίζει το χρώμα των σχημάτων
- Η εντολή **glOrtho** (*GLdouble left, GLdouble right, GLdouble bottom, GLdouble top, GLdouble nearVal, GLdouble farVal*) είναι απαραίτητη για την σωστή προβολή των γραφικών σε 2 διαστάσεις γιατί μετασχηματίζει το μητρώο των γραφικών σε μια παράλληλη ορθογραφική προβολή



5.1 Κουκίδες (points)

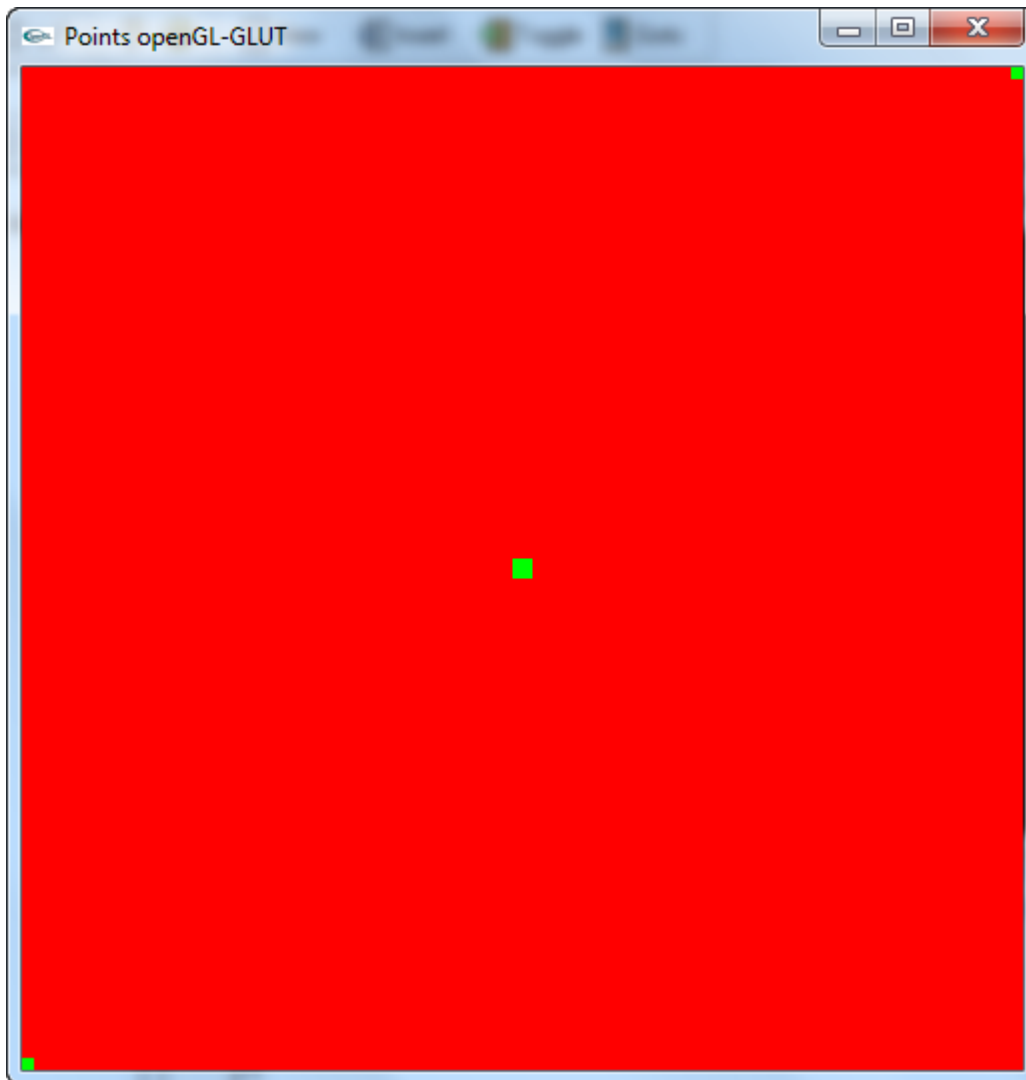
Παράδειγμα κώδικα:

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

void display (void) {
    /* Called when OpenGL needs to update the display */
    glClearColor (1.0,0.0,0.0,0.0); /* Set the window color*/
    glClear (GL_COLOR_BUFFER_BIT); /* Clear the window */
    glPointSize (10); /* Set the points size */
    glColor3f(0.0,1.0,0.0); /* Set the drawing color */
    glBegin (GL_POINTS); /* Draws points */
    glVertex2f (1.0, 1.0);
    glVertex2f (250.0, 250.0);
    glVertex2f (499.0, 499.0);
    glEnd ();
    glFlush(); /* Force update of screen */
}

int main (int argc, char **argv) {
    glutInit (&argc, argv); /* Initialise OpenGL */
    glutInitWindowSize (500, 500); /* Set the window size */
    glutInitWindowPosition (100, 100); /* Set the window position */
    glutCreateWindow ("Points openGL-GLUT"); /* Create the window */
    glutDisplayFunc (display); /* Register the "display" function */
    glOrtho (0.0, 500, 0.0, 500, -1.0, 1.0); /* creates a matrix for an orthographic projection */
    glutMainLoop (); /* Enter the OpenGL main loop */
    return 0;
}
```

Αποτέλεσμα εκτέλεσης κώδικα:



5.2 Γραμμές (lines)

Παράδειγμα κώδικα:

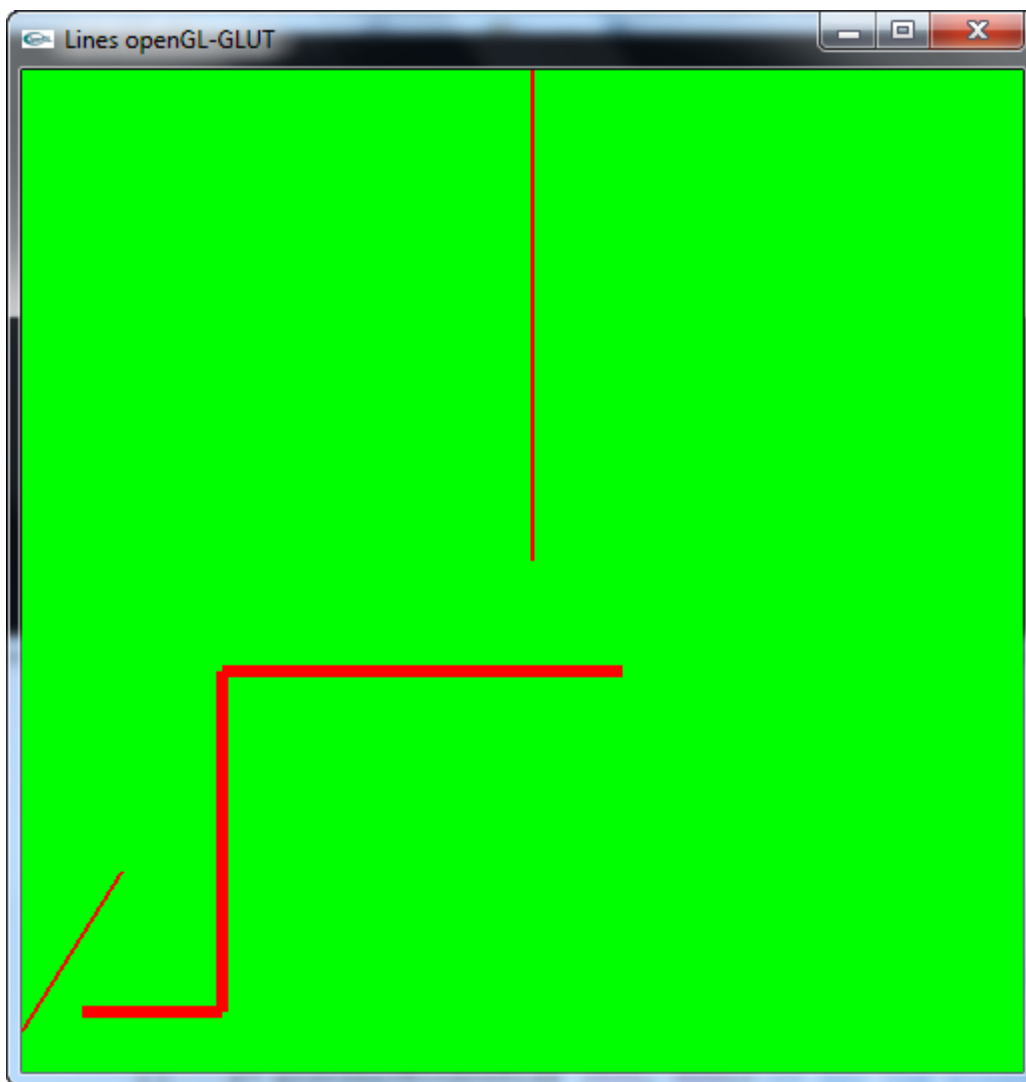
```
#include <GL\glut.h>

