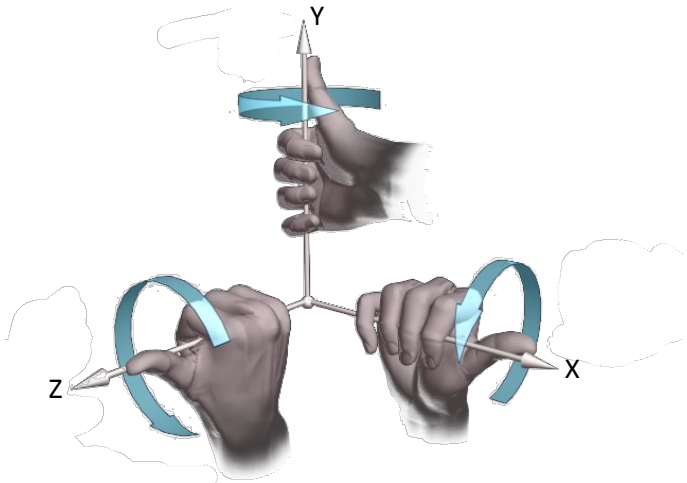
OpenGL Primitives

1. Εισαγωγή

Η σχεδίαση ενός αντικειμένου γίνεται με την βοήθεια σχεδίασης πρωταρχικών αντικειμένων που Υποστηρίζονται άμεσα από την βιβλιοθήκη γραφικών. Τα στοιχειώδη αντικείμενα γραφικών είναι τα σημεία (points) και τα ευθύγραμμα τμήματα (lines). Με την σύνθεση των παραπάνω προκύπτουν πιο πολύπλοκα αντικείμενα είτε πρωταρχικά που δημιουργούνται από αλγορίθμους της βιβλιοθήκης γραφικών ή είναι αντικείμενα που σχεδιάζονται από τον χρήστη. Για τον ορισμό και την σχεδίαση των αντικειμένων γραφικών χρησιμοποιείται η έννοια του **Vertex** (δηλαδή η κάθε κορυφή ενός γεωμετρικού αντικειμένου).

Όλα τα βασικά σχήματα στην OpenGL σχηματίζονται με **δήλωση** των κορυφών τους. Με τον όρο βασικά σχήματα αναφερόμαστε σε γραμμές, τρίγωνα και πολύγωνα. Πιο σύνθετα σχήματα, όπως κύκλοι και ελλείψεις προσεγγίζονται με τη σύνδεση πολλαπλών σημείων τους με ευθύγραμμα τμήματα. Στην OpenGL η σκηνή αναπαρίσταται στη γενική περίπτωση σε τρισδιάστατο καρτεσιανό σύστημα συντεταγμένων. Επιπλέον το σύστημα συντεταγμένων είναι δεξιόστροφο δηλαδή τα μοναδιαία διανύσματα $\vec{x}, \vec{y}, \vec{z}$ συνδέονται με σχέσεις εξωτερικού γινομένου ως εξής.

$$\vec{x} \times \vec{y} = \vec{z},\ \vec{y} \times \vec{z} = \vec{x},\ \vec{z} \times \vec{x} = \vec{y}\ .$$



Σχ. 1. Δεξιόστροφο σύστημα συντεταγμένων.

Κάθε σημείο αναπαρίσταται με τη χρήση **ομογενών συντεταγμένων**. Με τη χρήση ομογενών συντεταγμένων όλα τα σημεία αναπαρίστανται με τέσσερις συντεταγμένες κινητής υποδιαστολής (x, y, z, **w**). Αν η παράμετρος **w** είναι διαφορετική από το 0, τότε αυτές οι συντεταγμένες ανταποκρίνονται στο τρισδιάστατο σημείο (x/w, y/w, z/w). Η συντεταγμένη **w** μπορεί να καθοριστεί

στις εντολές της OpenGL, αλλά αυτό γίνεται σπανίως. Αν η συντεταγμένη w **δεν** είναι **καθορισμένη**, το σύστημα της αποδίδει την **τιμή 1**.

