

A decorative L-shaped line in a dark blue color, consisting of a vertical segment on the left and a horizontal segment extending to the right, intersecting at a right angle.

# Gráficos por Computador

Ana Gil Luezas  
Departamento de Sistemas Informáticos y Computación  
Facultad de Informática  
Universidad Complutense de Madrid

- ❑ Gráficos por computador:

Generar imágenes mediante un computador

**Open GL** (Open Graphics Library)

- ❑ Procesamiento de imágenes:

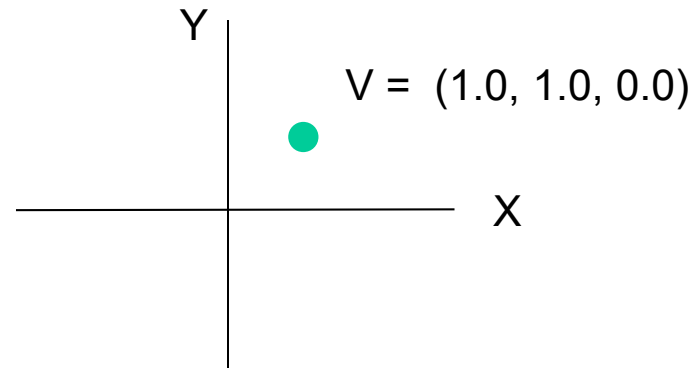
Mejorar o alterar imágenes previamente creadas,  
por una cámara de fotos, video, por computador,

...

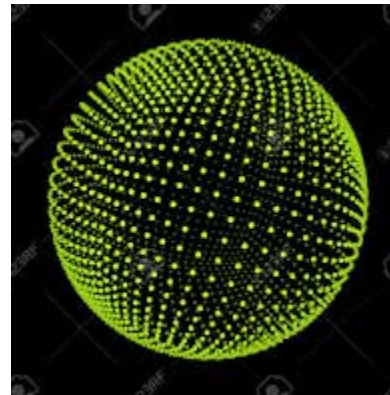
- ❑ Un gráfico se compone de una serie de elementos básicos, denominados **primitivas**.
  - ❑ Puntos
  - ❑ Poli-líneas
  - ❑ Regiones rellenas (triángulos)
  - ❑ Texto
  - ❑ Imágenes rasterizadas
  
- ❑ Cada primitiva tiene asociada **atributos**: color, grosor, ...

- ❑ Un **punto** se determina por las **coordenadas**  $(x, y, z)$  de un **vértice**.

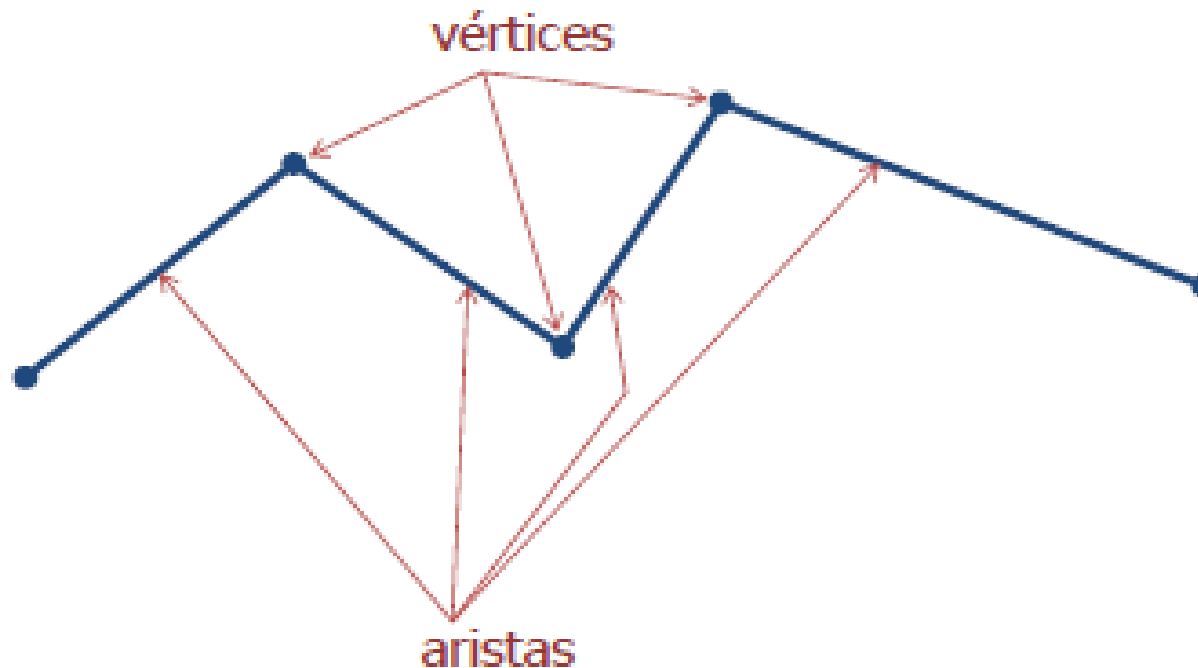
- ❑ **Atributos** asociados:  
grosor y color.



- ❑ Un gráfico construido a base de puntos se denomina **dibujo de puntos**.

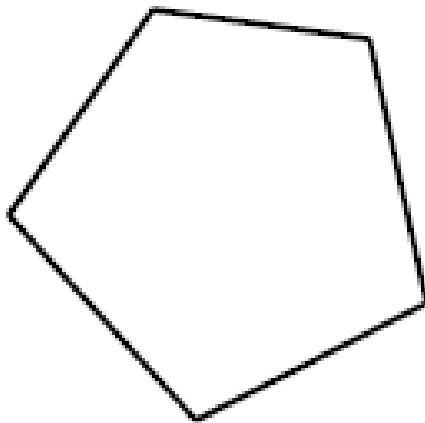


- ❑ Una **poli-línea** es una secuencia contigua de segmentos (aristas) determinados por **vértices**.
- ❑ La poli-línea más simple es un segmento.

