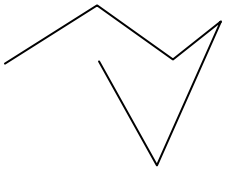
ΣΧΕΔΙΑΣΗ ΣΤΟ ΕΠΙΠΕΔΟ (2D)

Πρωταρχικά Αντικείμενα Γραφικών – (Graphics Primitives)

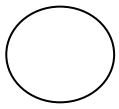
Η σχεδίαση ενός αντικειμένου γίνεται με την βοήθεια σχεδίασης «πρωταρχικών αντικειμένων» που υποστηρίζονται άμεσα από τη βιβλιοθήκη γραφικών.

- 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 το χρήστη.



primitive LineStrip



user's defined Circle

Σχ. 2-1

- Για τον ορισμό και την σχεδίαση των αντικειμένων γραφικών χρησιμοποιείται η έννοια του "**vertex**" (κορυφή γεωμετρικού αντικειμένου) το οποίο δηλώνει ένα «ουσιώδες» σημείο για την περιγραφή του αντικειμένου

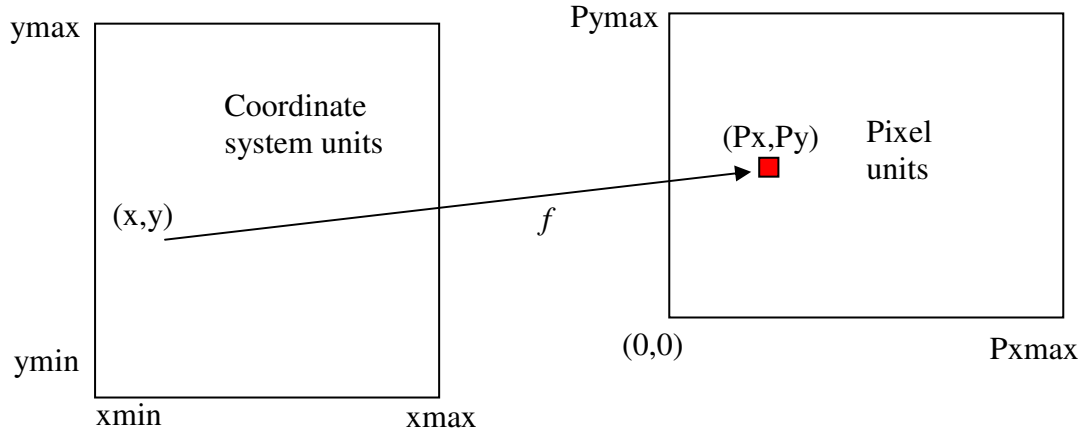
```
glVertex*(  )
```

- Η OpenGl συμπεριλαμβάνει συλλογή εντολών για άμεση σχεδίαση πιο πολύπλοκων αντικειμένων
  Π.χ. `glRect, gluSphere, gluCylinder, gluDisk`

1. Σύστημα συντεταγμένων

Μια `κορυφή` (vertex) δηλώνεται στο επίπεδο με τις συντεταγμένες *x,y* που παίρνουν τιμές με βάση το *σύστημα συντεταγμένων* που ορίζεται με την εντολή

```
gluOrtho2D(x_min, x_max, y_min, y_max)
```



World scene domain (A)

Window (B)

Σχ. 2-2

Ένα σημείο *r=(x,y)* του χώρου σχεδίασης απεικονίζεται στο pixel P=(Px,Py) μέσω μιας γραμμικής απεικόνισης *f:A→B*.

$$P = (r - r_{\min}) \frac{P_{\max}}{r_{\max} - r_{\min}}$$

όπου το P_{max}=(Pxmax,Pymax) καθορίζεται με την εντολή αρχικοποίησης

```
glutInitWindowSize(size_x, size_y)
```

με `Pxmax=size_x, Pymax=size_y`

2. Ιδιότητες – εφαρμογή αλγορίθμων - αναπαράσταση

- Οι ιδιότητες των αντικειμένων (attributes) δηλώνονται με αντίστοιχες εντολές. Από τη στιγμή που θα δηλωθούν στο πρόγραμμα (με την εκτέλεση της αντίστοιχης εντολής), παραμένουν ενεργές έως ότου αλλαχθούν. Τα attributes δηλώνονται με έναν ή περισσότερους αριθμούς (ακέραιοι ή πραγματικοί)

