CUDA + OpenGL

Alvaro Cuno

23/01/2010

OpenGL

- Biblioteca gráfica escalable y multiplataforma
  - Linux, Windows, Solaris, Amiga ...
- Permite el desarrollo de aplicaciones interactivas 2D y 3D
- Provee
  - Primitivas gráficas vectoriales y matriciales (imágenes)
  - Transformaciones e iluminación
  - Mapeamiento de texturas, etc.
- Varias implementaciones (Mesa,Microsoft,etc.)

OpenGL

- ¿Que no hace?
  - No hace administración de ventanas
  - No se preocupa con la interacción con el usuario
  - No se preocupa con archivos de entrada y salida
- No es un lenguaje de programación
  - Pero puede ser usada con varios lenguajes de programación: C, C++, Python, Java, Fortran, etc.

OpenGL

- Funciona como una máquina de estados
- La API tiene rutinas para
  - Diseñar primitivas geométricas e imágenes
  - Alterar variables de estado (ejemplo: color, material, pila de matrices, etc.)
  - Consultar variables de estado
- OpenGL es un estándar en evolución
  - Mecanismo estandarizado de extensiones
  - Las nuevas versiones son establecidas por un comite (ARB) de usuarios y fabricantes

OpenGL: APIs relacionadas

- OpenGL (Open Graphics Library)
  - Funcionalidad gráfica
- GLU (OpenGL Utility Library)
  - Funcionalidad extra
  - Parte del estándar OpenGL
  - Posee varias funciones para modelaje
    - Curvas, superficies cuadráticas y NURBs
- GLUT (OpenGL Utility Toolkit)
  - API portátil para acceso a los sistemas de ventanas

Hello OpenGL

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

///
/// Tipica funcion main de un programa OpenGL/GLUT
///
int main(int argc, char** argv) {

   // 1. Inicializacion del glut
   glutInit(&argc, argv);                               // Inicializar glut
   // 2. Inicializacion de la ventana
   glutInitDisplayMode(GLUT_DOUBLE|GLUT_RGB); // Double buffering y RGB
   glutInitWindowPosition(50, 0);                       // Posicion inicial de la ventana
   glutInitWindowSize(300, 300);                        // Tamanho inicial de la ventana
   // 3. Creación de la ventana
   glutCreateWindow(argv[0]);                           // Crear la ventana
   // 4. Invocar el loop principal
   glutMainLoop();                                      // Bucle de procesamiento
   return 0;
}
```

OpenGL: ejemplo 1

- Renderización de un rectángulo

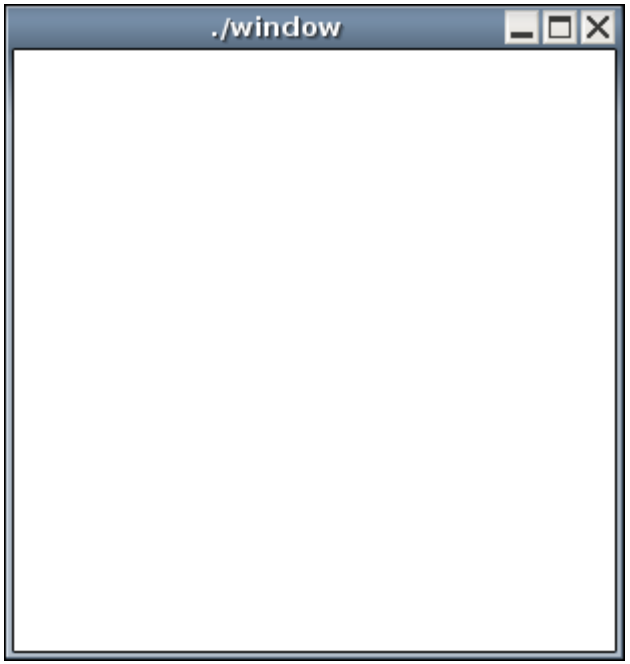
```
int main(int argc, char** argv) {

   // 1. Inicializacion del glut
   glutInit(&argc, argv);
   // 2. Inicializacion de la ventana
   glutInitDisplayMode(GLUT_DOUBLE|GLUT_RGB);
   glutInitWindowSize(300, 300);
   glutInitWindowPosition(50, 0);
   // 3. Creación de la ventana
   glutCreateWindow(argv[0]);
   // 4. Otras inicializaciones
   init();
   // 5. Registro de callbacks
   glutDisplayFunc(display);
   glutReshapeFunc(reshape);
   // 6. Invocar el bucle principal
   glutMainLoop();
   return 0;
}
```

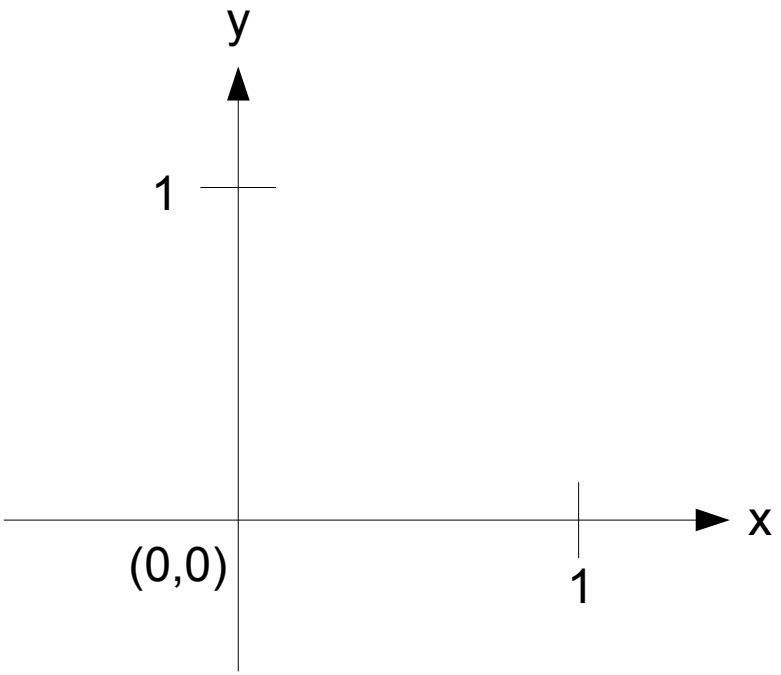
```
void reshape(int w, int h) {
    glViewport(0, 0, w, h);
   // espeficando campo de vision
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   gluOrtho2D(0.0, 1.0, 0.0, 1.0);
}
```

```
void init() {
   // especificando el color fondo
   glClearColor(0.0, 0.0, 0.0, 0.0);
}
```

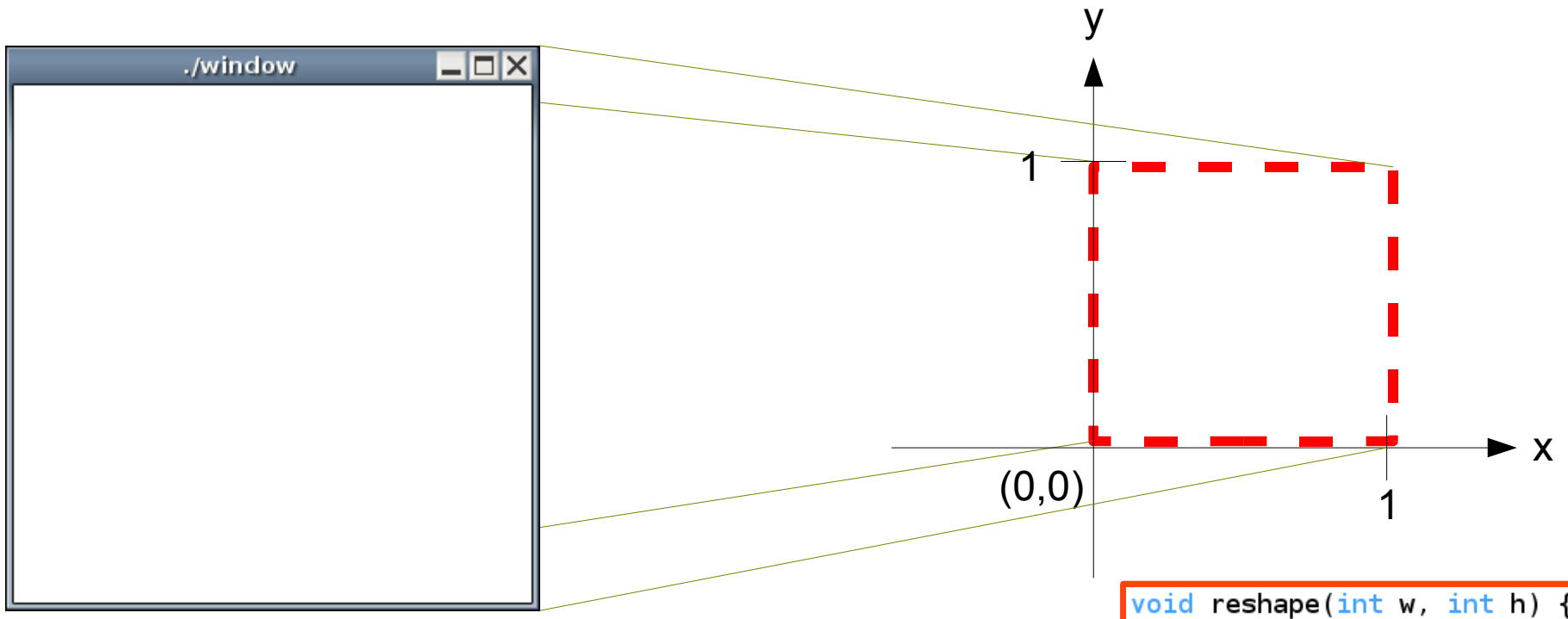
```
void display() {

   // limpia la ventana
   glClear (GL_COLOR_BUFFER_BIT);

   // disenha un rectangulo blanco
   glColor3f(1.0, 1.0, 1.0);
   glBegin(GL_POLYGON);
     glVertex2f(0.25, 0.25);
     glVertex2f(0.75, 0.25);
     glVertex2f(0.75, 0.75);
     glVertex2f(0.25, 0.75);
   glEnd();

