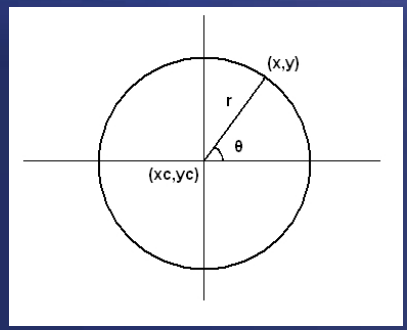
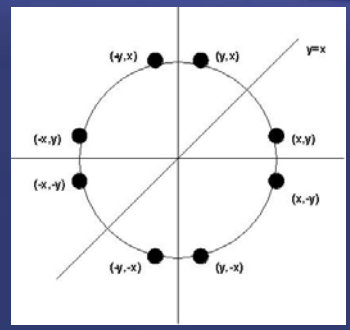
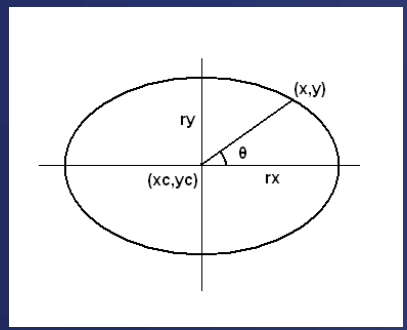
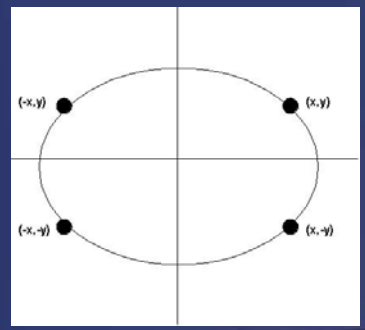
# Βασικές αρχές σχεδίασης (A)



# Περιεχόμενα ενότητας

- Πρωτογενείς τύποι δεδομένων
- Ονοματολογία – Συμβάσεις
- Η μηχανή καταστάσεων της OpenGL
- Περιβάλλον σχεδίασης
- Χρώμα
- Φιλοσοφία σχεδιάσης στην OpenGL
- Σχεδίαση σημείων και γραμμών

# Πρωτογενείς τύποι δεδομένων

| Τύπος της OpenGL | Τύπος δεδομένων | Αντίστοιχος τύπος στη C | Επίθημα |
|---|---|---|---|
| GLbyte | ακέραιος 8bits | signed char | b |
| GLshort | ακέραιος 16bits | short | s |
| GLint / GLsizei | ακέραιος 32bits | int/long | i |
| GLfloat / GLclampf | κινητής υποδιαστολής | float | f |
| GLdouble / GLclampd | κινητής υποδιαστολής διπλής ακρίβειας | double | d |
| GLubyte / GLboolean | ακέραιος 8bits χωρίς πρόσημο | unsigned char | ub |
| GLushort | ακέραιος 16bits χωρίς πρόσημο | unsigned short | us |
| GLuint/GLenum/GLbitfield | ακέραιος 32bits χωρίς πρόσημο | unsigned int unsigned long | ui |

# Ονοματολογία συμβολικών σταθερών στην OpenGL

Συμβολικές σταθερές: προκαθορισμένες σταθερές που συνήθως χρησιμοποιούνται ως ορίσματα σε εντολές τις OpenGL και εκπροσωπούν συγκεκριμένες ρυθμίσεις.

Συμβολικές σταθερές της κύριας βιβλιοθήκης (core library):
GL_(όνομα_σταθεράς)

Πχ
GL_COLOR_BUFFER_BIT
GL_LIGHTING

Συμβολικές σταθερές της της βιβλιοθήκης GLUT
GLUT_(όνομα_σταθεράς)

Π.χ.
GLUT_RGB

# Ονοματολογία εντολών στην OpenGL (1)

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

• τον τύπο των ορισμάτων που δέχονται (π.χ. ακέραιοι ή πραγματικοί),

• τις διαστάσεις του χώρου (π.χ. σχεδίαση σε δύο ή τρείς διαστάσεις)

• το χρωματικό μοντέλο

• τον τρόπο με τον οποίο περνάμε ορίσματα (call by value ή call by reference).

