C.U. jueves, 10 de Marzo de 2022

Computación Gráfica

The Course Presentation: OpenGL

Presentación del Curso: OpenGL

Mg. Johnny R. Avendaño Q.

Lic. John Ledgard Trujillo Trejo

"Una imagen vale más que mil palabras"
Anónimo

Universidad Nacional Mayor de San Marcos

Facultad de Ingeniería de Sistemas e Informática

MOTIVACION: Aplicaciones OpenGL

- **3DMax:** Aplicación profesional para el modelado 3D, animación y renderizado. Ejemplo de uso: El Último Samurai.

- **HollywoodFx:** Aplicación para realización de efectos de video digitales 3D, transiciones, composiciones y animación.

MOTIVACION: Aplicaciones OpenGL

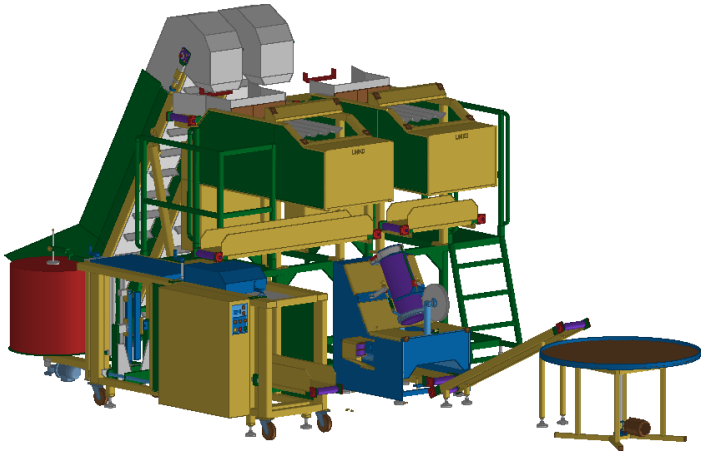
- **Inferno:** Aplicación comercial para la creación de efectos visuales 3D para películas.

- **Chief Architect:** Aplicación CAD orientada a objetos para el diseño arquitectónico.

MOTIVACION: Aplicaciones OpenGL

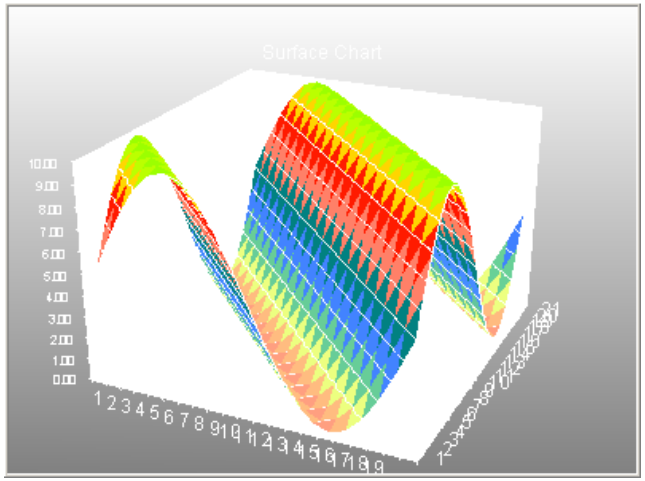
- **12d Model:** Programa de diseño para el modelado 3D de terrenos de construcción, civiles, etc.

- **VariCAD:** Aplicación CAD para la ingeniería mecánica.

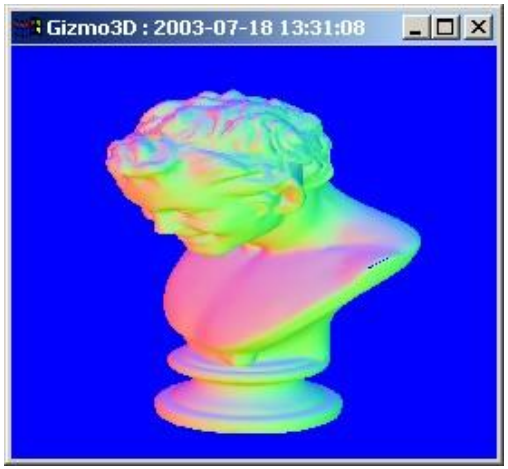


MOTIVACION: Aplicaciones OpenGL

- **3D Charting Toolkit:** Componentes ActiveX que usan OpenGL.



- **Gizmo3D:** Aplicación de desarrollo de escenas gráficas 3D mediante C++.



MOTIVACION: Aplicaciones OpenGL

- **OpenMind 3D Engine:** Motor de juego escrito en su mayoría en Java.

- **Chess Commander**

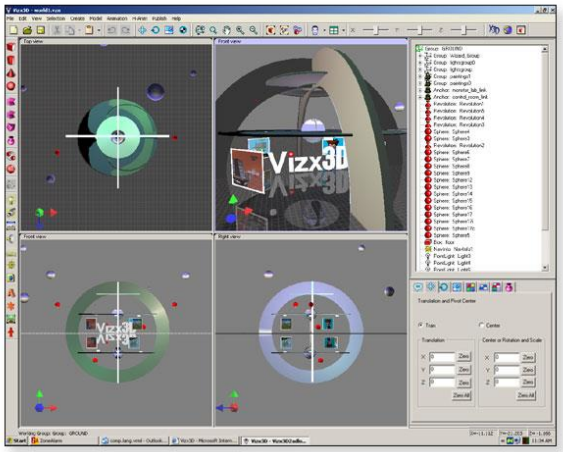
- **Deus EX**

- **HalfLife**

MOTIVACION: Aplicaciones OpenGL

- **Panorama3D:** Control ActiveX que permite la visualización de imágenes panorámicas directamente desde Internet.

- **Vizx3D:** Herramienta para el modelado visual 3D y la animación.



MOTIVACION: Aplicaciones OpenGL

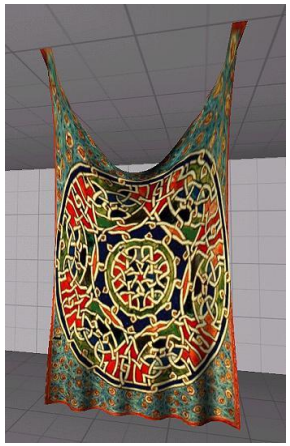
- **Cosmo Player:** Browser para web 3D.

- **Salva pantallas: AquaScape3D**

MOTIVACION: Aplicaciones OpenGL

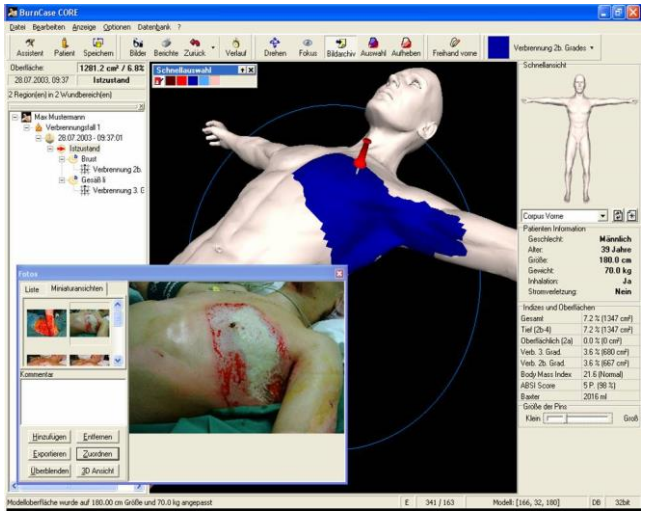
- **Aerofly:** Simulador de vuelo por radio-control.