void display (void) {
    /* Called when OpenGL needs to update the display */
    glClearColor (0.0,1.0,0.0,0.0); /* Set the window color*/
    glClear (GL_COLOR_BUFFER_BIT); /* Clear the window */
    glLineWidth (2); /* Set the lines size */
    glColor3f(1.0,0.0,0.0); /* Set the drawing color */
    glBegin (GL_LINES); /* or GL_LINE_STRIP or GL_LINE_LOOP */
    glVertex2f (0.0, 20.0);
    glVertex2f (50.0, 100.0);
    glVertex2f (255.0, 255.0);
    glVertex2f (255.0, 500.0);
    glEnd ();
    glLineWidth (6); /* Set the lines size */
}
```

```
glBegin (GL_LINE_STRIP); /* or GL_LINE_STRIP or GL_LINE_LOOP */
glVertex2f (30.0, 30.0);
glVertex2f (100.0, 30.0);
glVertex2f (100.0, 200.0);
glVertex2f (300.0, 200.0);
glEnd ();
glFlush(); /* Force update of screen */
}

int main (int argc, char **argv) {
glutInit (&argc, argv); /* Initialise OpenGL */
glutInitWindowSize (500, 500); /* Set the window size */
glutInitWindowPosition (100, 100); /* Set the window position */
glutCreateWindow ("Lines openGL-GLUT"); /* Create the window */
glutDisplayFunc (display); /* Register the "display" function */
glOrtho (0.0, 500, 0.0, 500, -1.0, 1.0); /* creates a matrix for an orthographic projection */
glutMainLoop (); /* Enter the OpenGL main loop */
return 0;
}
```

Αποτέλεσμα εκτέλεσης κώδικα:



5.3 Τρίγωνα (triangles)

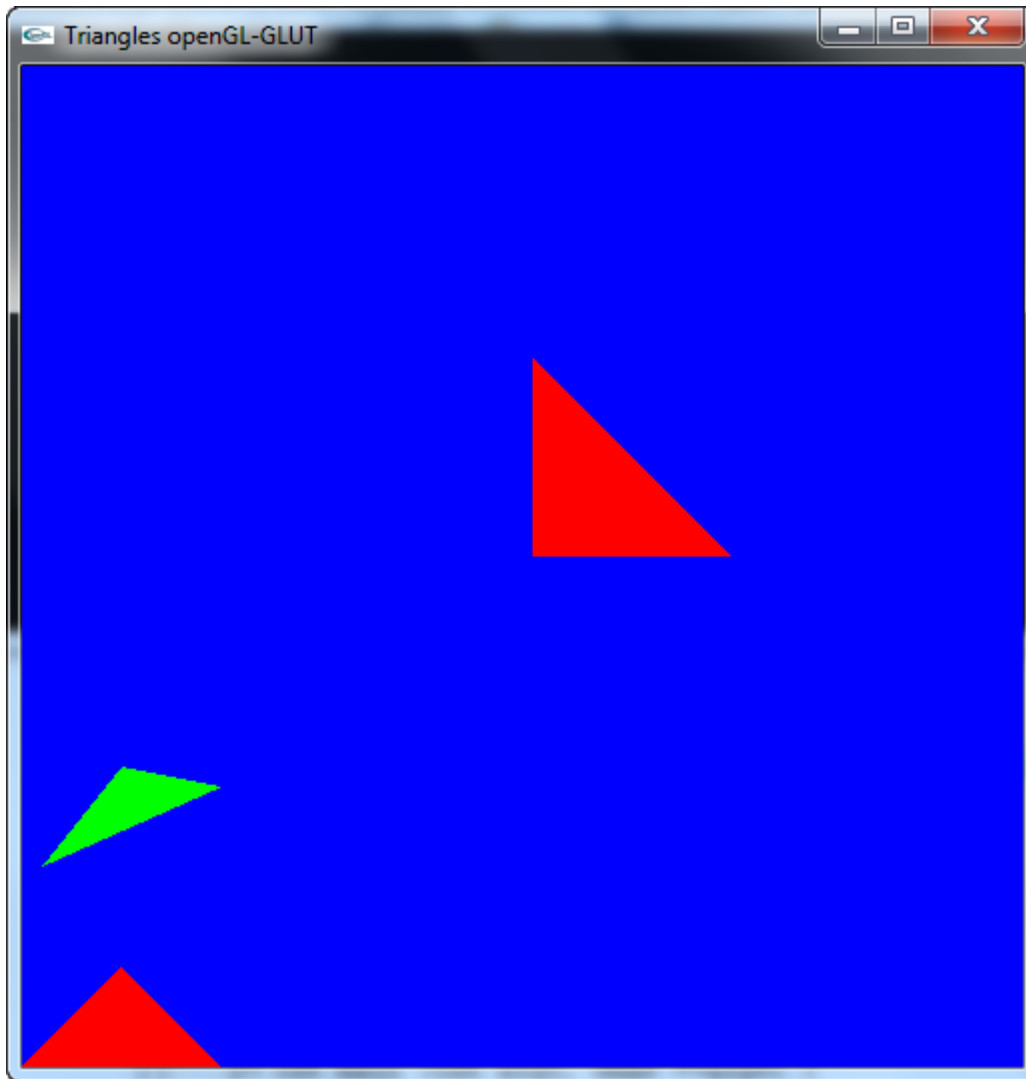
Παράδειγμα κώδικα:

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

void display (void) {
    /* Called when OpenGL needs to update the display */
    glClearColor (0.0,0.0,1.0,0.0); /* Set the window color*/
    glClear (GL_COLOR_BUFFER_BIT); /* Clear the window */
    glLineWidth (2); /* Set the lines size */
    glColor3f(1.0,0.0,0.0); /* Set the drawing color */
    glBegin (GL_TRIANGLES); /* Draws triangles */
    glVertex2f (0.0, 0.0);
    glVertex2f (50.0, 50.0);
    glVertex2f (100.0, 00.0);
    glVertex2f (255.0, 255.0);
    glVertex2f (255.0, 355.0);
    glVertex2f (355.0, 255.0);
    glEnd ();
    glLineWidth (6); /* Set the lines size */
    glColor3f(0.0,25.0,0.0); /* Set the drawing color */
    glBegin (GL_TRIANGLE_STRIP); /* Draws triangles */
    glVertex2f (10.0, 100.0);
    glVertex2f (50.0, 150.0);
    glVertex2f (100.0, 140.0);
    glEnd ();
    glFlush(); /* Force update of screen */
}

int main (int argc, char **argv) {
    glutInit (&argc, argv); /* Initialise OpenGL */
    glutInitWindowSize (500, 500); /* Set the window size */
    glutInitWindowPosition (100, 100); /* Set the window position */
    glutCreateWindow ("Triangles openGL-GLUT"); /* Create the window */
    glutDisplayFunc (display); /* Register the "display" function */
    glOrtho (0.0, 500, 0.0, 500, -1.0, 1.0); /* creates a matrix for an orthographic projection */
    glutMainLoop (); /* Enter the OpenGL main loop */
    return 0;
}
```

