ΤΕΙ Θεσσαλονίκης
Τμήμα Πληροφορικής

Γραφικά Υπολογιστών: OpenGL

Πασχάλης Ράπτης
http://aetos.it.teithe.gr/~praptis
praptis@it.teithe.gr

Περιεχόμενα

- Τι είναι η OpenGL;
- Μοντέλα αντικειμένων (object modeling)
- Φωτισμός και σκίαση (lighting και shading)
- Θέαση του Η/Υ (computer viewing)
- Απόδοση (rendering)
- Χαρτογράφηση υφής (texture mapping)

Τι είναι η OpenGL;

Η OpenGL είναι μια βιβλιοθήκη 2D/3D γραφικών ανεξάρτητη συσκευής (desktop, laptop, mobile phone) και ανεξάρτητη Λειτουργικού Συστήματος (Windows, Linux/Unix, Mac OS X)

Η OpenGL είναι μια διεπαφή API (*application programming interface*)

-- Συλλογή από σταθερές, τύπους δεδομένων και συναρτήσεις

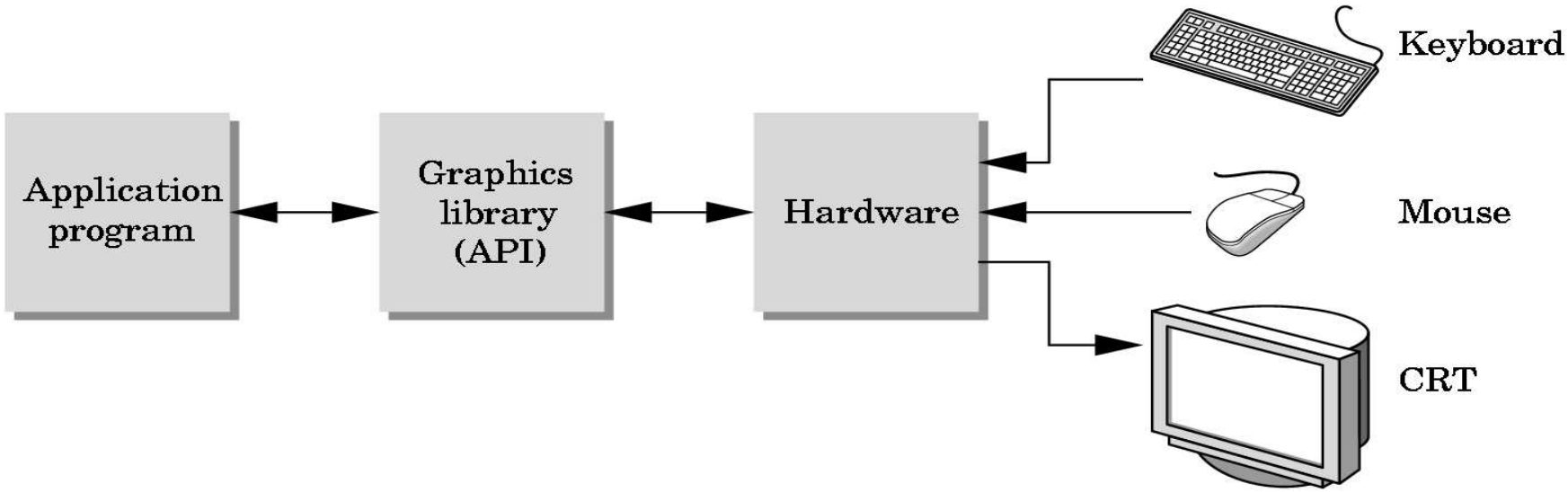
OpenGL

Προγραμματιστής βλέπει το σύστημα γραφικών μέσω των βιβλιοθήκων

-- OpenGL

-- Direct X (Microsoft)

-- Java 3D



OpenGL

- Η Silicon Graphics (SGI), έφερε την επανάσταση στα PC-γραφικά με την ανάπτυξη ενός γραφικού συστήματος που επιτρέπει την πρόσβαση στα graphics hardware με την υλοποίηση pipeline αγωγού σε (1982)
- Για να χρησιμοποιήσουν αυτό το σύστημα οι προγραμματιστές εφαρμογών έκαναν χρήση μιας βιβλιοθήκης με το όνομα GL (Graphics Library)
- Με την GL, ήταν σχετικά απλός ο προγραμματισμός διαδραστικών τριδιάστατων (3D) εφαρμογών

OpenGL

- Η επιτυχία της GL οδήγησε στην OpenGL το 1992, μια API βιβλιοθήκη ανεξάρτητη πλατφόρμας που ήταν:
  - -- Εύκολη στην χρήση
  - -- Συνεργάζεται στενά με το hardware με πολύ καλά αποτελέσματα απόδοσης (performance)
  - -- Εστιάζει στην απόδοση (rendering) γραφικών
  - -- Δεν προσφέρει «παράθυρα» (windowing) και «είσοδο» (input) για να αποφύγει εξάρτηση από ένα συγκεκριμένο παραθυρικό Λειτουργικό Σύστημα (MS, Unix/Linux, Mac)
  - -- Συνδέεται με C, C++, Java, Python, Processing, Ruby, κ.ά.

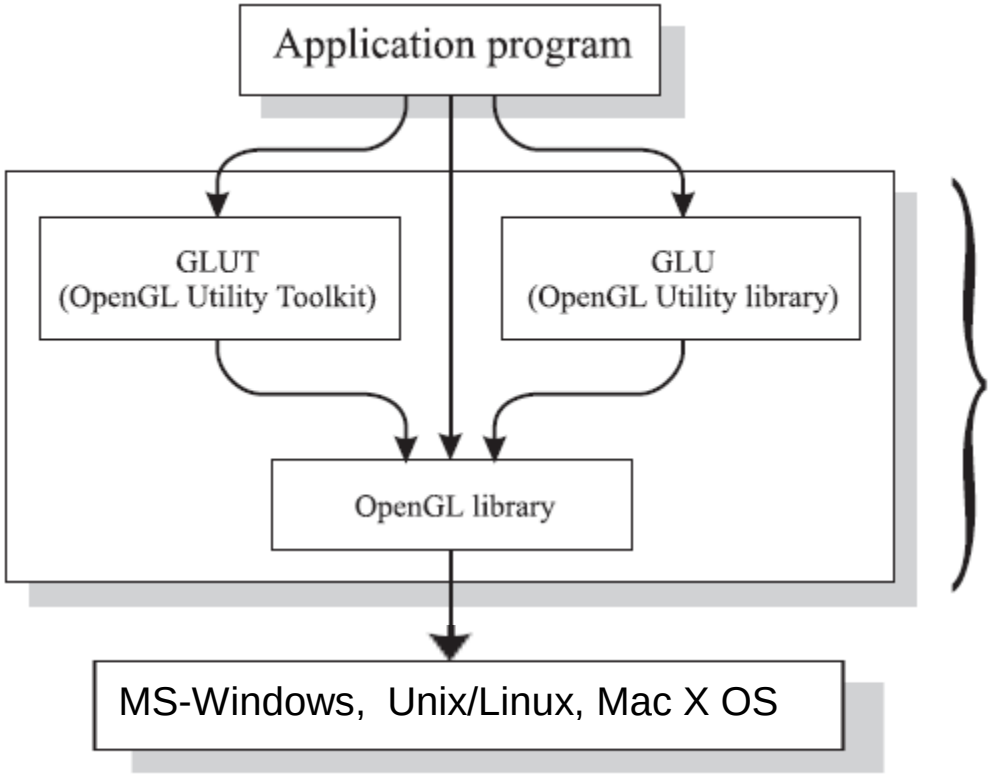
Η εξέλιξη της OpenGL

- Ελέγχεται από την Architectural Review Board (ARB)
  - Μέλη SGI, Microsoft, Nvidia, HP, 3DLabs, IBM, …
  - Αρκετά σταθερή μετά την version 1.4
- Η εξέλιξη αναπαράγει νέες δυναμικές δυνατότητες (capabilities) στο hardware
  - 3D σχεδίαση (mapping) υφής και αντικείμενα υφής
  - Προγράμματα κορυφών (Vertex programs)
  - Με επεκτάσεις ενσωματώνει συγκεκριμένα χαρακτηριστικά κάθε πλατφόρμας