   // actualizando la ventana
   glutSwapBuffers();
}
```

gluOrtho2D



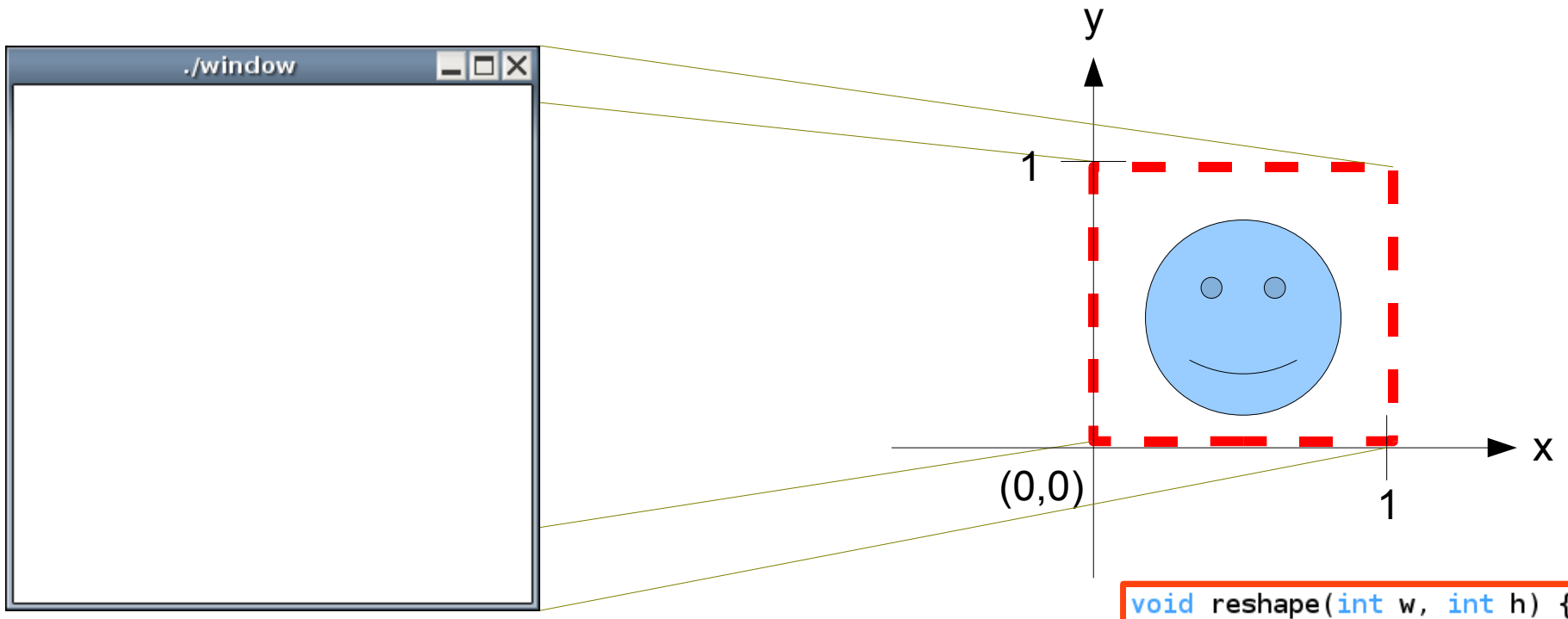


gluOrtho2D



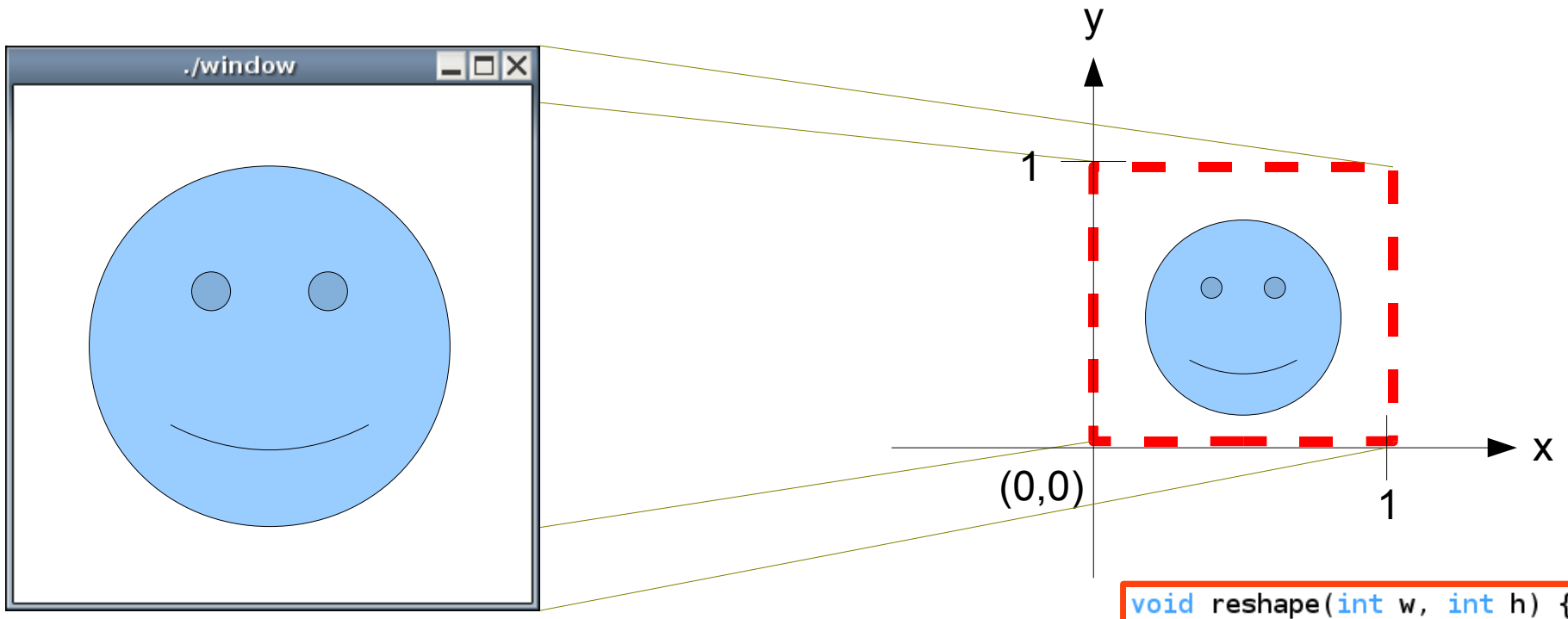

```
void reshape(int w, int h) {
    glViewport(0, 0, w, h);
   // espeficando campo de vision
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   gluOrtho2D(0.0, 1.0, 0.0, 1.0);
}
```

gluOrtho2D




```
void reshape(int w, int h) {
    glViewport(0, 0, w, h);
   // espeficando campo de vision
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   gluOrtho2D(0.0, 1.0, 0.0, 1.0);
}
```

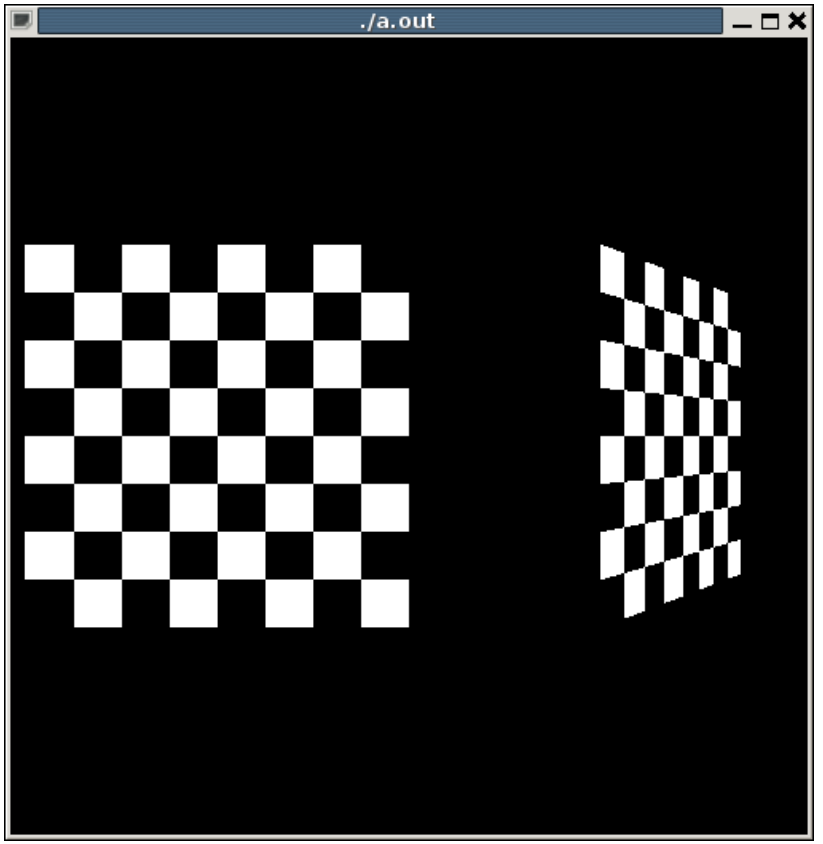
gluOrtho2D



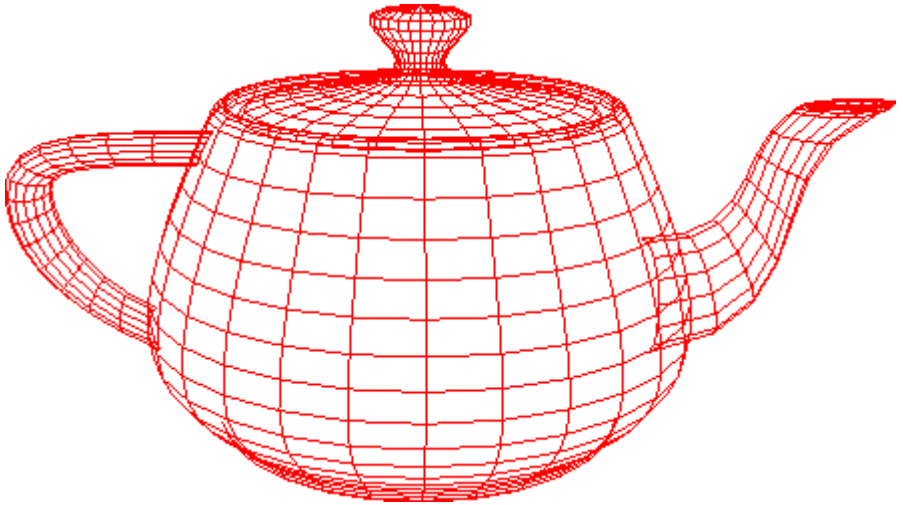