Αποτέλεσμα εκτέλεσης κώδικα:



5.4 Τετράπλευρα (quadrilaterals)

Παράδειγμα κώδικα:

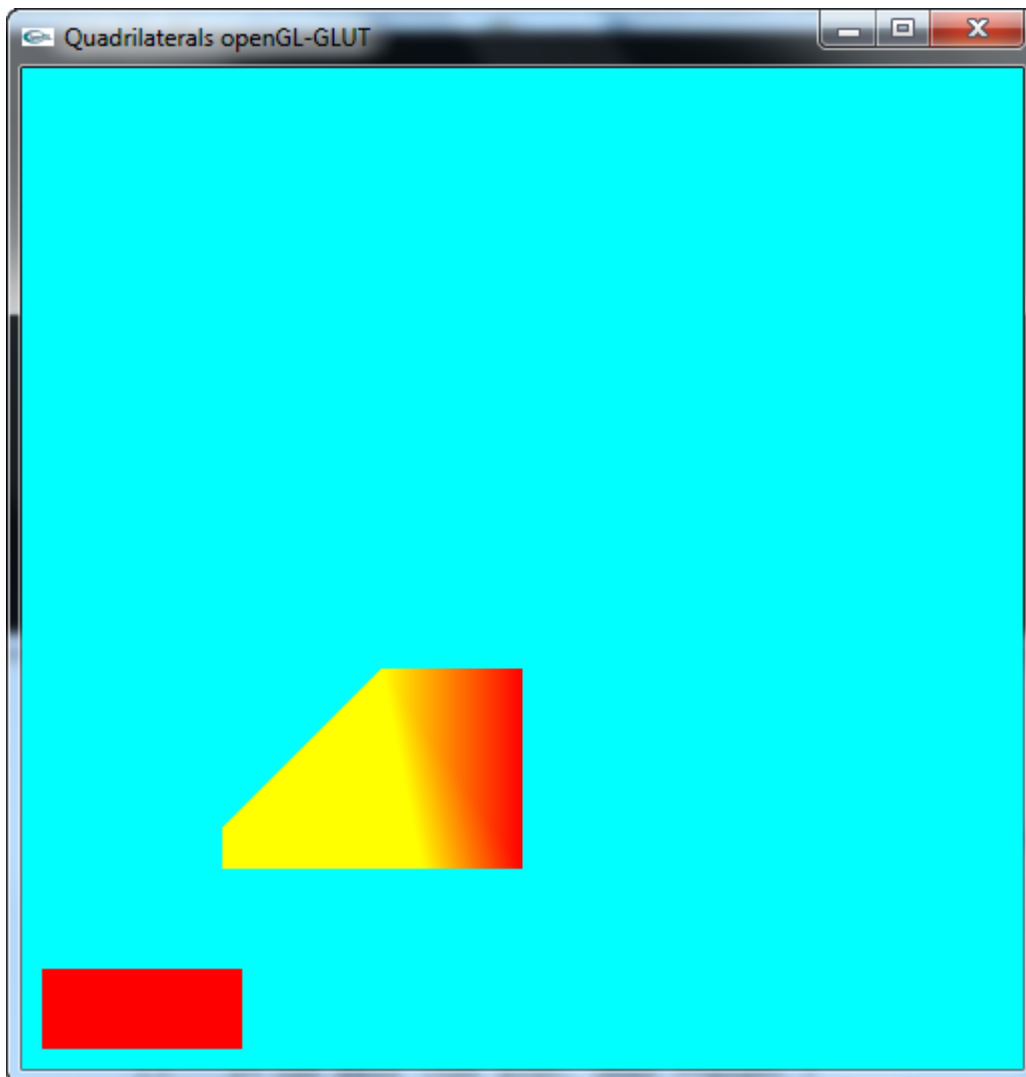
```
#include <GL\glut.h>

void display (void) {
    /* Called when OpenGL needs to update the display */
    glClearColor (0.0,204.0,204.0,0.0); /* Set the window color*/
    glClear (GL_COLOR_BUFFER_BIT); /* Clear the window */
    glColor3f(1.0,0.0,0.0); /* Set the drawing color */
    glBegin (GL_QUADS); /* or GL_QUAD_STRIP */
    glVertex2f (10.0, 10.0);
    glVertex2f (10.0, 50.0);
    glVertex2f (110.0, 50.0);
    glVertex2f (110.0, 10.0);
    glEnd ();
    glColor3f(1.0,5.0,0.0); /* Set the drawing color */
    glBegin(GL_QUAD_STRIP); /* Draws Quadrilaterals */
    glVertex2f (100.0, 100.0);
```

```
glVertex2f (100.0, 120.0);
glVertex2f (200.0, 100.0);
glVertex2f (180.0, 200.0);
glColor3f(1.0,0.0,0.0); /* Set the drawing color */
glVertex2f (250.0, 100.0);
glVertex2f (250.0, 200.0);
glEnd();
glFlush(); /* Force update of screen */
}

int main (int argc, char **argv) {
glutInit (&argc, argv); /* Initialise OpenGL */
glutInitWindowSize (500, 500); /* Set the window size */
glutInitWindowPosition (100, 100); /* Set the window position */
glutCreateWindow ("Quadrilaterals openGL-GLUT"); /* Create the window */
glutDisplayFunc (display); /* Register the "display" function */
glOrtho (0.0, 500, 0.0, 500, -1.0, 1.0); /* creates a matrix for an orthographic projection */
glutMainLoop (); /* Enter the OpenGL main loop */
return 0;
}
```

Αποτέλεσμα εκτέλεσης κώδικα:



5.5 Πολύγωνα (polygons)

