PROGRAMACIÓN 3D

Máster en Programación de Videojuegos

TEMA 1: Introducción y fundamentos básicos

Juan Mira Núñez

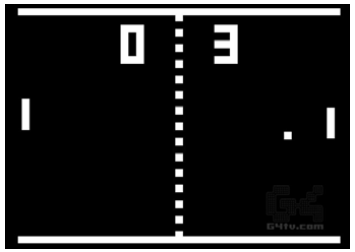
U-tad
CENTRO UNIVERSITARIO
DE TECNOLOGÍA Y ARTE DIGITAL

Índice

Historia de la informática gráfica

Décadas

- **Años 50:** MIT primeros CRT's
  - *Ivan Shuderland*
- **Años 60:** Industrias aeronáutica
  - *CAD/CAM*
  - *GKS (2-D) (ISO ANSI)*
- **Años 70. Generación "cero"**
  - *Gráficos por software*
  - *Gráficos vectoriales*
  - *Memoria muy cara*
  - *Consolas VJ básicas*
  - *"Evans and Sutherland SP1" Simulador de vuelo militar*



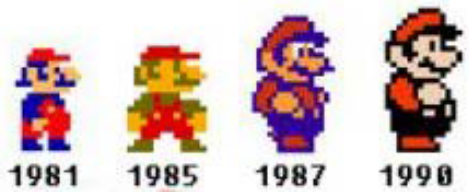
Décadas

- Años 80. Industria de animación.
  - *Librerías 3D profesionales*
    - 1988 GKS-3D, PHIGS
    - 1989 SGI IrisGL(“Terminator2”)
- Explosión computadores domésticos
  - *Videojuegos 2D*
  - *Muchos en modo texto*
- Apple Macintosh (1984)
  - *GUI y ratón*
- Atari ST (1985) "Jackintosh"
  - *Hardware dedicado “Sprites”*
  - *Innovador sonido (Multimedia)*

Décadas

- **Años 80. Industria del videojuego 2D**
- 1980 Space Invader

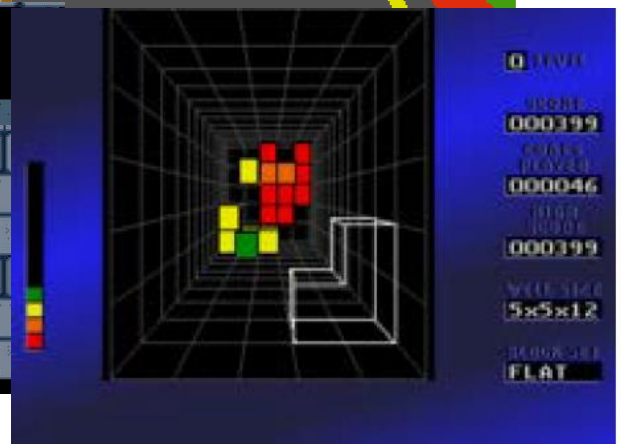
- 1981 Super Mario Bros



- 1983 Pole Position

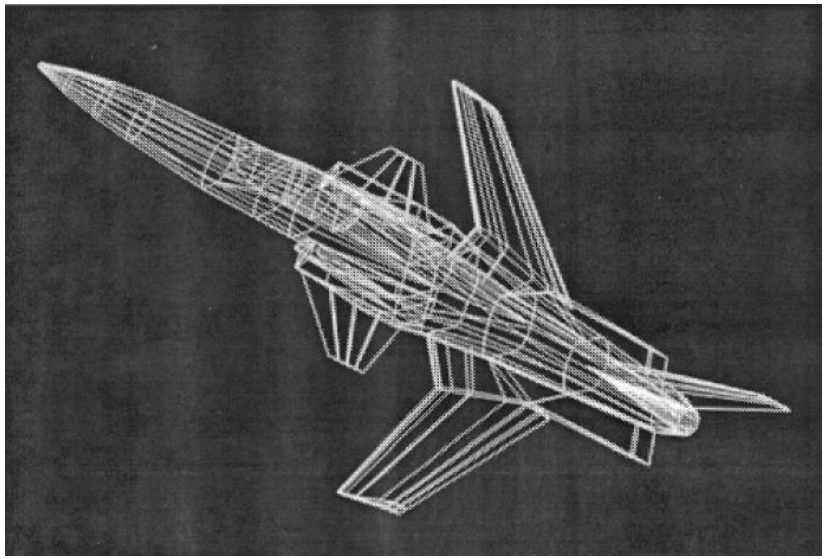
- 1989 Prince of Persia

- 1989 BlockOut (Tetris 3D)



Décadas

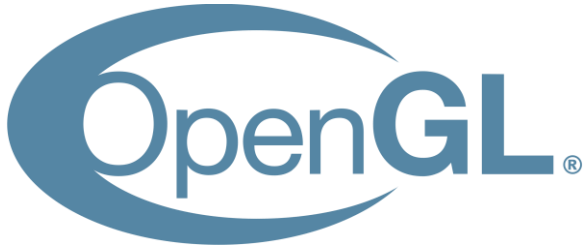
- Años 80. 3D
- ❖ 1ª Generación 3D – Gráficos alámbricos
  - *Software*
  - *Vertex: transform, clip, and project*
  - *Rasterization: color interpolation (points, lines)*
  - *Fragment: overwrite*
  - *Dates: prior to 1987*