- **ClothSim:** Simulador real de prendas de ropa.

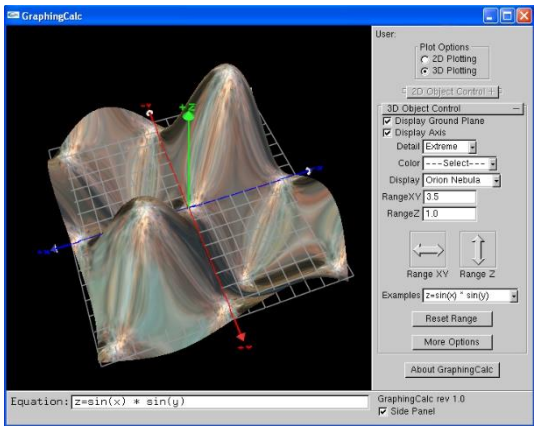


MOTIVACION: Aplicaciones OpenGL

- **Burn Case 3D:** Herramienta de modelado 3D de quemaduras en humanos.



- **GraphingCalc:** Programa de visualización matemática 2D, 3D.

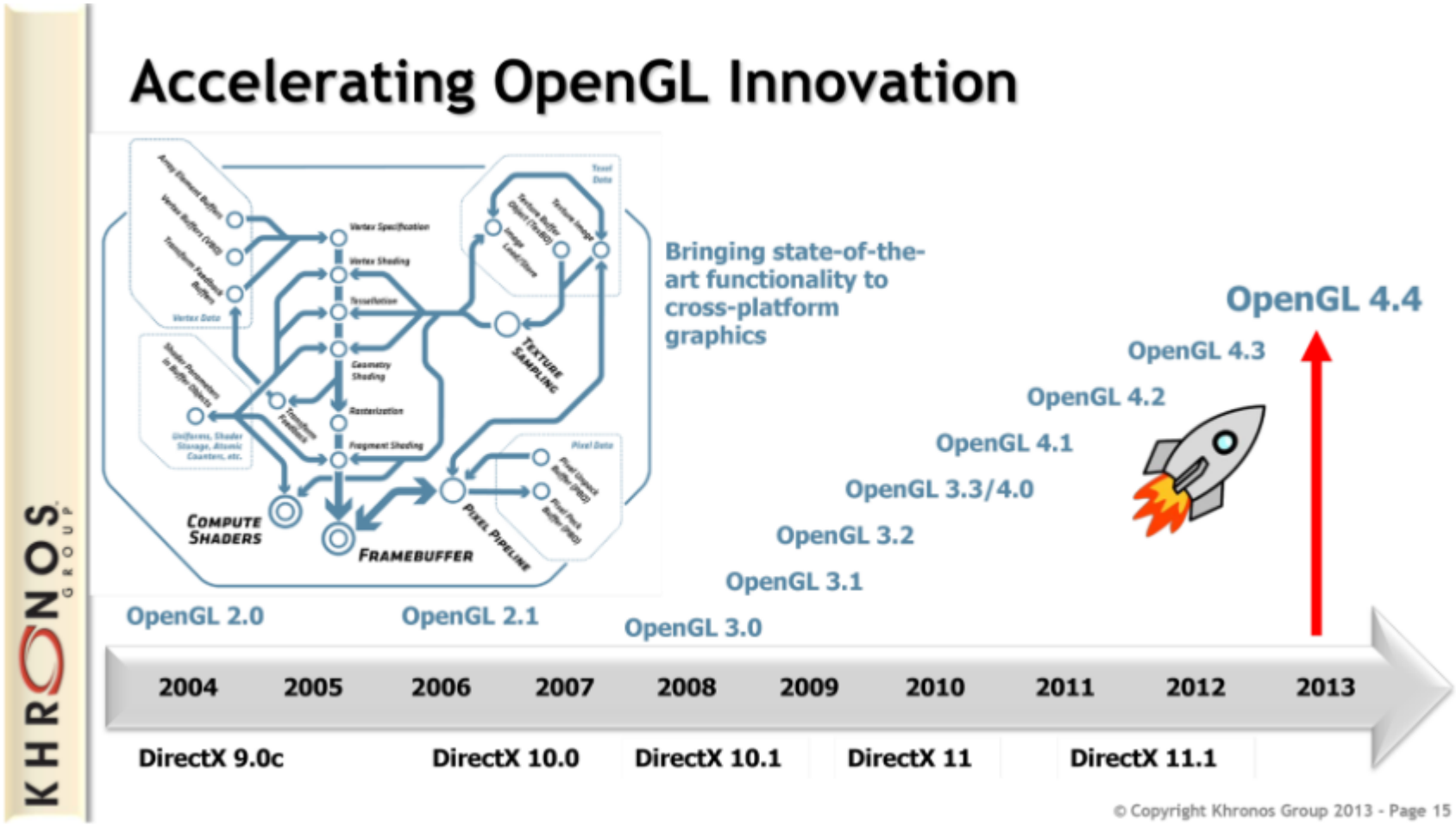


¿Que es OpenGL?

***OpenGL** es un Interfase de programación de aplicaciones (API) estándar. Es una biblioteca de trazado de gráficos de alto rendimiento, fácil de usar, intuitivo, portable, en el que el programador describe las llamadas necesarias a funciones y comandos para conseguir una escena, apariencia o efecto.*

- Desarrollada por Silicon Graphics en 1992.
- Evolución de la antigua Iris GL.
- Consiste en mas de 120 comandos de uso general que permiten crear escenas interactivas en 2D y 3D de alta calidad en tiempo real.

¿Evolución de OpenGL?



¿Evolución de OpenGL?

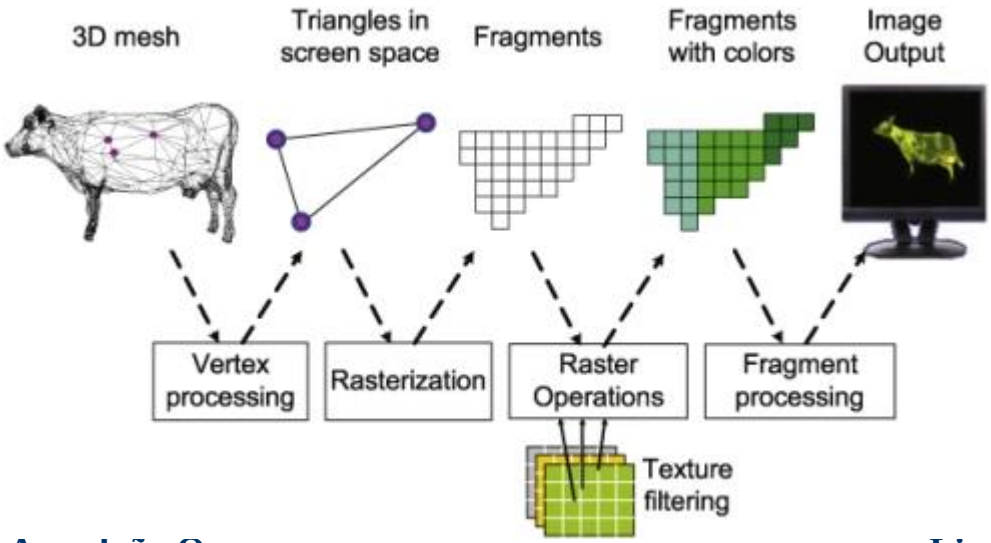
El hardware grafico moderno introdujo nuevas funcionalidades para crear un cambio de paradigma y aprovechar al máximo las capacidades de las GPUs actuales (Graphics Processing Units), pero manteniendo compatibilidad hacia atrás con las versiones previas.