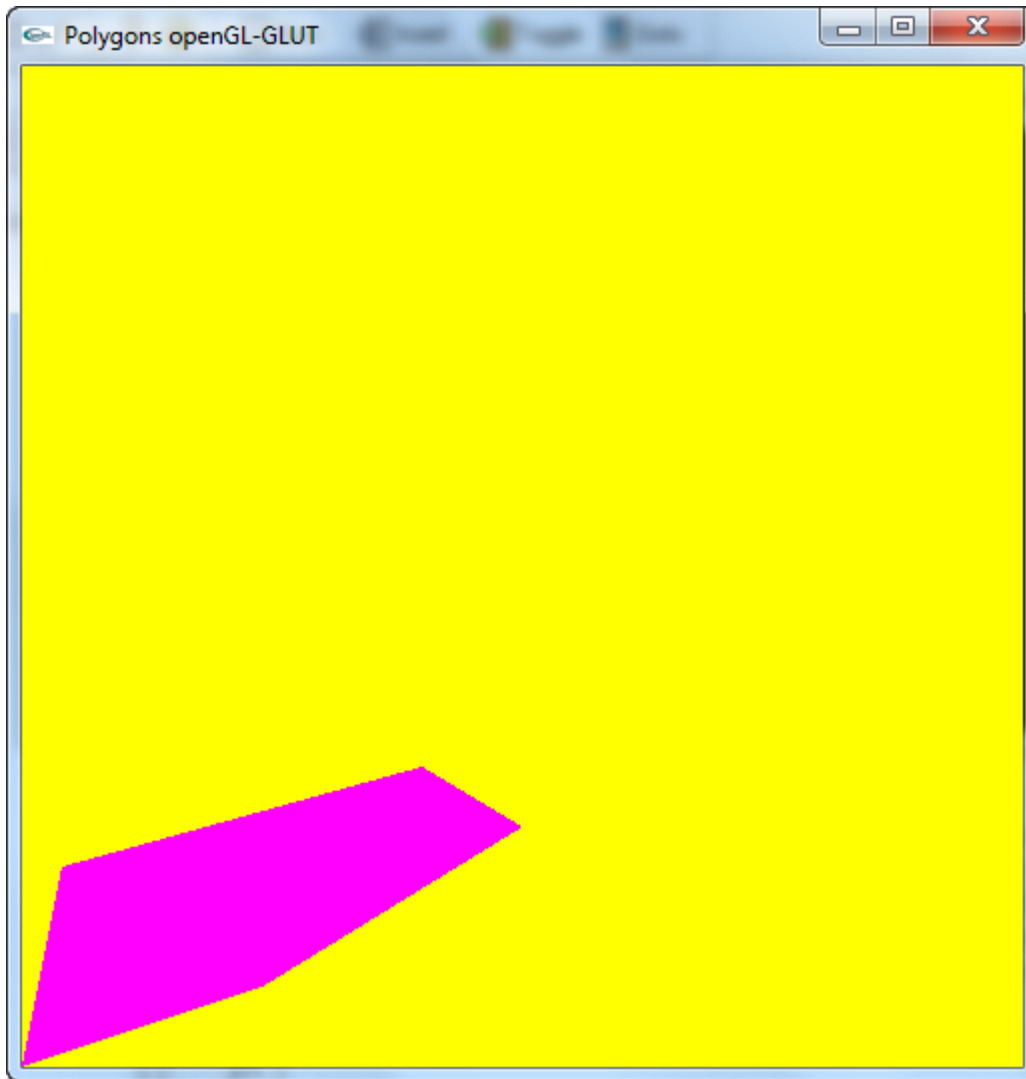
Παράδειγμα κώδικα:

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

void display (void) {
    /* Called when OpenGL needs to update the display */
    glClearColor (1.0,12.0,0.0,0.0); /* Set the window color*/
    glClear (GL_COLOR_BUFFER_BIT); /* Clear the window */
    glColor3f(1,0,1); /* Set the drawing color */
    glBegin (GL_POLYGON); /* Draws Polygons */
    glVertex2f (0.0, 0.0);
    glVertex2f (20.0, 100.0);
    glVertex2f (200.0, 150.0);
    glVertex2f (250.0, 120.0);
    glVertex2f (120.0, 40.0);
    glEnd ();
    glFlush(); /* Force update of screen */
}

int main (int argc, char **argv) {
    glutInit (&argc, argv); /* Initialise OpenGL */
    glutInitWindowSize (500, 500); /* Set the window size */
    glutInitWindowPosition (100, 100); /* Set the window position */
    glutCreateWindow ("Polygons openGL-GLUT"); /* Create the window */
    glutDisplayFunc (display); /* Register the "display" function */
    glOrtho (0.0, 500, 0.0, 500, -1.0, 1.0); /* creates a matrix for an orthographic projection */
    glutMainLoop (); /* Enter the OpenGL main loop */
    return 0;
}
```


Αποτέλεσμα εκτέλεσης κώδικα:



5.6 Κείμενο (text)

- Η εντολή *glRasterPos* ()ορίζει τις συντεταγμένες στις οποίες θα γίνει εν συνεχεία εμφάνιση pixels γραφικών
- Η εντολή *glutBitmapCharacter*(void *font, int character) εμφανίζει ένα bitmap χαρακτήρα

Παράδειγμα κώδικα:

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

void drawString (void *font, float x, float y, char *str) {
/* Draws string 'str' in font 'font', at world (x,y,0) */
char *ch;
glRasterPos3f(x, y, 0.0); /* Sets the drawing position */
for (ch= str; *ch; ch++)
glutBitmapCharacter(font, (int)*ch); /* Renders a bitmap character */
}
```

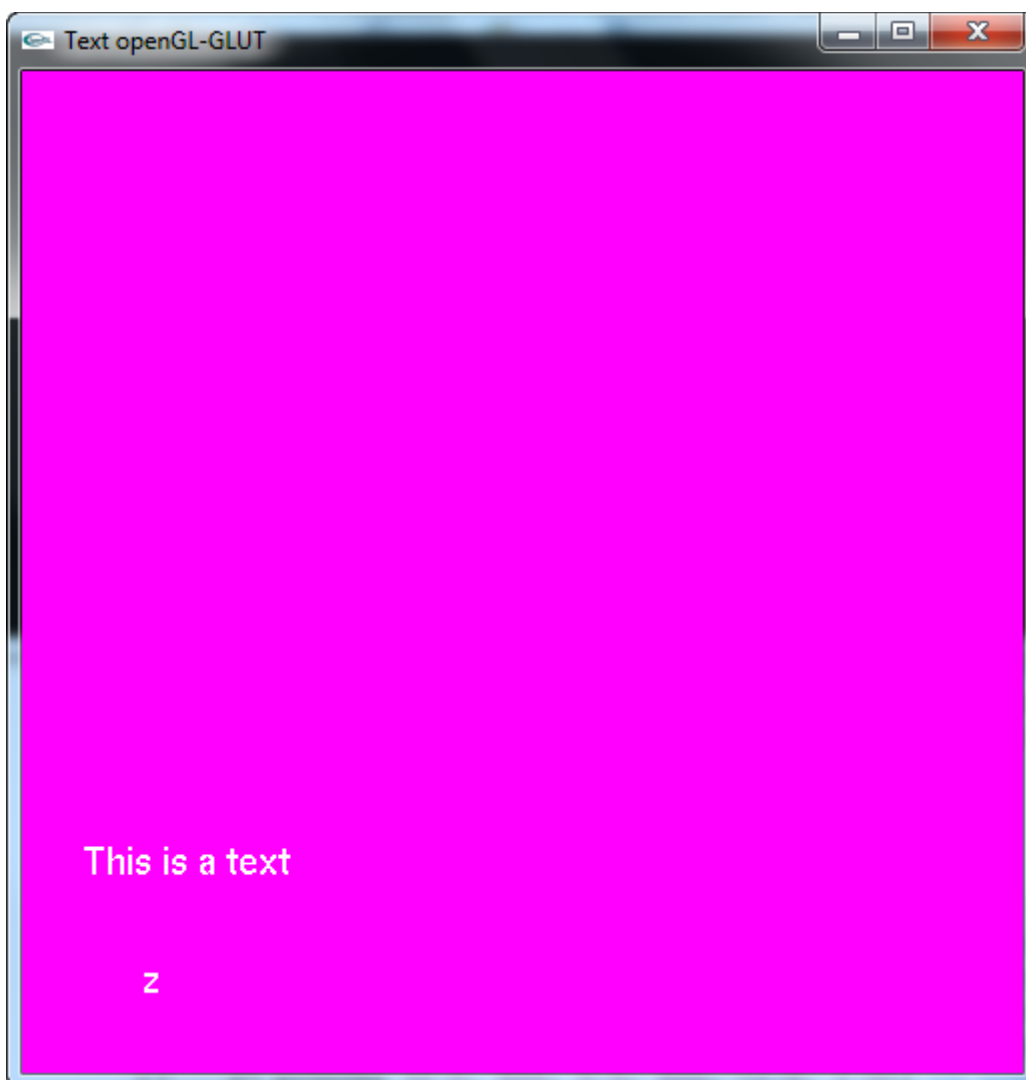
```

void display (void) {
/* Called when OpenGL needs to update the display */
glClearColor (1.0,0.0,1.0,0.0); /* Set the window color*/
glClear (GL_COLOR_BUFFER_BIT); /* Clear the window */
glRasterPos3f(60.0, 40.0, 0.0); /* Sets the drawing position */
glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18,122); /* Renders a bitmap character */
drawString (GLUT_BITMAP_HELVETICA_18, 30.0, 100.0, "This is a text");
glFlush(); /* Force update of screen */
}

int main (int argc, char **argv) {
glutInit (&argc, argv); /* Initialise OpenGL */
glutInitWindowSize (500, 500); /* Set the window size */
glutInitWindowPosition (100, 100); /* Set the window position */
glutCreateWindow ("Text openGL-GLUT"); /* Create the window */
glutDisplayFunc (display); /* Register the "display" function */
glOrtho (0.0, 500, 0.0, 500, -1.0, 1.0); /* creates a matrix for an orthographic projection */
glutMainLoop (); /* Enter the OpenGL main loop */
return 0;
}