- Κάθε αντικείμενο γραφικών μπορεί να υποστεί μια διαδικασία επεξεργασίας πριν την παρουσίασή του στην οθόνη. Μια τέτοια επεξεργασία γίνεται μέσω κάποιου αλγορίθμου ο οποίος ενεργοποιείται μέσω της συνάρτησης ελέγχου

  ```
  glEnable(arg_int)
  ```

  Το όρισμα `arg_int` της συνάρτησης είναι ένας ακέραιος που αντιστοιχεί στον αλγόριθμο που πρόκειται να ενεργοποιηθεί. Η απενεργοποίηση του αλγορίθμου γίνεται με την εντολή `glDisable`.

Το Μπλοκ Σχεδίασης (Rendering)

Η σχεδίαση ενός ή περισσοτέρων αντικειμένων ταυτόχρονα γίνεται μέσα στο block σχεδίασης

glBegin(*GL_type*)

decleration_of_vertices

glEnd()

GL_type : : Το είδος γραφικού που θα σχεδιαστεί

decleration_of_vertices : ένα σύνολο από κορυφές (vertices) τα οποία δηλώνουν το αντικείμενο ή τα αντικείμενα προς σχεδίαση

3. Στοιχειώδη αντικείμενα γραφικών

GL_type = GL_POINTS : Σχεδίαση σημείων. Ένα σημείο ορίζεται από ένα μοναδικό vertex και καταλαμβάνει στην οθόνη χώρο ίσο με ένα pixel. Άρα κάθε vertex στο block σχεδίασης ορίζει και ένα σημείο.

ΙΔΙΟΤΗΤΕΣ ΣΗΜΕΙΩΝ

`glColor*(....)` : χρώμα
`glPointSize(GLfloat x)` : μέγεθος

Επεξεργασία ομαλοποίησης : `glEnable(GL_POINT_SMOOTH)` :
(βλ κώδικα code20.cpp)

GL_type = GL_LINES : Σχεδίαση ευθυγράμμων τμημάτων (ή, απλά, γραμμή). Μια γραμμή ορίζεται από δύο vertices. Άρα, κάθε ζεύγος από vertices ορίζει και μια γραμμή.

πχ

```
glBegin(GL_POINTS);
     glVertex2i(10,10);
     glVertex2i(30,80);
     glVertex2i(90,10);
     glVertex2i(40,90);
     glVertex2i(90,110);
glEnd();
```

```
glBegin(GL_LINES);
     glVertex2i(10,10);
     glVertex2i(30,80);
     glVertex2i(90,10);
     glVertex2i(40,90);
     glVertex2i(90,110);
glEnd();
```

ΙΔΙΟΤΗΤΕΣ ΓΡΑΜΜΩΝ

`glColor*(....)` : χρώμα

`glLineWidth(GLfloat)` : πάχος

`glLineStipple(GLint factor, GLushort pattern)*` : στυλ γραμμής

*** Απαιτείται η ενεργοποίηση : `glEnable(GL_LINE_STIPPLE);`**
Π.χ. `glLineStipple(n, 255)` : Διακεκομμένη γραμμή σε κλίμακα *n*.

- Ο αλγόριθμος "antialiasing" μπορεί να ενεργοποιηθεί με την

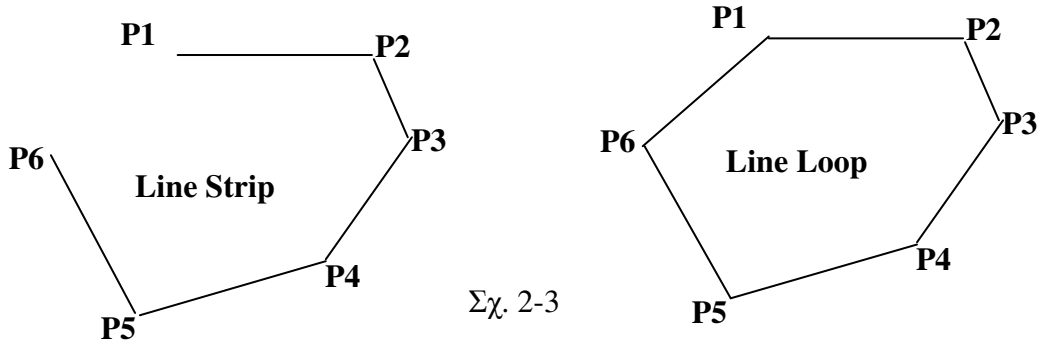
`glEnable(GL_LINE_SMOOTH);`

(βλ κώδικα code21.cpp)

4α. Σύνθετοι τύποι γραμμών

GL_LINE_STRIP : Σχεδιάζει συνεχόμενες γραμμές από κορυφή σε κορυφή.