OpenGL 1.0	1.0	1.1	1.2	1.3	1.4	1.5	(1992-2003)
OpenGL 2.0	2.0	2.1					(2004-2007)
OpenGL 3.0	3.0	3.1	3.2	3.3			(2008-2010)
OpenGL 4.0	4.0	4.1	4.2	4.3	4.4		(2010-201*)

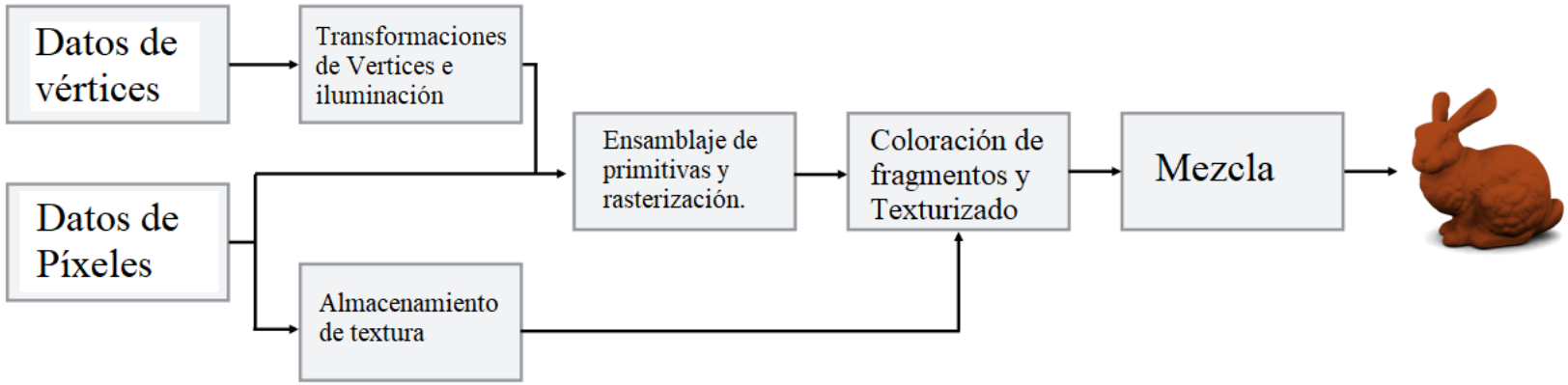
Figura 1: *Evolución de las versiones de OpenGL a través del tiempo. Actualizado a enero 2014.*

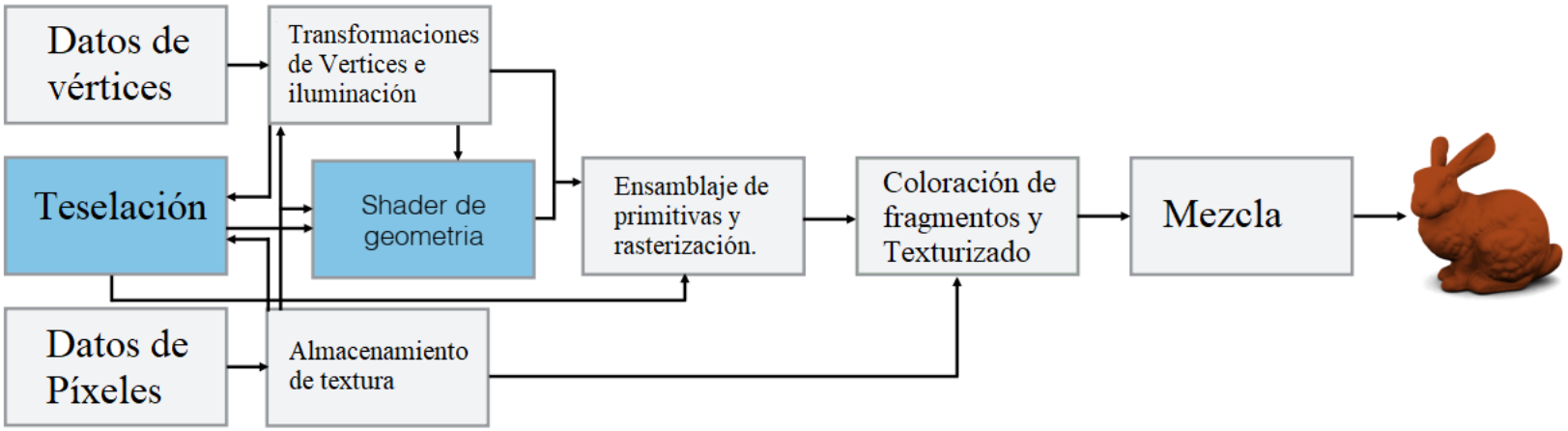
¿Evolución de OpenGL?

La versión inicial de OpenGL (versión 1.0) incluía un pipeline de funcionalidad fija. Esto significa que todas las funciones soportadas por OpenGL están bien definidas y una aplicación solo puede modificar los parámetros de entrada a dichas funciones como los valores de posición, color, entre otros. En un pipeline de funcionalidad fija, el orden de ejecución de las etapas operacionales es fijo.