# Ονοματολογία εντολών στην OpenGL (2)

•Οι εντολές στην OpenGL έχουν επίθημα που καθορίζει το πλήθος και τύπο των ορισμάτων που δέχονται

Σύμβαση: Τα επιθήματα συνδυάζονται ως εξής

gl

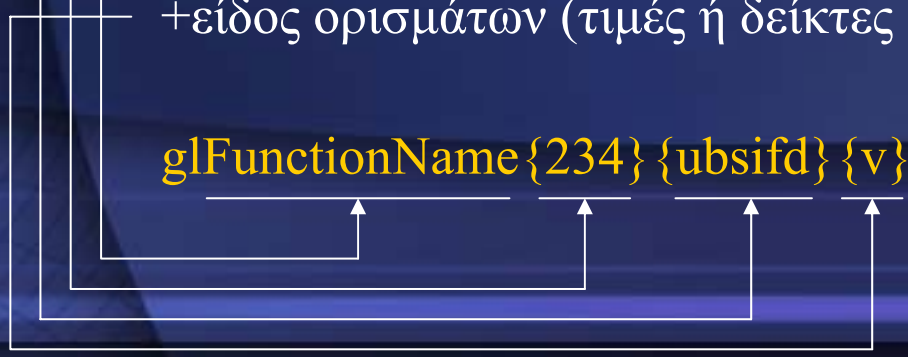
+όνομα εντολής

+διάσταση χώρου ή πλήθος χρωματικών τιμών {2,3,4}

+επίθημα πρωτογενούς τύπου δεδομένων {ubsifd}

+είδος ορισμάτων (τιμές ή δείκτες σε μητρώα) {-/v}

glFunctionName{234}{ubsifd}{v}



# Παράδειγμα ονοματολογίας: glVertex

Δήλωση σημείου στο διδιάστατο χώρο {2} με συντεταγμένες που δίνονται ως πραγματικοί αριθμοί απλής ακρίβειας {f}

glVertex2f (GLfloat x, GLfloat y);

Δήλωση σημείου στον τρισδιάστατο χώρο {3} με συντεταγμένες που δίνονται ως προσημασμένοι ακέραιοι {i}

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

Για πραγματικές συντεταγμένες {f} ενός σημείου στον τρισδιάστατο χώρο {3} που δίνονται με τη μορφή μητρώου (όρισμα δείκτης σε πίνακα) {v}

GLfloat coord[ ]={1,2,3};

glVertex3fv(const GLfloat  *coord);

# Η OpenGL ως μηχανή καταστάσεων

•Το περιβάλλον της OpenGL λειτουργεί ως μια μηχανή καταστάσεων.

•Μηχανή καταστάσεων: Ένα περιβάλλον, το οποίο, σε κάθε χρονική στιγμή, λειτουργεί βάσει προκαθορισμένων ιδιοτήτων, των μεταβλητών κατάστασης (state variables).

•Μεταβλητές κατάστασης: Έχουν μια προκαθορισμένη αρχική τιμή η οποία μπορεί να μεταβληθεί κατά την πορεία της εκτέλεσης του κώδικα από τον προγραμματιστή.

•Οι τρέχουσες τιμές των μεταβλητών παραμένουν ενεργές.

# Παραδείγματα μεταβλητών κατάστασης

•Ορισμένα παραδείγματα μεταβλητών κατάστασης:

1) τρέχον χρώμα σχεδίασης
2) πάχος γραμμών
3) χρώμα καθαρισμού της οθόνης (φόντου)

•Είναι σημαντικό ο προγραμματιστής να αρχικοποιεί τις μεταβλητές κατάστασης, όποτε αυτό είναι απαραίτητο και να παρακολουθεί τις τιμές τους.