```
void reshape(int w, int h) {
    glViewport(0, 0, w, h);
   // espeficando campo de vision
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   gluOrtho2D(0.0, 1.0, 0.0, 1.0);
}
```

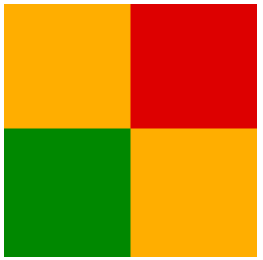
OpenGL: ejemplo 2

- Rendering de una textura (ejemplo del redbook)

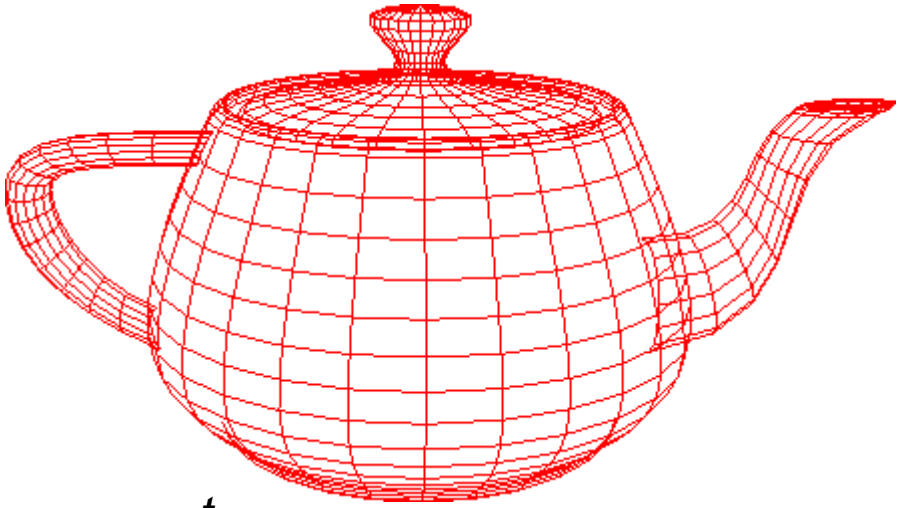


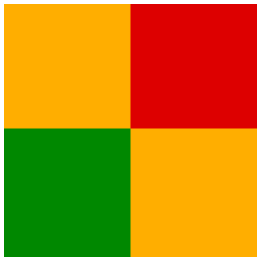
OpenGL: Texturas

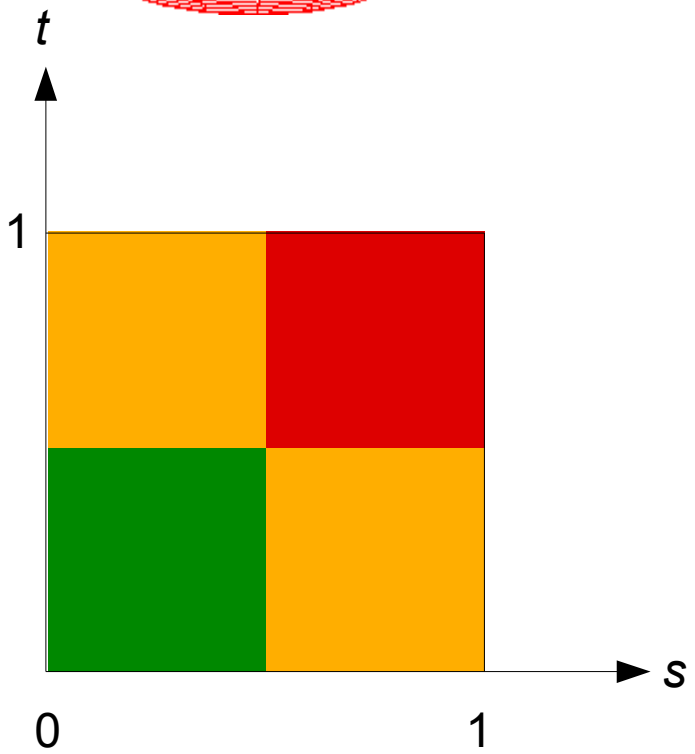




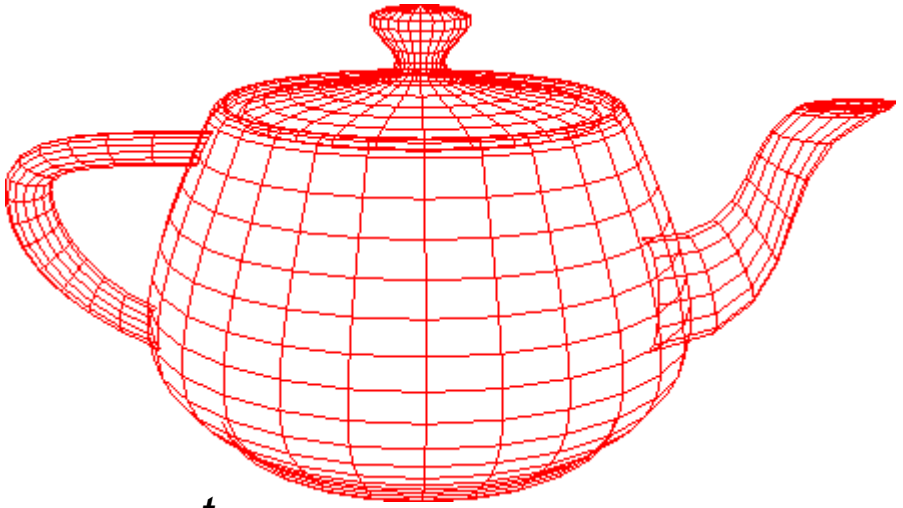
OpenGL: Texturas

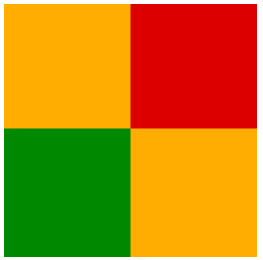


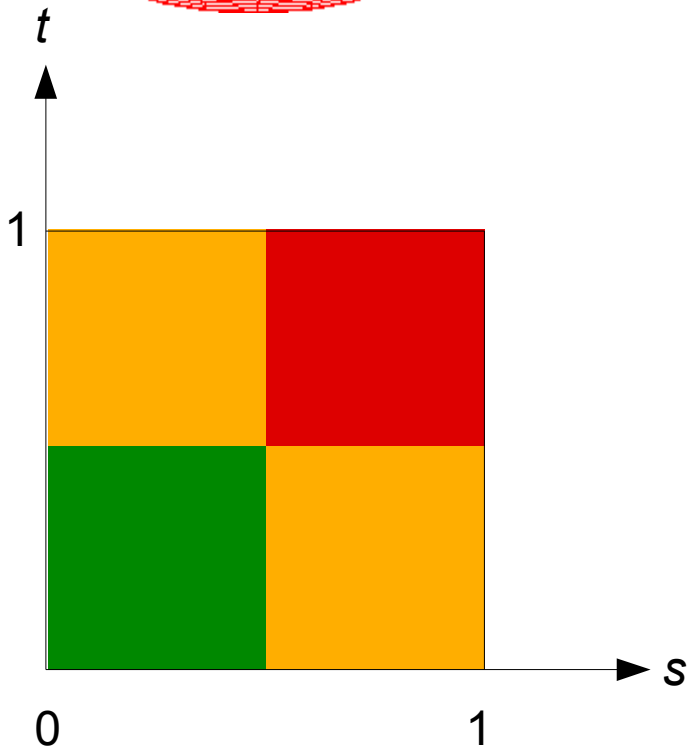


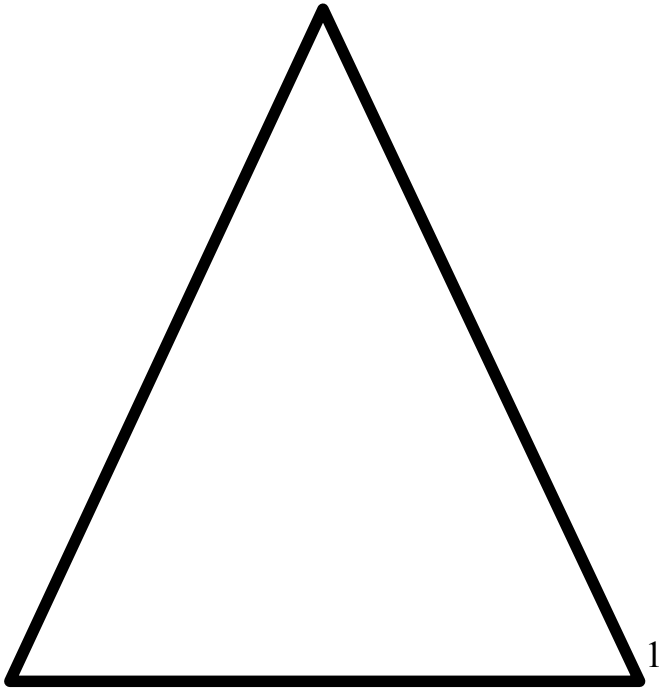


OpenGL: Texturas

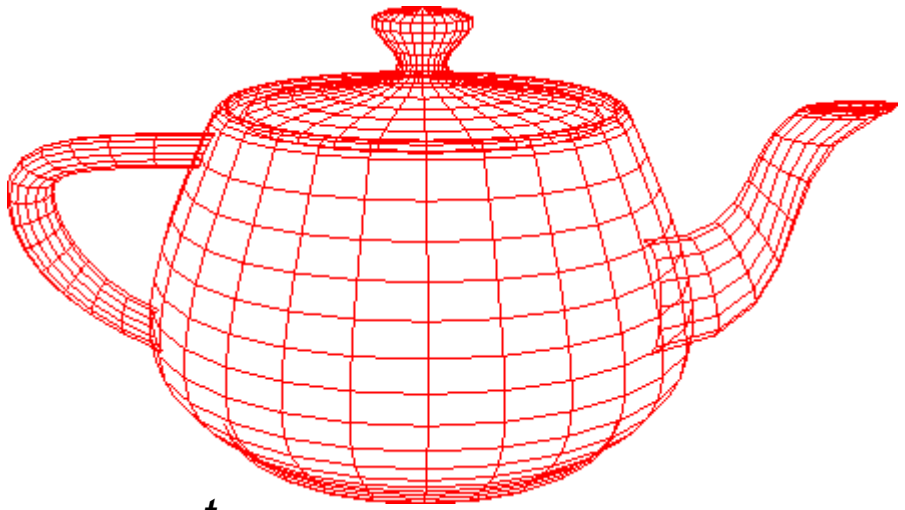


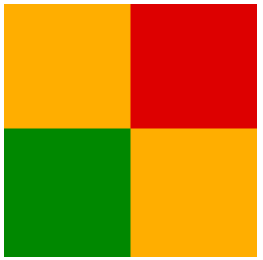


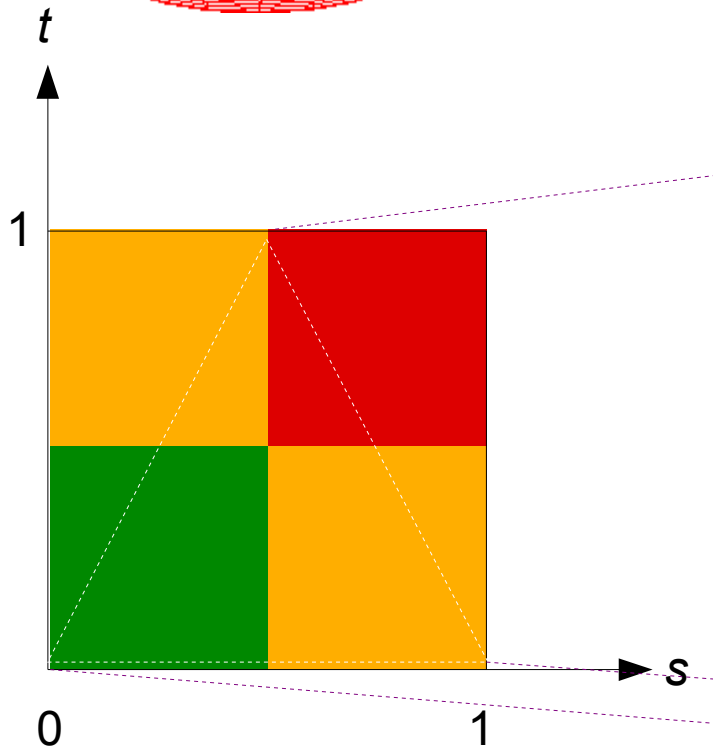


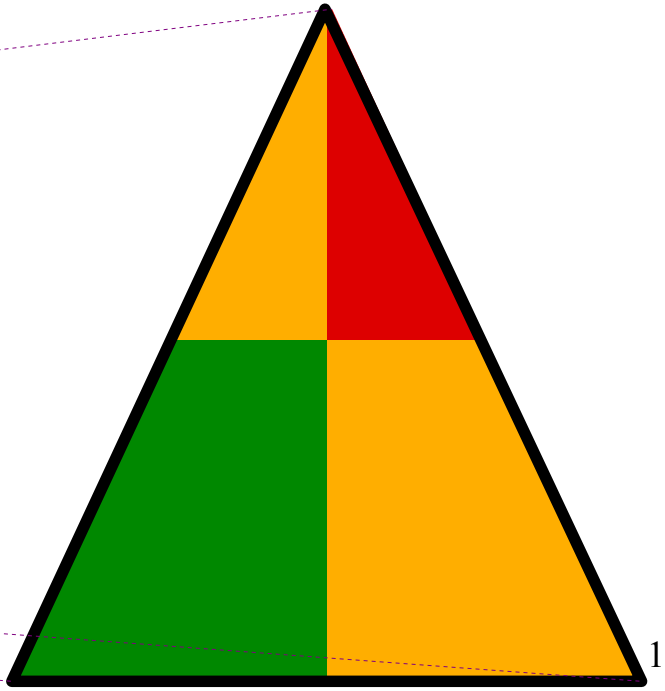


OpenGL: Texturas



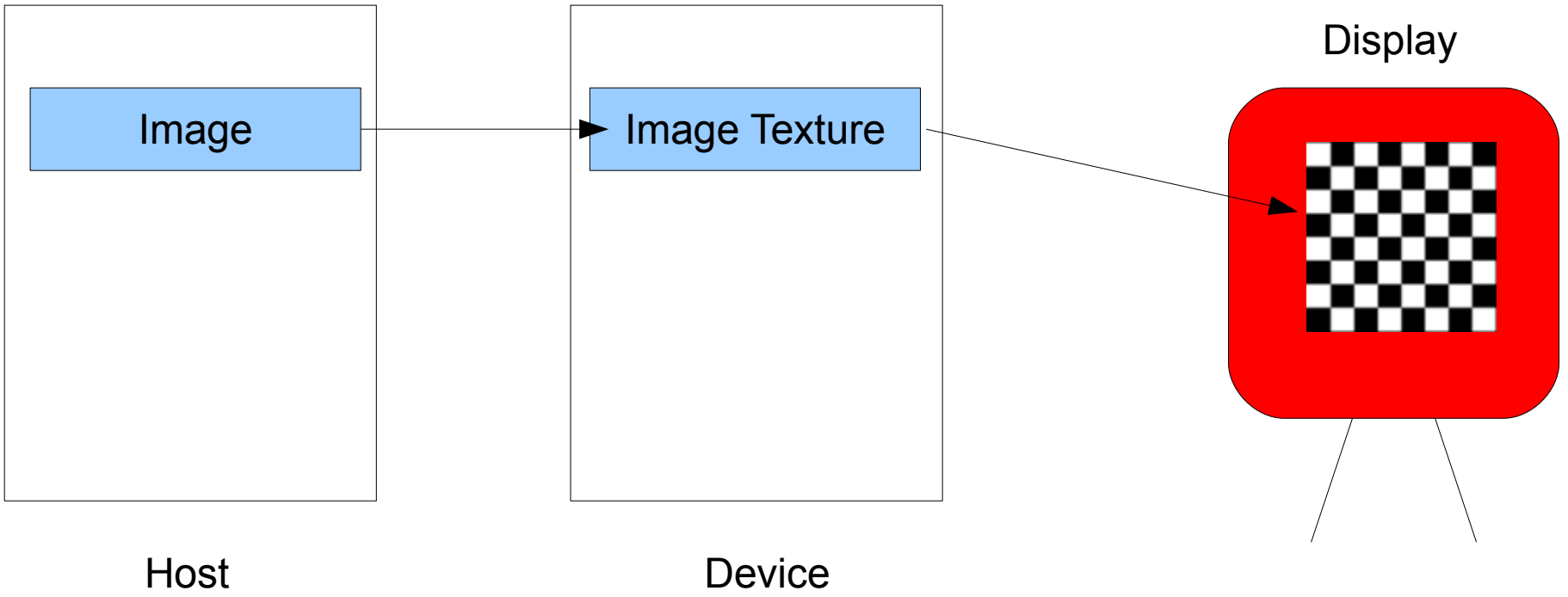






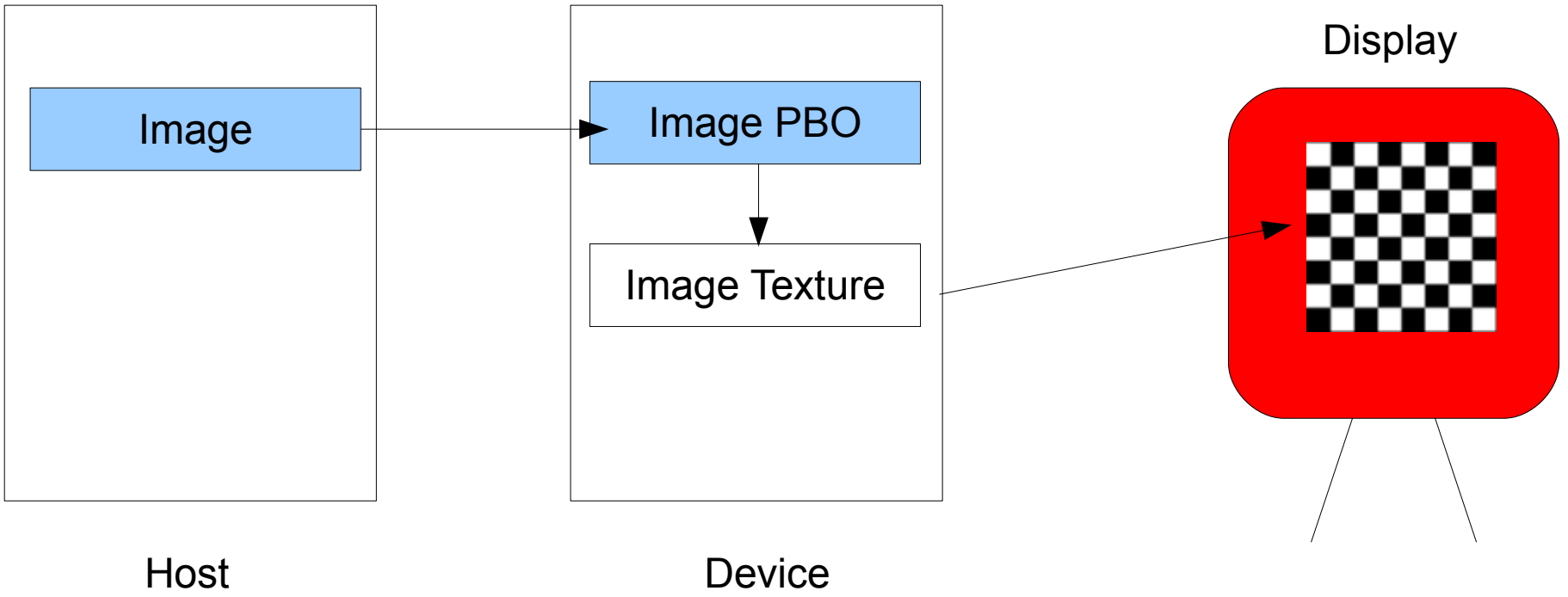
OpenGL: ejemplo 3

- Rendering de una textura (ejemplo modificado del redbook)



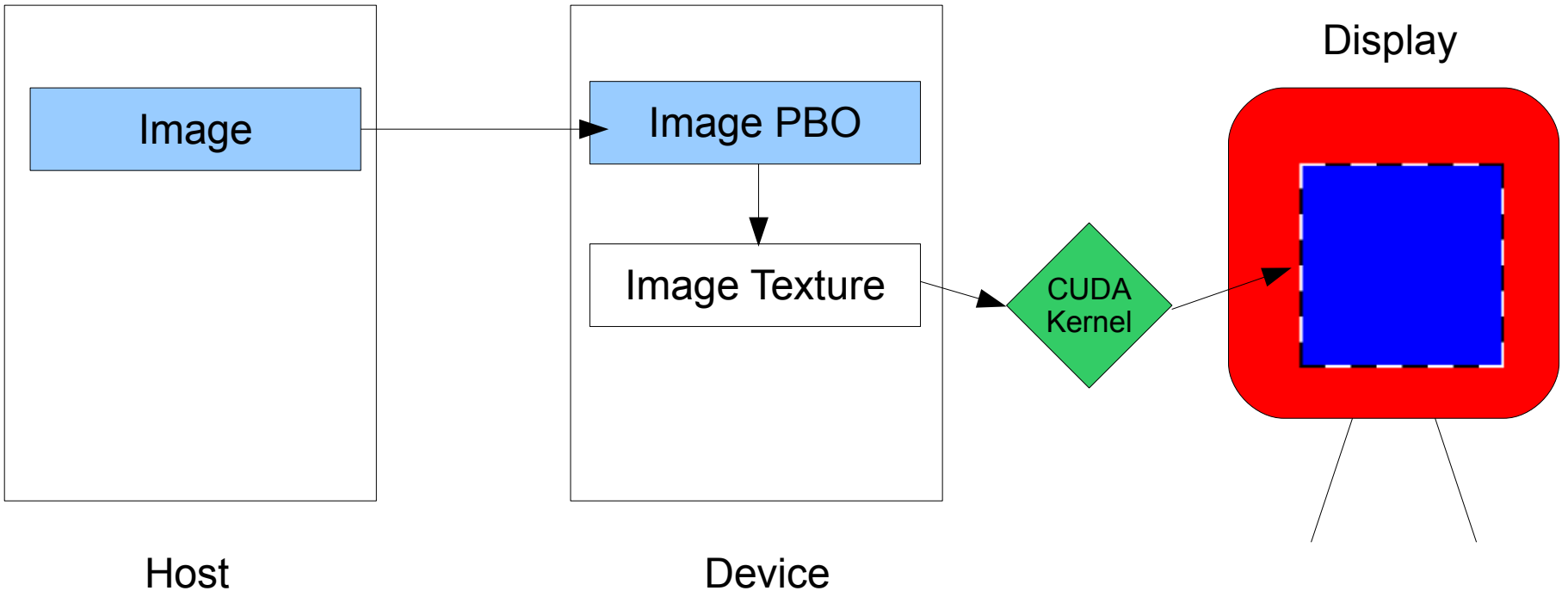
OpenGL: ejemplo 4

- Rendering de una textura (ejemplo del redbook usando PBO)



OpenGL: ejemplo 4

- Ejercicio: Modificar los valores de los texels usando un kernel



OpenGL: ejemplo 5

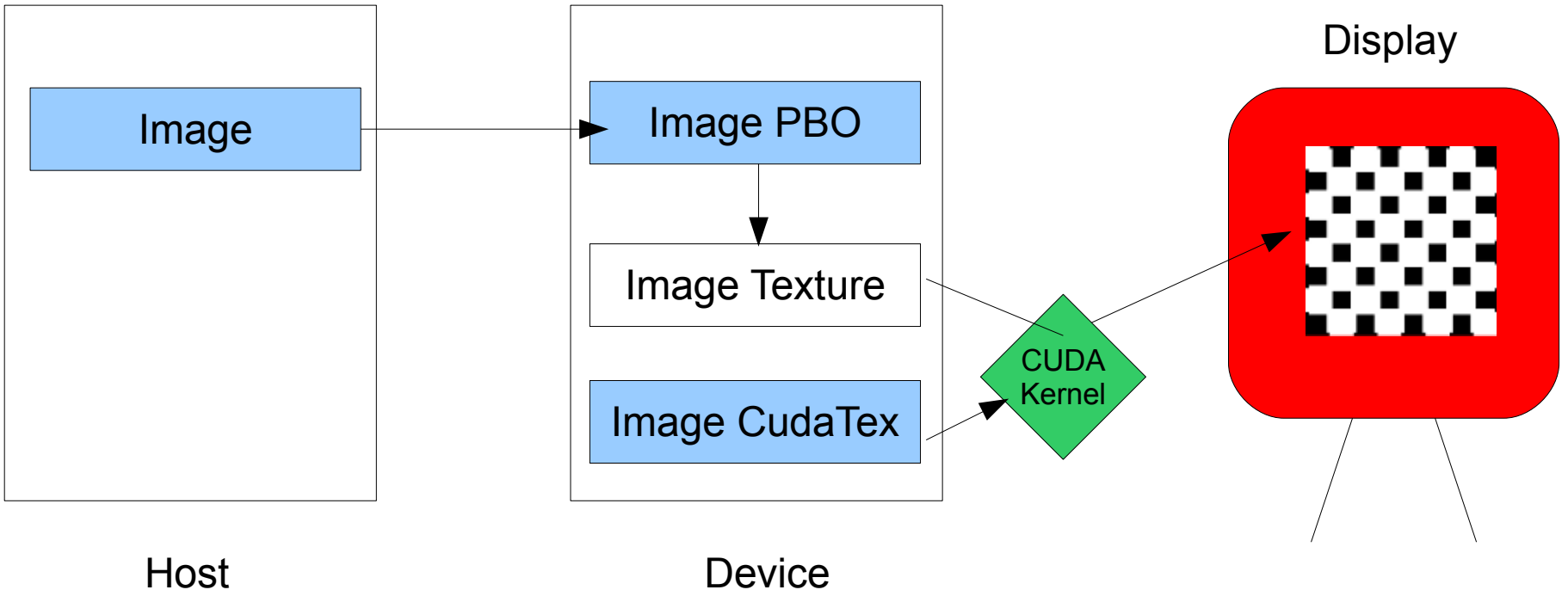
- Ejercicio: codificar un kernel de filtraje

0	1	0
1	1	1
0	1	0

OpenGL: ejemplo 5

- Ejercicio: codificar un kernel de filtraje

0	1	0
1	1	1
0	1	0



OpenGL: Texturas

- Rendering de una textura (a partir de una imagen)

OpenGL: Imágenes

- Sin compresión
- Con compresión

OpenGL: Imágenes

- Sin compresión: son usados para reducir el espacio para almacenamiento y aumentar la velocidad de transmisión
- Compresión sin pérdida: cada píxel original puede ser reproducido exactamente
  - Ejemplo: formato GIF (8 bits)
- Compresión con pérdida: cada píxel es reproducido aproximadamente
  - Permite mejores tasa de compresión
  - Ejemplo: formato JPEG (8/24 bits). No soporta transparencia

OpenGL: Imágenes

- Sin compresión: son usados para reducir el espacio para almacenamiento y aumentar la velocidad de transmisión
- Compresión sin pérdida: cada píxel original puede ser reproducido exactamente
  - Ejemplo: formato GIF (8 bits)
- Compresión con pérdida: cada píxel es reproducido aproximadamente
  - Permite mejores tasa de compresión
  - Ejemplo: formato JPEG (8/24 bits). No soporta transparencia

OpenGL: Imágenes

- Sin compresión: familia PNM (portable anymap format)
  - PPM (pixmap)
    - 1 bit por píxel (P1/P4)
  - PGM (graymap)
    - 8 bits por píxeles (P2/P5)
  - PBM (bitmap)
    - 24 bits por píxeles (P3/P6)

OpenGL: Imágenes

- Sin compresión: familia PNM
  - PPM (pixmap)
    - 1 bit por píxel (P1/P4)