Πληροφορίες και υλικό http://www.opengl.org

OpenGL βιβλιοθήκες

-- OpenGL Library
-- OpenGL Utility Library (GLU)
-- OpenGL Utility Toolkit (GLUT)



Οι τρείς αυτές βιβλιοθήκες συνήθως αναφέρονται ως OpenGL

OpenGL βιβλιοθήκες

OpenGL βασική (core) βιβλιοθήκη, έχει σχεδιαστεί ως μια βελτιωμένη διεπαφή ανεξάρτητη από το hardware

-- OpenGL32 για Windows

-- GL για Unix/Linux συστήματα

GLU παρέχει πολλά από τα χαρακτηριστικά μοντέλων, όπως quadric επιφάνειες και NURBS καμπύλες και επιφάνειες.

-- Παρέχει λειτουργικότητα στην βασική OpenGL αλλά δεν χρειάζεται να ξαναγράψουμε κώδικα

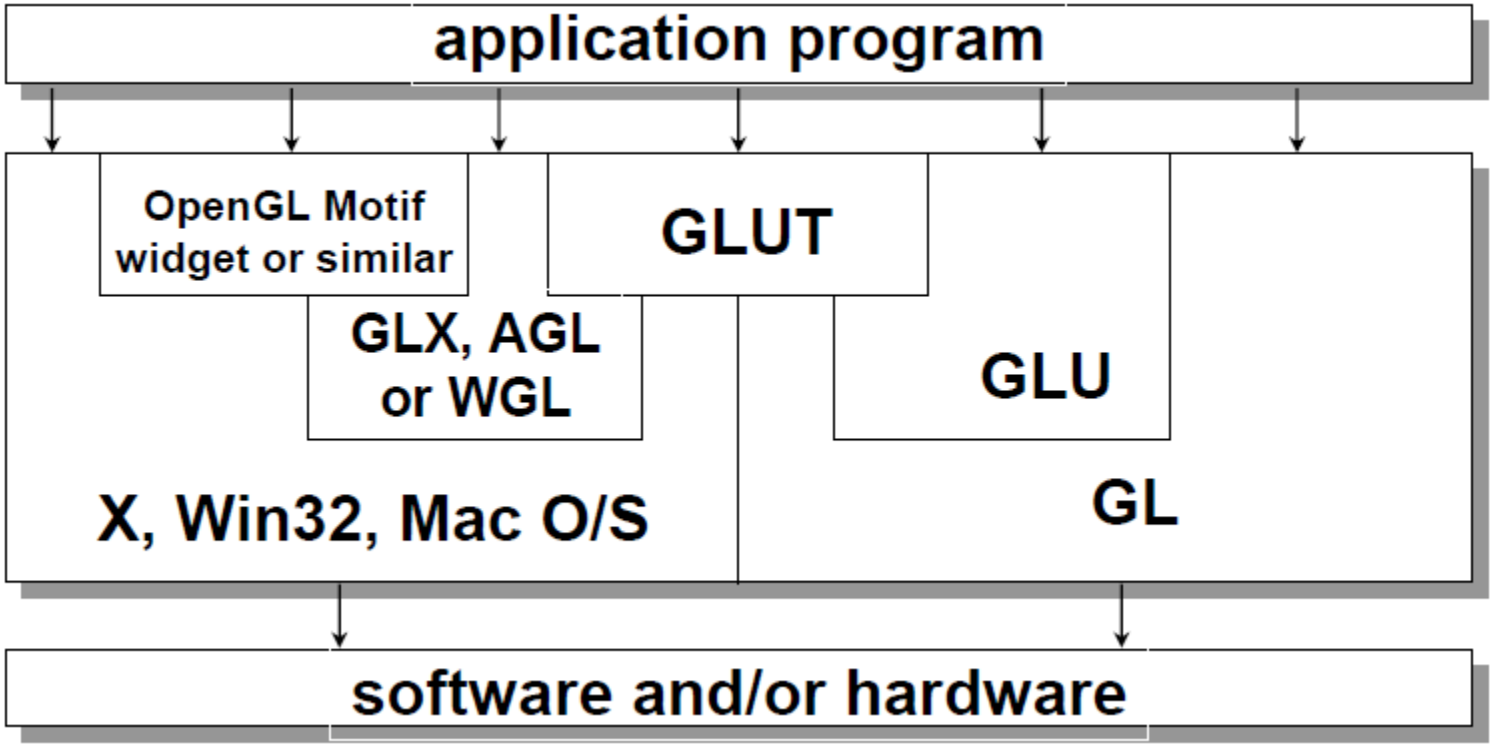
GLUT συνδέει την OpenGL με το παραθυρικό σύστημα

-- GLX για X συστήματα

-- WGL για Windows

-- AGL για Macintosh

Οργάνωση Λογισμικού
(Software Organization)



Windowing with OpenGL

OpenGL είναι ανεξάρτητη από παραθυρικό σύστημα. Για κάθε παραθυρικό σύστημα υπάρχει μια βιβλιοθήκη που συνδέει την OpenGL (rendering) με το παραθυρικό σύστημα:

-- X windows (GLX) για μηχανές H/Y που χρησιμοποιούν X Window System, παρέχεται η GLX (επέκταση της OpenGL για το X Window System as an adjunct to OpenGL. Οι GLX συναρτήσεις χρησιμοποιούν το πρόθεμα **`glx.`**

-- Για τα Microsoft Windows, οι WGL συναρτήσεις (routines) παρέχουν το interface των Windows με την OpenGL. Οι WGL συναρτήσεις χρησιμοποιούν το πρόθεμα **`wgl.`**

-- Για το Mac OS υπάρχουν τρείς διεπαφές (interfaces): AGL (με πρόθεμα **`agl`**), CGL (**`cgl`**), και Cocoa (**`NSOpenGL`** classes).

GLUT (OpenGL Utility Toolkit) = freeglut

- GLUT παρέχει ένα portable API για την δημιουργία παραθύρων και την επικοινωνία με τις συσκευές εισόδου/εξόδου (I/O devices)
  - -- GUI για διαφορετικά Λ.Σ.
  - -- Χειρισμός γεγονότων (events), κλικ ποντικιού, keys πληκτρολόγιου, αλλαγές στο μέγεθος παραθύρου, επιλογή μενού, κλπ
- Η βιβλιοθήκη GLUT είναι ανεξάρτητη από παραθυρικό συστημα (αρχικά γραμμένο από τον Mark Kilgard) που κρύβει την περιπλοκότητα των διαφορετικών APIs των παραθυρικών συστημάτων
- Η **freeglut** είναι μια υλοποίηση ανοικτού κώδικα που επεκτείνει την λειτουργικότητα της αρχικής GLUT.
- Οι GLUT συναρτήσεις χρησιμοποιούν το πρόθεμα **glut**.