# Ιδιότητες δύο καταστάσεων

•Έχουν δύο πιθανές τιμές (TRUE ή FALSE)

•Καθορίζουν την ενεργοποίηση ή μη εξειδικευμένων λειτουργιών (μίξη χρωμάτων, φωτορεαλισμός, απόδοση υφής κ.λ.π).

•Δίνοντας στον προγραμματιστή τη δυνατότητα ενεργοποίησης λειτουργιών, η μηχανή της OpenGL βελτιστοποιεί το υπολογιστικό φορτίο.

# Ιδιότητες δύο καταστάσεων

•Οι υποστηριζόμενες λειτουργίες της μηχανής καταστάσεων της OpenGL ενεργοποιούνται και απενεργοποιούνται με τις εντολές glEnable και glDisable αντίστοιχα.

void glEnable(GLenum cap);
void glDisable(Glenum cap);

cap: Η ιδιότητα που ενεργοποιείται η απενεργοποιείται

Πχ
glEnable(GL_BLEND); Ενεργοποίηση μίξης χρωμάτων

•Κάθε ιδιότητα που ενεργοποιείται παραμένει ενεργή σε όλη τη διάρκεια εκτέλεσης του προγράμματος

# Επισκόπηση ιδιοτήτων δύο καταστάσεων

•Προκειμένου να ελέγξουμε αν μια ιδιότητα δύο καταστάσεων είναι ενεργοποιημένη ή απενεργοποιημένη χρησιμοποιούμε την εντολή glIsEnabled:

GLboolean glIsEnabled(GLenum capability);

•Επιστρέφει GL_TRUE ή GL_FALSE, ανάλογα με το αν η εξεταζόμενη ιδιότητα είναι ενεργοποιημένη ή όχι.

# Ιδιότητες κατάστασης πολλαπλών τιμών

•Υπάρχουν μεταβλητές κατάστασης που μπορούν να πάρουν περισσότερες από δύο τιμές (πχ τρέχον χρώμα σχεδίασης ή πάχος γραμμών)

•Οι τιμές των συνθετων ιδιοτήτων κατάστασης ρυθμίζονται με ξεχωριστές εντολές της OpenGL η καθεμιά.

# Επισκόπηση σύνθετων ιδιοτήτων κατάστασης

Οι εντολές glGet{Type}v χρησιμοποιούνται για την επισκόπηση της τρέχουσας τιμής μεταβλητών κατάστασης

```
void glGetBooleanv(GLenum parameterName, GLboolean *parameters);
void glGetIntegerv(GLenum parameterName, GLint *parameters);
void glGetFloatv(GLenum parameterName, GLfloat *parameters);
void glGetDoublev(GLenum parameterName, GLdouble *parameters);
```

parameterName: συμβολική σταθερά που καθορίζει την ιδιότητα

parameters: δείκτης σε μητρώο όπου αποθηκεύεται η τιμή ή το σύνολο τιμών που προσδιορίζουν την ιδιότητα.

# Παράδειγμα: Επισκόπηση τρέχοντος χρώματος σχεδίασης

Το μητρώο parameters θα περιέχει τις τιμές τριών χρωματικών συνιστωσών (μοντέλο RGB, διάσταση 3)

GLfloat colorVal[3];

Παράμετρος = τρέχον χρώμα σχεδίασης

parameterName = GL_CURRENT_COLOR

Αποθήκευση των τρεχουσών χρωματικών συνιστωσών στο μητρώο colorVal:

glGetFloatv( GL_CURRENT_COLOR, colorVal );

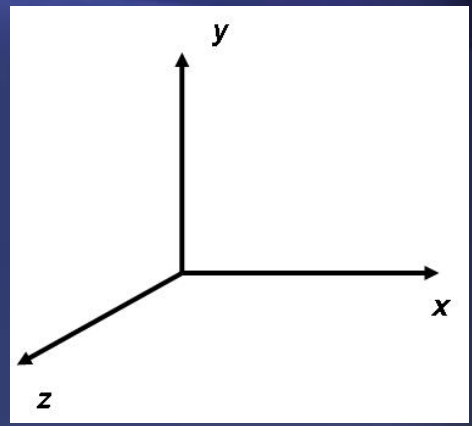
# Σχεδίαση στην OpenGL

•Όλα τα γεωμετρικά σχήματα ορίζονται με δήλωση των κορυφών τους

•Σύνθετα σχήματα προσεγγίζονται από στοιχειώδη βασικά σχήματα.