GL_LINE_LOOP : Όπως η GL_LINE_STRIP αλλά προστίθεται και μια επιπλέον γραμμή που ενώνει την τελευταία κορυφή με την αρχική.



Σχ. 2-3

4β. Πολύγωνα

GL_TRIANGLES : Χρησιμοποιεί τρεις συνεχόμενες κορυφές για τη σχεδίαση ενός τριγώνου. Δηλώνοντας *N* κορυφές σχεδιάζονται [*N*/3] τρίγωνα.

GL_QUADS: Χρησιμοποιεί τέσσερις συνεχόμενες κορυφές για τη σχεδίαση ενός τετράπλευρου (όχι απαραιτήτως κανονικού). Δηλώνοντας *N* κορυφές σχεδιάζονται [*N*/4] τετράπλευρα.

GL_POLYGON: Χρησιμοποιεί όλες τις συνεχόμενες κορυφές που δίνονται για τη σχεδίαση ενός πολυγώνου (όχι απαραιτήτως κανονικού).

- Κατάσταση Σχεδίασης

```
glPolygonMode(GLenum face,  GLenum mode )
```

***face*:**

GL_FRONT for front-facing polygons,
GL_BACK for back-facing polygons, or
GL_FRONT_AND_BACK for front- and back-facing polygons.

***mode*:**

GL_POINT : κορυφές του πολυγώνου
GL_LINE : πλευρές του πολυγώνου
GL_FILL : εσωτερικό του πολυγώνου (default value)

Ορισμός Πρόσοψης

Η φορά με την οποία σχεδιάζεται ένα πολύγωνο είναι είτε με τη φορά των δεικτών του ρολογιού ή αντίστροφα ("clockwise" ή "Counterclockwise"). Το ποια θα είναι η μπροστινή πρόσοψη (front face) καθορίζεται από την εντολή-ιδιότητα

`glFrontFace(GL_CW)` ή `glFrontFace(GL_CCW)`.

Στην 1η περίπτωση μπροστινή πρόσοψη είναι αυτή στην οποία βλέπουμε το πολύγωνο να γράφεται με διεύθυνση "clockwise", ενώ στη δεύτερη ισχύει το αντίθετο (Default).

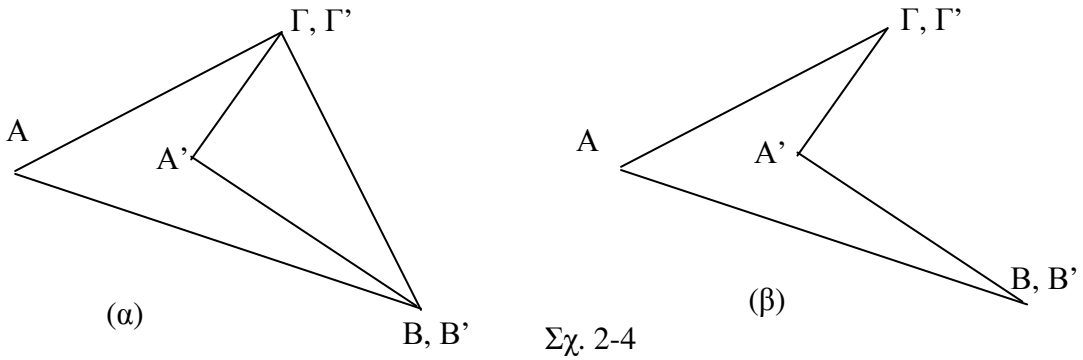
«Συνοριακές» (ή μη) **πλευρές - κορυφές**

Στην κατάσταση σχεδίασης πολυγώνου "`GL_FILL`" οι πλευρές και οι κορυφές δεν σχεδιάζονται (γεμίζει μόνο με το τρέχον χρώμα το εσωτερικό του πολυγώνου). Αντίθετα στη κατάσταση `GL_LINE` ή `GL_POINT` σχεδιάζονται όλες οι πλευρές ή όλα τα σημεία. Αυτό σχετίζεται με την ιδιότητα

`glEdgeFlag ( B ), B=TRUE ή FALSE`

Αν η `glEdgeFlag` είναι `TRUE` (default value) τότε η κορυφή ή η πλευρά χαρακτηρίζεται ως «συνοριακή» και σχεδιάζεται διαφορετικά δεν σχεδιάζεται.