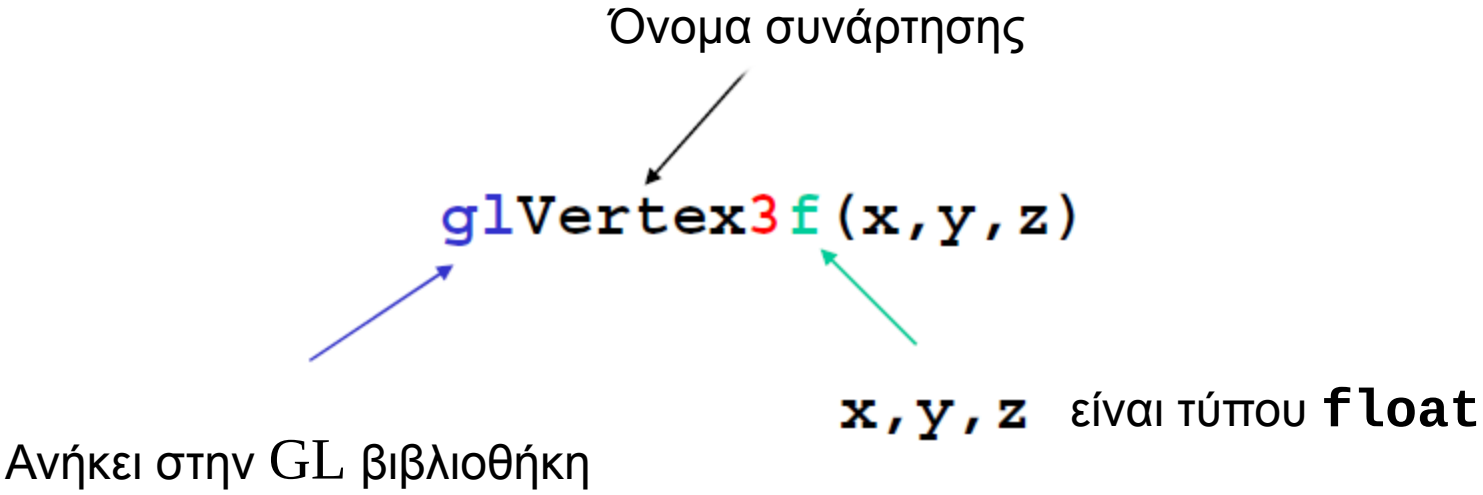
```
glutInit()
glutInitWindowPosition(int x, int y)
glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGBA | GLUT_DEPTH)
```

Τύποι δεδομένων της OpenGL

Εσωτερικοί τύποι για μεγαλύτερη μεταφερσιμότητα (portability)

Suffix	Data type	Typical C or C++ type	OpenGL type name
b	8-bit integer	signed char	GLbyte
s	16-bit integer	short	GLshort
i	32-bit integer	int or long	GLint, GLsizei
f	32-bit floating point	float	GLfloat, GLclampf
d	64-bit floating point	double	GLdouble, GLclampd
ub	8-bit unsigned number	unsigned char	GLubyte, GLboolean
us	16-bit unsigned number	unsigned short	GLushort
ui	32-bit unsigned number	unsigned int or unsigned long	GLuint, GLenum, GLbitfield

Μορφή των συναρτήσεων της OpenGL (function format)




```
glVertex3fv(p)
```

p είναι ένας δείκτης σε **float** πίνακα

Ονόματα στην OpenGL

- Συναρτήσεις έχουν πρόθεμα (prefix) το **`gl`**,
  - **`glBegin`**, **`glClear`**, **`glClearColor`**
- Σταθερές είναι με κεφαλαία γράμματα και η υπογράμμιση χρησιμοποιείται ως διαχωριστής
  - **`GL_2D`**, **`GL_LINES`**, **`GL_TRIANGLES`**
- Ενσωματωμένοι τύποι δεδομένων αρχίζουν με **`GL`**
  - **`GLbyte`**, **`GLshort`**, **`GLint`**, **`GLboolean`**

Περιεχόμενα της API

- Συναρτήσεις που ορίζουν τι χρειαζόμαστε για να σχηματίσουμε (form) μια εικόνα
  - -- Αντικείμενα (objects) κατασκευάζονται από θεμελιώδη γεωμετρικά – σημεία, γραμμές – που ορίζονται από τις κορυφές (vertices)
  - -- Θεατής (viewer, camera)
  - -- Πηγές φωτός (light sources)
  - -- Υλικά (materials)
- Άλλες πληροφορίες
  - -- Είσοδο από συσκευές όπως ποντίκι και πληκτρολόγιο
  - -- Δυνατότητες του συστήματος
  - -- *framebuffer,* η οποία κρατάει όλες τις πληροφορίες που η οθόνη γραφικών χρειάζεται για να ελέγχει το χρώμα και την ένταση όλων των pixel στην οθόνη

OpenGL State

- OpenGL είναι μια μηχανή καταστάσεων
- OpenGL διαθέτει συναρτήσεις δυο τύπων:
  - -- Δημιουργία βασικών (primitive)
    - Εμφάνιση βασικών εάν είναι ορατά
    - Επεξεργασία κορυφών και εμφάνιση βασικών με έλεγχο της καταστασης
  - -- Αλλαγή καταστάσεων (state changing)
    - Συναρτήσεις μετασχηματισμών
    - Συναρτήσεις χαρακτηριστικών (attribute)

OpenGL #defines

Οι περισσότερες σταθερές ορίζονται στα αρχεια κεφαλίδων **`gl.h`**, **`glu.h`** και **`glut.h`**

Με την **`#include <glut.h>`** ή **`#include <freeglut.h>`** θα πρέπει αυτόματα να συμπεριλαμβάνονται και τα άλλα

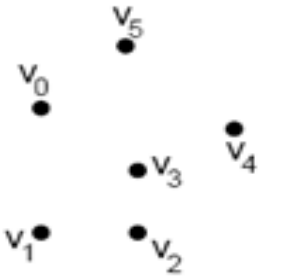
Παραδείγματα

```
glBegin(GL_POLYGON)
glClear(GL_COLOR_BUFFER_BIT)
```

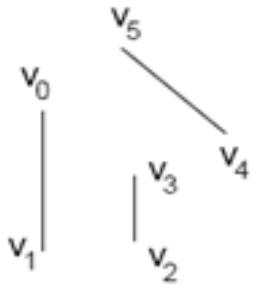
Στα αρχεία κεφαλίδων ορίζονται και οι τύποι δεδομένων της OpenGL:

`Glfloat`, **`Gldouble`**, ...