```
P1
# This is an example bit map file j.pbm
6 10
0 0 0 0 1 0
0 0 0 0 1 0
0 0 0 0 1 0
0 0 0 0 1 0
0 0 0 0 1 0
0 0 0 0 1 0
1 0 0 0 1 0
0 1 1 1 0 0
0 0 0 0 0 0
0 0 0 0 0 0
```

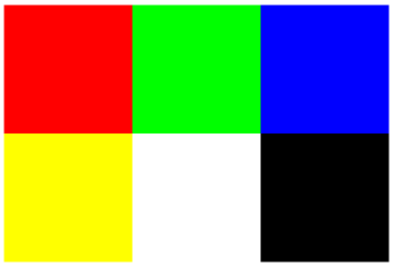
OpenGL: Imágenes

- Sin compresión: familia PNM
  - PGM (graymap)
    - 8 bits por píxeles (P2/P5)


```
P2
# feep.pgm from NetPBM man page on PGM
24 7
15
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  3  3  3  3  0  0  7  7  7  7  0  0 11 11 11 11  0  0 15 15 15 15  0
0  3  0  0  0  0  0  7  0  0  0  0  0 11  0  0  0  0  0 15  0  0 15  0
0  3  3  3  0  0  0  7  7  7  0  0  0 11 11 11  0  0  0 15 15 15 15  0
0  3  0  0  0  0  0  7  0  0  0  0  0 11  0  0  0  0  0 15  0  0  0  0
0  3  0  0  0  0  0  7  7  7  7  0  0 11 11 11 11  0  0 15  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
```

OpenGL: Imágenes

- Sin compresión: familia PNM
  - PBM (bitmap)
    - 24 bits por píxeles (P3/P6)



```
P3
#the P3 means colors are in ascii, then 3 columns and 2 rows, then 255 for max color, then RGB triplets
3 2
255
255 0 0
0 255 0
0 0 255
255 255 0
255 255 255
0 0 0
```