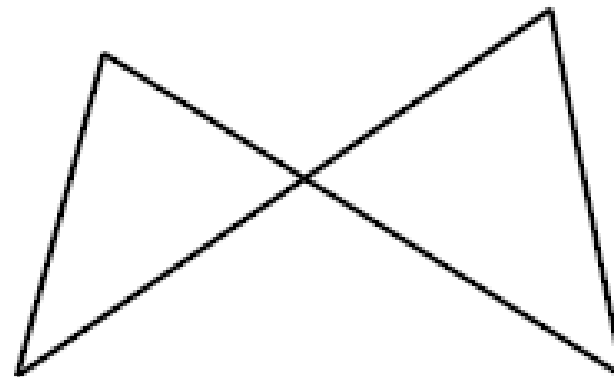


- ❑ Un gráfico construido a base de poli-líneas se denomina **dibujo de líneas**.
- ❑ Un **polígono** es una poli-línea en la que el primer y último vértice están conectados mediante una arista.

Es **simple** si no existen dos aristas que se crucen.

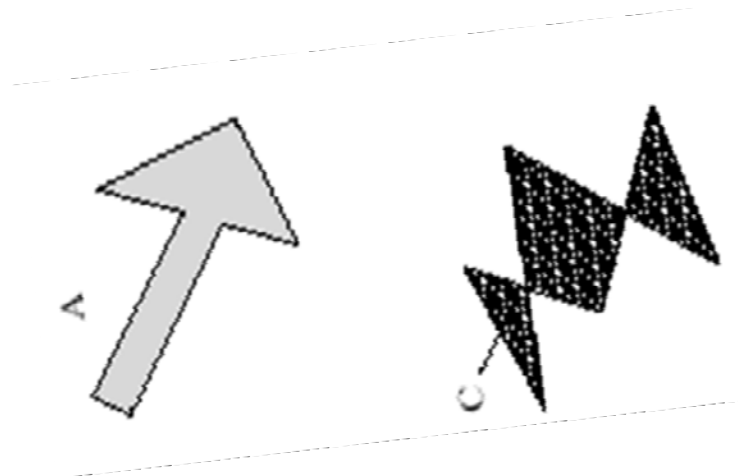


Polígono simple



Polígono no simple

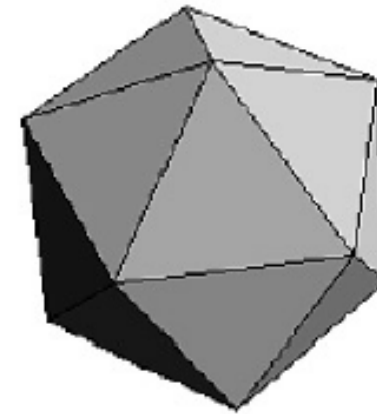
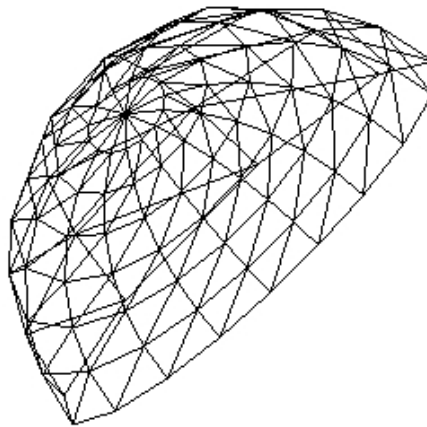
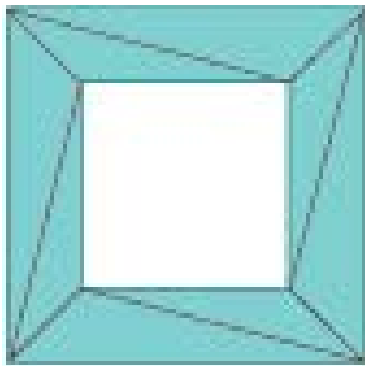
- ❑ Son formas cerradas, rellenas con algún color, patrón o textura.
- ❑ En general el borde que la delimita es un polígono simple, plano y convexo.
- ❑ **Atributos** asociados: color, patrón de relleno, textura



- ❑ Los **triángulos** son fundamentales en informática gráfica:

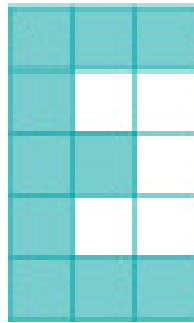
Son fáciles de definir, son simples, convexos y planos, por lo que muchos de los algoritmos de “rendering” (proceso por el cual un ordenador muestra una imagen a partir de un modelo) funcionan de forma óptima para triángulos.

- ❑ Se utilizan para aproximar superficies y componer otros polígonos.





- ❑ Los **caracteres** pueden definirse mediante una poli-línea o un mapa de bits.
- ❑ **Atributos** asociados: **color**, **tamaño**, **espaciado**, **fuente**