2. Καθορισμός χρώματος της OpenGL

Με την OpenGL, η περιγραφή του σχήματος ενός αντικειμένου και ο χρωματισμός του είναι ανεξάρτητα. Κάθε φορά που δίνουμε **εντολή σχεδίασης** ενός συγκεκριμένου γεωμετρικού σχήματος, το τρέχον επιλεγμένο χρώμα καθορίζει και το χρώμα με το οποίο το σχήμα σχεδιάζεται. Για να καθορίσουμε ένα χρώμα, χρησιμοποιούμε την εντολή **glColor3f()**. Η εντολή αυτή παίρνει τρεις παραμέτρους, οι οποίες είναι όλες αριθμοί κινητής υποδιαστολής ή διπλής ακρίβειας όπως φαίνεται στον πίνακα 1.

Πίνακας 1

Εντολή	Σχόλιο
glColor3ub(GLubyte red, GLubyte green, GLubyte blue);	$(0 \le red, green, blue \le 255)$
glColor3f(GLfloat red, GLfloat green, GLfloat blue);	$(0 \le red, green, blue \le 1)$
glColor3d(GLdouble red, GLdouble green, GLdouble blue);	$(0 \le red, green, blue \le 1)$

Οι παράμετροι red, green και blue αντιστοιχούν στις τιμές των χρωματικών συνιστωσών του χρώματος στο μοντέλο RGB. π.χ. η εντολή

glColor3f(1,0,0);

επιστρέφει το πιο έντονο κόκκινο που μπορεί να αποδώσει το σύστημα. Στον πίνακα 2 ορίζονται ορισμένα βασικά χρώματα.

Πίνακας 2

Εντολή	Χρώμα
glColor3f(0, 0, 0);	Μαύρο
glColor3f(1, 0, 0);	Κόκκινο
glColor3f(0, 1, 0);	Πράσινο
glColor3f(0, 0, 1);	Μπλε
glColor3f(1, 0, 1);	Ματζέντα
glColor3f(0, 1, 1);	Κυανό
glColor3f(1, 1, 0);	Κίτρινο
glColor3f(1, 1, 1);	Λευκό

3. Καθορισμός των κορυφών

Στην OpenGL, όλα τα γεωμετρικά σχήματα περιγράφονται δηλώνοντας τις κορυφές τους. Για τον καθορισμό μιας κορυφής χρησιμοποιείται η εντολή **gl**Vertex*. Η συνάρτηση δέχεται ως όρισμα μέχρι τέσσερις συντεταγμένες (x, y, z, w) για μία συγκεκριμένη κορυφή ή και μέχρι δύο (x, y) χρησιμοποιώντας την κατάλληλη εκδοχή της εντολής για το επίπεδο. Η εντολή **gl**Vertex*() πρέπει να εκτελείται μεταξύ των εντολών **gl**Begin() και **gl**End(). Στη συνέχεια δίνονται μερικά παραδείγματα με χρήση της **gl**Vertex*.

glVertex2s(2,3); *Δήλωση σημείου με συντεταγμένες (x,y) = (2,3)*
glVertex3d(0,0,3.14); *Δήλωση σημείου με συντεταγμένες (x,y,z) = (0,0,3.14)*
glVertex4f(2.3, 1.0, -2.2, 2.0);

Στο πρώτο παράδειγμα αναπαριστάται μία κορυφή με συντεταγμένες σε τρεις διαστάσεις (2, 3, 0). Οι συντεταγμένες (0.0, 0.0, 3.14) στο δεύτερο παράδειγμα είναι αριθμοί κινητής υποδιαστολής με διπλή ακρίβεια. Στο τρίτο παράδειγμα αναπαριστάται μία κορυφή με τρισδιάστατες συντεταγμένες (1.15, 0.5, -1.1) διότι οι συντεταγμένες x, y, και z διαιρούνται με την συντεταγμένη **w**.

4. Η δομή glBegin() - glEnd()

Ο ορισμός των κορυφών κάθε γεωμετρικού σχήματος περικλείεται μεταξύ δύο εντολών, των **gl**Begin() και **gl**End():

glBegin(GL_type);

ορισμός των κορυφών

glEnd();

Το είδος του γεωμετρικού σχήματος που σχεδιάζεται, εξαρτάται από την **παράμετρο** που εισάγεται στο όρισμα **GL_type** της εντολής **gl**Begin.