```
P6
#any comment string
3 2
255
!@#$%^&*()_+|{}:"<
```

OpenGL: Imágenes

- Ejercicio
  - Codifique una rutina para cargar una imagen de un archivo PPM
  - Codifique filtros para procesamiento de imágenes

0	0	0	0	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	0	0	0	0

Blur

	0	1	0	
	1	-4	1	
	0	1	0	

Edge detect

0	0	0	0	0
0	0	-1	0	0
0	-1	5	-1	0
0	0	-1	0	0
0	0	0	0	0

Sharpen

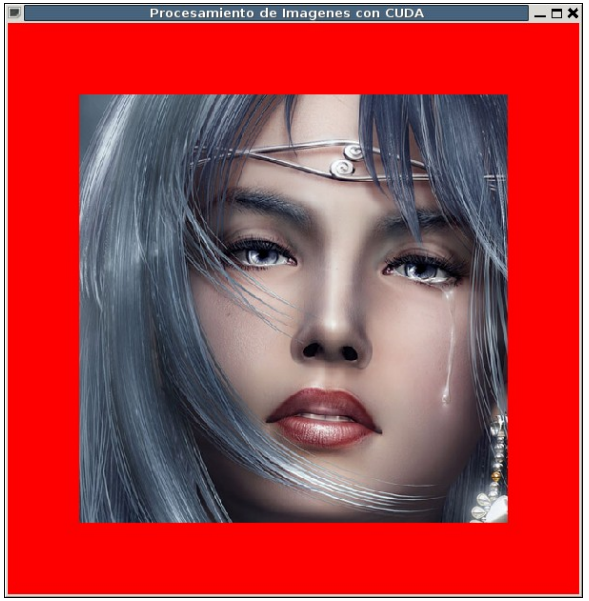
	0	0	0	
	-1	1	0	
	0	0	0	

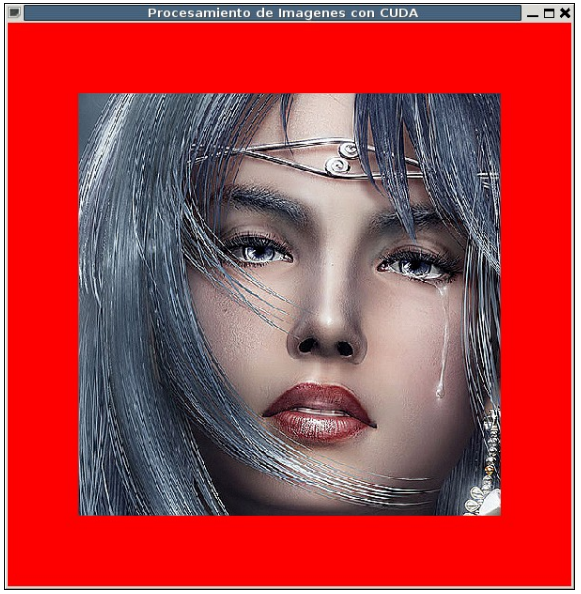
Edge enhance

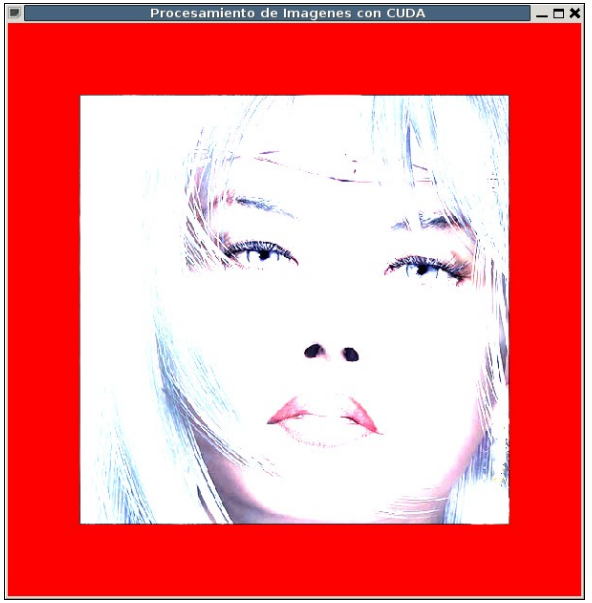
	-2	-1	0	
	-1	1	1	
	0	1	2	

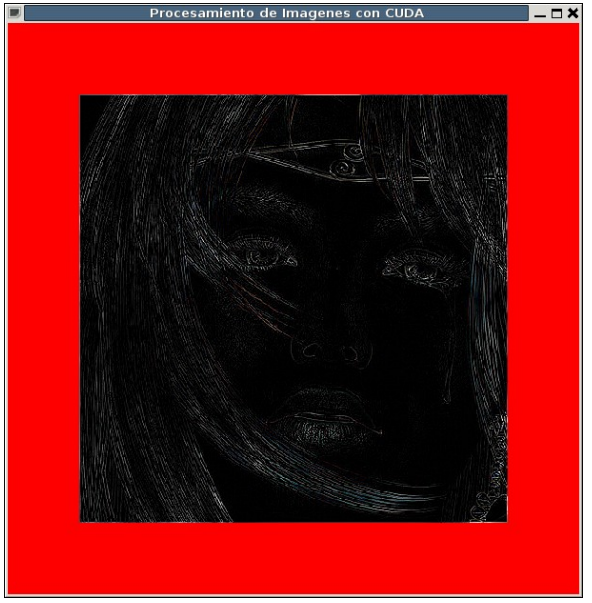
Emboss

OpenGL: Imágenes









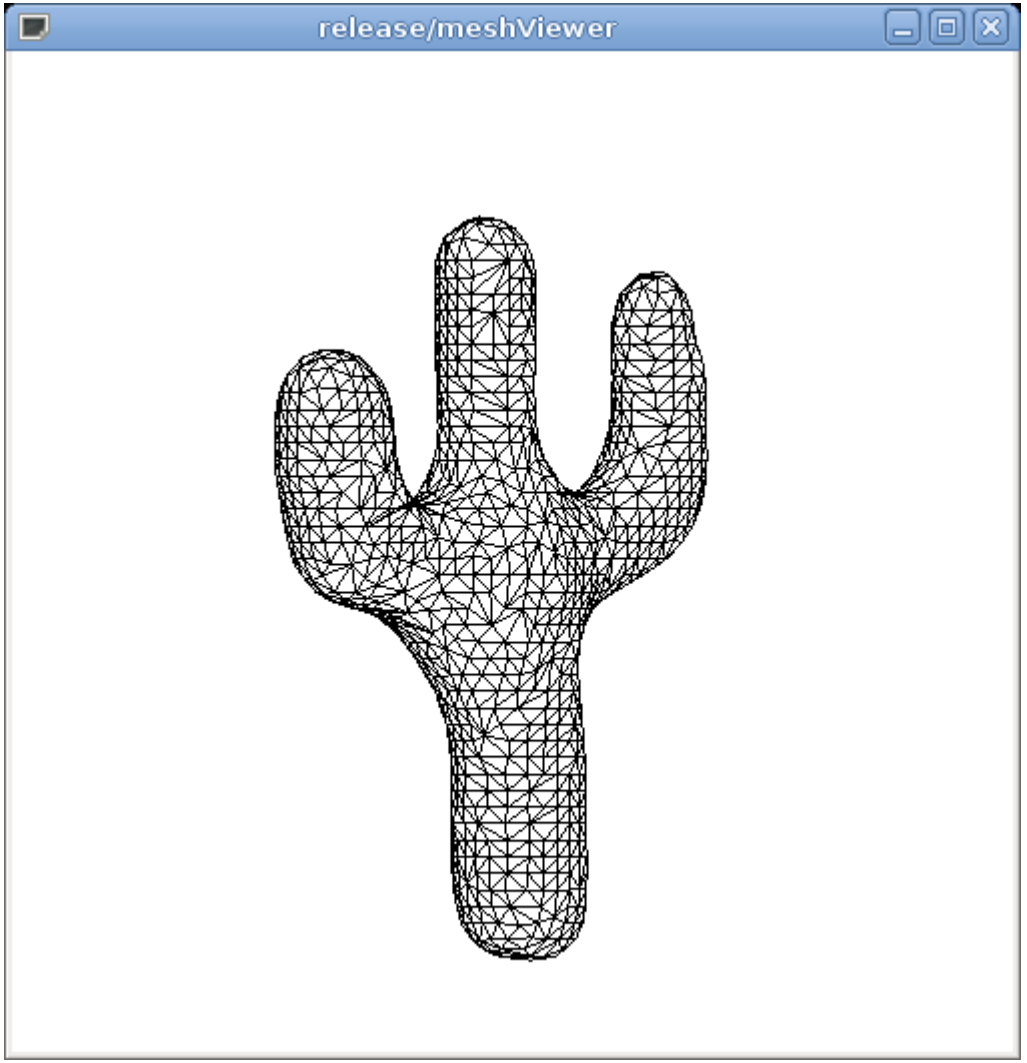
OpenGL 3D

OpenGL 3D

- Modificar la geometría (los vértices) de un modelo

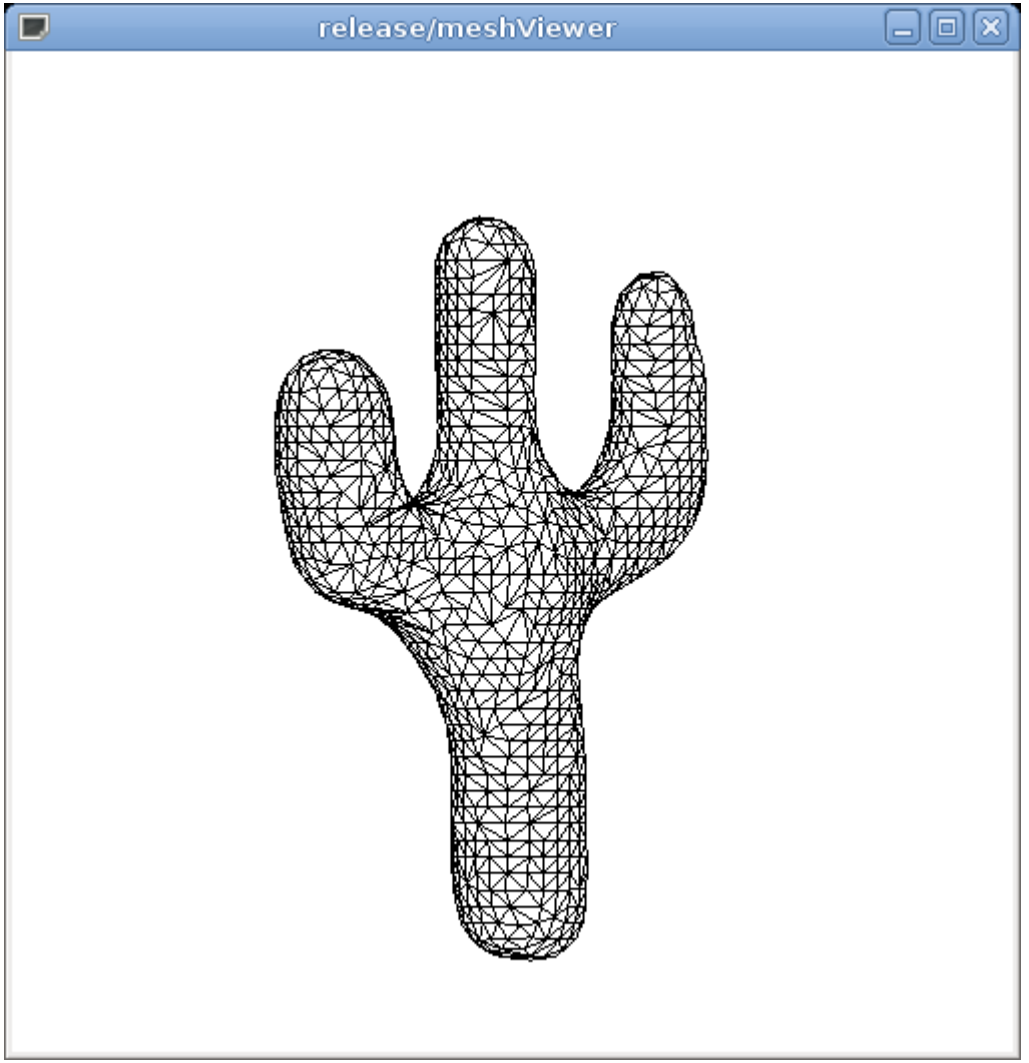
OpenGL 3D

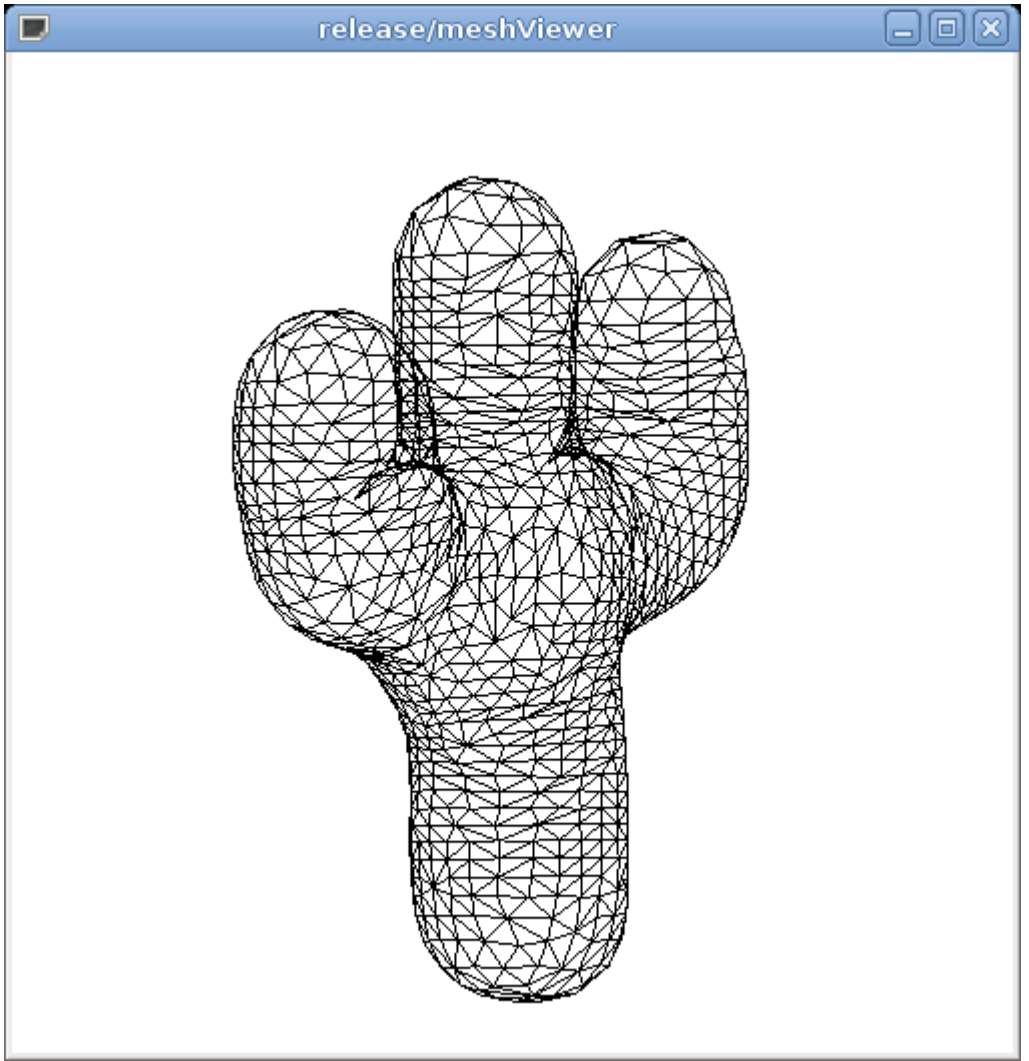
- Modificar la geometría (los vértices) de un modelo



OpenGL 3D

- Modificar la geometría (los vértices) de un modelo

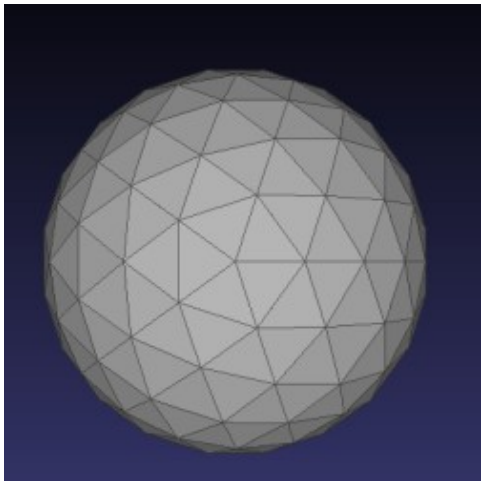


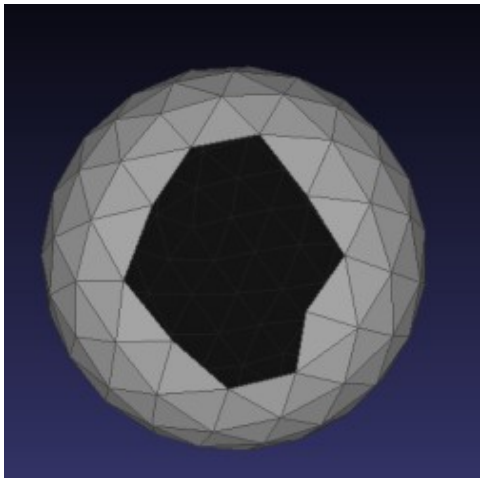


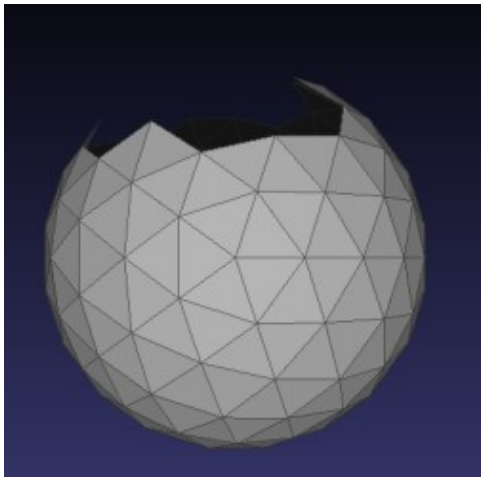
Modelos 3D

Modelos 3D

Modelos 3D



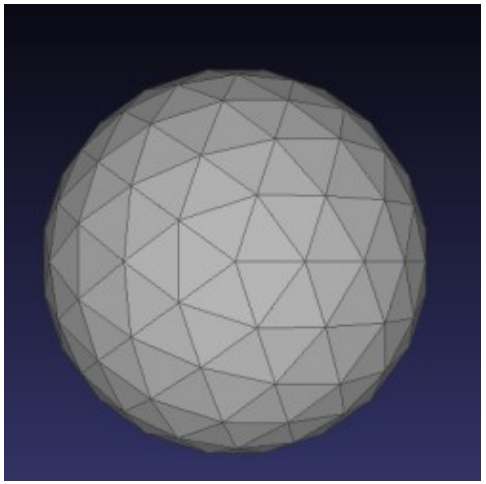


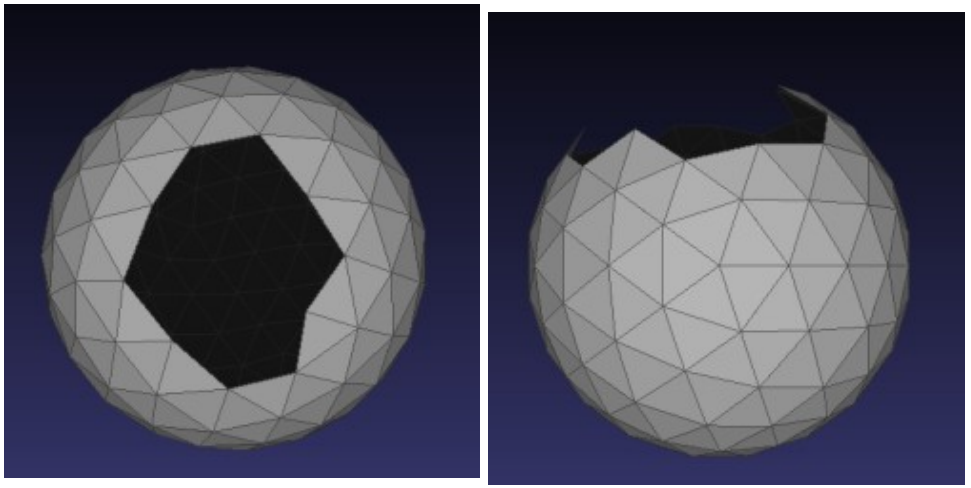


Modelos de superficie

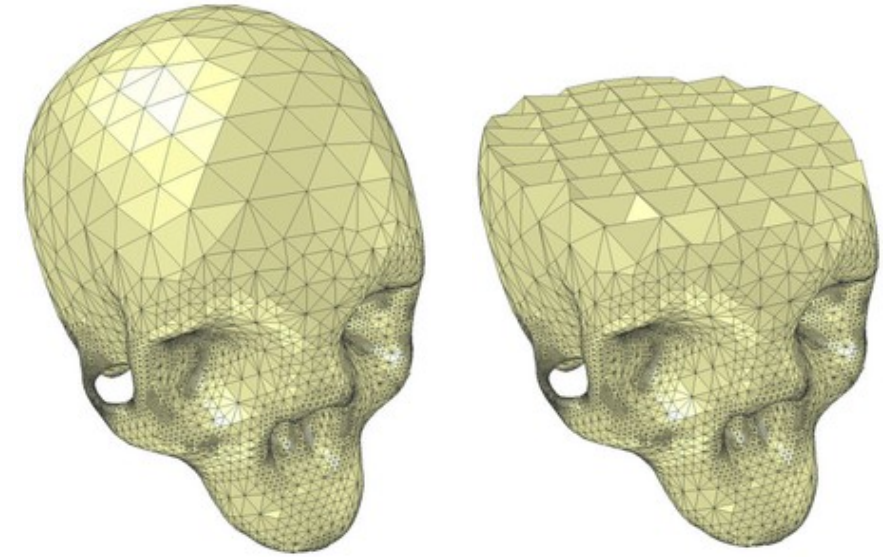
Modelos volumétricos

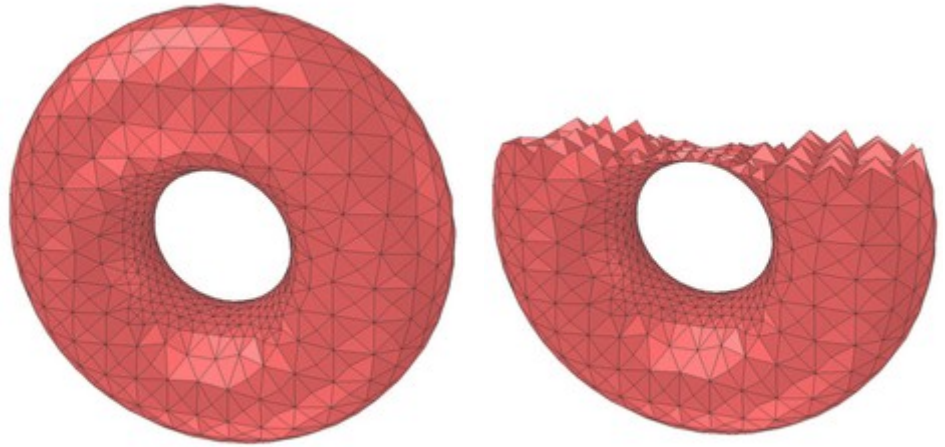
Modelos 3D