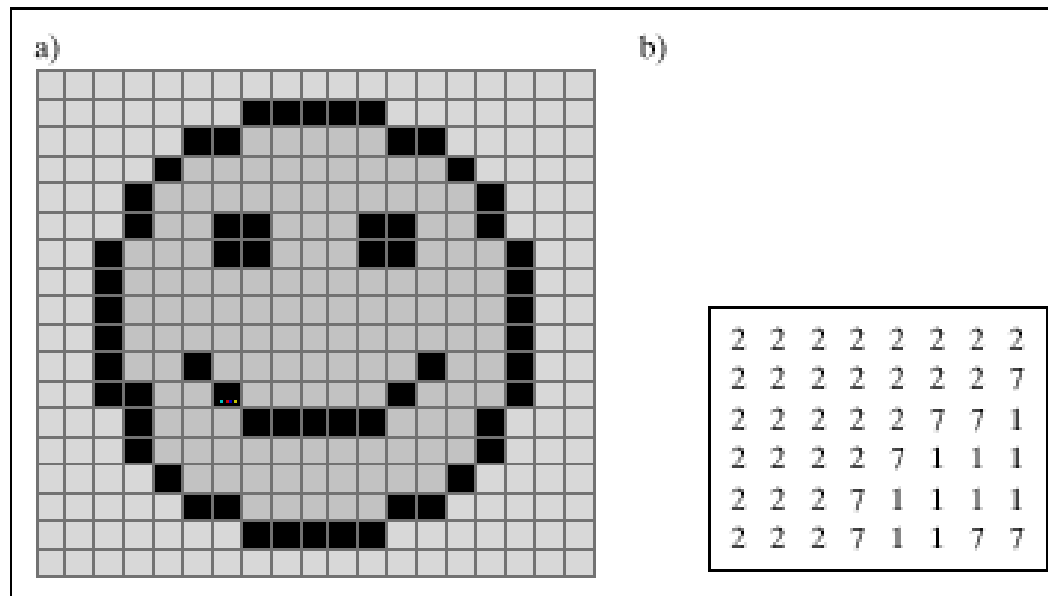
Bitmapped



Stroke

## Imágenes rasterizadas

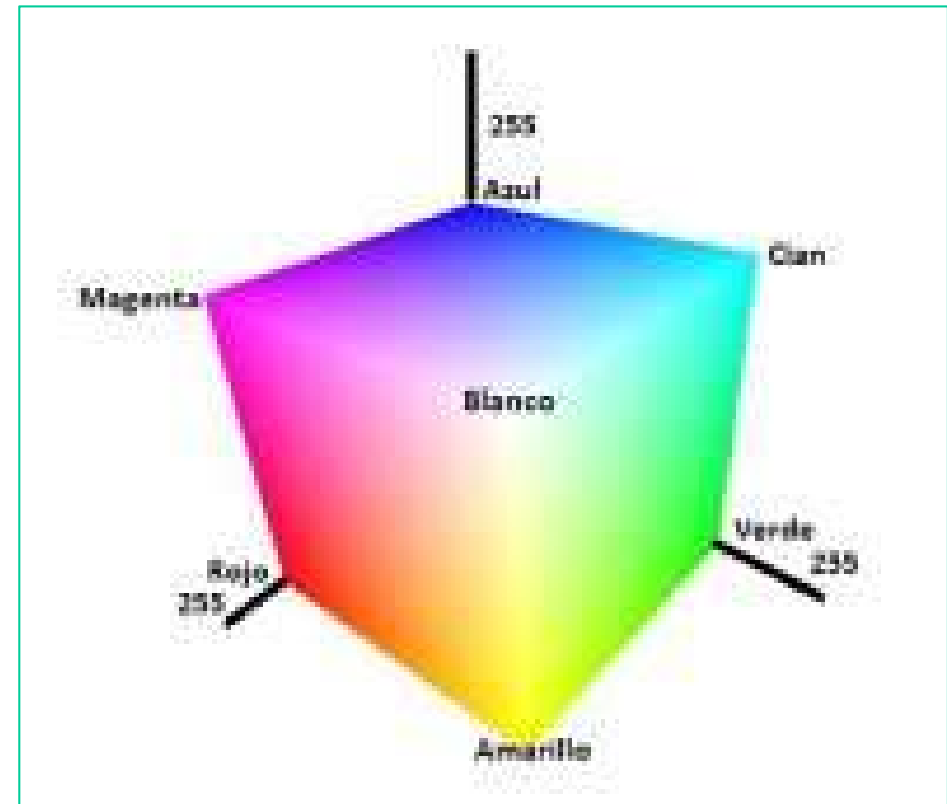
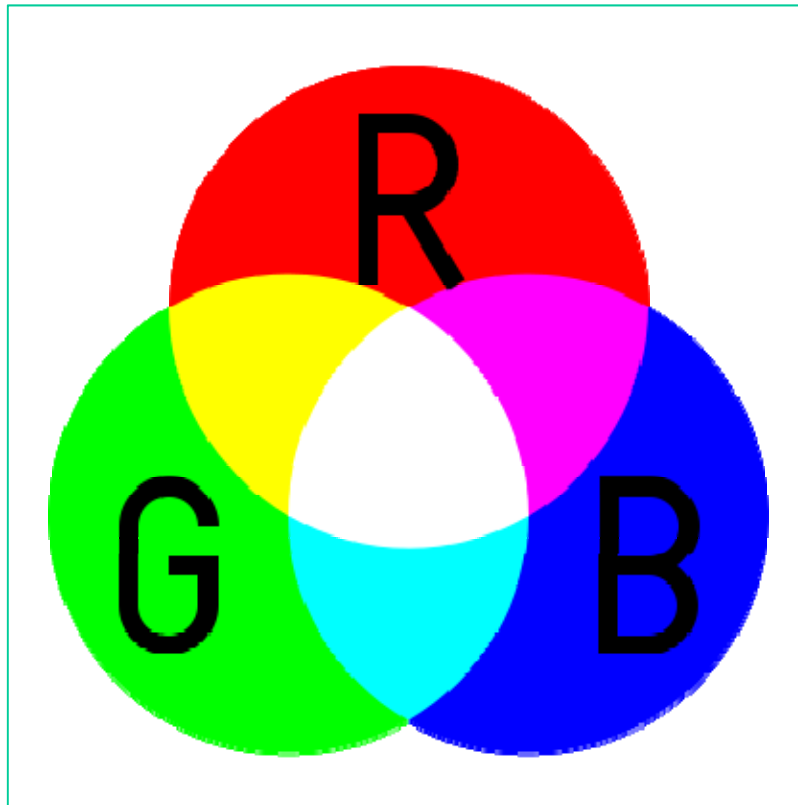
- ❑ Son **imágenes** previamente generadas **que aparecen en el gráfico**. Por ejemplo, podemos incorporar una imagen almacenada en un fichero **bmp** dentro de nuestro gráfico.
- ❑ Una **imagen rasterizada** se almacena en el ordenador como una **matriz de valores numéricos (bitmap)**.