Στο σχήμα 2-4(α) έχουμε δύο τρίγωνα ΑΒΓ και Α'Β'Γ' (με ΒΓ==Β'Γ') με όλες τις πλευρές συνοριακές. Στο σχήμα 2-4(β) οι πλευρές ΒΓ και Β'Γ' έχουν χαρακτηριστεί με την ιδιότητα `glEdgeFlag(FALSE)` και δεν σχεδιάζονται.



Σχ. 2-4

Σημείωση : το πολύγωνο του σχήματος 2-4(β) μπορεί να σχεδιαστεί ευκολότερα ως ένα **μη-κυρτό** τετράεδρο με κορυφές τα Α, Β, Α' και Γ. Όμως στην σχεδίαση τα μη-κυρτά πολύγωνα πρέπει να αποφεύγονται γιατί πολλές ιδιότητες δεν μπορούν να αποδοθούν σωστά.

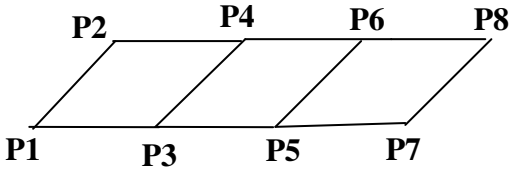
(βλ κώδικα code22.cpp)

4γ. Συνεχόμενη σχεδίαση (strips and fans)

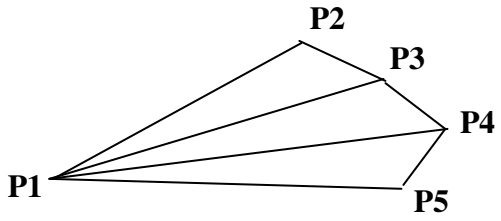
GL_TRIANGLE_STRIP
Σχεδιάζει μια συνδεμένη ομάδα τριγώνων σχηματίζοντας μια ζώνη. Το πρώτο τρίγωνο σχεδιάζεται με τις κορυφές 1,2 και3, το δεύτερο τρίγωνο με τις κορυφές 2,3 και 4, κ.ο.κ.

GL_TRIANGLE_FAN
Σχεδιάζει μια συνδεμένη ομάδα τριγώνων με κοινή κορυφή. Η πρώτη κορυφή αποτελεί την κοινή κορυφή. Έτσι το πρώτο τρίγωνο έχει κορυφές την 1, 2 και 3, το δεύτερο τρίγωνο έχει τις 1, 3 και 4, κ.ο.κ.

GL_QUAD_STRIP
Σχεδιάζει μια συνδεμένη ομάδα τετράεδρων σχηματίζοντας μια ζώνη. Το πρώτο τετράεδρο σχεδιάζεται με τις κορυφές 1,2,4 και3, το δεύτερο με τις κορυφές 3,4, 6 και 5, κ.ο.κ. Γενικά το *n* τετράεδρο σχηματίζεται από τις κορυφές $2n - 1$, $2n$, $2n + 2$, και $2n + 1$.



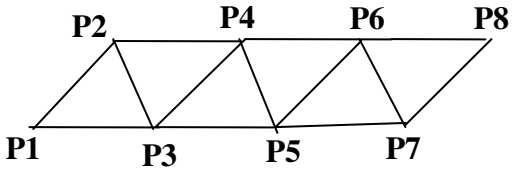
Quadrilateral strip



Triangle Fan

Σχ. 2-5

Triangle Strip



(βλ κώδικα code23.cpp)

5. Χρωματισμός Αντικειμένων

Το χρώμα (RGB) ορίζεται με την συνάρτηση

`glColor3*(R,G,B)` ή `glColor4*(R,G,B,alpha)`

με ορίσματα στο διάστημα [0,1]. Όταν δίνεται η παραπάνω εντολή, το δηλωμένο χρώμα συνδέεται με τις επόμενες κορυφές (vertices) που θα δοθούν μέχρι να αλλάξει το χρώμα με την χρησιμοποίηση μιας άλλης εντολής glColor.
Μπορεί να επιλεγεί ένα από τα δύο βασικά μοντέλα χρωματισμού : το `GL_FLAT` ή `GL_SMOOTH` με την εντολή