Modelos de superficie

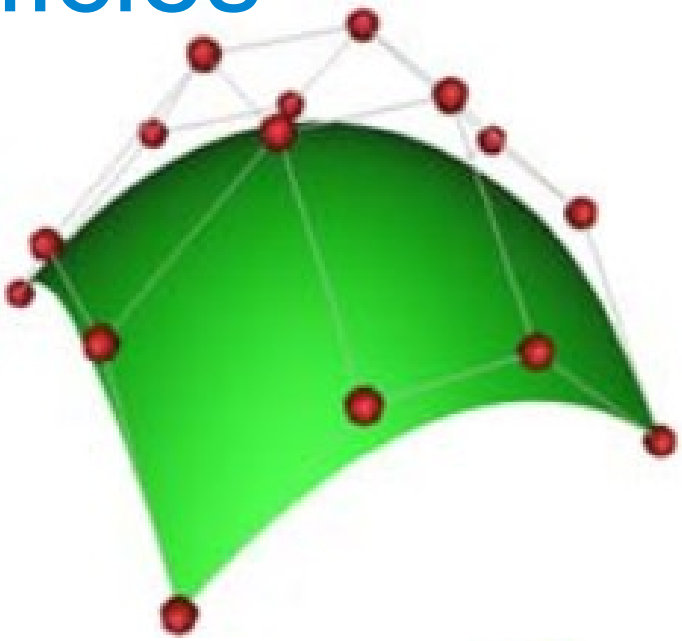


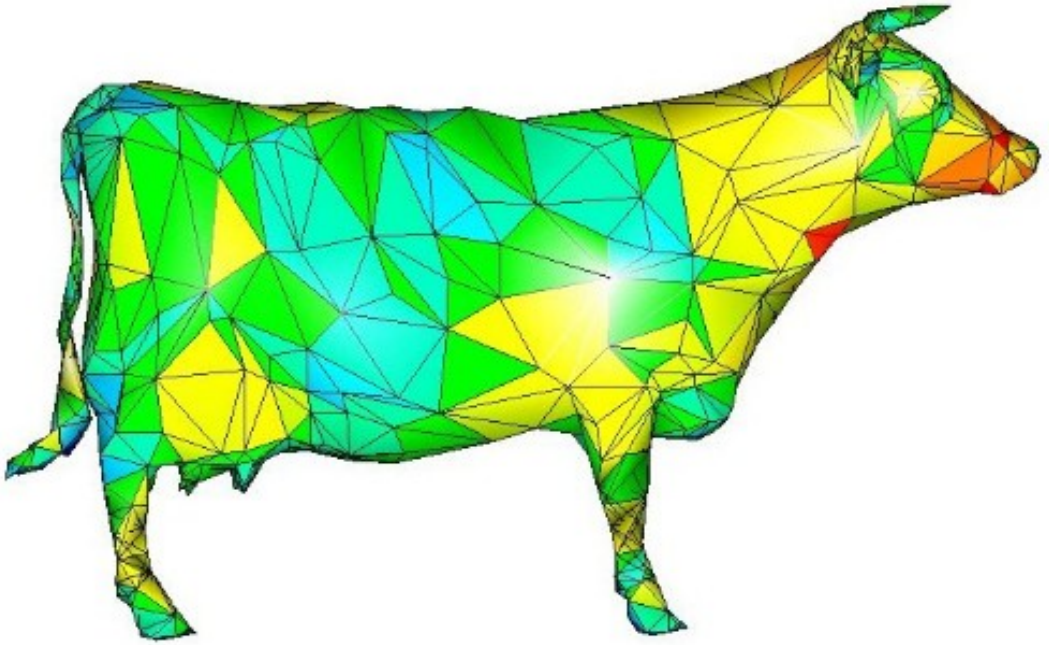


Modelos volumétricos

Modelos de superficies

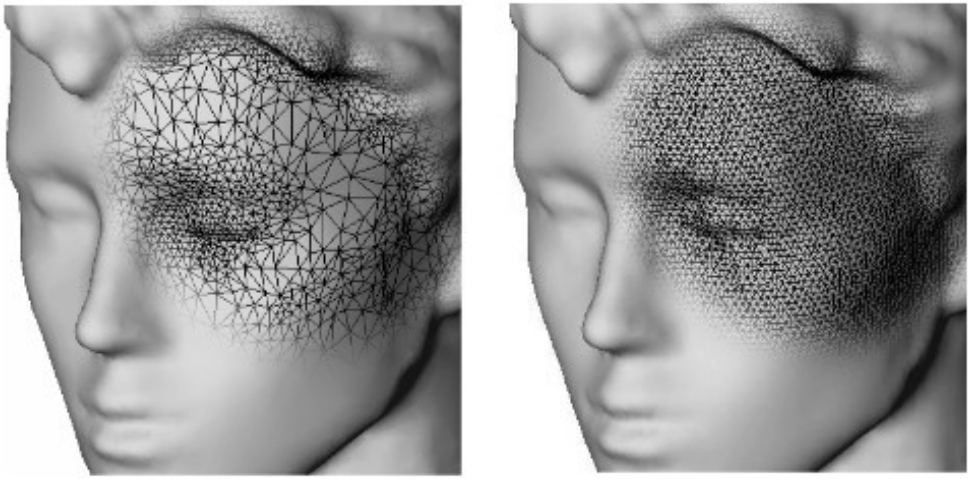
- Paramétricas
- Puntos
- Polígonos (mallas)





Mallas

- Ventajas
  - Generalidad para aproximar objetos de topología arbitraria
    - Exactitud arbitraria
  - Hardware especializado
- Desventajas
  - Gran número de polígonos (memoria)
    - Superficies curvas complejas y con muchos detalles
  - Superficies son aproximadas dentro de una precisión fija

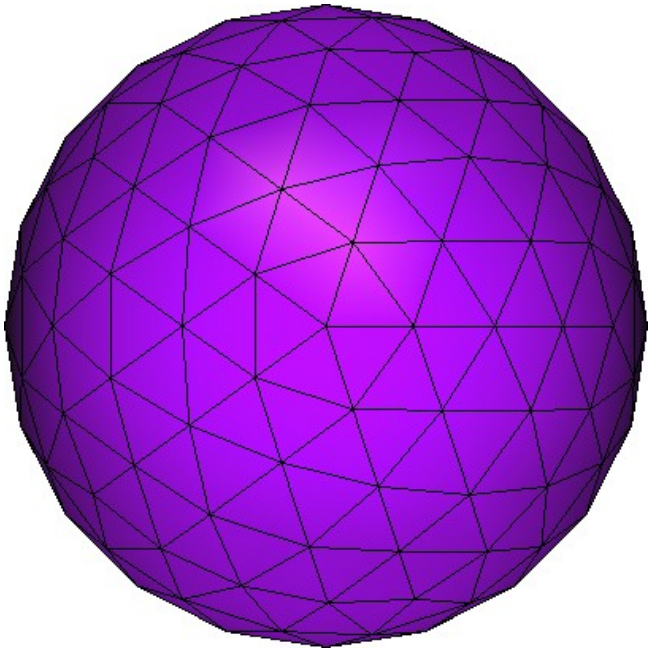


Mallas

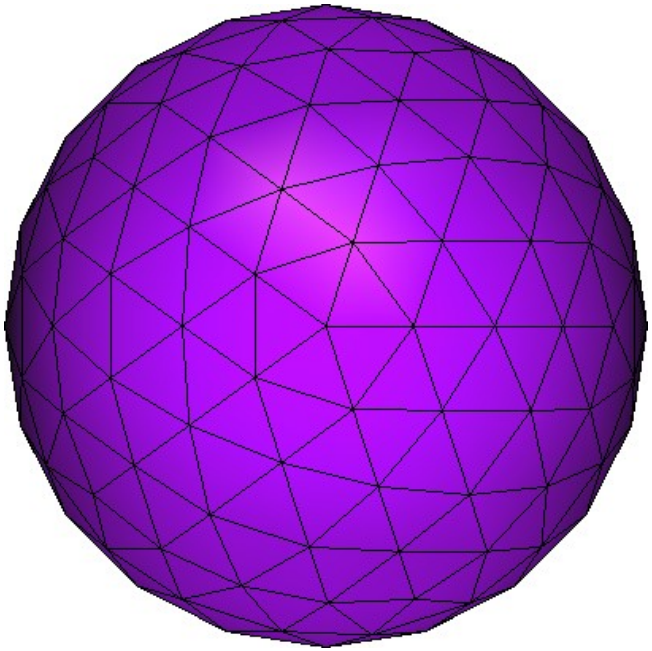
- Aproximación lineal por partes de superficies complejas
- Hoy en día las mallas de triángulos son altamente populares
- Buena alternativa para superficies spline tradicionales
  - Su simplicidad conceptual permite:
    - Procesamiento flexible y altamente eficiente
  - Bastante usadas en CAD (computer-aided design)
  - Juegos por computador y producción de cine

Mallas

- Consiste de dos componentes



Mallas

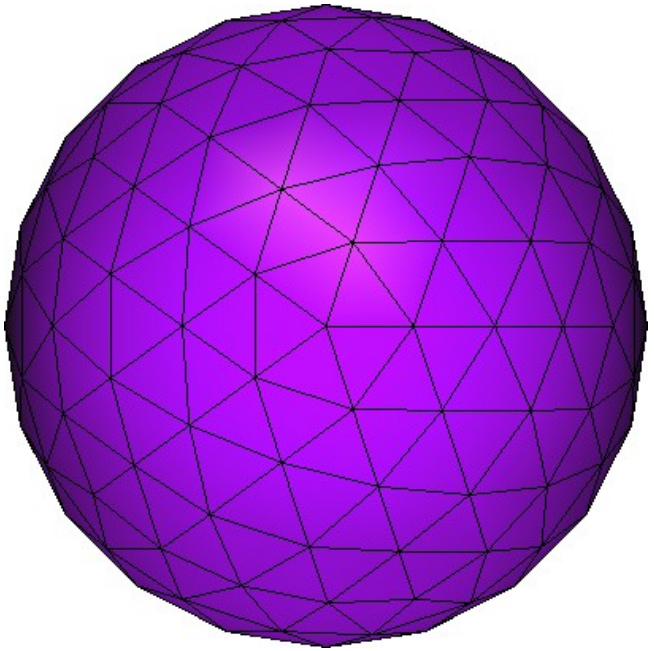


- Consiste de dos componentes
  - Geométrico

$$\mathcal{P} = \{\mathbf{p}_1, \ldots, \mathbf{p}_V\}, \quad \mathbf{p}_i := \mathbf{p}(v_i) = \begin{pmatrix} x(v_i) \\ y(v_i) \\ z(v_i) \end{pmatrix} \in \mathbb{R}^3$$

Las posiciones de los vértices definen la geometría de la superficie