Cada valor numérico indica el color de un *texel* (pixel).

- ❑ **Modelo de color RGB** (Red, Green, Blue). Utilizado en monitores. Los colores se obtienen mediante la **mezcla aditiva (sobre negro)** de la intensidad de los colores **rojo, verde y azul** (colores primarios).
- ❑ **Modelo aditivo**. Los colores se obtienen sumando las tres componentes. **Colores secundarios: cian, magenta y amarillo**.
  - El **cian** se obtiene sumando el verde y el azul
  - El **magenta** se obtiene sumando el rojo y el azul
  - El **amarillo** se obtiene sumando el rojo y el verde
  - El **blanco** es la suma de los tres primarios
- ❑ ¿Cuál son los colores primarios? Relación entre el modelo matemático y un **espacio de color** (gama de colores absolutos)

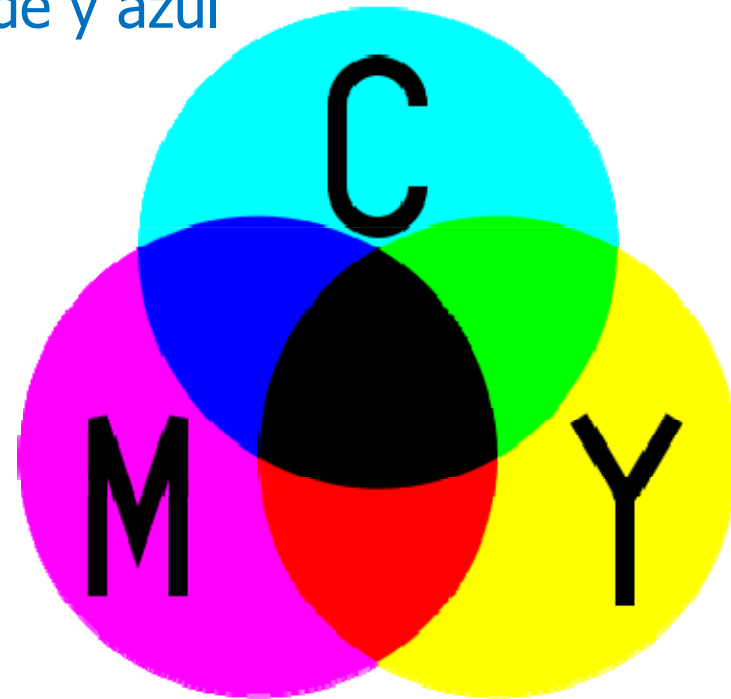
### Modelo aditivo de colores rojo, verde y azul



- ❑ El **modelo de color CMY** (Cyan, Magenta, Yellow).  
Utilizado en impresoras. Los colores se obtienen mediante la **mezcla sustractiva** de los colores **cian**, **magenta** y **amarillo** (colores primarios). Quitando el rojo al blanco queda cian.

Colores secundarios: rojo, verde y azul

- ❑ **CMYK** añade el color negro (Key=negro)



- ❑ **Modelo de color RGB** (Red, Green, Blue). Cada valor de color es un triple ordenado  $(r, g, b)$ , que representa las **intensidades de rojo, verde y azul** respectivamente.
- ❑ El intervalo de **valores para indicar la intensidad** de cada componente suele ser:
  - un **entero entre 0 y 255** (1 byte)
  - un **real entre 0 y 1** (float/double)
- ❑ La **profundidad de color** se define como la **suma de los bits asociados a las componentes  $r, g, b$** . Por ejemplo, si utilizamos 8 bits para el rojo, 8 para el verde y 8 para el azul, tendremos una profundidad de color de 24 bits por píxel, y una gama de colores de  $2^{24} = 16777216$  posibles colores.
- ❑ La profundidad de color de 24 bits por píxel, se denomina **color verdadero**. La profundidad de 32 bits agrega un **canal alfa (RGBA)** que representa la **transparencia**.

## Representación del color

- ❑ En hexadecimal, cada componente de 1 byte son dos dígitos

	0xHexa.	Decimal	3 int/bytes	3 float/double
Rojo:	0xFF0000	16711680	(255, 0, 0)	(1.0, 0.0, 0.0)
Verde:	0x00FF00	65280	(0, 255, 0)	(0.0, 1.0, 0.0)
Azul:	0x0000FF	255	(0, 0, 255)	(0.0, 0.0, 1.0)
Cian:	0x00FFFF	65535	(0, 255, 255)	(0.0, 1.0, 1.0)
Magenta:	0xFF00FF	16711935	(255, 0, 255)	(1.0, 0.0, 1.0)
Amarillo:	0xFFFF00	16776960	(255, 255, 0)	(1.0, 1.0, 0.0)
Blanco:	0xFFFFFFFF	16777215	(255, 255, 255)	(1.0, 1.0, 1.0)
Gris:	0x808080	8421504	(128, 128, 128)	(0.5, 0.5, 0.5)
Negro:	0x000000	0	(0, 0, 0)	(0.0, 0.0, 0.0)