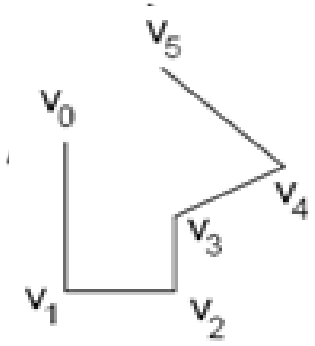
Θεμελιώδη σχήματα της OpenGL (Primitives)



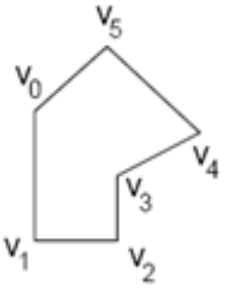
GL_POINTS



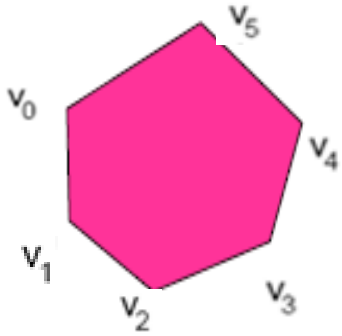
GL_LINES



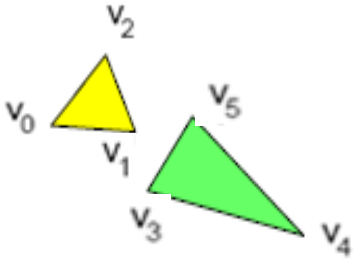
GL_LINE_STRIP



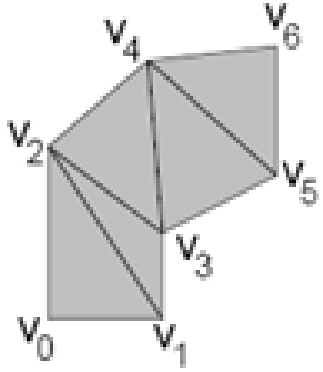
GL_LINE_LOOP



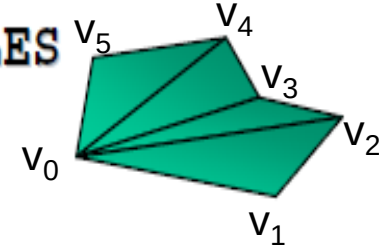
GL_POLYGON



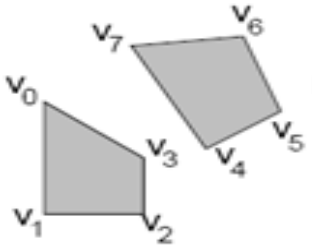
GL_TRIANGLES



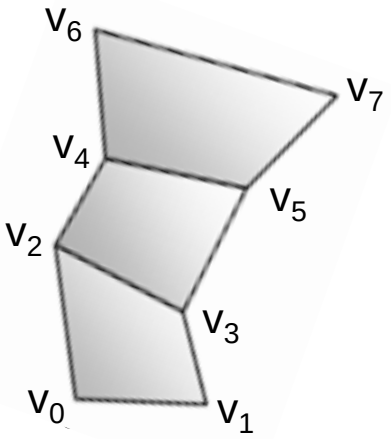
GL_TRIANGLE_STRIP



GL_TRIANGLE_FAN



GL_QUADS



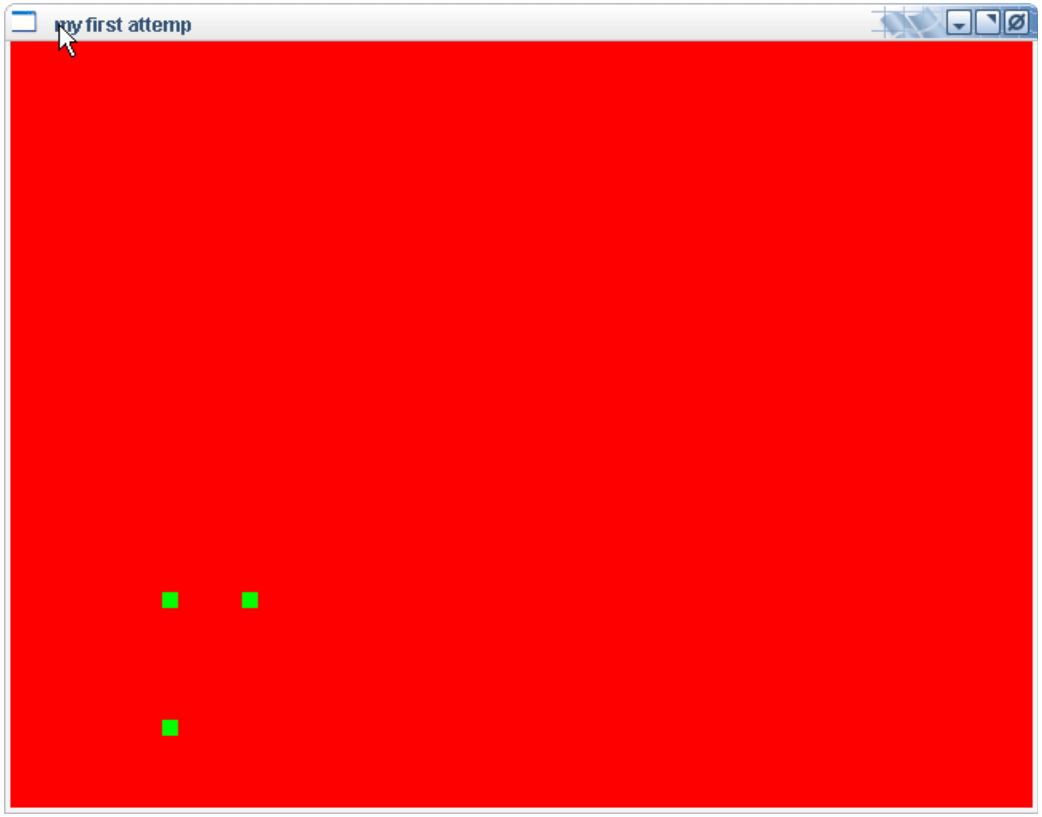
GL_QUAD_STRIP

Σχεδίαση σημείων

Παράδειγμα:

Το πρόγραμμα δεν είναι διαδραστικό. Περιλαμβάνει τρείς συναρτήσεις:

`main`, **`myDisplay`**, **`myInit`**



Σχεδίαση σημείων

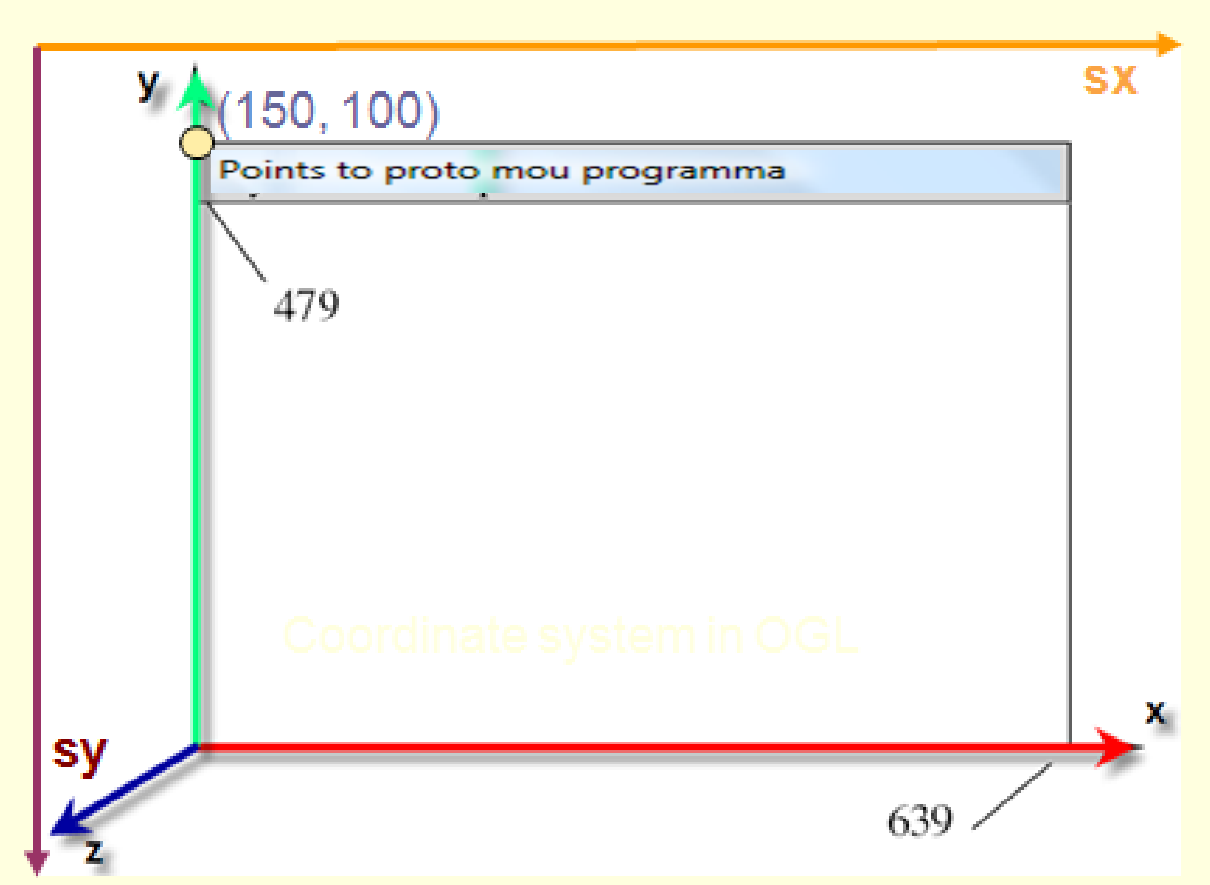
Παράδειγμα:

Συνάρτηση: **`main`**

```
int main(int argc, char** argv)
{
        glutInit(&argc, argv);          // ενεργοποίηση της glut
        glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB); // set the display mode
        glutInitWindowSize(640, 480); // set το μέγεθος του παραθύρου
        glutInitWindowPosition(100, 150); // set την θέση του παραθύρου
        glutCreateWindow("Σημεία"); // δημιουργία του παραθύρου
        glutDisplayFunc(myDisplay); // εγγραφή της συνάρτησης εμφάνισης
        myInit();               // επιπλέον ενεργοποιήσεις
        glutMainLoop();    // ατέρμονη θηλιά
        return(0);
}
```

Δημιουργία Παραθύρου

Οι πέντε πρώτες γραμμές είναι κλήσεις σε συναρτήσεις της GLUT για την δημιουργία παραθύρου οπου θα γίνει η σχεδίαση



Σχεδίαση σημείων

Παράδειγμα:

Σχεδίαση θεμελιωδών σχηματων στην συνάρτηση:

`myDisplay`

```
void myDisplay()
{
        glClear(GL_COLOR_BUFFER_BIT); // καθαρισμός οθόνης
        glBegin(GL_POINTS);
                glVertex2i(100, 50);      // σχεδίαση τριών σημείων
                glVertex2i(100, 130);
                glVertex2i(150, 130);
        glEnd();
        glFlush();              // αποστολή για εμφάνιση (display)
}
```

Σχεδίαση σημείων

Παράδειγμα:

Αρχικοποιήσεις στην: **`myInit`**

```
void myInit()
{
        glClearColor(1.0, 0.0, 0.0, 0.0); // χρώμα υπόβαθρου κόκκινο
        glColor3f(0.0, 1.0, 0.0); // χρώμα σχεδίασης (drawing color)
        glPointSize(10.0);   // ένα σημείο (dot) είναι 10 επί 10 pixels
        // Οι επόμενες γραμμές establish το σύστημα συντεταγμένων
        // Θα αναλυθούν αργότερα.
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        gluOrtho2D(0, 640, 0, 480);
}
```

Σχεδίαση Γραμμών

Παράδειγμα:

Αρχικοποιήσεις στην: **`myInit`**

```
glLineWidth(2.0);                    // set line thickness
glBegin(GL_LINES);
        glVertex2i(10, 20);          // first horizontal line
        glVertex2i(40, 20);
        glVertex2i(20, 10);          // first vertical line
        glVertex2i(20, 40);
        // four more calls to glVertex here for the other two lines
glEnd();
```

Παράδειγμα:
Σχεδίαση ενός τόξου Arc

- Δίδεται κύκλος με ακτίνα radius *r*, και κέντρο στο σημείο (*x*,*y*). Να σχεδιασθεί ένα τόξο του κύκλου που sweeps out μια γωνία angle θ.

$$(x, y) = (x_0 + r\cos\theta, y_0 + r\sin\theta),$$

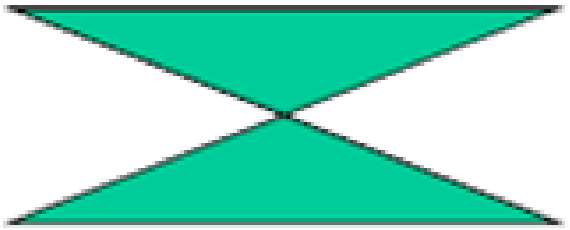
για $0 \le \theta \le 2\pi$

Χρήση της Line Strip

```
void drawArc(float x, float y, float r,float t0,
             float sweep)
{
   float t, dt; /* angle */
   int n = 30; /* # of segments */
   int i;
   t = t0 * PI/180.0; /* radians */
   dt = sweep * PI/(180*n); /* increment */
   glBegin(GL_LINE_STRIP);
      for(i=0; i<=n; i++, t += dt)
         glVertex2f(x + r*cos(t), y + r*sin(t));
   glEnd();
}
```

Πολύγωνα
(Polygon issues)

- Η OpenGL εμφανίζει τα πολύγωνα σωστά μόνο όταν είναι:
  -- *Απλά* δηλ. οι πλευρές δεν τέμνονται
  -- *Κυρτά*
  -- *Επίπεδα*: όλες οι κορυφές βρίσκονται στο ίδιο επίπεδο
- Το πρόγραμμα του χρήστη πρέπει να ελέγχει τα παραπάνω.
- Τα τρίγωνα ικανοποιούν όλες τις συνθήκες



Μη-κυρτό πολύγωνο

Μη-κυρτό πολύγωνο

Χαρακτηριστικά (Attributes)

Τα attributes είναι μέρος της OpenGL και ορίζουν την εμφάνιση των αντικειμένων

-- Χρώμα-color (σημείων, γραμμών, πολυγώνων)

-- Μέγεθος-size και πλάτος-width (σημείων, γραμμων)