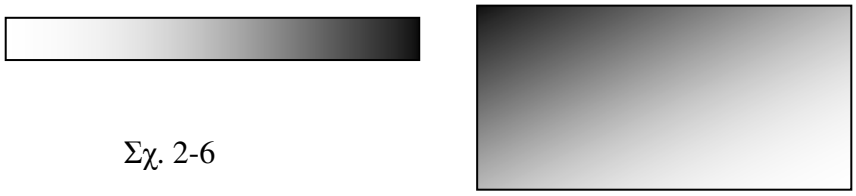
`glShadeModel(GL_FLAT)` ή `glShadeModel(GL_SMOOTH)`

Στο μοντέλο `GL_FLAT` χρησιμοποιείτε το σταθερό (solid) χρώμα για τη σχεδίαση του αντικειμένου που ακολουθεί το οποίο επιλέγεται από κάποια κορυφή ανάλογα με το αντικείμενο που σχεδιάζεται πχ

- μια γραμμή σχεδιάζεται με το χρώμα της τελευταίας κορυφής,
- ένα τρίγωνο ή τετράπλευρο σχεδιάζεται με το χρώμα της τελευταίας κορυφής,
- ένα πολύγωνο σχεδιάζεται με το χρώμα της πρώτης κορυφής.

Στο μοντέλο `GL_SMOOTH` γίνεται ομαλοποίηση του χρώματος από κορυφή σε κορυφή με «γραμμική μεταβολή»

(βλ κώδικα code25.cpp)



Σχ. 2-6

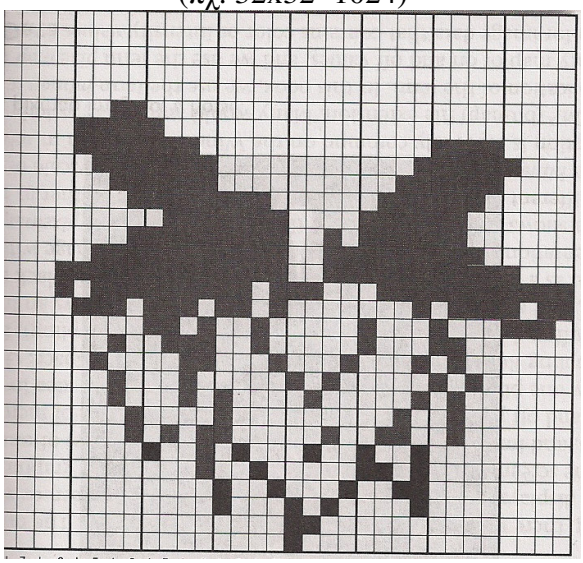
(βλ κώδικα code25.cpp)

BITMAPS

Bitmap ονομάζεται ένας πίνακας με *L* στοιχεία τα οποία μπορούν να έχουν την τιμή 0 ή 1. Εν' γένει θεωρούμε τα παραπάνω στοιχεία σε διδιάστατη διάταξη ως πίνακα *N*x*M* (=L).

Ένα Bitmap αναπαρίσταται γραφικά ως ένα ορθογώνιο παραλληλεπίπεδο από pixels που το καθένα μπορούμε να δώσουμε ένα από δύο δυνατά χρώματα.

(πχ. 32x32=1024)



Ένα Bitmap στον υπολογιστή σχηματίζεται από ένα πίνακα από bytes (8bit) τα οποία μπορούν (πχ) να αντιπροσωπεύσουν τους ακεραίους από 0 έως 255.

Πχ 13 = 00001101 =

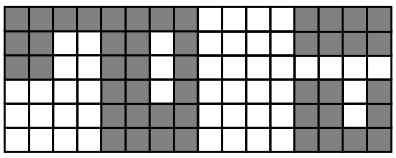
Πχ 205 = 11001101 =

Πχ 15 = 00001111 =

GLubyte Mybitmap[12]={127,127,127,13,13,13, 205,0,205,127,255,127}

4x24 pixels (4x3 bytes)

6x16 pixels (6x2 bytes)



* Η διαχείριση των bitmaps από τον Η/Υ γίνεται με την αποθήκευσή τους ως memory blocks γι'αυτό στην πράξη έχουν πλευρές (πλατος και ύψος) που είναι δυνάμεις του 2. Αν δημιουργήσουμε ένα bitmap με μέγεθος 500x800 αυτό θα καταλάβει στη μνήμη μέγεθος 512x1024 (τα επιπλέον 65536 bytes μνήμης θα δεσμευτούν άσκοπα).