4.1. Σημεία

Ένα σημείο αναπαρίσταται από ένα σύνολο αριθμών κινητής υποδιαστολής που ονομάζεται κορυφή. Όλοι οι εσωτερικοί υπολογισμοί γίνονται σαν οι κορυφές να είναι τρισδιάστατες. Κορυφές που καθορίζονται από το χρήστη ως δισδιάστατες (δηλαδή δίνονται μόνο οι τιμές για τις x και y συνιστώσες) παίρνουν από την OpenGL την τιμή z=0. Η **σχεδίαση σημείων** επιτυγχάνεται δίνοντας ως όρισμα στην **gl**Begin( ) τη σταθερά **GL_POINTS**. Μία χαρακτηριστική **ιδιότητα** των σημείων που μπορούν να μεταβληθούν είναι το πάχος τους. Για τη ρύθμιση του μεγέθους ενός σημείου, χρησιμοποιείται η εντολή **gl**PointSize( ):

glPointSize(**GL**float **size**);

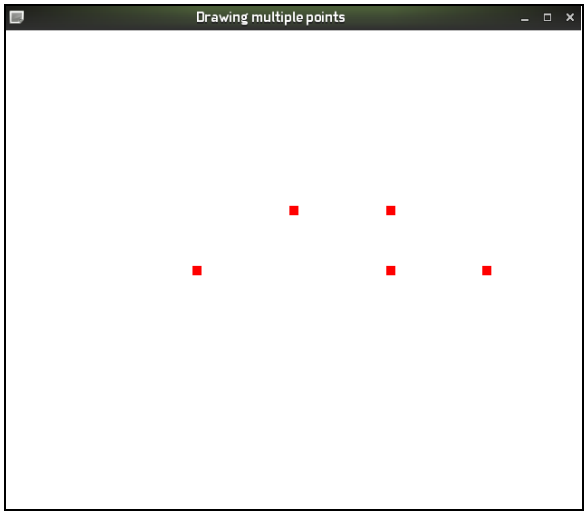
όπου **size** το πλάτος του σημείου σε pixels. Προφανώς, πρέπει να είναι μεγαλύτερο από 0 και η προκαθορισμένη τιμή του είναι **1**. Εξ' ορισμού ένα σημείο απεικονίζεται στην οθόνη σαν ένα **pixel**. Ομαλοποίηση των σημείων επιτυγχάνεται δίνοντας ως όρισμα την παράμετρο GL_POINT_SMOOTH στην εντολή **gl**Enable( ):

glEnable(GL_POINT_SMOOTH);

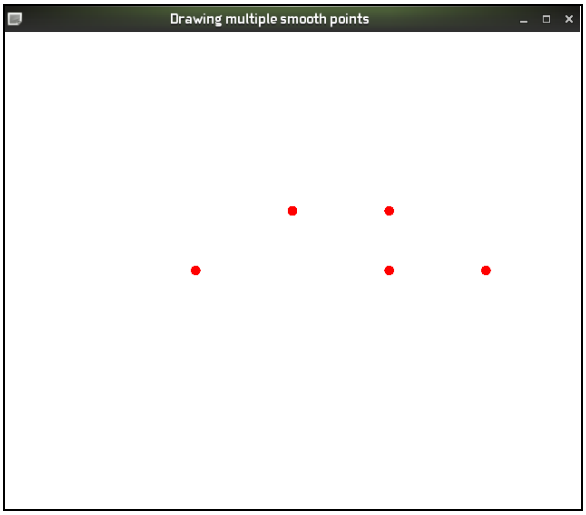
4.2. Εφαρμογή για σχεδίαση σημείων

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

void draw_points()
{
glClearColor(0,0,0,0);
glClear(GL_COLOR_BUFFER_BIT);
glColor3f(1,0,0);
glPointSize(3);
glBegin(GL_POINTS);
      glVertex2i(10,10);
      glVertex2i(15,15);
      glVertex2i(20,10);
      glVertex2i(20,15);
      glVertex2i(25,10);
```