•Κάθε σημείο της σκηνής αναπαρίσταται, στη γενική περίπτωση, σε τρισδιάστατο καρτεσιανό σύστημα συντεταγμένων

•Το σύστημα συντεταγμένων είναι δεξιόστροφο.




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

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

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

# Ομογενείς συντεταγμένες

•Όλα τα σημεία αναπαρίστανται με τέσσερις συντεταγμένες κινητής υποδιαστολής $(x, y, z, w)$.

•Η δήλωση ενός σημείου με ομογενείς συντεταγμένες $(x, y, z, w)$ ορίζει στον τρισδιάστατο χώρο χώρο ένα σημείο με συντεταγμένες $(x/w, y/w, z/w)$

•Αν δε δίνεται η συντεταγμένη z, το σύστημα θεωρεί την τιμή 0.

•Αν δε δίνεται η συντεταγμένη w, το σύστημα θεωρεί την τιμή 1.

# Καθαρισμός οθόνης

•Πριν το σχεδιασμό μιας  σκηνής, απαιτείται ο καθαρισμός του ενταμιευτή χρωματικών τιμών (color buffer) του υπολογιστή,

Ενταμιευτής χρωματικών τιμών: περιοχή μνήμης όπου αποθηκεύονται οι χρωματικές πληροφορίες για τη σχεδιαζόμενη σκηνή

•“Καθαρισμός” οθόνης = αρχικοποίηση των ενταμιευτών με κάποια προκαθορισμένη τιμή.

•Ο καθαρισμός τηε οθόνης γίνεται με το χρώμα φόντου που επιλέγει ο προγραμματιστής.

# Ρύθμιση χρώματος καθαρισμού

•Το χρώμα καθαρισμού της οθόνης είναι μεταβλητή κατάστασης και ορίζεται με την εντολή glClearColor.

glClearColor(GLfloat red, GLfloat green, GLfloat blue, GLfloat alpha);

red, green, blue, alpha: τα βάρη του χρώματος στο χρωματικό μοντέλο RGBA

•Το χρώμα καθαρισμού, ως μεταβλητή κατάστασης, διατηρεί την τελευταία τιμή που του ανατέθηκε.

# Καθαρισμός ενταμιευτών

•Ο καθαρισμός ενταμιευτών αυτός καθεαυτός γίνεται με την εντολή glClear.

•void glClear(GLEnum buffer);

•buffer: ο ενταμιευτής που θέλουμε να καθαρίσουμε

•Η μηχανή της OpenGL περιέχει πολλούς ενταμιευτές, οπότε πρεπει να καθορίσουμε το είδος του ενταμιευτή που επιθυμούμε να καθαρίσουμε

•GL_COLOR_BUFFER_BIT: καθαρισμός ενταμιευτή χρωματικών τιμών (colour buffer)

•GL_DEPTH_BUFFER_BIT: καθαρισμός ενταμιευτή βάθους (depth buffer)

# Παράδειγμα: Καθαρισμός ενταμιευτή χρωματικών τιμών

Καθαρισμός της οθόνης με μαύρο χρώμα

```
glClearColor(0.0, 0.0, 0.0, 0.0);
glClear(GL_COLOR_BUFFER_BIT);
```

•Ορίζουμε το χρώμα καθαρισμού στην αρχή του προγράμματός μας και κατόπιν καθαρίζουμε τους ενταμιευτές όσο συχνά χρειάζεται.

# Καθαρισμός πολλαπλών ενταμιευτών

•Η εντολή glClear επιτρέπει επίσης τον καθορισμό πολλαπλών ενταμιευτών με μία μόνο κλήση της.

•Δίνουμε ως παραμέτρους πολλαπλούς ενταμιευτές διαχωρισμένους με τελεστές OR ( | ).