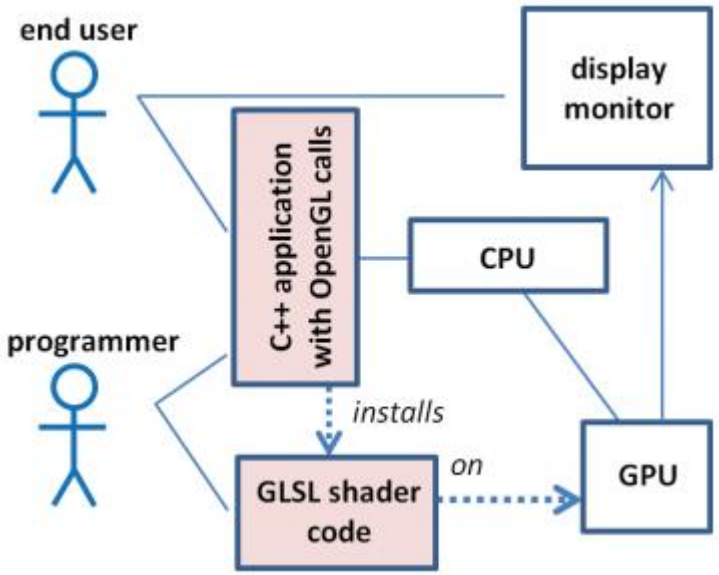


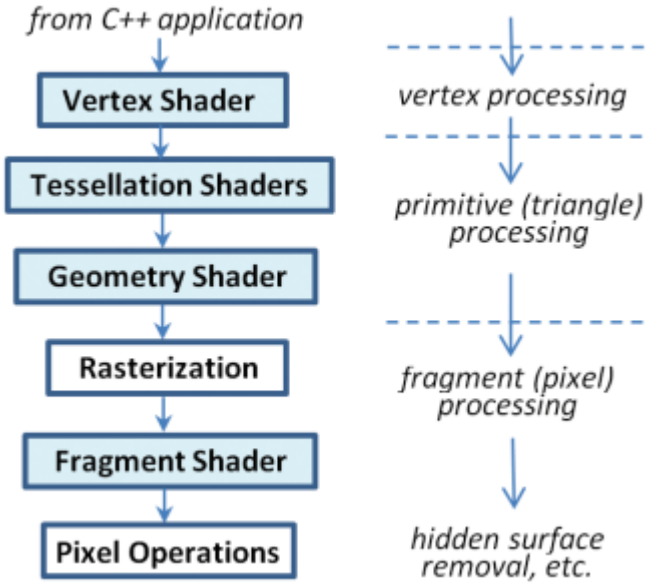
¿Evolución de OpenGL?



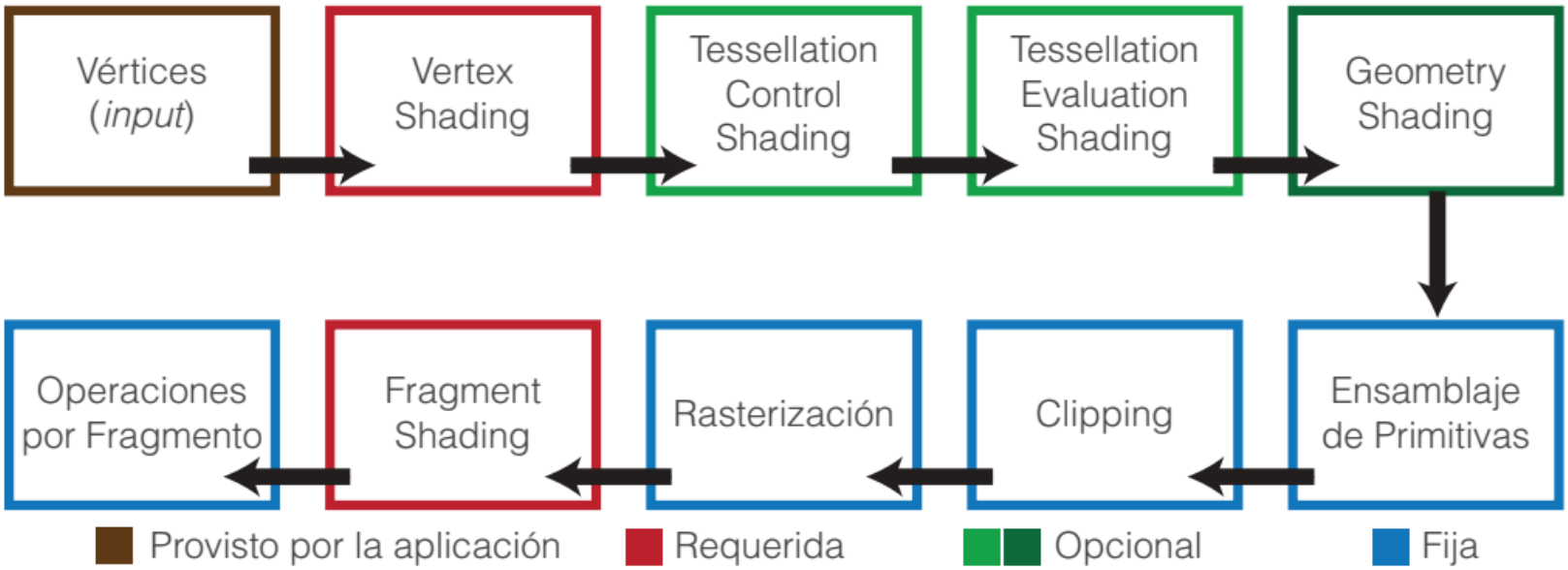


¿Evolución de OpenGL?





¿Evolución de OpenGL?



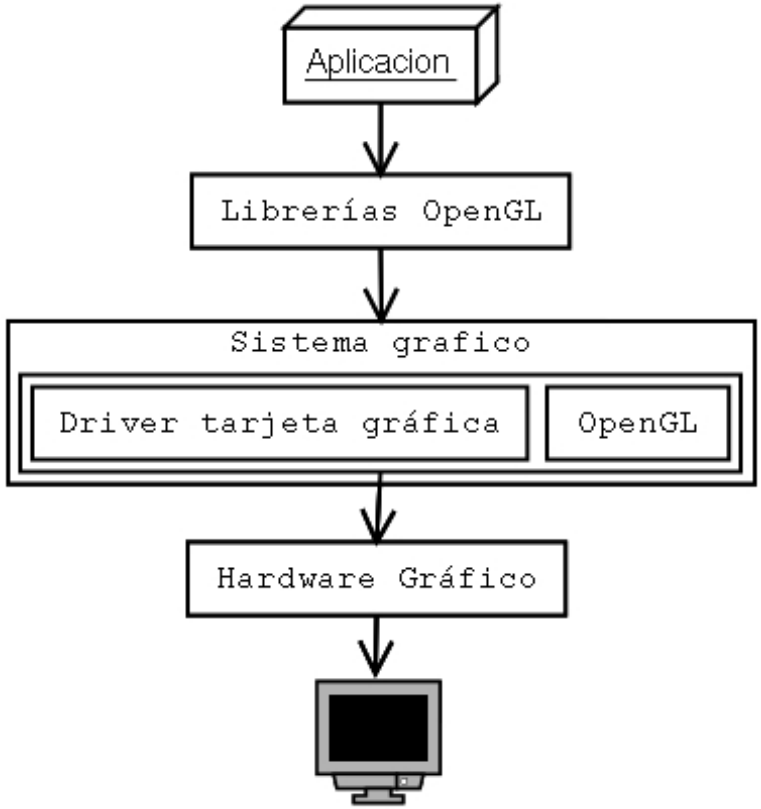
Una representación simplificada del pipeline gráfico de OpenGL mostrando cada una de sus etapas.

¿Qué no es OpenGL?

- Un sistema de ventanas
- Un manejador de eventos
- Un sistema de animación de objetos
- Un sistema que controle la interacción entre los objetos
- Un sistema de base de datos de objetos tridimensionales
- Etc, etc.

¿Características de OpenGL?

- **Portabilidad:** OpenGL es por diseño independiente del hardware, sistema operativo o sistema de ventanas.