```
glEnd();
glFlush();
}

int main(int argc, char** argv)
{
glutInit(&argc,argv);
glutInitWindowPosition(50,50);
glutInitWindowSize(640,512);
glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
glutCreateWindow("Drawing multiple points");
glMatrixMode(GL_PROJECTION);
gluOrtho2D(0,30,-10,25);
glutDisplayFunc(draw_points);
glutMainLoop();
return 0;
}
```



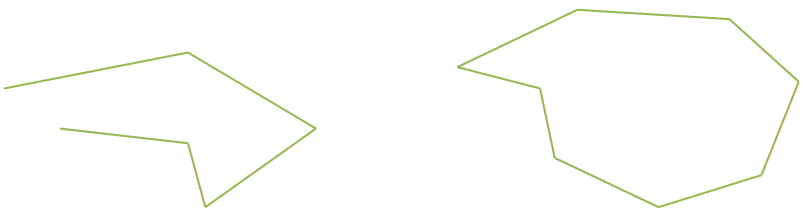
Σχ. 2. Σημεία στο επίπεδο



Σχ. 3. Ομαλοποιημένα (smoothed) σημεία στο επίπεδο

4.3. Σχεδίαση γραμμών

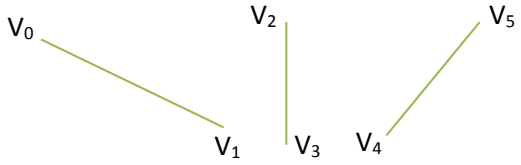
Στην OpenGL, μία γραμμή σημαίνει ένα ευθύγραμμο τμήμα και όχι η μαθηματική έννοια που εκτείνεται στο άπειρο και στις δύο κατευθύνσεις. Υπάρχουν εύκολοι τρόποι για να καθοριστεί μία σειρά από συνδεμένα ευθύγραμμα τμήματα ή μία κλειστή σειρά από συνδεμένα ευθύγραμμα τμήματα (σχ. 2). Σε όλες τις περιπτώσεις όμως, οι γραμμές που αποτελούν τις συνδεμένες σειρές καθορίζονται με τις συντεταγμένες των άκρων τους.



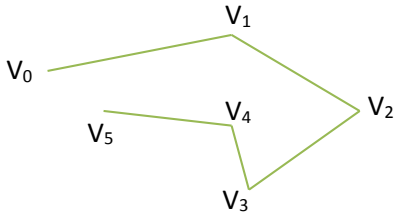
Σχ. 4. Σχεδίαση ευθύγραμμων τμημάτων

Ανάλογα με τον τύπο της παραμέτρου που δίνεται στη είσοδο της **gl**Begin() μπορούν να επιλεχθούν διαφορετικοί τρόποι σχεδίασης γραμμών.

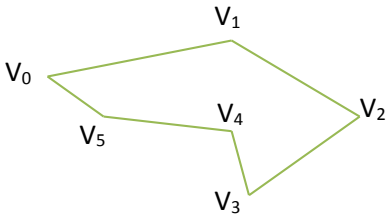
- GL_LINES, τα δοθέντα σημεία ορίζουν κατά ζεύγη ευθύγραμμα τμήματα



- GL_LINE_STRIP, δίνοντας τα σημεία $(v_0, v_1, \ldots, v_{n-1})$, σχεδιάζονται διαδοχικά τα ευθύγραμμα τμήματα $(v_0, v_1), (v_1, v_2), \ldots, (v_{n-2}, v_{n-1})$
-



- GL_LINE_LOOP, όπως και η GL_LINE_STRIP με τη μόνη διαφορά ότι το **πρώτο** και το **τελευταίο** σημείο συνενώνονται, σχεδιάζοντας κλειστή γραμμή



4.3.1. Ιδιότητες των γραμμών

Όσον αφορά τις ιδιότητες των γραμμών υπάρχει δυνατότητα τροποποίησης του πάχους και της διάστιξή τους.

- το πάχος της γραμμής τροποποιείται με τη χρήση της εντολής

  glLineWidth(**GL**float **width**);