•Π.χ. για τον ταυτόχρονο καθαρισμό του ενταμιευτή χρωματικών τιμών (color buffer) και του ενταμιευτή τιμών βάθους (depth buffer) δίνουμε:

```
glClearColor(0.0, 0.0, 0.0, 0.0);
glClearDepth(0.0);
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
```

# Καθορισμός χρωμάτων

•Κάθε φορά που δίνουμε εντολή σχεδιάσης ενός συγκεκριμένου γεωμετρικού σχήματος, του αποδίδουμε το τρέχον χρώμα σχεδίασης

•Το χρώμα σχεδίασης, ως μεταβλητή κατάστασης, διατηρεί την τελευταία τιμή του.

•Για να καθορίσουμε ένα χρώμα, χρησιμοποιούμε την εντολή glColor3{ub,f,d}.

void glColor3ub(GLubyte red, GLubyte green, GLubyte blue); 0<r,g,b<255
void glColor3f(GLfloat red, GLfloat green, GLfloat blue); 0<r,g,b<1
void glColor3d(GLdouble red, GLdouble green, GLdouble blue); 0<r,g,b<1

# Καθορισμός κορυφών

•Στην OpenGL, όλα τα γεωμετρικά σχήματα περιγράφονται δηλώνοντας τις κορυφές τους.

•Για τον καθορισμό κάθε μίας κορυφής χρησιμοποιούμε την εντολή glVertex*.

•void glVertex{234}{sifd}[v](TYPE coords);

•Με τις παραλλαγές της εντολής glVertex, μπορούμε να δώσουμε από δύο (x, y) μέχρι και τέσσερις συντεταγμένες (x, y, z, w) για μία κορυφή.

# Παραδείγματα καθορισμού κορυφών

glVertex2s(2,3);

Δήλωση σημείου με ακέραιες 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 );

Δήλωση σημείου *( 1.15, 0.5, -1.1 )* σε ομογενείς συντεταγμένες *( w=2 )*

GLdouble dvect[3] = {5,9,1};

glVertex3dv(dvect);

(Δήλωση σημείου που οι τιμές του ορίζονται στο μητρώο dvect)

# Η δομή glBegin/glEnd

•Ο ορισμός των κορυφών κάθε γεωμετρικού σχήματος περικλείεται μεταξύ δύο εντολών, των glBegin και glEnd.

```
void glBegin(GLEnum mode);
void glEnd();
```

•Μεταξύ των εντολών glBegin και glEnd περικλείονται εντολές δήλωσης σημείων ορίζουν ένα γεωμετρικό σχήμα ή μία ομάδα γεωμετρικών σχημάτων.

•Το είδος του γεωμετρικού σχήματος που ορίζεται, εξαρτάται από την συμβολική σταθερά mode που δίνουμε ως παράμετρο στην εντολή glBegin.

# Σχεδίαση μεμονωμένων σημείων (GL_POINTS)

- Δίνουμε ως όρισμα στην glBegin τη σταθερά GL_POINTS.

- Παράδειγμα

```
glBegin(GL_POINTS);
glVertex2i(10,10);
glVertex2i(20,10);
glEnd();
```

Ορισμός δύο σημείων με συντεταγμένες *(10,10)* και *(20,10)*.

# Πάχος σημείων

•Για τη ρύθμιση του μεγέθους ενός σημείου, χρησιμοποιούμε την εντολή glPointSize:

void glPointSize(GLfloat size);

size: το πλάτος του σημείου σε pixels

• Η προκαθορισμένη τιμή του πάχπυς γραμμής είναι 1.

•Το πάχος σημείων είναι μεταβλητή κατάστασης, άρα διατηρεί την τελευταία του τιμή.

# Σχεδίαση γραμμών

•Οι γραμμές χαράσσονται ορίζοντας τις συντεταγμένες των άκρων τους.