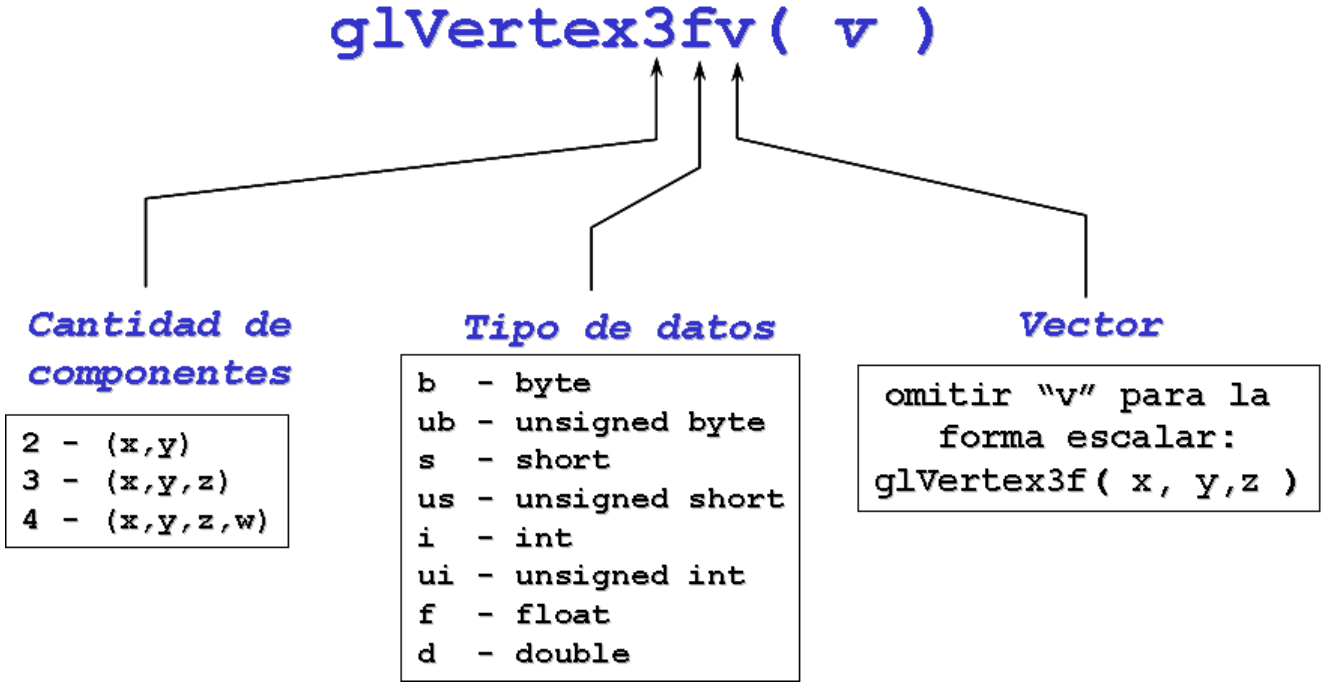


¿Características de OpenGL?

- **Perceptiva a la red:** Es posible separar la aplicación OpenGL en un servidor y un cliente que verdaderamente produzca los gráficos. Existe un protocolo para mover por la red los comandos OpenGL entre el servidor y el cliente.
- **Funciones para crear gráficos 2D y 3D:** OpenGL tiene funciones o primitivas con las que se podrá realizar modelado 3D, transformaciones, utilización de color e iluminación, sombras, mapeado de texturas, animación, movimiento borroso, etc.
- **Ausencia de comandos para modelos complejos:** no tiene comandos para describir modelos complejos (mapas, pájaros, moléculas, etc) sino que tiene primitivas que permiten dibujar puntos, líneas y polígonos.
- **Ejecución Inmediata:** Con OpenGL cualquier comando es ejecutado inmediatamente.

¿Características de OpenGL?

- **Sintaxis Común:** Todos los comandos de OpenGL utilizan la misma sintaxis. Todos ellos usan el prefijo **gl** y después la palabra que forma el comando con la primera letra en mayúscula.



¿Características de OpenGL?

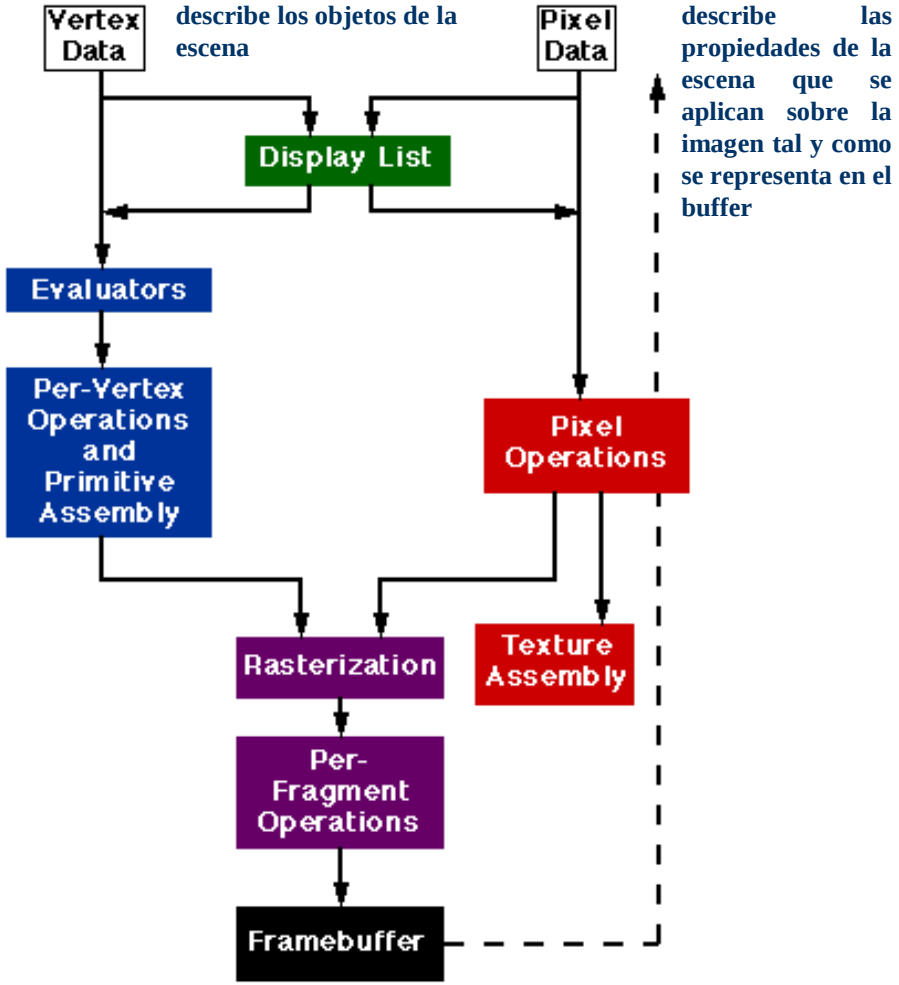
- **Máquina de estados:** Las aplicaciones se pueden poner en varios estados que permanecen en efecto hasta que se cambian. Por ejemplo el color, se puede poner un color y dibujar un objeto, y ese color seguirá activo hasta que se ponga otro color.

  OpenGL tiene las siguientes variables de estado:
  - Color
  - Vista actual
  - Transformaciones de proyecciones
  - Posiciones
  - Luces
  - Propiedades de material.

  Algunas de estas variables de estado se pueden habilitar o deshabilitar con los comandos glEnable() o glDisable(). Cada variable de estado tiene un valor por defecto disponible en todo momento a la consulta por parte del programador.