  όπου **width** το πάχος γραμμής σε pixels. Η παράμετρος width πρέπει να είναι μεγαλύτερη από το 0 και εξ' ορισμού είναι ίση με 1. Το πάχος γραμμής είναι μεταβλητή κατάστασης και συνεπώς διατηρεί την τιμή που της ανατέθηκε την τελευταία φορά
- στην OpenGL, οι γραμμές σχεδιάζονται ως συνεχή ευθύγραμμα τμήματα που ενώνουν τα δύο άκρα τους, ωστόσο υπάρχουν περιπτώσεις σχεδίασης με διακεκομμένες γραμμές διαφόρων μορφών. Στην OpenGL η δυνατότητα χρήσης διάστικτων γραμμών αποτελεί μεταβλητή κατάστασης, αυτό σημαίνει ότι θα πρέπει να τη δηλώσουμε ρητά στο πρόγραμμά μας. Η κατάσταση ενεργοποίησης διάστιξης γραμμών ενεργοποιείται δίνοντας την παράμετρο GL_LINE_STIPPLE στην εντολή **gl**Enable( )

  glEnable(GL_LINE_STIPPLE);

  στη συνέχεια η μορφή των διάστικτων γραμμών καθορίζεται με την εντολή **gl**LineStipple( )

  glLineStipple (**GL**int **factor**, **GL**ushort **pattern**);

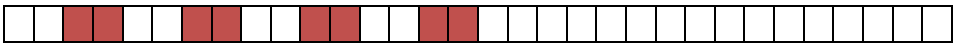
  Το όρισμα pattern καθορίζει το **μοτίβο** των ευθειών και δίνεται δε δεκαεξαδική μορφή. Οι θέσεις των **άσσων** (1) καθορίζουν τα pixels της γραμμής που θα σχεδιαστούν, ενώ οι θέσεις

των **μηδενικών** (0) καθορίζουν τα pixels που θα παραμείνουν κενά. Έτσι για την τιμή **0x00AA** του μοτίβου (factor ίσο με 1) δίνοντας την εντολή θα σχεδιαζόταν το παρακάτω μοτίβο

glLineStipple(1,0x00AA);

Η έναρξη του μοτίβου γίνεται από το λιγότερο σημαντικό ψηφίο του pattern

Η παράμετρος **factor** αναπαράγει κάθε δυαδικό ψηφίο όσες φορές δηλώνει η τιμή της. Οπότε, στο παραπάνω παράδειγμα, αν η τιμή του factor είναι 2, θα δημιουργηθεί η ακόλουθη διάστιξη



Στον πίνακα 3 φαίνονται σχεδιασμένες γραμμές με διαφορετικά μοτίβα και factors

Πίνακας 3

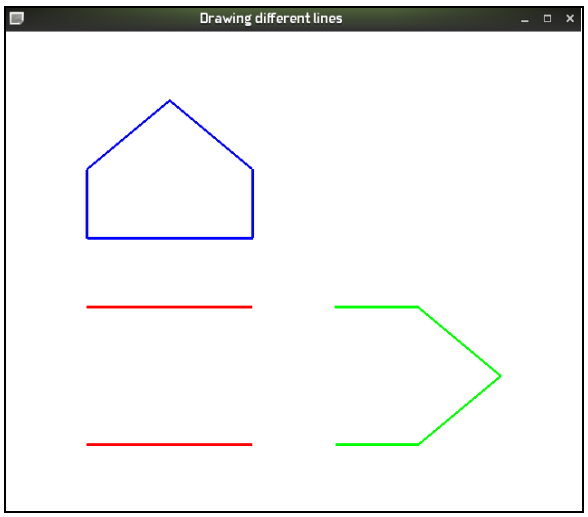
Μοτίβο	Factor	Εμφάνιση γραμμής
0x00FF	1	
0x00FF	2	
0x0C0F	1	
0x0C0F	3	
0xAAAA	1	
0xAAAA	2	
0xAAAA	3	
0xAAAA	4	

4.4. Εφαρμογές για σχεδίαση γραμμών

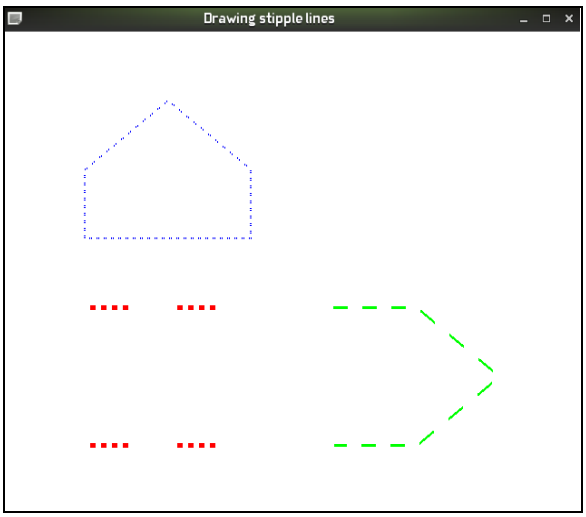
```
#include <GL/glut.h>