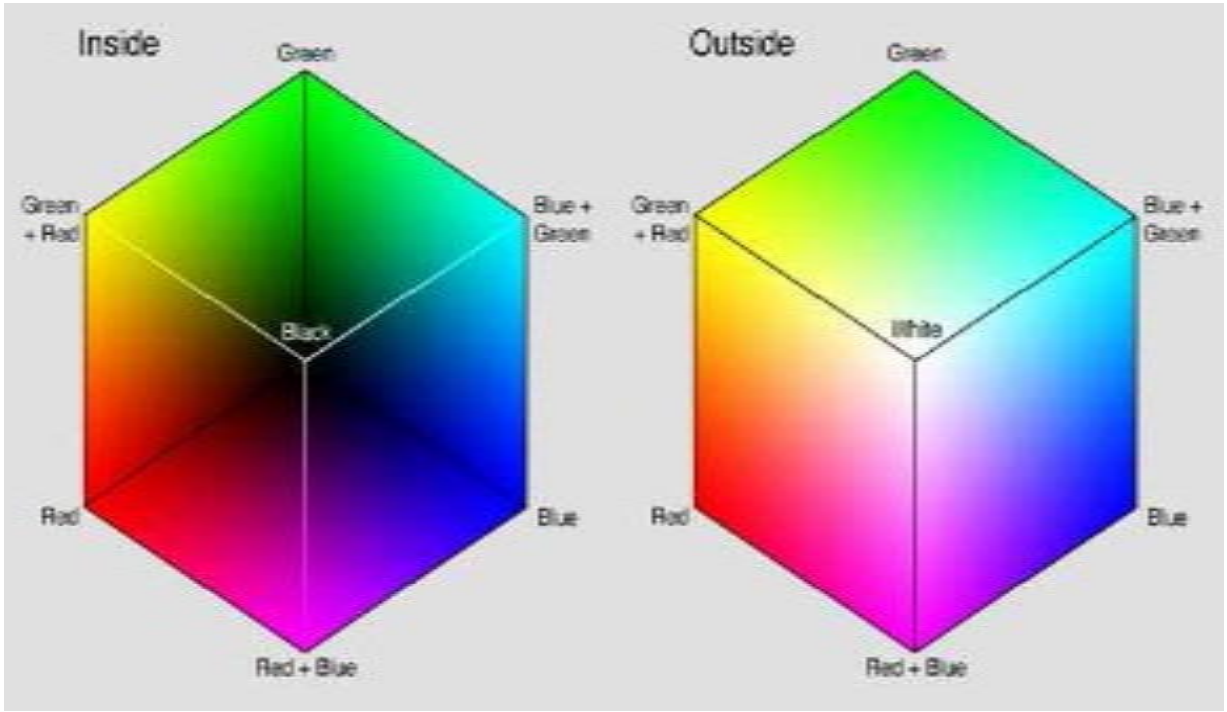
-- Σχεδιαση με κουκίδες - Stipple pattern (lines, polygons)

-- Πολυγωνικός τροπος - Polygon mode

- Εμφάνιση όπως γεμίστηκε (solid color or stipple pattern
- Εμφάνιση ακμών (display edges)

RGB χρώμα

- Κάθε συστατικό (component) χρώματος αποθηκευεται χωριστά (συνήθως 8 bits ανά component)
- Στην OpenGL οι τιμές χρώματος κυμαίνονται (range) από 0.0 (none) έως 1.0 (all).



Texture Mapping

Περιορισμοί της γεωμετρικής μοντελοποίησης
(The Limits of Geometric Modeling)

Παρ' όλο που οι κάρτες γραφικών μπορούν να αποδώσουν (render) πανω από 10 (million) εκατομμύρια πολύγωνα ανά δευτερόλεπτο, αυτός ο αριθμός είναι ανεπαρκής για πολλά φαινόμενα όπως:

-- Σύννεφα (clouds)

-- Γρασίδι (grass)

-- Επιφάνεια εδαφους (terrain)

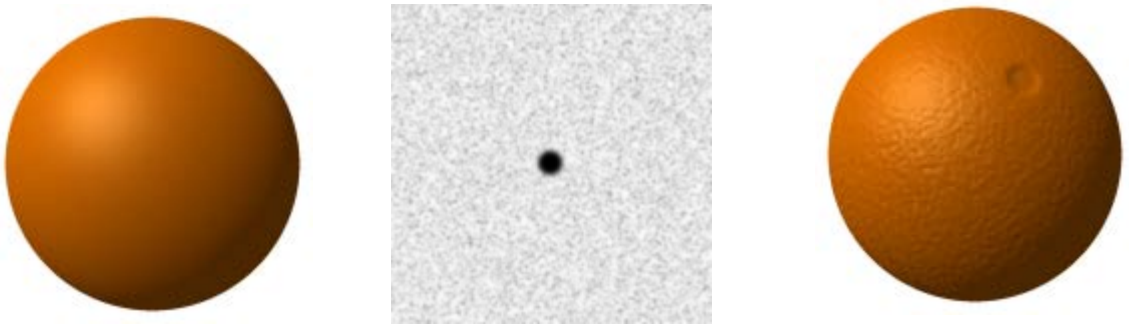
-- Δέρμα (skin)

Μοντελοποίηση ενός πορτοκαλιού (1)
(Modeling an Orange)

- Θεωρήστε το πρόβλημα της μοντελοποίησης ενός πορτοκαλιού
- Αρχίστε με μια σφαίρα χρωματισμένη πορτοκαλί
  - -- Δεν «πιάνει» (capture) τα χαρακτηριστικά της επιφάνειας μικρά λακκάκια (dimples)
  - -- Χρειάζονται πάρα πολλά πολύγωνα να μοντελοποιείστε όλα τα λακκάκια

Μοντελοποίηση ενός πορτοκαλιού (2)
(Modeling an Orange)

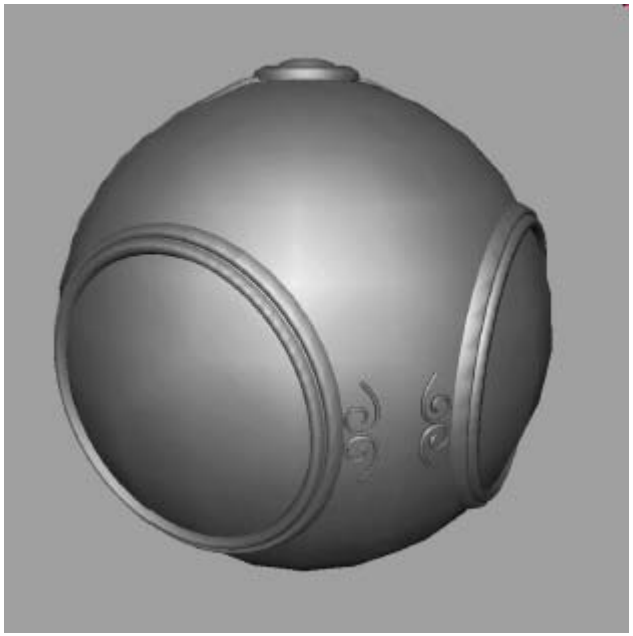
- Πάρτε μια φωτογραφία ενός πραγματικού πορτοκαλιού (εάν είναι αναλογική σαρώστε την –scan it) και επικολλήστε την “paste” στο απλό γεωμετρικό μοντέλο
  - -- Αυτή η διαδικασία ονομάζεται χαρτογράφηση υφής (*texture mapping*)
- Πιθανόν αυτό να μη είναι αρκετό επειδή η επεξεργασμένη επιφάνεια να είναι λεία
  - -- Χρειάζεται να αλλάξει το τοπικά σχήμα (local shape)
  - -- Χαρτογράφηση εξογκομάτων-φουσκοματων (bump mapping)



Τύποι Χαρτογράφησης
(Types of Mapping)