¿Características de OpenGL?

- **Variedad de lenguajes:** OpenGL puede ser invocado desde cualquiera de los siguientes lenguajes C, C++, Fortran, Ada, Java.
- **Pipeline de renderizado:** Con OpenGL se debe seguir un orden para la realización de las operaciones graficas necesarias para renderizar una imagen en la pantalla.



¿Características de OpenGL?

- **Tipos de datos propios:** Todos los tipos de Opengl tienen el sufijo **gl** y a continuación el tipo de C correspondiente.

Tipo en OpenGL	Tipo de C	Representación interna
GLbyte	char con signo	Entero de 8 bits
GLshort	short	Entero de 16 bits
GLint, GLsizei	int	Entero de 32 bits
GLfloat, GLclampf	float	Coma flotante de 32bits
GLdouble, GLclampd	double	Coma flotante de 64 bits
GLubyte, GLboolean	unsigned char	Entero de 8 bits sin signo
GLushort	unsigned short	Entero de 16 bits sin signo
GLuint, GLenum, GLbitfield	unsigned long	Entero de 32 bits sin signo

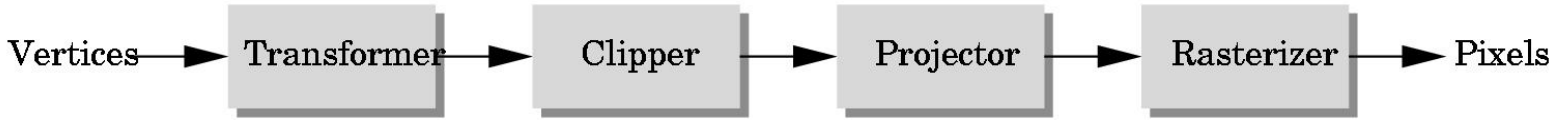
¿Qué provee la API OpenGL?

Una API debería proveer 6 grupos de funciones:

- Funciones de Primitivas (qué?)
- Funciones de Atributos (cómo?)
- Funciones de Visión (viewing)
- Funciones de Transformación
- Funciones de Entrada
- Funciones de Control (sistema de ventanas, errores)

¿Qué provee la API OpenGL?

- **Un conjunto de funciones** que controlan una maquina de estados la que determina como deben dibujarse los objetos simples (puntos, líneas, triángulos y polígonos) en pantalla.
- **Un sistema de proyección** que permite especificar objetos en 3 dimensiones y llevarlos a coordenadas de pantalla.
- **Un sistema de transformaciones** que permiten posicionar los objetos en el espacio.
- **Un sistema de asociación de imágenes a polígonos** para crear escenas realistas (textureado).
- En la version 2.0 provee un **lenguaje específico** para programar el hardware directamente (pixel y vertex shaders).



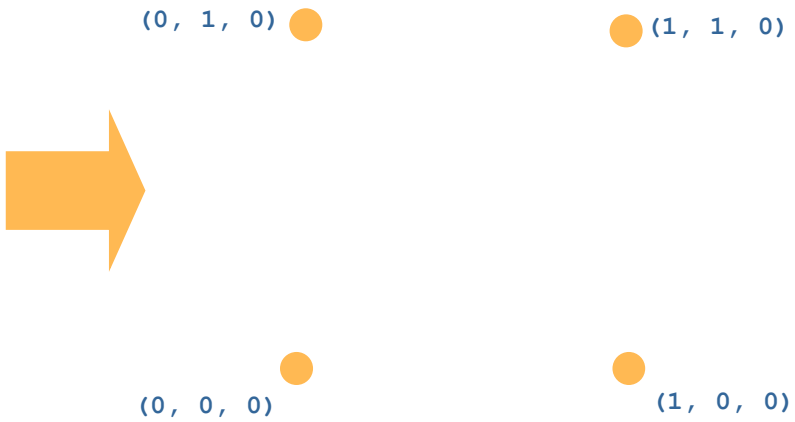
Librerías auxiliares de OpenGL

Biblioteca	*Archivos*	*Descripción*
Gl	Gl.h Opengl32.lib	Funciones del núcleo de OpenGL.
Glu	Glu.h Glu32.lib	**GLU (Graphics Utility Library)** contiene funciones para objetos comunes a dibujar como esferas o toros así como funciones de control de la cámara, entre otras. Estas funciones empiezan con las letras "glu".
Glut	Glut.h Glut32.dll Glut32.lib	**GLUT (GL Utility Toolkit)** contiene todas aquellas funciones que permitirán a nuestro programa ser interactivo, es decir, manejable desde el ratón y el teclado: Manejo de ventanas, Renderización de texto, Entrada por teclado, Menús. Estas funciones empiezan con las letras "glut".

Primitivas geométricas básicas

Modo	Descripción
GL_POINTS	Puntos individuales aislados
GL_LINES	Cada par de puntos corresponde a una recta
GL_LINE_STRIP	Segmentos de recta conectados
GL_LINE_LOOP	Segmentos de recta conectados y cerrados
GL_TRIANGLES	Cada tres puntos un triángulo
GL_QUADS	Cada cuatro puntos un cuadrado
GL_POLYGON	Un polígono

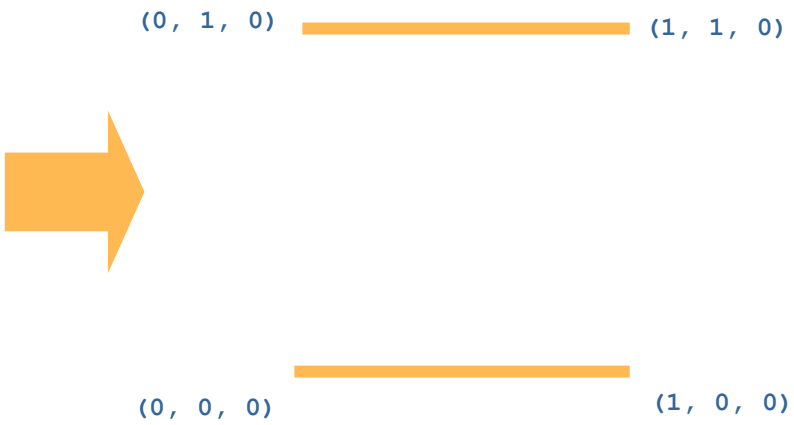
```
glBegin(GL_POINTS);
glVertex3f(0, 0, 0);
glVertex3f(0, 1, 0);
glVertex3f(1, 1, 0);
glVertex3f(1, 0, 0);
glEnd();
```



Primitivas geométricas básicas

Modo	Descripción
GL_POINTS	Puntos individuales aislados
GL_LINES	Cada par de puntos corresponde a una recta
GL_LINE_STRIP	Segmentos de recta conectados
GL_LINE_LOOP	Segmentos de recta conectados y cerrados
GL_TRIANGLES	Cada tres puntos un triángulo
GL_QUADS	Cada cuatro puntos un cuadrado
GL_POLYGON	Un polígono