```

Αποτέλεσμα εκτέλεσης κώδικα:



6. Τύποι μεταβλητών

C Type	Bitdepth	Description
GLboolean	1+	A boolean value, either <code>GL_TRUE</code> or <code>GL_FALSE</code>
GLbyte	8	Signed, 2's complement binary integer
GLubyte	8	Unsigned binary integer
GLshort	16	Signed, 2's complement binary integer
GLushort	16	Unsigned binary integer
GLint	32	Signed, 2's complement binary integer
GLuint	32	Unsigned binary integer
GLfixed	32	Signed, 2's complement 16.16 integer
GLint64	64	Signed, 2's complement binary integer
GLuint64	64	Unsigned binary integer
GLsizei	32	A non-negative binary integer, for sizes.
GLenum	32	An OpenGL enumerator value
GLintptr	ptrbits1	Signed, 2's complement binary integer
GLsizeiptr	ptrbits1	Non-negative binary integer size, for pointer offsets and ranges
GLsync	ptrbits1	Sync Object handle
GLbitfield	32	A bitfield value
GLhalf	16	An IEEE-754 floating-point value
GLfloat	32	An IEEE-754 floating-point value
GLclampf	32	An IEEE-754 floating-point value, clamped to the range [0,1]
GLdouble	64	An IEEE-754 floating-point value
GLclampd	64	An IEEE-754 floating-point value, clamped to the range [0,1]

7. Αλληλεπίδραση με το πληκτρολόγιο

- Η εντολή *glutKeyboardFunc* (*function*) καλεί την συνάρτηση που περιέχει τον κώδικα αλληλεπίδρασης με το πληκτρολόγιο
- Η συνάρτηση *void function (unsigned char key, int x, int y)* διαβάζει το πλήκτρο που πατάει ο χρήστης και το αποθηκεύει στη μεταβλητή *key* με τιμές από 0 έως 255 του κώδικα ASCII

Παράδειγμα κώδικα:

```
#include <GL\glut.h>
#include <stdio.h> /* library for input/output */
#include <stdlib.h> /* include rand function */

void drawRandomPoint () {
    /* Draws point randomly after key pressed */
    int X, Y;
    int R, G, B;
    X = rand() % 500 + 1; /* random number for X position */
    Y = rand() % 500 + 1; /* random number for Y position */
    R = rand() % 255; /* random color for Red */
    G = rand() % 255; /* random color for Green */
    B = rand() % 255; /* random color for Blue */
    printf ("X = %i", X);
    printf (" Y = %i\n", Y);
    glPointSize (20); /* Set the points size */
    glColor3ub(R,G,B); /* Set the drawing color */
    glBegin (GL_POINTS); /* Draws points */
    glVertex2f (X, Y);
    glEnd ();
    glFlush(); /* Force update of screen */
}

void keyboard (unsigned char key, int x, int y) {
    /* Called when a key is pressed */
    if (key == 27) exit (0); /* 27 is the Escape key */
    else
    {
        printf ("You pressed %c\n", key);
        drawRandomPoint();
    }
}

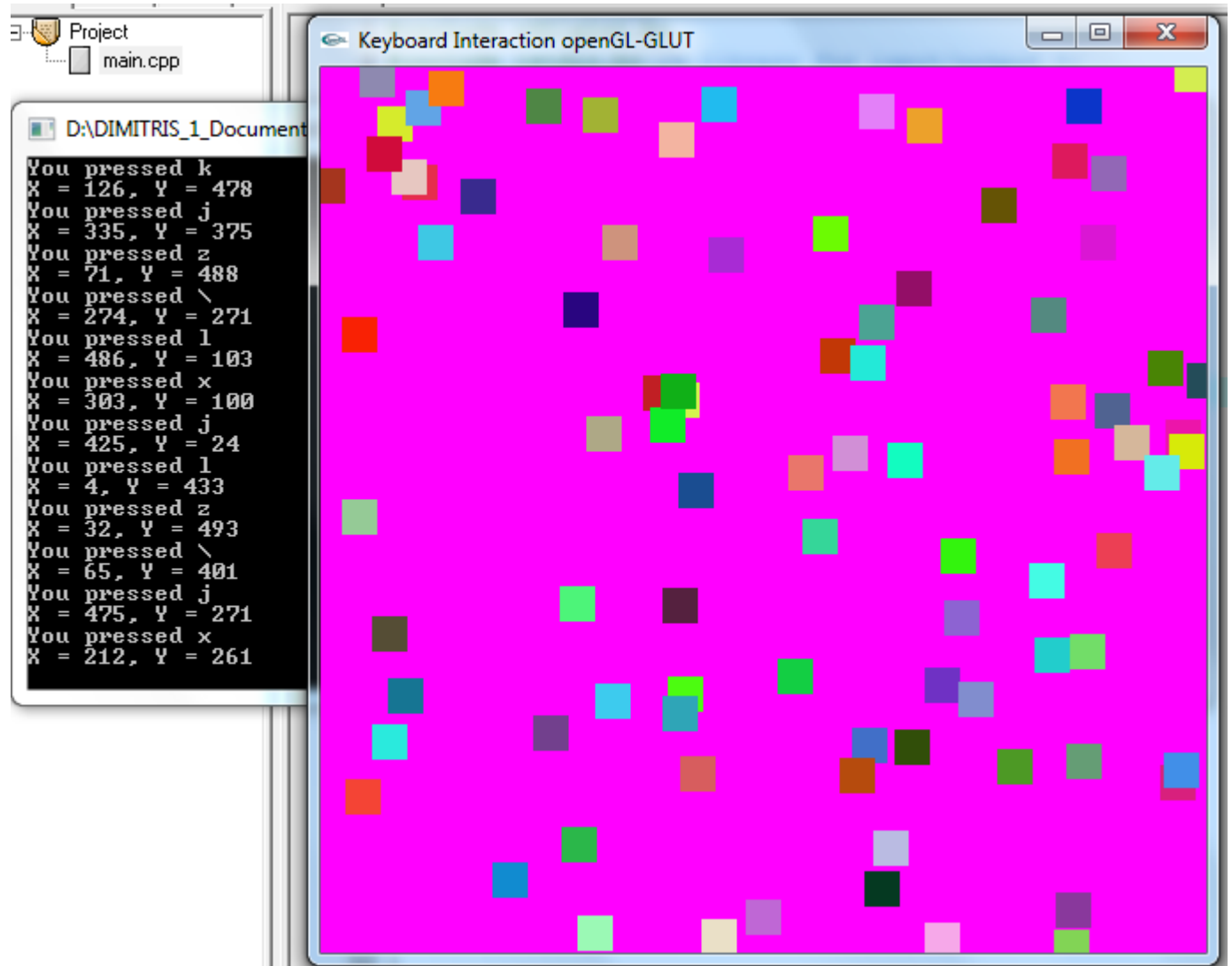
void display (void) {
    /* Called when OpenGL needs to update the display */
    glClearColor (1.0,0.0,1.0,0.0); /* Set the window color*/
    glClear (GL_COLOR_BUFFER_BIT); /* Clear the window */
    glFlush(); /* Force update of screen */
}

int main (int argc, char **argv) {
    glutInit (&argc, argv); /* Initialise OpenGL */
    glutInitWindowSize (500, 500); /* Set the window size */
    glutInitWindowPosition (100, 100); /* Set the window position */
    glutCreateWindow ("Keyboard Interaction openGL-GLUT"); /* Create the window */
    glutDisplayFunc (display); /* Register the "display" function */
    glutKeyboardFunc (keyboard); /* Register the "keyboard" function */
    glOrtho (0.0, 500, 0.0, 500, -1.0, 1.0); /* creates a matrix for an orthographic projection */
}
```

```
glutMainLoop (); /* Enter the OpenGL main loop */  
return 0;  
}
```