void draw_lines()
{
glClearColor(1,1,1,0);
glClear(GL_COLOR_BUFFER_BIT);

glLineWidth(3);
glColor3f(1,0,0);

glBegin(GL_LINES);
    glVertex2i(0,0);
    glVertex2i(10,0);
    glVertex2i(0,10);
    glVertex2i(10,10);
glEnd();

glColor3f(0,1,0);

glBegin(GL_LINE_STRIP);
    glVertex2i(15,10);
```

```
    glVertex2i(20,10);
    glVertex2i(25,5);
    glVertex2i(20,0);
    glVertex2i(15,0);
glEnd();

glColor3f(0,0,1);

glBegin(GL_LINE_LOOP);
    glVertex2i(0,15);
    glVertex2i(0,20);
    glVertex2i(5,25);
    glVertex2i(10,20);
    glVertex2i(10,15);
glEnd();

glFlush();
}
int main(int argc, char** argv)
{
glutInit(&argc,argv);
glutInitWindowPosition(150,150);
glutInitWindowSize(640,512);
glutInitDisplayMode(GLUT_SINGLE|GLUT_RGBA);
glutCreateWindow("Drawing different lines");
glMatrixMode(GL_PROJECTION);
gluOrtho2D(-5,30,-5,30);
glutDisplayFunc(draw_lines);
glutMainLoop();
return 0;
}


#include <GL/glut.h>

void draw_lines()
{
glClearColor(1,1,1,0);
glClear(GL_COLOR_BUFFER_BIT);

glColor3f(1,0,0);
glLineWidth(5);
glLineStipple(1,0x00AA);

glBegin(GL_LINES);
    glVertex2i(0,0);
    glVertex2i(10,0);
    glVertex2i(0,10);
    glVertex2i(10,10);
glEnd();

glColor3f(0,1,0);
glLineWidth(3);
glLineStipple(2,0x00FF);

glBegin(GL_LINE_STRIP);
```

```
    glVertex2i(15,10);
    glVertex2i(20,10);
    glVertex2i(25,5);
    glVertex2i(20,0);
    glVertex2i(15,0);
glEnd();

glColor3f(0,0,1);
glLineWidth(2);
glLineStipple(1,0x0A0A);

glBegin(GL_LINE_LOOP);
    glVertex2i(0,15);
    glVertex2i(0,20);
    glVertex2i(5,25);
    glVertex2i(10,20);
    glVertex2i(10,15);
glEnd();
glFlush();
}

int main(int argc, char** argv)
{
glutInit(&argc,argv);
glutInitWindowPosition(200,200);
glutInitWindowSize(640,512);
glutInitDisplayMode(GLUT_SINGLE|GLUT_RGBA);
glutCreateWindow("Drawing stippled lines");
glMatrixMode(GL_PROJECTION);
gluOrtho2D(-5,30,-5,30);
glEnable(GL_LINE_STIPPLE);
glutDisplayFunc(draw_lines);
glutMainLoop();
return 0;
}
```



Σχ. 5. Σχεδιασμένες γραμμές με διαφορετικό χρώμα



Σχ. 6. Σχεδιασμένες γραμμές με διαφορετικά πρότυπα