Οι εντολές σχεδιάσης γραμμών μας επιτρέπουν να σχεδιάσουμε

α) Ευθύγραμμα τμήματα

β) Αλυσίδες ευθυγράμμων τμημάτων

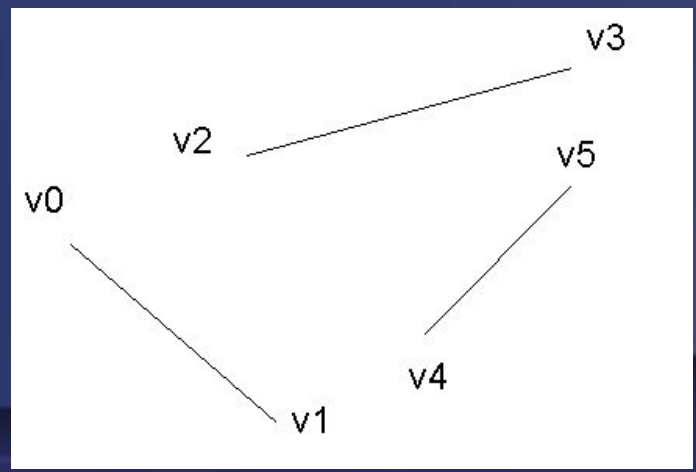
γ) Βρόχους ευθυγράμμων τμημάτων

# Ευθύγραμμα τμήματα (GL_LINES)

glBegin(GL_LINES);

- Τα δοθέντα σημεία ορίζουν ανά ζεύγη ευθύγραμμα τμήματα.
- Όταν ο αριθμός των σημείων είναι περιττός, το τελευταίο σημείο αγνοείται.

```
glBegin(GL_LINES);
glVertex*(v0);
glVertex*(v1);
glVertex*(v2);
glVertex*(v3);
glVertex*(v4);
glVertex*(v5);
glEnd();
```

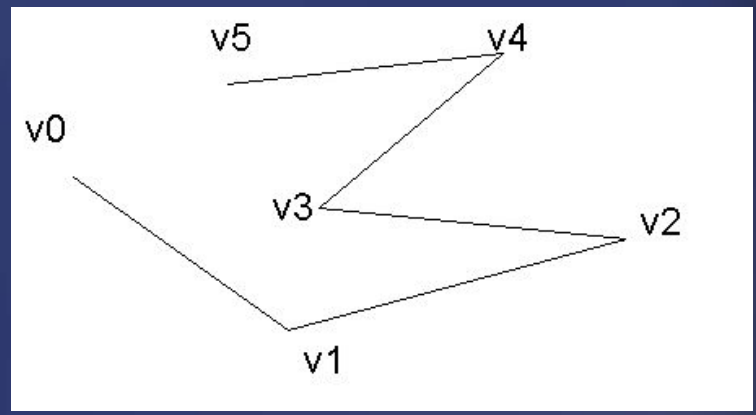


# Αλυσίδα ευθυγράμμων τμημάτων

glBegin(GL_LINE_STRIP);

Τα σημεία ορίζουν διαδοχικά ευθύγραμμα τμήματα.

```
glBegin(GL_LINE_STRIP);
glVertex*(v0);
glVertex*(v1);
glVertex*(v2);
glVertex*(v3);
glVertex*(v4);
glVertex*(v5);
glEnd();
```

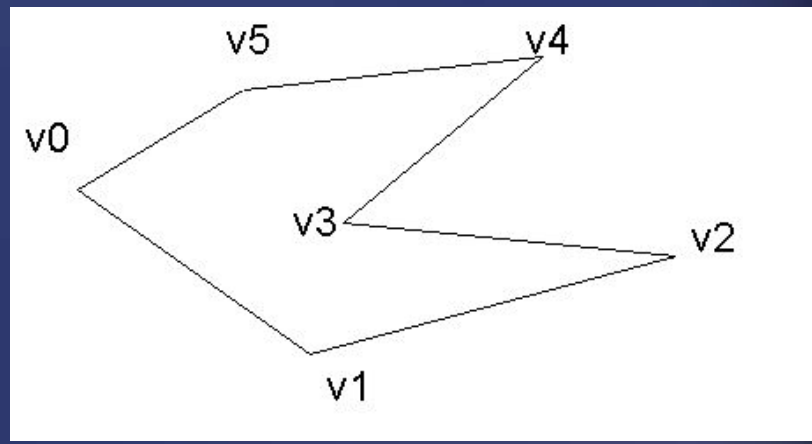


# Βρόχος ευθυγράμμων τμημάτων

glBegin(GL_LINE_LOOP):