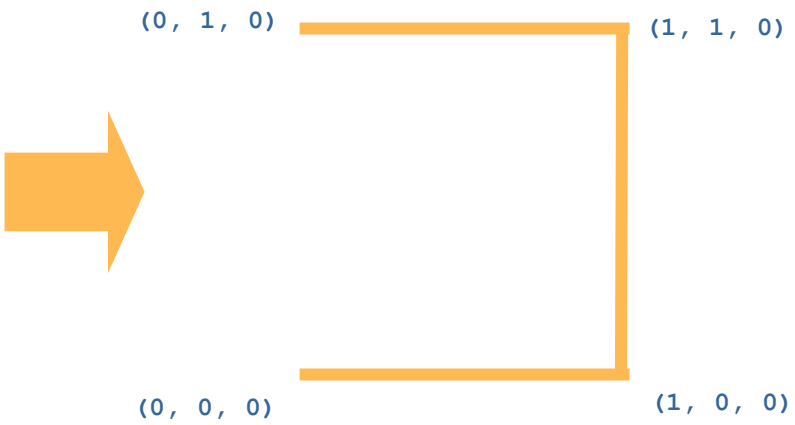
```
glBegin(GL_LINES);
glVertex3f(0, 0, 0);
glVertex3f(0, 1, 0);
glVertex3f(1, 1, 0);
glVertex3f(1, 0, 0);
glEnd();
```



Primitivas geométricas básicas

Modo	Descripción
GL_POINTS	Puntos individuales aislados
GL_LINES	Cada par de puntos corresponde a una recta
GL_LINE_STRIP	Segmentos de recta conectados
GL_LINE_LOOP	Segmentos de recta conectados y cerrados
GL_TRIANGLES	Cada tres puntos un triángulo
GL_QUADS	Cada cuatro puntos un cuadrado
GL_POLYGON	Un polígono

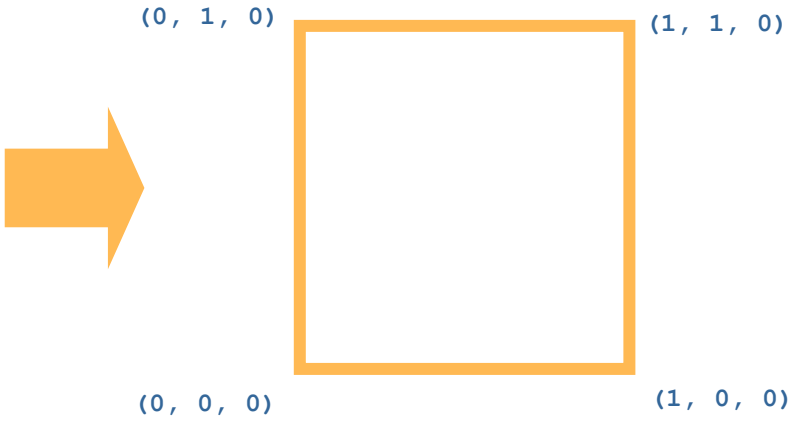
```
glBegin(GL_LINE_STRIP);
glVertex3f(0, 0, 0);
glVertex3f(0, 1, 0);
glVertex3f(1, 1, 0);
glVertex3f(1, 0, 0);
glEnd();
```



Primitivas geométricas básicas

Modo	Descripción
GL_POINTS	Puntos individuales aislados
GL_LINES	Cada par de puntos corresponde a una recta
GL_LINE_STRIP	Segmentos de recta conectados
GL_LINE_LOOP	Segmentos de recta conectados y cerrados
GL_TRIANGLES	Cada tres puntos un triángulo
GL_QUADS	Cada cuatro puntos un cuadrado
GL_POLYGON	Un polígono

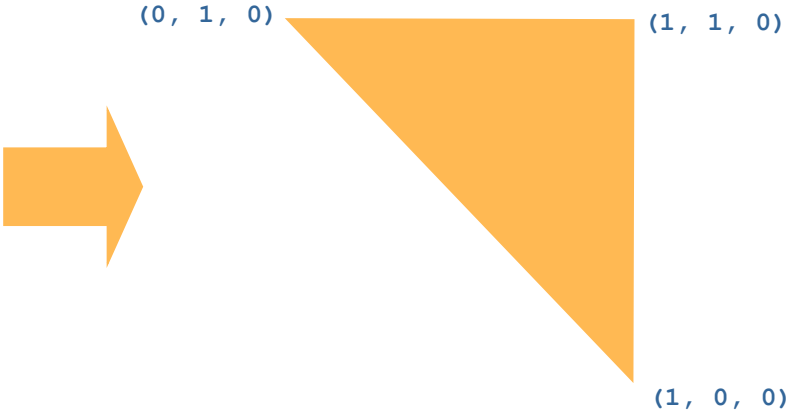
```
glBegin(GL_LINE_LOOP);
glVertex3f(0, 0, 0);
glVertex3f(0, 1, 0);
glVertex3f(1, 1, 0);
glVertex3f(1, 0, 0);
glEnd();
```



Primitivas geométricas básicas

Modo	Descripción
GL_POINTS	Puntos individuales aislados
GL_LINES	Cada par de puntos corresponde a una recta
GL_LINE_STRIP	Segmentos de recta conectados
GL_LINE_LOOP	Segmentos de recta conectados y cerrados
GL_TRIANGLES	Cada tres puntos un triángulo
GL_QUADS	Cada cuatro puntos un cuadrado
GL_POLYGON	Un polígono

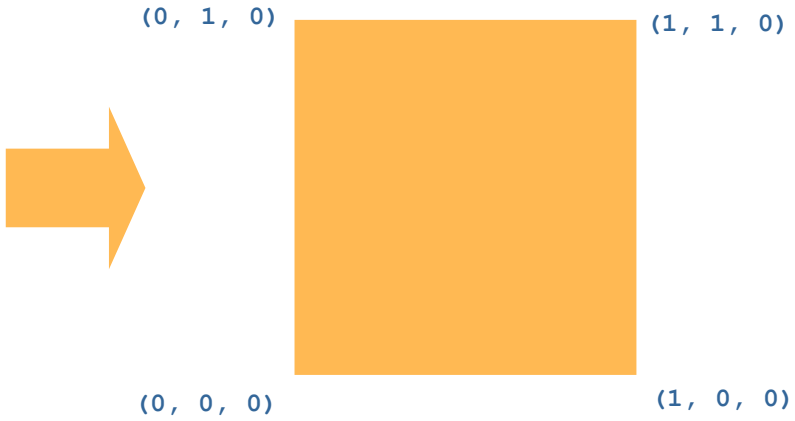
```
glBegin(GL_TRIANGLES);
glVertex3f(0, 1, 0);
glVertex3f(1, 1, 0);
glVertex3f(1, 0, 0);
glEnd();
```



Primitivas geométricas básicas

Modo	Descripción
GL_POINTS	Puntos individuales aislados
GL_LINES	Cada par de puntos corresponde a una recta
GL_LINE_STRIP	Segmentos de recta conectados
GL_LINE_LOOP	Segmentos de recta conectados y cerrados
GL_TRIANGLES	Cada tres puntos un triángulo
GL_QUADS	Cada cuatro puntos un cuadrado
GL_POLYGON	Un polígono