Mallas



- Consiste de dos componentes
  - Geométrico

$$\mathcal{P} = \{\mathbf{p}_1, \ldots, \mathbf{p}_V\}, \quad \mathbf{p}_i := \mathbf{p}(v_i) = \begin{pmatrix} x(v_i) \\ y(v_i) \\ z(v_i) \end{pmatrix} \in \mathbb{R}^3$$

Las posiciones de los vértices definen la geometría de la superficie

  - Topológico

$$\mathcal{V} = \{v_1, \ldots, v_V\}$$

$$\mathcal{F} = \{f_1, \ldots, f_F\}, \quad f_i \in \mathcal{V} \times \mathcal{V} \times \mathcal{V}$$

$$\mathcal{E} = \{e_1, \ldots, e_E\}, \quad e_i \in \mathcal{V} \times \mathcal{V}$$

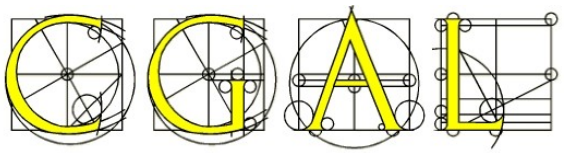
La adyacencia de vértices, aristas y polígonos define la topología de la superficie

Mallas

- Estructuras de datos
  - Punteros para lista de vértices
  - Punteros para lista de aristas
  - Winged-Edge
  - Half-Edge
  - Face-Edge, Quad-Edge, Radial-Edge

Mallas

- La descripción de la ED halfedge es simple, sin embargo, su implementación eficiente (tiempo/memoria), robusta y fácil de usar no lo es
- Se sugiere usar implementaciones opensource
  - Bibliotecas C++
    - CGAL (www.cgal.org)
    - OpenMesh (www.openmesh.org)



Mallas

- Formatos de archivos
  - Categorías: Ascii o Binario
  - Tipos
    - PLY, STL, OFF, OBJ, 3DS, COLLADA, PTX, V3D, PTS, APTS, XYZ, GTS, TRI, ASC, X3D, X3DV, VRML, ALN

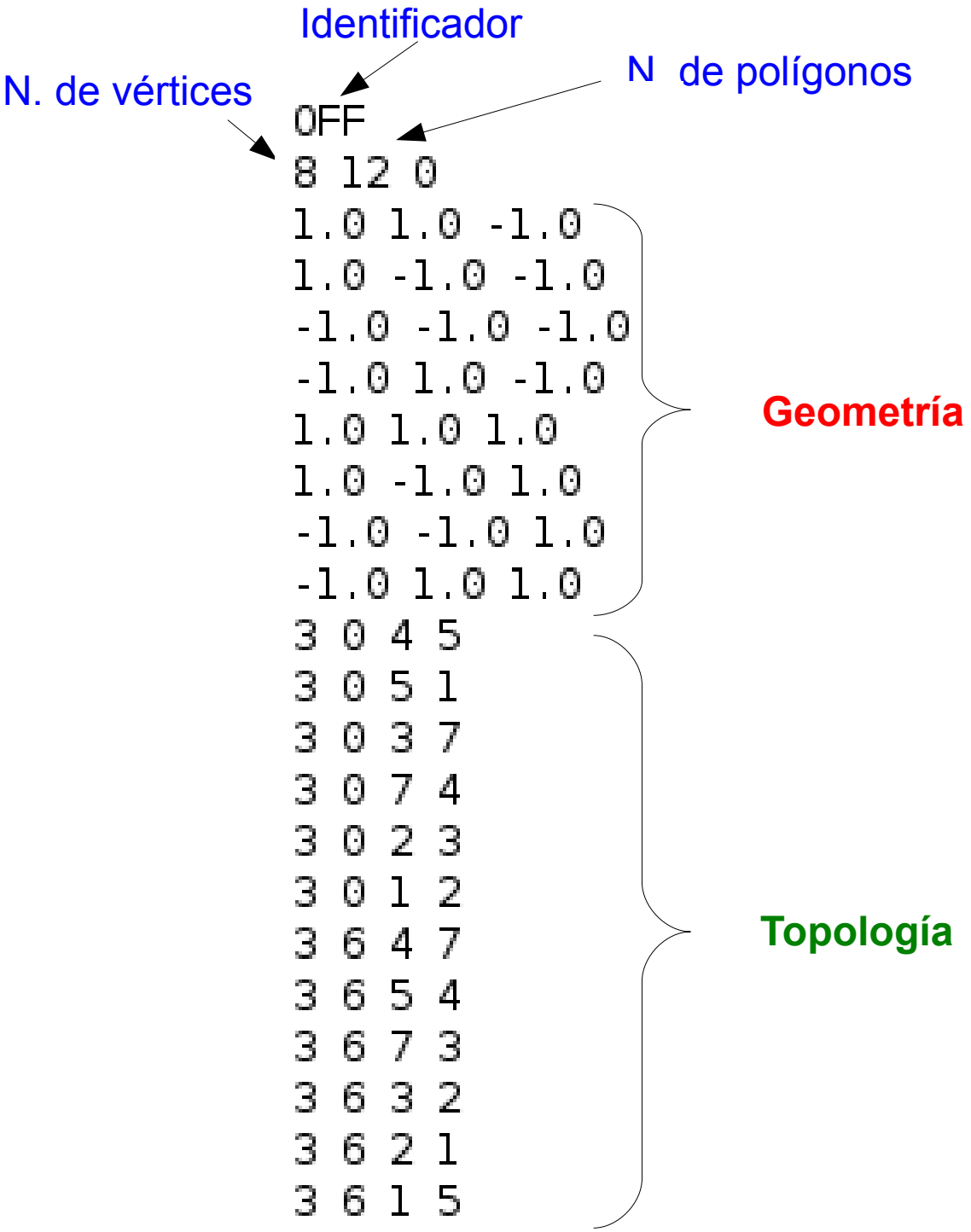
Mallas

- Formato OFF