Τα σημεία ορίζουν διαδοχικά ευθύγραμμα τμήματα. Το τελευταίο σημείο ενώνεται με το αρχικό.

```
glBegin(GL_LINE_LOOP);
glVertex*(v0);
glVertex*(v1);
glVertex*(v2);
glVertex*(v3);
glVertex*(v4);
glVertex*(v5);
glEnd();
```



# Ιδιότητες γραμμών

α) Πάχος γραμμών

Τροποποιείται με τη χρήση της εντολής glLineWidth:

```
void glLineWidth(GLfloat width);
```

width: το πάχος της γραμμής σε pixels

β) Διάστιξη γραμμών

Χρήσιμη για τη σχεδιάση διακεκομμένων γραμμών

Η διάστιξη γραμμών ενεργοποιείται δίνοντας την παράμετρο GL_LINE_STIPPLE στην εντολή glEnable( ):

```
glEnable(GL_LINE_STIPPLE);
```

# Ρύθμιση διάστιξης γραμμών

void glLineStipple (GLint factor, GLushort pattern);

Το όρισμα pattern καθορίζει το μοτίβο των ευθειών. Δίνεται σε δεκαεξαδική μορφή (πχ 0x0A0A).

Αναπαράσταση δεκαεξαδικών συμβόλων στο δυαδικό σύστημα:

| | | | |
|---|---|---|---|
| 0 = 0000 | 4 = 0100 | 8 = 1000 | C = 1100 |
| 1 = 0001 | 5 = 0101 | 9 = 1001 | D = 1101 |
| 2 = 0010 | 6 = 0110 | A = 1010 | E = 1110 |
| 3 = 0011 | 7 = 0111 | B = 1011 | F = 1111 |

# Σχηματισμός μοτίβου

•Αναπαριστούμε την αριθμητική τιμή του ορίσματος pattern σε δυαδική μορφή

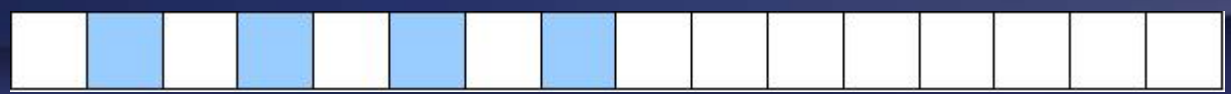
Π.χ. ο δεκαεξαδικός αριθμός 0x00AA αναπαρίσταται ως

0 0 A A

0000 0000 1010 1010 ⟶ 0000000010101010

•Οι θέσεις των μονάδων στη δυαδική ακολουθία καθορίζουν τα pixels της γραμμής που θα σχεδιαστούν. Οι θέσεις των μηδενικών καθορίζουν τα pixels που θα παραμείνουν κενά.

Π.χ. Ο δεκαεξαδικός 0x00AA (για factor=1) ορίζει το μοτίβο:



•Το μοτίβο ξεκινάει από το λιγότερο σημαντικό ψηφίο της δυαδικής ακολουθίας.

# Κλιμάκωση μοτίβου

void glLineStipple (GLint factor, GLushort pattern);

factor: κλιμακώνει το μοτίβο. Αναπαράγει κάθε ψηφίο της δυαδικής ακολουθίας όσες φορές δηλώνει η τιμή του.

Πχ για factor ίσο με 2 και pattern 0x00AA

glLineStipple(2, 0x00AA);

Παράγουμε την ακόλουθη διάστιξη

•Η διάστιξη γραμμών, ως παράμετρος κατάστασης, παραμένει ενεργή.

# Τέλος ενότητας!