- ❑ Operaciones suma, resta, producto, combinación lineal

Dados:  $C1 = (r1, g1, b1)$  y  $C2 = (r2, g2, b2) : 0 \leq r, g, b \leq 1$

$$C1 * C2 = (r1*r2, g1*g2, b1*b2) : 0 \leq r, g, b \leq 1$$

$$C * s = (r*s, g*s, b*s) : 0 \leq r, g, b, s \leq 1$$

$$C1 + C2 = (r1+r2, g1+g2, b1+b2) : 0 \leq r, g, b$$

$$C1 - C2 = (r1-r2, g1-g2, b1-b2) : r, g, b \leq 1$$

$$C1 * a + C2 * b :$$

$$\text{clamp}(\text{Valor}, \text{Vmin}, \text{Vmax}), \text{max}(V1, V2), \text{min}(V1, V2)$$

$$\text{clamp}(R, 0, 1) \quad \text{max}(G, 0) \quad \text{min}(B, 1)$$



- interpolación lineal (mix)

$$\text{mix}(C1, C2, a) = C1 * (1-a) + C2 * a : 0 \leq r, g, b \leq 1$$

$$\text{si } 0 \leq a \leq 1$$

- La **luminancia** (escala de grises) de un color RGB sería  
 $\text{lum} = (R + G + B) / 3$ , y el

color **gris** asociado sería  $(\text{lum}, \text{lum}, \text{lum})$

Pero debido a como el ser humano percibe la luz, se define con distintos pesos:

$$\text{lum} = R * 0.299 + G * 0.587 + B * 0.114$$

$$\text{lum} = R * 0.2125 + G * 0.7154 + B * 0.0721 \quad (\text{sRGB})$$

- ❑ Se utiliza una **tabla de colores** que ofrece una asociación configurable entre el valor de cada píxel (índice) y el color final que representa.
- ❑ Supongamos que cada píxel consta de **b** bits. Estos bits se usan como un índice de una tabla de  $2^b$  entradas.

Por lo tanto la imagen sólo puede constar de  $2^b$  colores diferentes.

- ❑ Cada elemento de la tabla contiene información sobre un color. La profundidad del color (**w**) es independiente del tamaño reservado para los índices (**b**).

La tabla se puede configurar sobre una gama de  $2^w$  colores diferentes.

- ❑ Para índices de 8 bits ( $b=8$ ) y profundidad de color de 24 bits ( $w=24$ ), tendríamos  $2^{24}$  posibles colores (color verdadero), aunque en una imagen sólo podrían aparecer  $2^8 = 256$  colores.

Una imagen de 1280 x 1024

Con índices de 8 bits (256 posibles colores), necesita

1280 x 1024 x 8 bits de memoria

+ la tabla ( $n^{\circ}$  de colores diferentes de la imagen \* 24 bits)

Y en modo RGB de 24 bits ( $2^{24}$  posibles colores):

1280 x 1024 x 24 bits de memoria.

### ❑ Formato **PNG** (Portable Network Graphics).

Utiliza un algoritmo de compresión sin pérdida para **bitmaps**: En blanco y negro, en color RGB y con paleta de colores.

Permite transparencias (canal o componente **Alpha**).

**Rango total de opciones de color soportados**

Profundidad de bits por canal	1	2	4	8	16
Imagen indexada (1 canal)	1	2	4	8	
Escala de grises (1 canal)	1	2	4	8	16
Escala de grises con alfa (2 canales)				16	32
Color verdadero (RGB) (3 canales)				24	48
Color verdadero con alfa (RGBA) (4 canales)				32	64

Generar imágenes mediante un computador

- ❑ Proceso de visualización (renderizado)
  - ❑ Tubería gráfica (OpenGL pipeline)
- ❑ Escena, cámara y puerto de vista

## Proceso de visualización (renderizado)

- ❑ Dada una escena
  - ❑ Hacer y revelar una foto
  - ❑ Determinar el color de cada píxel
- ❑ Técnicas para determinar el color de cada píxel
  - ❑ **Interactivas**: procesamiento de primitivas (triángulos) dadas por vértices y posterior relleno
    - ❑ Tuberías gráficas: **OpenGL**, DirectX
  - ❑ Realistas: por píxel
    - ❑ Trazado de rayos (Ray Tracing), Radiosidad, ...

### ☐ Objetos

- ☐ Descripción local: Geometría y materiales
- ☐ Disposición en la escena: Posición y orientación

### ☐ Luces

- ☐ Descripción lumínica
- ☐ Disposición en la escena

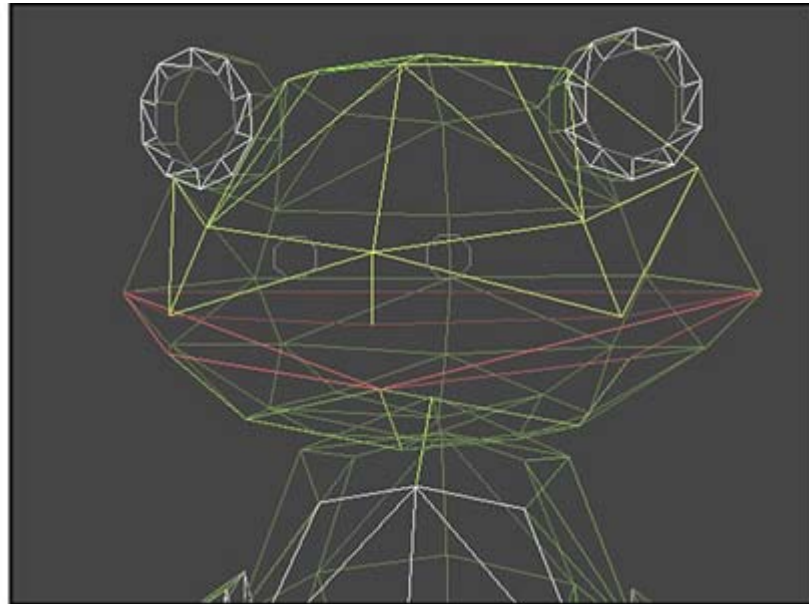
### ☐ Cámara

- ☐ Descripción óptica: Volumen de vista y proyección
- ☐ Disposición en la escena

### ☐ Puerto de vista

- ☐ Posición y dimensiones en píxeles

- ❑ Diseñamos el objeto (**escena**).