Η OpenGL διαχειρίζεται bitmaps m(32x32) (=πίνακας με m*128 Bytes) (?).

Απεικόνιση Bitmap στο παράθυρο σχεδίασης

`glRasterPos*(X_pos, Y_pos) :`
Ορίζεται η τρέχουσα θέση όπου θα εμφανιστεί το bitmap (η κάτω αριστερή γωνία). Τα `X_pos, Y_pos` δίνονται στο σύστημα συντεταγμένων που έχει οριστεί.

`glWindowPos*( X_pos, Y_pos) :`
Ορίζεται η τρέχουσα θέση όπου θα εμφανιστεί το bitmap (η κάτω αριστερή γωνία). Τα `X_pos, Y_pos` δίνονται σε pixels, δηλαδή στις φυσικές συντεταγμένες του παραθύρου.

`glBitmap(width, height, x_origin, y_origin, x_move, y_move, *bitmap):` Σχεδιάζει το bitmap στο παράθυρο με το τρέχον χρώμα.

`width, height` : πλάτος και ύψος του bitmap σε pixels
`x_origin, y_origin` : Ορίζει την «αρχή» του bitmap

`x_move, y_move` : ορίζει την μετατόπιση του RasterPos που θα πάρει μέρος μετά την σχεδίαση του bitmap.

`*bitmap` : Δείκτης στον πίνακα που περιέχει το bitmap.

(βλ κώδικα code28.cpp)

Bitmap Fonts

Η OpenGl υποστηρίζει σύνολα από bitmaps χαρακτήρες

```
glutBitmapCharacter(font, char)
```

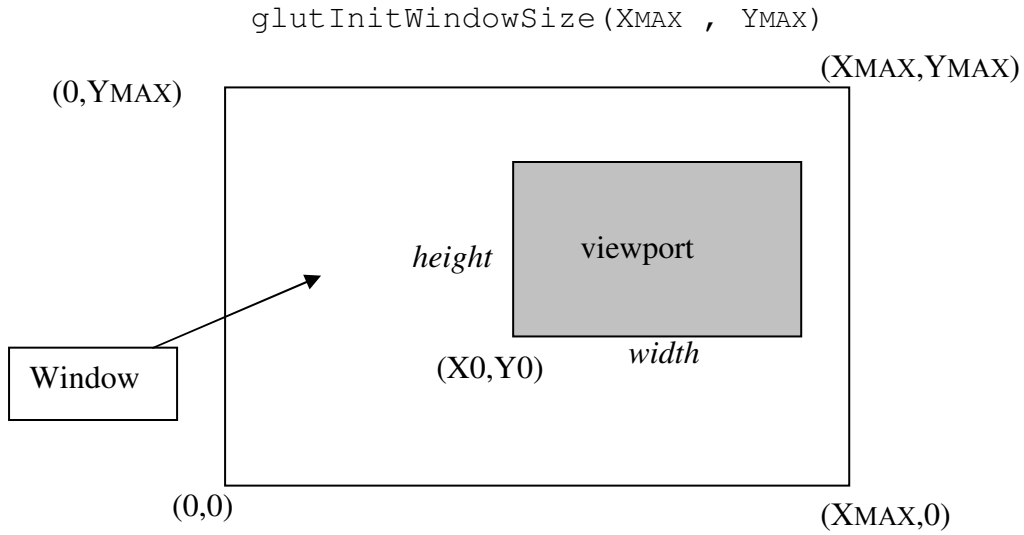
Ως fonts μπορούν να χρησιμοποιηθούν :

- GLUT_BITMAP_8_BY_13
- GLUT_BITMAP_9_BY_15
- GLUT_BITMAP_TIMES_ROMAN_10
- GLUT_BITMAP_TIMES_ROMAN_24
- GLUT_BITMAP_HELVETICA_10
- GLUT_BITMAP_HELVETICA_12
- GLUT_BITMAP_HELVETICA_18

(βλ κώδικα code29.cpp)