Αποτέλεσμα εκτέλεσης κώδικα:

Εμφανίζει με το πάτημα οποιουδήποτε πλήκτρου μια κουκίδα σε τυχαίο σημείο με τυχαίο χρώμα, εάν πατηθεί το πλήκτρο escape τερματίζει το πρόγραμμα. Παράλληλα στη κονσόλα εξόδου εμφανίζονται ποιο πλήκτρο πατήθηκε και οι τυχαίες συντεταγμένες στις οποίες θα εμφανιστεί η κουκίδα.



8. Πίνακας χαρακτήρων ASCII

Βασικός πίνακας:

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040	 	Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	\$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	((72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051))	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	:	91	5B	133	[[123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174	|	
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135]]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177		DEL

Επεκταμένος πίνακας για ελληνικούς χαρακτήρες:

Dec	Hex	Char	Description	Dec	Hex	Char	Description
128	80	Α	Greek upper case alpha	192	C0	⊞	Box drawings light up and right
129	81	Β	Greek upper case beta	193	C1	⊞	Box drawings light up and horizontal
130	82	Γ	Greek upper case gamma	194	C2	⊞	Box drawings light down and horizontal
131	83	Δ	Greek upper case delta	195	C3	⊞	Box drawings light vertical and right
132	84	Ε	Greek upper case epsilon	196	C4	—	Box drawings light horizontal
133	85	Ζ	Greek upper case zeta	197	C5	⊞	Box drawings light vertical and horizontal
134	86	Η	Greek upper case eta	198	C6	⊞	Box drawings vertical single and right double
135	87	Θ	Greek upper case theta	199	C7	⊞	Box drawings vertical double and right single
136	88	Ι	Greek upper case iota	200	C8	⊞	Box drawings double up and right
137	89	Κ	Greek upper case kappa	201	C9	⊞	Box drawings double down and right
138	8A	Λ	Greek upper case lamda	202	CA	⊞	Box drawings double up and horizontal
139	8B	Μ	Greek upper case mu	203	CB	⊞	Box drawings double down and horizontal
140	8C	Ν	Greek upper case nu	204	CC	⊞	Box drawings double vertical and right
141	8D	Ξ	Greek upper case xi	205	CD	=	Box drawings double horizontal
142	8E	Ο	Greek upper case omicron	206	CE	⊞	Box drawings double vertical and horizontal
143	8F	Π	Greek upper case pi	207	CF	⊞	Box drawings up single and horizontal double
144	90	Ρ	Greek upper case rho	208	D0	⊞	Box drawings up double and horizontal single
145	91	Σ	Greek upper case sigma	209	D1	⊞	Box drawings down single and horizontal double
146	92	Τ	Greek upper case tau	210	D2	⊞	Box drawings down double and horizontal single
147	93	Υ	Greek upper case upsilon	211	D3	⊞	Box drawings up double and right single
148	94	Φ	Greek upper case phi	212	D4	⊞	Box drawings up single and right double
149	95	Χ	Greek upper case chi	213	D5	⊞	Box drawings down single and right double
150	96	Ψ	Greek upper case psi	214	D6	⊞	Box drawings down double and right single
151	97	Ω	Greek upper case omega	215	D7	⊞	Box drawings vertical double and horizontal single
152	98	α	Greek lower case alpha	216	D8	⊞	Box drawings vertical single and horizontal double
153	99	β	Greek lower case beta	217	D9	⊞	Box drawings light up and left
154	9A	γ	Greek lower case gamma	218	DA	⊞	Box drawings light down and right
155	9B	δ	Greek lower case delta	219	DB	■	Full block
156	9C	ε	Greek lower case epsilon	220	DC	■	Lower half block
157	9D	ζ	Greek lower case zeta	221	DD	■	Left half block
158	9E	η	Greek lower case eta	222	DE	■	Right half block
159	9F	θ	Greek lower case theta	223	DF	■	Upper half block
160	A0	ι	Greek lower case iota	224	E0	ω	Greek lower case omega
161	A1	κ	Greek lower case kappa	225	E1	ᾱ	Greek lower case alpha with tonos
162	A2	λ	Greek lower case lamda	226	E2	ἕ	Greek lower case epsilon with tonos
163	A3	μ	Greek lower case mu	227	E3	ἥ	Greek lower case eta with tonos
164	A4	ν	Greek lower case nu	228	E4	ϊ	Greek lower case iota with dialytika
165	A5	ξ	Greek lower case xi	229	E5	ἰ	Greek lower case iota with tonos
166	A6	ο	Greek lower case omicron	230	E6	ὀ	Greek lower case omicron with tonos
167	A7	π	Greek lower case pi	231	E7	ῦ	Greek lower case upsilon with tonos
168	A8	ρ	Greek lower case rho	232	E8	ῦ	Greek lower case upsilon with dialytika
169	A9	σ	Greek lower case sigma	233	E9	ὦ	Greek lower case omega with tonos
170	AA	ς	Greek lower case final sigma	234	EA	Ᾱ	Greek upper case alpha with tonos
171	AB	τ	Greek lower case tau	235	EB	Ἔ	Greek upper case epsilon with tonos
172	AC	υ	Greek lower case upsilon	236	EC	Ἡ	Greek upper case eta with tonos
173	AD	φ	Greek lower case phi	237	ED	Ἰ	Greek upper case iota with tonos
174	AE	χ	Greek lower case chi	238	EE	Ὢ	Greek upper case omicron with tonos
175	AF	ψ	Greek lower case psi	239	EF	Ὶ	Greek upper case upsilon with tonos
176	B0	░	Light shade	240	F0	Ω	Greek upper case omega with tonos
177	B1	▒	Medium shade	241	F1	±	Plus-minus sign
178	B2	▓	Dark shade	242	F2	≥	Greater-than or equal to
179	B3		Box drawings light vertical	243	F3	≤	Less-than or equal to
180	B4	├	Box drawings light vertical and left	244	F4	Ἰ	Greek upper case iota with dialytika
181	B5	┤	Box drawings vertical single and left double	245	F5	Ὶ	Greek upper case upsilon with dialytika
182	B6	┥	Box drawings vertical double and left single	246	F6	÷	Division sign
183	B7	┦	Box drawings down double and left single	247	F7	≈	Almost equal to
184	B8	┧	Box drawings down single and left double	248	F8	°	Degree sign
185	B9	┨	Box drawings double vertical and left	249	F9	·	Bullet operator
186	BA	┩	Box drawings double vertical	250	FA	·	Middle dot
187	BB	┪	Box drawings double down and left	251	FB	√	Square root
188	BC	┫	Box drawings double up and left	252	FC	ⁿ	Superscript latin lower case n
189	BD	┬	Box drawings up double and left single	253	FD	²	Superscript two
190	BE	┭	Box drawings up single and left double	254	FE	■	Black square
191	BF	┮	Box drawings light down and left	255	FF		No-break space