```
OFF
8 12 0
1.0 1.0 -1.0
1.0 -1.0 -1.0
-1.0 -1.0 -1.0
-1.0 1.0 -1.0
1.0 1.0 1.0
1.0 -1.0 1.0
-1.0 -1.0 1.0
-1.0 1.0 1.0
3 0 4 5
3 0 5 1
3 0 3 7
3 0 7 4
3 0 2 3
3 0 1 2
3 6 4 7
3 6 5 4
3 6 7 3
3 6 3 2
3 6 2 1
3 6 1 5
```

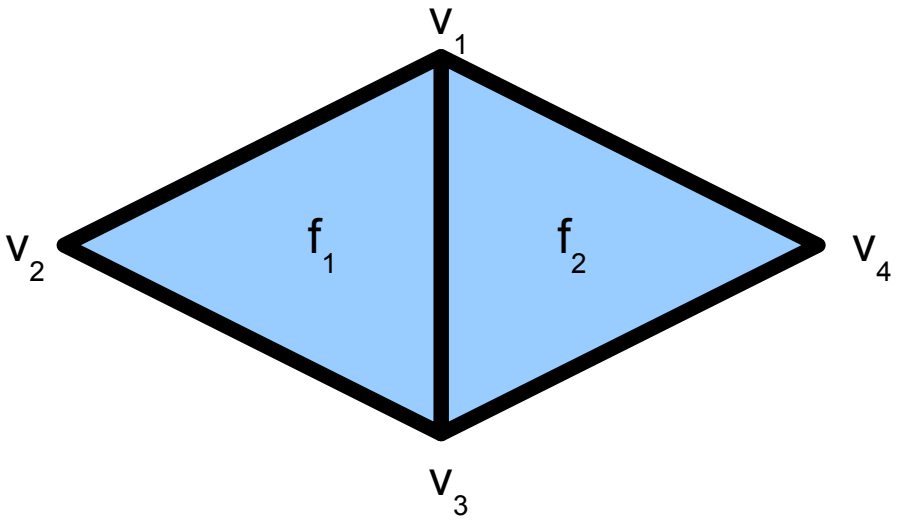
Mallas

- Formato OFF



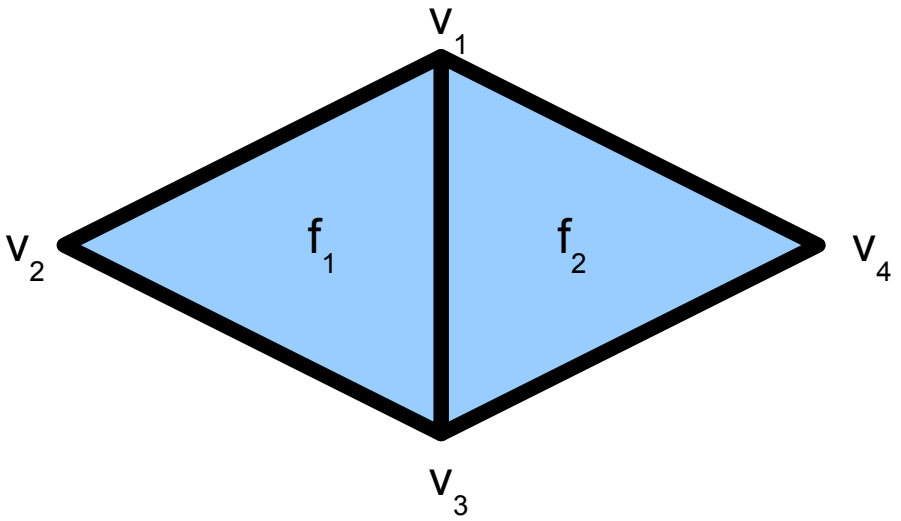
Mallas

- ED simple



Mallas

- ED simple



V = {v1={x1,y1,z1}, v2={x2,y2,z2}, v3={x3,y3,z3}, v4={x4,y4,z4}}

F = {idv1, idv2, idv3, idv1, idv3, idv4}

FV = {idv1={f1,f2}, idv2={f1}, idv3={f2,f1}, idv4={f2}}

El futuro

El presente

- Tradicionalmente, para mejorar el desempeño solo se requería cambiar de hardware
  - Sin embargo, en la actualidad ...
    - ~~Hardware~~
    - Software (oportunidad para programadores!)
    - http://arstechnica.com/articles/paedia/gpu-sweeney-interview.ars
    - http://www.sciencedaily.com/releases/2008/09/080916155058.htm

Recursos

- Libros online
  - GPU Gems: Programming Techniques, Tips, and Tricks for Real-Time Graphics
    - http://http.developer.nvidia.com/GPUGems/gpugems_part01.html
  - GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation
    - http://http.developer.nvidia.com/GPUGems2/gpugems2_part01.html
  - The Cg Tutorial
    - http://http.developer.nvidia.com/CgTutorial/cg_tutorial_chapter01.html

Recursos

- Code
  - http://www.gpgpu.org/
  - http://www.graphicshardware.org/
  - http://www.shadertech.com