Viewports, Scissors και Επανασχεδίαση

* Η σχεδίαση αντικειμένων, ο ορισμός του συστήματος συντεταγμένων καθώς και οι γεωμετρικοί μετασχηματισμοί λαμβάνουν χώρα σε μια **παραλληλόγραμμη περιοχή** του παραθύρου σχεδίασης που ονομάζεται **viewport.**
* αν δεν έχει οριστεί κάποιο viewport τότε θεωρείται όλο το παράθυρο σχεδίασης που ορίζεται από την



Ορισμός viewport >> χώρος σχεδίασης

```
glViewport(X0, Y0, width, height)
```

`X0,Y0` μεταβλητές τύπου `GLint`, (pixels)
`width,height` μεταβλητές τύπου `GLsizei` (pixels)

** Μπορούμε να κάνουμε χρήση του Viewport για την δημιουργία αντιγράφων

Ορισμός Ψαλιδιού (Scissor) >> περιοχή σάρωσης (rendering)

```
glScissor(X0, Y0, width, height)
```

`X0,Y0` μεταβλητές τύπου `GLint`, (pixels)
`width,height` μεταβλητές τύπου `GLsizei` (pixels)

* Απαιτείται ενεργοποίηση αλγορίθμου : `glEnable(GL_SCISSOR_TEST)`

(βλ κώδικα code26.cpp)

>> Το Scissor ορίζει μια παραλληλόγραμμη περιοχή. Μπορεί να οριστεί όμως ένα οποιοδήποτε διάτρητο πρότυπο (στάμπο) με τη χρήση του **Stencil Buffer**.

Επανασχεδίαση

1. Κατά απαίτηση του χρήστη

```
glutPostRedisplay();
```

2. κατά την μεταβολή μεγέθους

Δήλωση callback function :

```
glutReshapeFunc(MyReshape);
```

Ορισμός callback function :

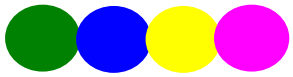
```
MyReshape(GLsizei width, GLsizei height) {..}
```

(βλ κώδικα code27.cpp)

Ασκήσεις

2-1. Σχεδιάστε τα παρακάτω αντικείμενα χρησιμοποιώντας σημεία

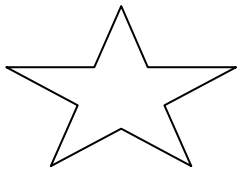
α) 

β) 

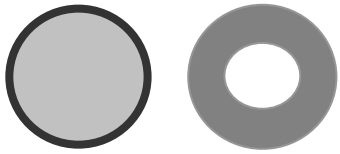
2-2. Να βρεθεί η αντίστροφη απεικόνιση της f (βλ. σχήμα 2-2): Για δοσμένο pixel (px,py) να βρίσκονται οι συντεταγμένες (x,y).
Να χρησιμοποιηθεί η παραπάνω απεικόνιση για να σχεδιαστούν (πχ με κοκκινο χρώμα) όλα τα pixels του ενός παραθύρου με σύστημα συντεταγμένων xmin=-1, xmax=1, ymin=0, ymax=1.

2-3. Με τη χρήση της GL_LINE_LOOP σχεδιάστε ένα κανονικό πολύγωνο με *n* γωνίες. Ο κώδικας να πραγματοποιηθεί σε μια ξεχωριστή συνάρτηση "void myisopolygon(unsigned int n)".

2-4. Σχεδιάστε
Το αστέρι του διπλανού σχήματος χρησιμοποιώντας πέντε τρίγωνα α) χωρίς γέμισμα (μόνο γραμμές) β) με γέμισμα (χρώμα κόκκινο το μπροστά, μπλέ το πίσω)



2-5. Σχεδιάστε , όπως φαίνεται στο διπλανό σχήμα, έναν κυκλικό δίσκο (με περιφέρεια) και έναν δακτύλιο (Ο δακτύλιος στη μέση είναι τρύπιος!).



2-6. Σχεδιάστε στο ίδιο σκηνικό αρκετά αστέρια της άσκησης 2-4 διαφορετικού μεγέθους αλλάζοντας "viewports".