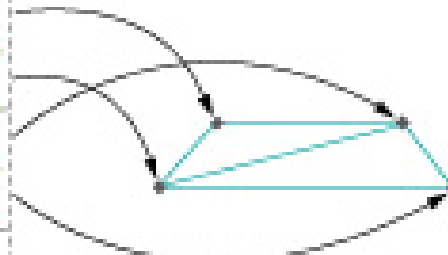


- ❑ Diseñamos el objeto (**escena**) en un **sistema de coordenadas local**

**Malla:** Coordenadas de los vértices, componentes del color, coordenadas de vectores normales, coordenadas de textura

Vertex Arrays

	Vertex coords	Color	Normal	Texture coords
-Vertex 0	$x\ y\ z$	$r\ g\ b$		
-Vertex 1	$x\ y\ z$	$r\ g\ b$		
-Vertex 2	$x\ y\ z$	$r\ g\ b$		
-Vertex 3	$x\ y\ z$	$r\ g\ b$		
	$\vdots$	$\vdots$	$\vdots$	$\vdots$

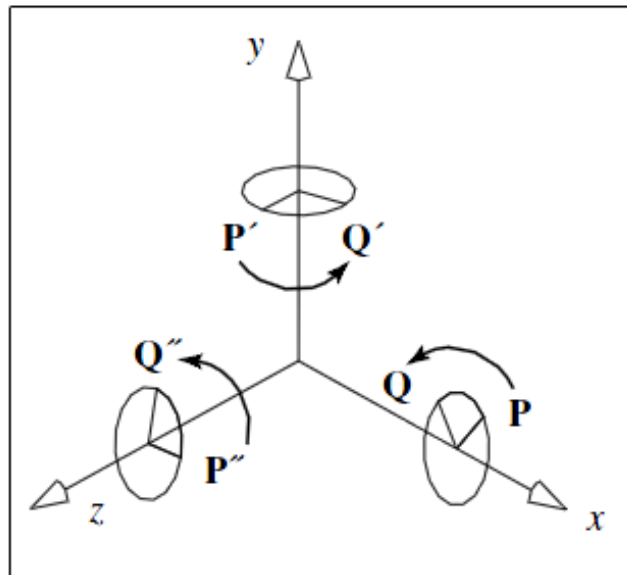


- ❑ **Matriz de modelado**

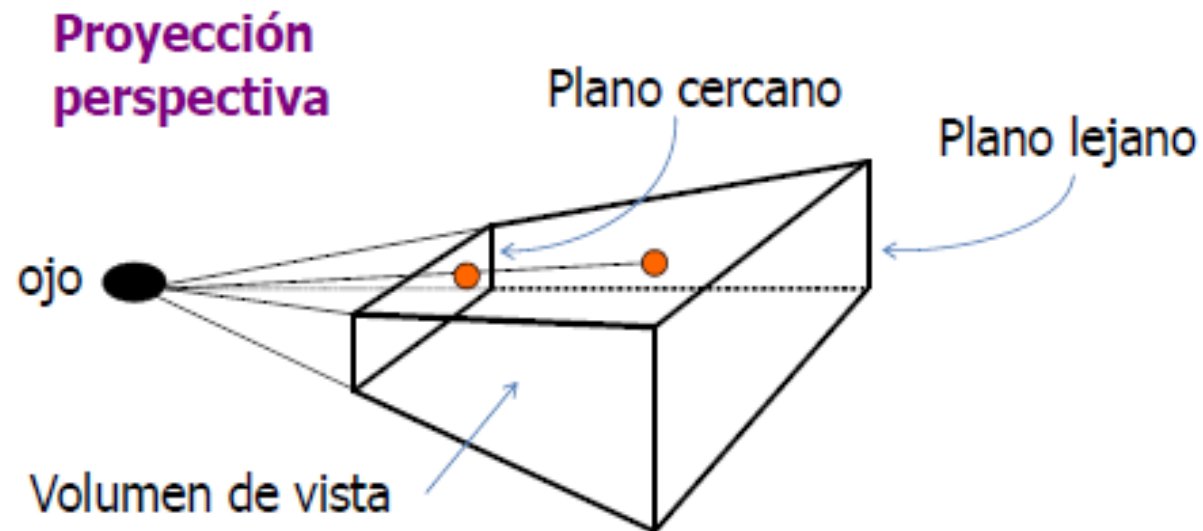
Y se dispone en la escena mediante **transformaciones afines**:  
 translación, rotación

## Transformaciones afines

- ❑ **Traslaciones:** vector  $(t_x, t_y, t_z)$
- ❑ **Escalas:** factor  $S=(s_x, s_y, s_z)$
- ❑ **Rotaciones** sobre los ejes:

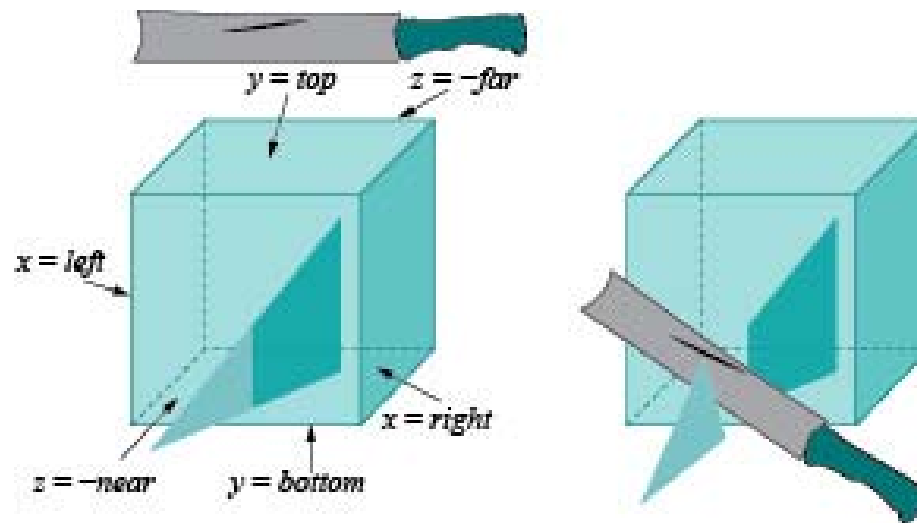


- ❑ Posicionamos la **cámara** y elegimos el tipo de óptica (**proyección**) que deseamos (ortogonal o perspectiva).
- ❑ Establecemos el **volumen de vista**. Los objetos dentro del volumen serán visibles. El resto de la escena no será visible.