- Χαρτογράφηση υφής (texture mapping)
  -- Χρήση εικόνων για το εσωτερικό γέμισμα των πολυγώνων
- Περιβάλλοντος (environmental reflection mapping)
  -- Χρήση μιας εικόνας του περιβάλλοντος για for texture maps
  -- Επιτρέπει την προσομοίωση των επιφανειών υψηλής κατοπτρικής αντανακλασης
- Χαρτογράφηση εξογκομάτων-φουσκοματων (Bump mapping)
  -- Προσομοιώνει τα εξογκόματα αλλαζοντας τα κανονικά διανυσματα (normal vectors) κατά την διαδικασία της αποδοσης (rendering)

Texture Mapping

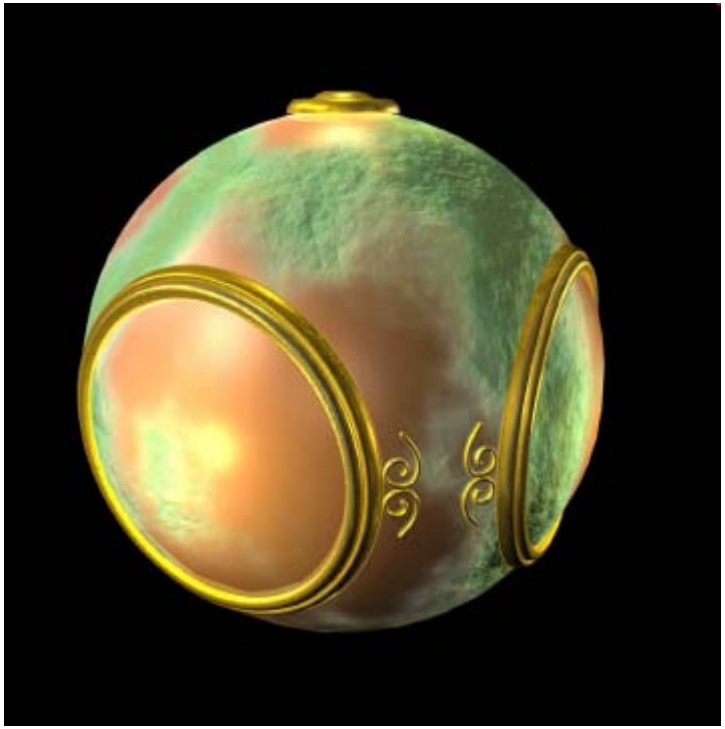


Γεωμετρικό μοντέλο
(geometric model)

Χαρτογραφηση υφής
(texture mapped)

Environment Mapping

Bump Mapping

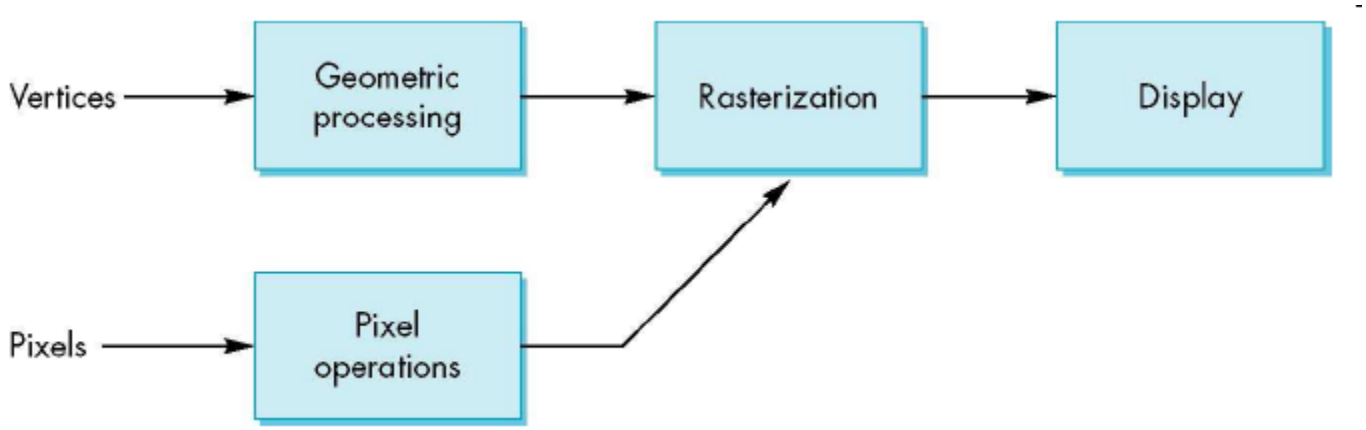


Χαρτογράφηση (mapping)

Που γίνεται η χαρτογράφηση;

Οι τεχνικές χαρτογράφησης υλοποιούνται στο τέλος της γραμμής απόδοσης (pipeline rendering)

Πολύ αποτελεσματικό επειδή λίγα πολύγωνα περνούν την γραμμή της επεξεργασίας γεωμετρίας



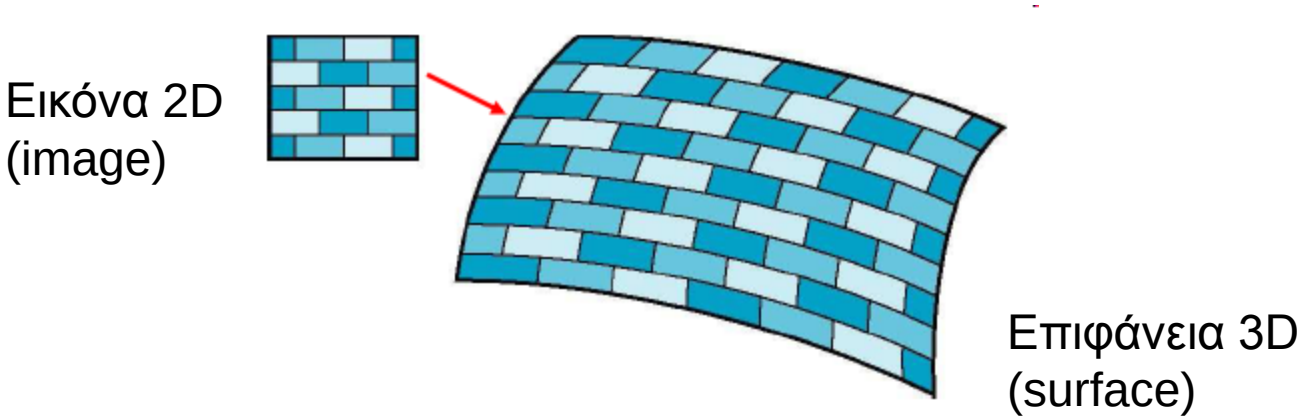
Βασική Στρατηγική

Τρία βήματα για την εφαρμογή υφής

1. Καθόρισε την υφή
   - διάβασε ή δημιούργησε την εικόνα
   - δώσε την υφή (assign to texture)
   - ενεργοποιησε την υφή (enable texturing)
2. Δώσε τις συντεταγμένες υφής στις κορυφές (vertices)
   - Επιλογή της συνάρτησης αντιστοίχησης επαφίεται στην εφαρμογή
3. Καθόρισε τις παραμέτρους υφής
   - τύλιγμα (wrapping), φιλτραρισμα (filtering)