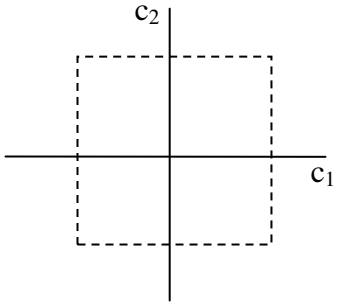
*2-7. Να κατασκευαστεί ένα πρόγραμμα που να διαβάζει ένα αρχείο με (x,y) σημεία και να σχεδιάζει την γραφική τους παράσταση. Να διαθέτει τις παρακάτω δυνατότητες.
α) αυτόματος υπολογισμός των ορίων του συστήματος συντεταγμένων.
β) Τα σημεία να ενώνονται ή όχι με γραμμές (πχ JoinPoints=True or False)
γ) Σχεδιασμός αξόνων με κλίμακα και ετικέτες

*2-8α. Σχεδιάστε την λεκάνη έλξης του συνόλου Mandelbrot σύμφωνα με τα παρακάτω:

Θεωρούμε στο επίπεδο O*xy* την απεικόνιση

$$x' \to x^2 - y^2 + c_1, \qquad y' \to 2xy + c_2$$

Για κάποιο ζεύγος σταθερών (c_1,c_2) εκτελούμε επαναληπτικά την απεικόνιση ξεκινώντας με αρχική συνθήκη την (x,y)=(0,0). Αν για N επαναλήψεις (N αρκετά μεγάλο) τα σημεία της απεικόνισης δεν ξεφεύγουν από τον τόπο $T = \{(x, y); |x| < 2, |y| < 2\}$ τότε θεωρούμε ότι οι παραπάνω τιμές των (c_1,c_2) ανήκουν στο ελκτικό σύνολο του Mandelbrot.



Θεωρούμε τον επίπεδο τόπο Oc_1c_2 με

$$c1\min \le c1 < c1\max \quad \text{και} \quad c2\min \le c2 < c2\max$$

και ένα grid XSIZE x YSIZE pixels μέσα στον παραπάνω τόπο. Έτσι κάθε pixel του grid αντιστοιχεί ένα ζεύγος τιμών (c_1,c_2). Αν το ζεύγος αυτό ανήκει στο σύνολο του Mandelbrot τότε ζωγραφίζουμε το αντίστοιχο pixel.

II) Χρησιμοποιείστε τον κώδικα 2-8a και χρησιμοποιώντας "RGB Color" και αποδώστε το σύνολο Mandelbrot με χρώμα ανάλογο του βήματος στο οποίο παρατηρείται διαφυγή (escape).

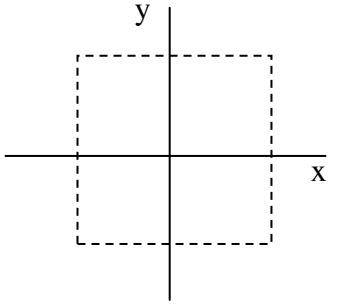
---

*2-8b. Σχεδιάστε την λεκάνη έλξης του συνόλου Julia σύμφωνα με τα παρακάτω:

Θεωρούμε στο επίπεδο O*xy* την απεικόνιση

$$x' \to x^2 - y^2 + c_1, \qquad y' \to 2xy + c_2$$

όπου c_1, c_2 σταθερές. (x_o,y_o)
Για κάθε σημείο (x_o,y_o) εκτελούμε επαναληπτικά την απεικόνιση. Αν για N επαναλήψεις (N αρκετά μεγάλο) τα σημεία της απεικόνισης δεν ξεφεύγουν από τον τόπο $T = \{(x, y); |x| < 2, |y| < 2\}$ τότε θεωρούμε ότι το σημείο (x_o,y_o) ανήκει στο ελκτικό σύνολο του Julia.



Θεωρούμε τον επίπεδο τόπο O*xy* με

$$X\min \le x < Xm\max \quad \text{και} \quad Y\min \le y < Y\max$$

και ένα grid XSIZE x YSIZE pixels μέσα στον παραπάνω τόπο. Έτσι κάθε pixel του grid αντιστοιχεί ένα σημείο (x_o,y_o). Αν το ζεύγος αυτό ανήκει στο σύνολο του Julia τότε ζωγραφίζουμε το αντίστοιχο pixel.

II) Χρησιμοποιείστε τον κώδικα 2-8b και χρησιμοποιώντας "RGB Color" και αποδώστε το σύνολο Julia με χρώμα ανάλογο του βήματος στο οποίο παρατηρείται διαφυγή (escape).
(δοκιμαστική τιμή σταθερών c1=-0.194, c2=0.6557)