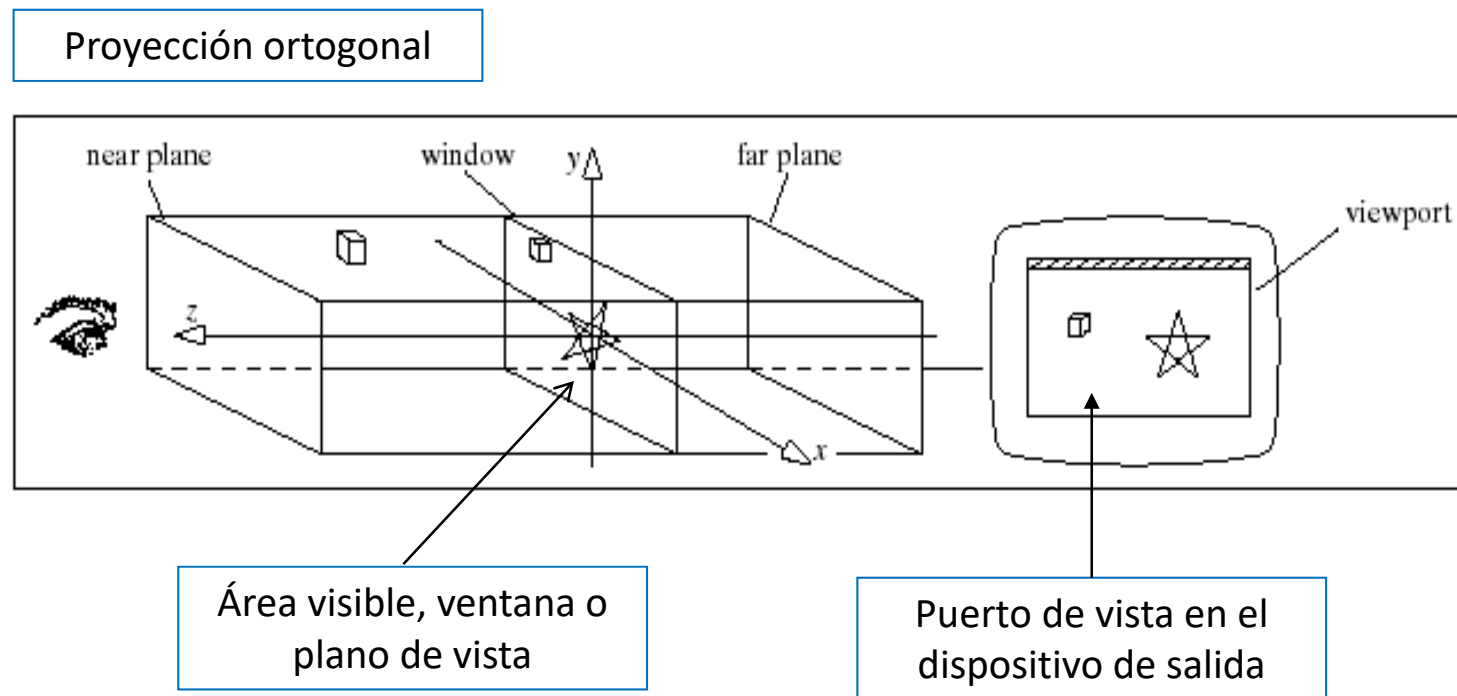
- ❑ **Volumen de vista.** Los objetos dentro del volumen serán visibles. El resto de la escena no será visible.

Algunos objetos pueden ser recortados (**clipping**)

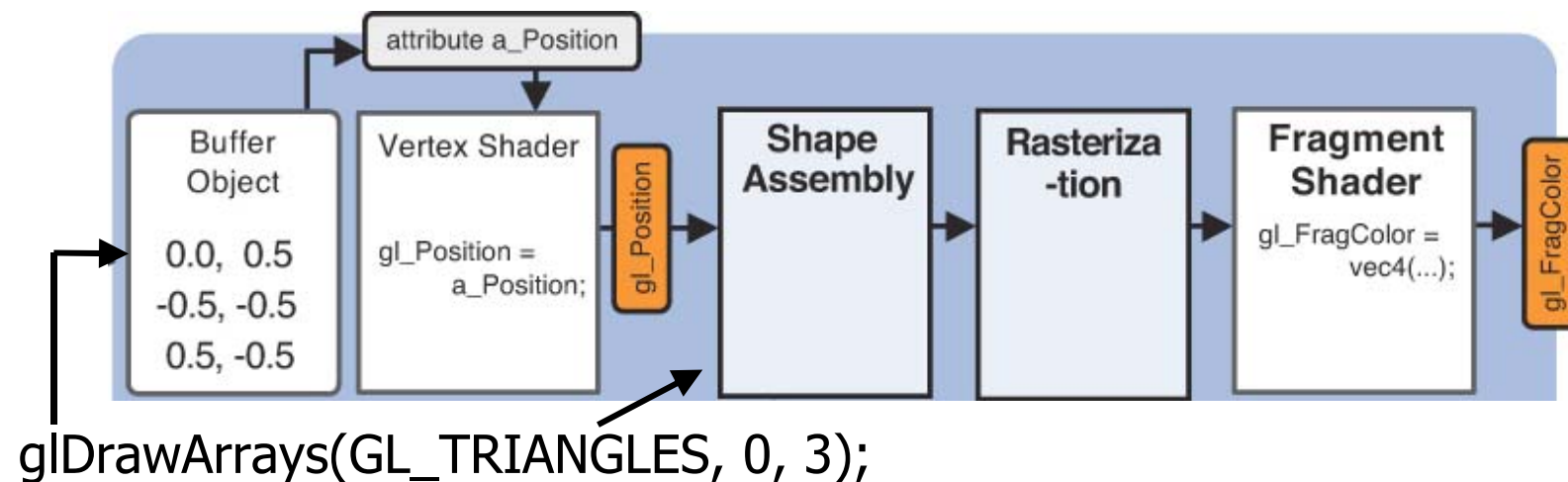


## ❑ Proyección y puerto de vista:

La proyección obtenida en el **plano de vista** se traslada al puerto de vista establecido en la ventana de visualización