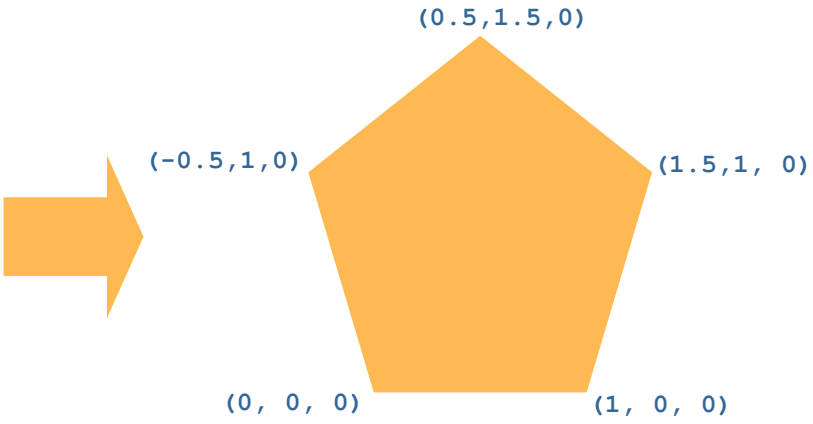
```
glBegin(GL_QUADS);
glVertex3f(0, 0, 0);
glVertex3f(0, 1, 0);
glVertex3f(1, 1, 0);
glVertex3f(1, 0, 0);
glEnd();
```



Primitivas geométricas básicas

Modo	Descripción
GL_POINTS	Puntos individuales aislados
GL_LINES	Cada par de puntos corresponde a una recta
GL_LINE_STRIP	Segmentos de recta conectados
GL_LINE_LOOP	Segmentos de recta conectados y cerrados
GL_TRIANGLES	Cada tres puntos un triángulo
GL_QUADS	Cada cuatro puntos un cuadrado
GL_POLYGON	Un polígono

```
glBegin(GL_QUADS);
glVertex3f(0, 0, 0);
glVertex3f(-0.5, 1, 0);
glVertex3f(0.5, 1.5, 0);
glVertex3f(1.5, 1, 0);
glVertex3f(0, 1, 0);
glEnd();
```



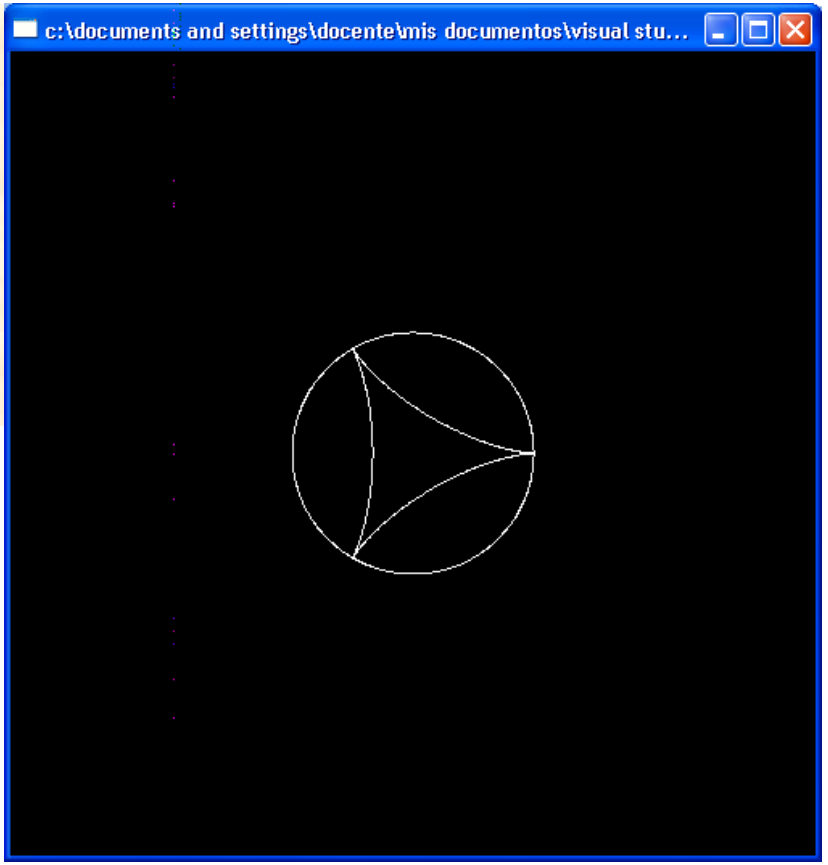
Un programa en OpenGL


```
#include <stdlib.h>
#include <math.h>
#include <GL/glut.h>
#define GL_PI 3.1416f
void init(void);
void display(void);
void reshape(int,int);
int main(int argc, char** argv)
{
glutInit(&argc, argv);
glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
glutInitWindowSize(500,500);
glutInitWindowPosition(100,100);
glutCreateWindow(argv[0]);
init();
glutDisplayFunc(display);
glutReshapeFunc(reshape);
glutMainLoop();
return 0;
}
```


```
void init(void)
{
glClearColor(0.0,0.0,0.0,0.0);  //parametros: rojo, amarillo y azul, el cuarto es el parametro alpha
glShadeModel(GL_FLAT);
glColor3f(1.0,1.0,1.0);
}
void display(void)
{
GLfloat ang, rm = 1.0f, rma = 3*rm, x, y, xc, yc;
glClear(GL_COLOR_BUFFER_BIT);
glPushMatrix();                                    // salva el estado actual de la matriz
glBegin(GL_POINTS);
for (ang = 0.0f; ang < 2 * GL_PI; ang += 0.01f)
{
xc = rma * sin(ang);
yc = rma * cos(ang);
glVertex2f(xc,yc);
x = (rma - rm)*cos(ang) + rm*cos((rma - rm)/rm* ang);
y = (rma - rm)*sin(ang) - rm*sin((rma - rm)/rm* ang);
glVertex2f(x,y);
}
glEnd();
glPopMatrix();          // reecupera el estado del matriz
glFlush();
}
```

Un programa en OpenGL


```
void reshape(int w, int h)
{
glViewport(0,0,(GLsizei)w, (GLsizei)h);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glOrtho(-10.0,10.0,-10.0,10,-10.0,10.0);
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
}
```



Bibliografía

Computación Gráfica

1. **Computer Graphics: Principles and Practice.** Foley J., Van Dame A., Feiner S., Hughes J., Phillips R. Addison – Wesley Publishing Company, Massachusetts. 1996
2. **Curvas y superficies para modelado geométrico.** Cordero Valle J., Cortes Parejo. Alfaomega Grupo Editor 2003
3. **Fundamentals of Computer Aided Geometric Design.** Hoschek J., Lasser D. A.K. Peters Ltd. Wellesley Massachusetts. 1993
4. **Gráficas por computadora.** Hearn D., Baker M.P. Prentice - Hall Hispanoamericana. 1998
5. **Introduction to Computing with Geometry Notes.** Shene C.K. Department of Computer Science. Michigan Technological University. 1997 www.cs.mtu.edu/~shene/COURSES/CS3621/NOTES/notes.html
6. **OpenGL Silicon Graphics Inc.** www.opengl.org
7. **Programación de graficos en 3D**. M. Escribano. Add. Wesley.
8. **Fundamentos de Dibujo en Ingeniería. Con una introducción a las Graficas por computadora interactiva para Diseño y Producción** - LUZADDER Warren J. - DUFF Jon M.

Discusión, preguntas...