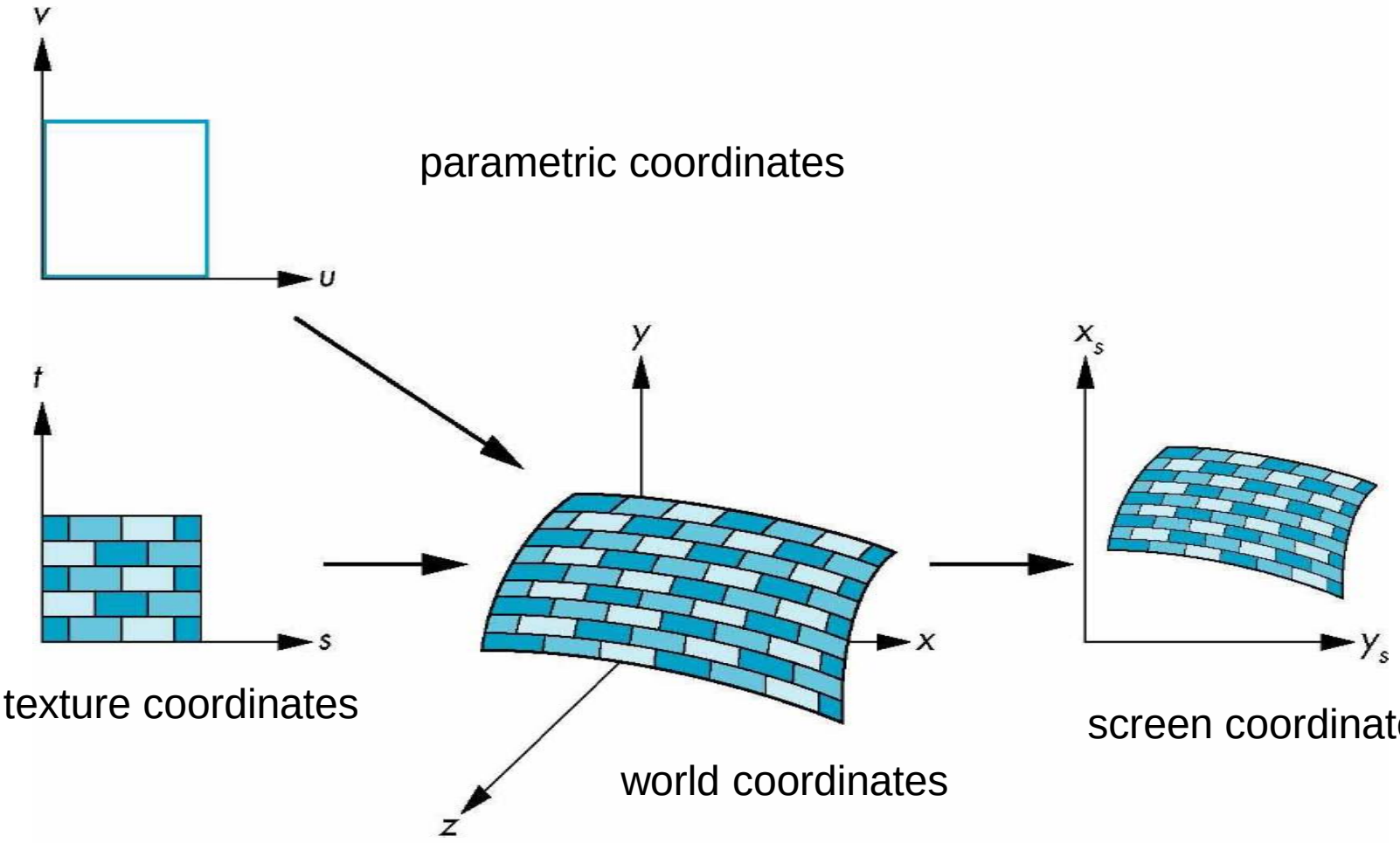
Χαρτογράφηση (Mapping)

Παρ' όλο που η ιδέα είναι απλή --- *αντιστοίχιση (map) μιας εικόνας σε μια επιφάνεια* -- υπάρχουν 3 ή 4 συστήματα συντεταγμένων που εμπλέκονται (involved)

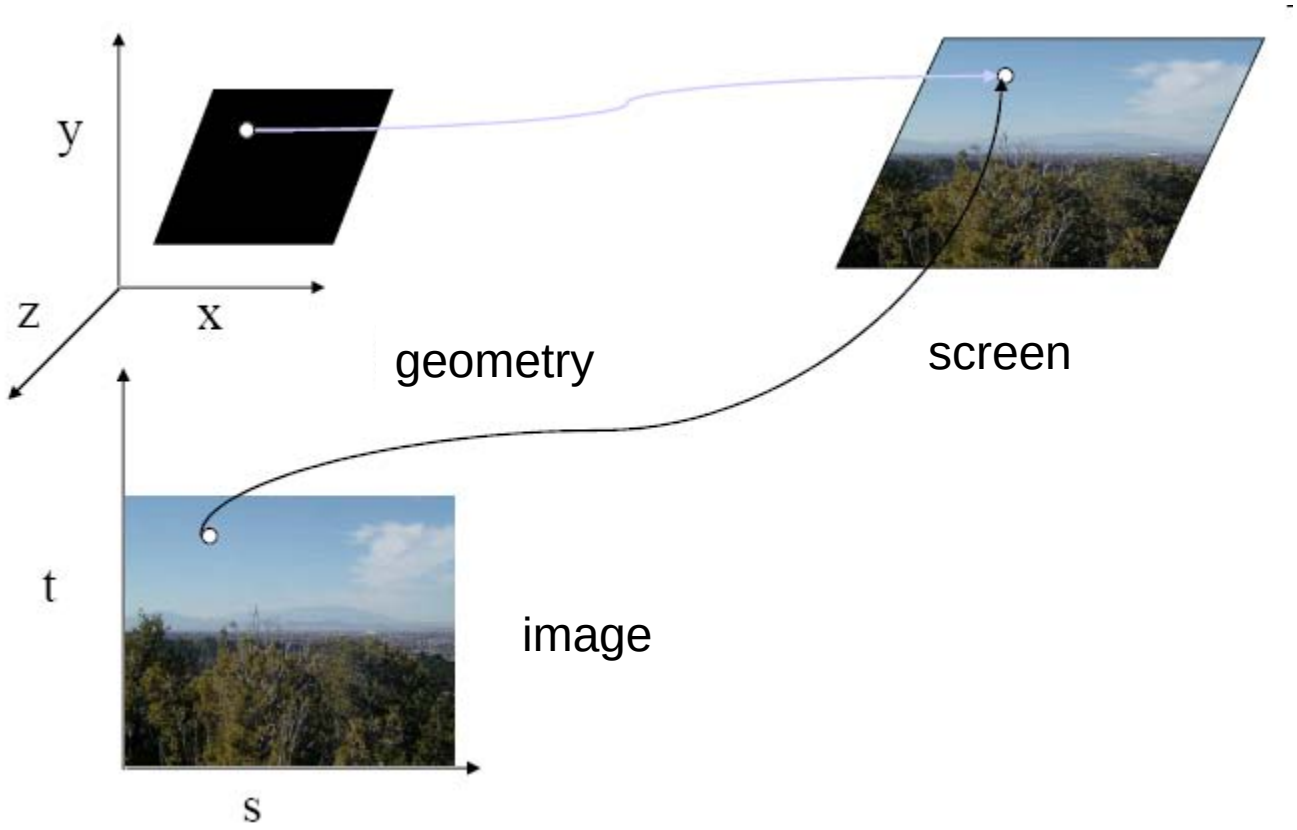


Texture Mapping

Συστήματα συντεταγμένων



Texture Mapping



Τέλος