- ❑ **Etapas de la tubería gráfica.** Una vez que la geometría (vértices y atributos de la primitiva) se encuentra en la GPU (Graphics Processing Unit) se ejecutan varios procesos
  - ❑ **Vertex Shader:** transforma las coordenadas y atributos de cada vértice
  - ❑ **Rasterization:** Se generan los fragmentos del interior de la primitiva
  - ❑ **Fragment Shader:** obtiene el color de cada fragmento



- ❑ **Vertex Shader:** Cada vértice sufre una serie de transformaciones.

Cada transformación se efectúa multiplicando las coordenadas actuales del vértice por una **matriz**:

- ❑ **Matriz de modelado y vista (V.M)**

- ❑ *Matriz de modelado (M):* recoge las distintas transformaciones (traslaciones, rotaciones, ...) que establece el objeto en la escena.

- ❑ *Matriz de vista (V):* recoge el sistema de coordenadas fijado por la cámara.

- ❑ **Matriz de proyección (Pr):** Proyecta la escena 3D sobre el plano de vista, de acuerdo a la proyección elegida.

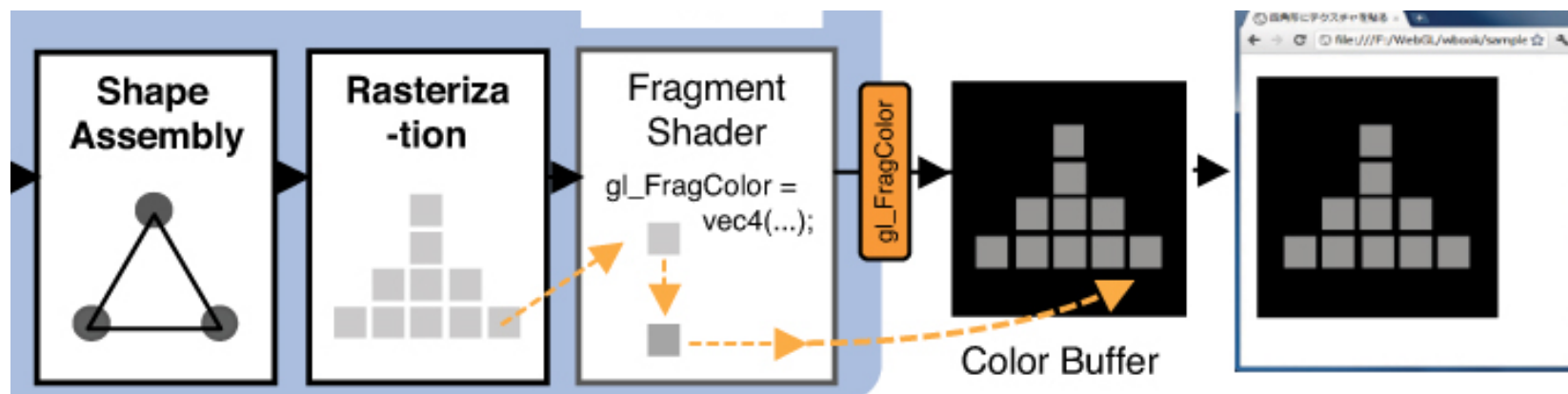
- ❑ **Matriz del puerto de vista (Vp):** Ajusta la imagen proyectada a la parte de la ventana especificada por el puerto de vista.

- ❑ **Rasterization:** genera los fragmentos del interior de la primitiva realizando un proceso de **interpolación** sobre los valores de los vértices de la primitiva.



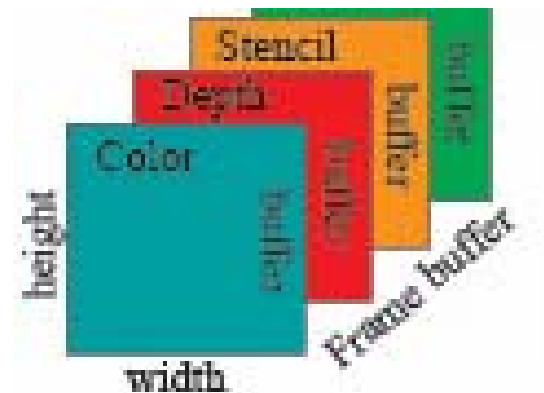


- ❑ **Fragment Shader:** Determina el color de cada fragmento escribiendo el resultado en el **Color Buffer** (BACK / FRONT).
- ❑ También se utiliza el **Depth Buffer** para determinar la visibilidad de cada fragmento.



- ❑ Los colores de los fragmentos se pueden combinar para obtener el color final que se visualizará.

- ❑ **Frame Buffer:** guarda información sobre los píxeles en la GPU
  - ❑ **Color Buffer** (FRONT y BACK): componentes RGBA del fragmento
  - ❑ **Depth Buffer** (Z-buffer): distancia del fragmento al plano de vista
  - ❑ **Stencil Buffer:** marcas para restringir los fragmentos a procesar



- ❑ **Doble buffer:** Mientras se realiza el proceso se muestra un buffer y se escribe en el otro. Al finalizar el proceso se intercambian.