9. Αλληλεπίδραση με το ποντίκι

- Η εντολή *glutMouseFunc(function)* καλεί την συνάρτηση που περιέχει τις εντολές αλληλεπίδρασης με το ποντίκι
- Η συνάρτηση *void function (int button, int state, int x, int y)* διαβάζει την κατάσταση του ποντικιού και την αποθηκεύει σε μεταβλητές. Η μεταβλητή *button* παίρνει την τιμή του κουμπιού που πατήθηκε (GLUT_LEFT_BUTTON, GLUT_MIDDLE_BUTTON, GLUT_RIGHT_BUTTON). Η μεταβλητή *state* παίρνει την τιμή της κατάστασης του κουμπιού του ποντικιού (GLUT_UP, GLUT_DOWN). Οι μεταβλητές *x, y* παίρνουν τις τιμές των συντεταγμένων όπου βρίσκεται το ποντίκι την στιγμή που πατήθηκε ένα κουμπί του.
- Η εντολή *glutMotionFunc(function)* καλεί την συνάρτηση που περιέχει τις εντολές αλληλεπίδρασης με το ποντίκι καθώς αυτό κινείται
- Η συνάρτηση *void function (int x, int y)* διαβάζει την κίνηση του ποντικιού και την αποθηκεύει σε μεταβλητές. Οι μεταβλητές *x, y* παίρνουν τις τιμές των συντεταγμένων όπου βρίσκεται το ποντίκι

Παράδειγμα κώδικα:

```
#include <GL\glut.h>
#include <stdio.h> /* library for input/output */

void drawMousePoint (int pos_x, int pos_y) {
    /* Draws point following mouse move */
    glPointSize (5); /* Set the points size */
    glColor3f(0.0,0.0,1.0); /* Set the drawing color */
    glBegin (GL_POINTS); /* Draws points */
    glVertex2f (pos_x, pos_y);
    glEnd ();
    glFlush(); /* Force update of screen */
}

// mouse callback
void mouseClicked(int button, int state, int x, int y) {
    if( button == GLUT_LEFT_BUTTON && state == GLUT_DOWN ){
        printf("Left mouse button pressed\n");
    }
    if( button == GLUT_RIGHT_BUTTON && state == GLUT_DOWN ){
        printf("Right mouse button pressed\n");
    }
}

// mouse motion callback
void mouseMotion(int x, int y) {
    printf("X = %i, Y = %i\n",x, 500-y);
    drawMousePoint(x,500-y);
}

void display (void) {
    /* Called when OpenGL needs to update the display */
    glClearColor (1.0,1.0,0.0,0.0); /* Set the window color*/
    glClear (GL_COLOR_BUFFER_BIT); /* Clear the window */
    glFlush(); /* Force update of screen */
}

int main (int argc, char **argv) {
```



```

glutInit (&argc, argv); /* Initialise OpenGL */
glutInitWindowSize (500, 500); /* Set the window size */
glutInitWindowPosition (100, 100); /* Set the window position */
glutCreateWindow ("Mouse Interaction openGL-GLUT"); /* Create the window */
glutDisplayFunc (display); /* Register the "display" function */
glutMouseFunc (mouseClick); /* Register the "mouseClick" function */
glutMotionFunc(mouseMotion); /* Register the "mouseMotion" function */
glOrtho (0.0, 500, 0.0, 500, -1.0, 1.0); /* creates a matrix for an orthographic projection */
glutMainLoop (); /* Enter the OpenGL main loop */
return 0;
}

```

Αποτέλεσμα εκτέλεσης κώδικα:

Εάν γίνει κλικ σε κάποιο κουμπί του ποντικιού εμφανίζεται στην κονσόλα εξόδου μήνυμα με το κουμπί που πατήθηκε. Όταν ένα κουμπί του ποντικιού είναι πατημένο τότε σχεδιάζεται κουκίδα στο σημείο που βρίσκεται το ποντίκι, με τον τρόπο αυτό μπορούμε να σχεδιάσουμε στην οθόνη.



10. Δημιουργία μενού

- Η εντολή ***glutCreateMenu(function)*** δημιουργεί ένα pop up μενού και καλεί την συνάρτηση η οποία περιέχει τις εντολές που καλούνται με την επιλογή του κάθε στοιχείου του μενού
- Η συνάρτηση ***void function (int value)*** περιέχει τις εντολές που καλούνται με την επιλογή του κάθε στοιχείου του μενού. Δέχεται ως όρισμα ένα ακέραιο αριθμό που είναι ο αύξων αριθμός του στοιχείου του μενού ξεκινώντας από το 1
- Η εντολή ***glutAddMenuEntry(char *name, int value)*** προσθέτει ένα στοιχείο στο μενού, δέχεται ως όρισμα το εμφανιζόμενο κείμενο και τον αύξοντα αριθμό του στοιχείου
- Η εντολή ***glutAttachMenu(int button)*** συνδέει το μενού με ένα από τα κουμπιά του ποντικού, δέχεται ως όρισμα το κουμπί με τιμές GLUT_LEFT_BUTTON, GLUT_MIDDLE_BUTTON, GLUT_RIGHT_BUTTON

Παράδειγμα κώδικα:

```
#include <GL\glut.h>
#include <stdio.h> /* library for input/output */

void drawMousePoint (int pos_x, int pos_y) {
    /* Draws point following mouse move */
    glPointSize (5); /* Set the points size */
    glBegin (GL_POINTS); /* Draws points */
    glVertex2f (pos_x, pos_y);
    glEnd ();
    glFlush(); /* Force update of screen */
}

// mouse callback
void mouseClicked(int button, int state, int x, int y) {
    if( button == GLUT_LEFT_BUTTON && state == GLUT_DOWN ){
        printf("Left mouse button pressed\n");
    }
    if( button == GLUT_RIGHT_BUTTON && state == GLUT_DOWN ){
    }
}

// mouse motion callback
void mouseMotion(int x, int y) {
    printf("X = %i, Y = %i\n",x, 500-y);
    drawMousePoint(x,500-y);
}

//create a menu
void mymenu(int value) {
    if(value == 1)
        glClear(GL_COLOR_BUFFER_BIT);
        glFlush(); /* Force update of screen */
    if(value == 2)
        glColor3f(0.0,0.0,1.0); /* Set the drawing color for drawing*/
    if(value == 3)
        glColor3f(1.0,1.0,1.0); /* Set the drawing color for erasing*/
    if(value == 4)
        exit(0);
}
```

```

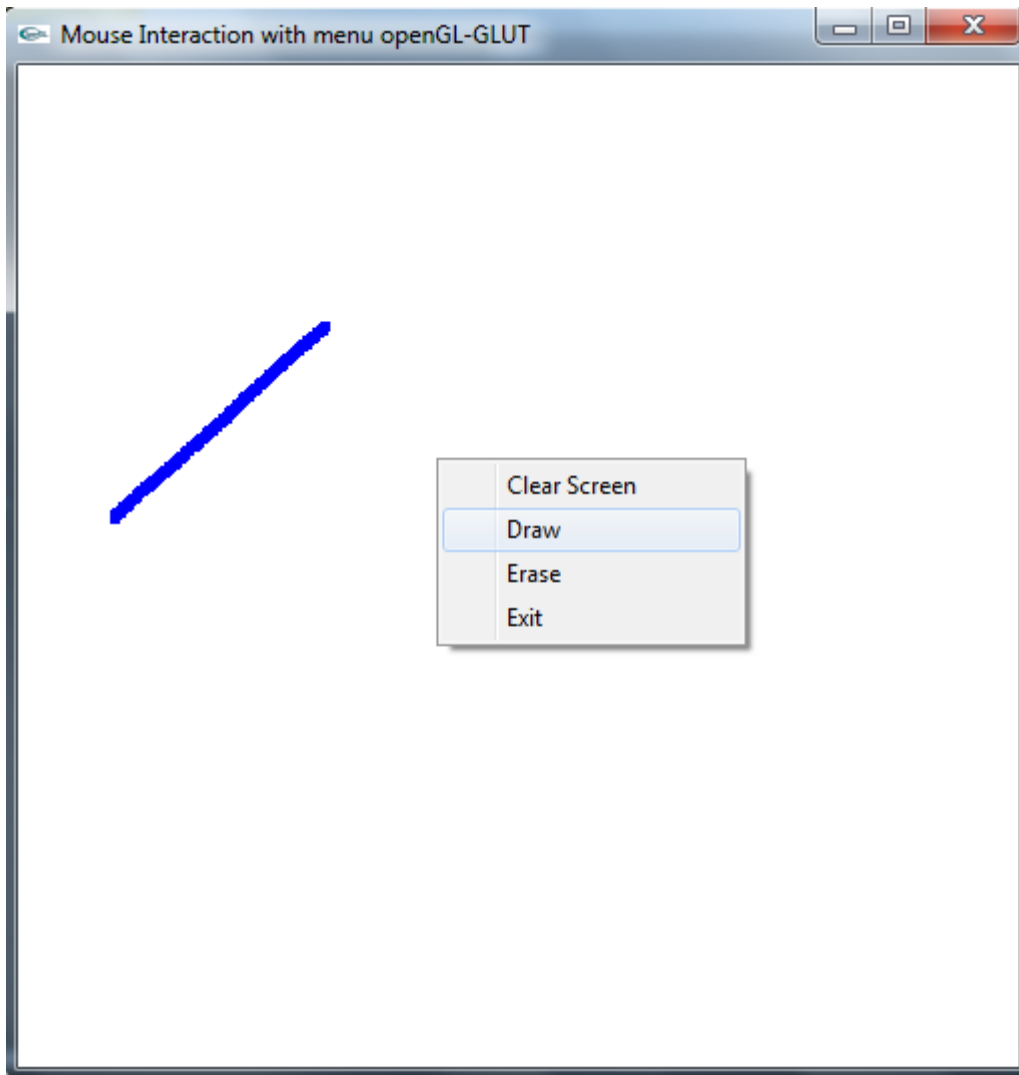
void display (void) {
/* Called when OpenGL needs to update the display */
glClearColor (1.0,1.0,1.0,0.0); /* Set the window color*/
glClear (GL_COLOR_BUFFER_BIT); /* Clear the window */
glColor3f(0.0,0.0,1.0); /* Set the drawing color for drawing*/
glFlush(); /* Force update of screen */
}

int main (int argc, char **argv) {
glutInit (&argc, argv); /* Initialise OpenGL */
glutInitWindowSize (500, 500); /* Set the window size */
glutInitWindowPosition (100, 100); /* Set the window position */
glutCreateWindow ("Mouse Interaction with menu openGL-GLUT"); /* Create the window */
glutDisplayFunc (display); /* Register the "display" function */
glutMouseFunc (mouseClick); /* Register the "mouseClick" function */
glutMotionFunc(mouseMotion); /* Register the "mouseMotion" function */
glOrtho (0.0, 500, 0.0, 500, -1.0, 1.0); /* creates a matrix for an orthographic projection */
glutCreateMenu(mymenu); // single menu, no need for id
glutAddMenuEntry("Clear Screen", 1);
glutAddMenuEntry("Draw", 2);
glutAddMenuEntry("Erase", 3);
glutAddMenuEntry("Exit", 4);
glutAttachMenu(GLUT_RIGHT_BUTTON);
glutMainLoop (); /* Enter the OpenGL main loop */
return 0;
}

```

Αποτέλεσμα εκτέλεσης κώδικα:

Εάν πατηθεί το δεξί κουμπί του ποντικιού εμφανίζεται μενού επιλογής λειτουργιών. Οι επιλογές του παραδείγματος είναι εκκαθάριση της οθόνης, σχεδίαση, σβήσιμο και έξοδος από το πρόγραμμα. Έτσι ο χρήστης μπορεί να σχεδιάσει με το ποντίκι, να σβήσει αυτό που σχεδίασε και να καθαρίσει την οθόνη.



Παραπομπές

- Λίστα εντολών OpenGL: <https://www.opengl.org/sdk/docs/man4/>
 1. **glClearColor** : <https://www.opengl.org/sdk/docs/man4/html/glClearColor.xhtml>
 2. **glClear** : <https://www.opengl.org/sdk/docs/man4/html/glClear.xhtml>
 3. **glFlush** : <https://www.opengl.org/sdk/docs/man4/html/glFlush.xhtml>
 4. **glBegin** : <https://www.opengl.org/sdk/docs/man2/xhtml/glBegin.xml>
 5. **glEnd** : <https://www.opengl.org/sdk/docs/man2/xhtml/glBegin.xml>
 6. **glVertex** : <https://www.opengl.org/sdk/docs/man2/xhtml/glVertex.xml>
 7. **glPointSize** : <https://www.opengl.org/sdk/docs/man4/html/glPointSize.xhtml>
 8. **glLineWidth** : <https://www.opengl.org/sdk/docs/man4/html/glLineWidth.xhtml>
 9. **glColor** : <https://www.opengl.org/sdk/docs/man2/xhtml/glColor.xml>
 10. **glOrtho** : <https://www.opengl.org/sdk/docs/man2/xhtml/glOrtho.xml>
 11. **glRasterPos** : <https://www.opengl.org/sdk/docs/man2/xhtml/glRasterPos.xml>
- Λίστα εντολών GLUT: <https://www.opengl.org/documentation/specs/glut/spec3/spec3.html>
 1. **glutInit** : <https://www.opengl.org/documentation/specs/glut/spec3/node10.html#SECTION00031000000000000000>
 2. **glutCreateWindow** : <https://www.opengl.org/documentation/specs/glut/spec3/node16.html#SECTION00051000000000000000>
 3. **glutDisplayFunc** : <https://www.opengl.org/documentation/specs/glut/spec3/node46.html#SECTION00081000000000000000>
 4. **glutMainLoop** : <https://www.opengl.org/documentation/specs/glut/spec3/node14.html#SECTION00041000000000000000>
 5. **glutInitWindowSize** : <https://www.opengl.org/documentation/specs/glut/spec3/node11.html#SECTION00032000000000000000>
 6. **glutInitWindowPosition** : <https://www.opengl.org/documentation/specs/glut/spec3/node11.html#SECTION00032000000000000000>
 7. **glutBitmapCharacter** : <https://www.opengl.org/documentation/specs/glut/spec3/node76.html>
 8. **glutKeyboardFunc** : <https://www.opengl.org/documentation/specs/glut/spec3/node49.html>
 9. **glutMouseFunc** : <https://www.opengl.org/documentation/specs/glut/spec3/node50.html>
 10. **glutMotionFunc** : <https://www.opengl.org/resources/libraries/glut/spec3/node51.html>
 11. **glutCreateMenu** : <https://www.opengl.org/documentation/specs/glut/spec3/node36.html>
 12. **glutAddMenuEntry** : <https://www.opengl.org/resources/libraries/glut/spec3/node39.html>
 13. **glutAttachMenu** : <https://www.opengl.org/documentation/specs/glut/spec3/node44.html>