- ❖ 2ª Generación 3D –Sombreado sólido
- Hardware Gráfico Fijo
- Vertex: lighting calculation
- Rasterization: Depth interpolation(triangles)
- Fragment: Depth buffer, color blending
- Dates: 1987 -1992

Décadas

- Años 90: Industria del videojuego (Gráficos 3D)
- Tarjetas Gráficas económicas: NVIDIA, ATI, ...
- 1992 SGI -> OpenGL



- 1992 Wolfenstien 3D
- 1993 Doom 3D

- 1995 Microsoft DirectX 1.0

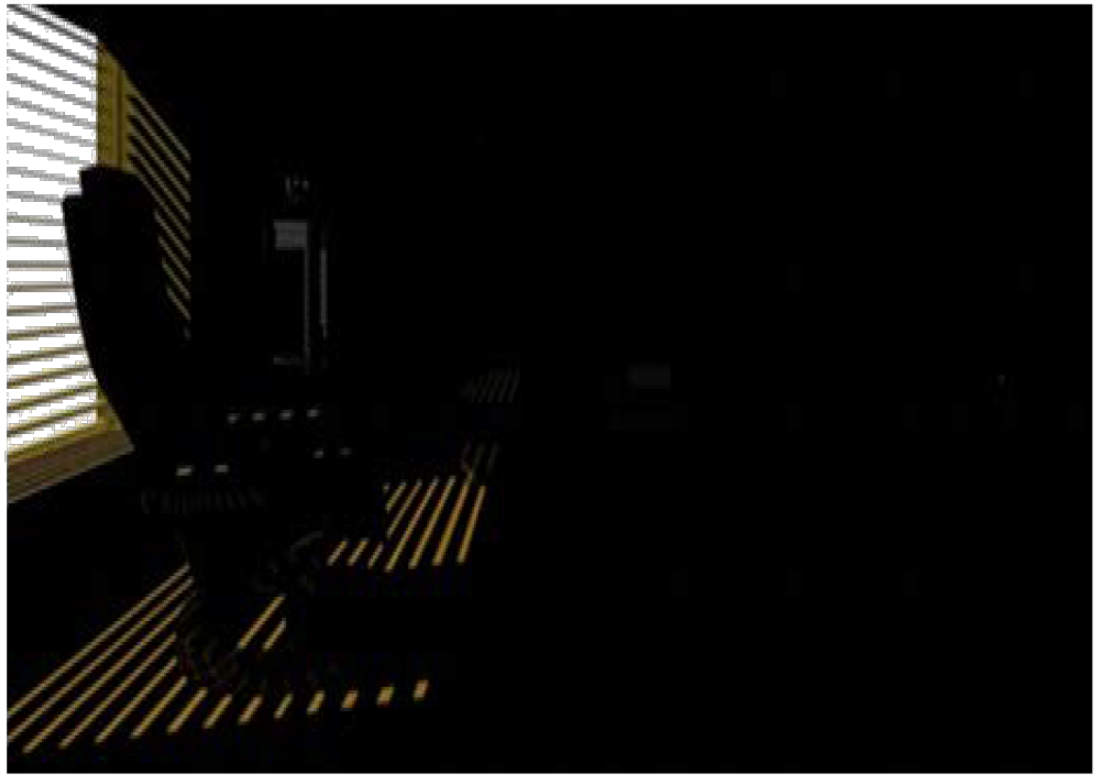
- 1995 RenderMan (Pixar)
  - *Desarrollo ToyStory*

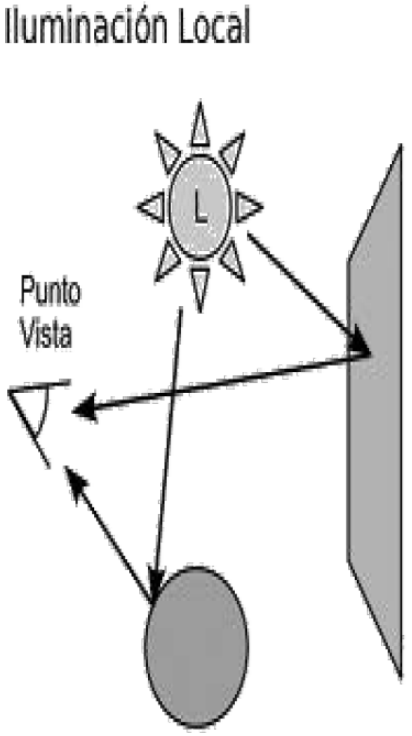
Décadas

- 3ª Generacióon 3D – Mapeado Texturas
  - ***Hardware Gráfico Configurable***
  - *Vertex: texture transformation*
  - *Rasterization: texture interpolation*
  - *Fragment: texture evaluation, antialiasing*
  - *Dates:1992 -2000*

Décadas

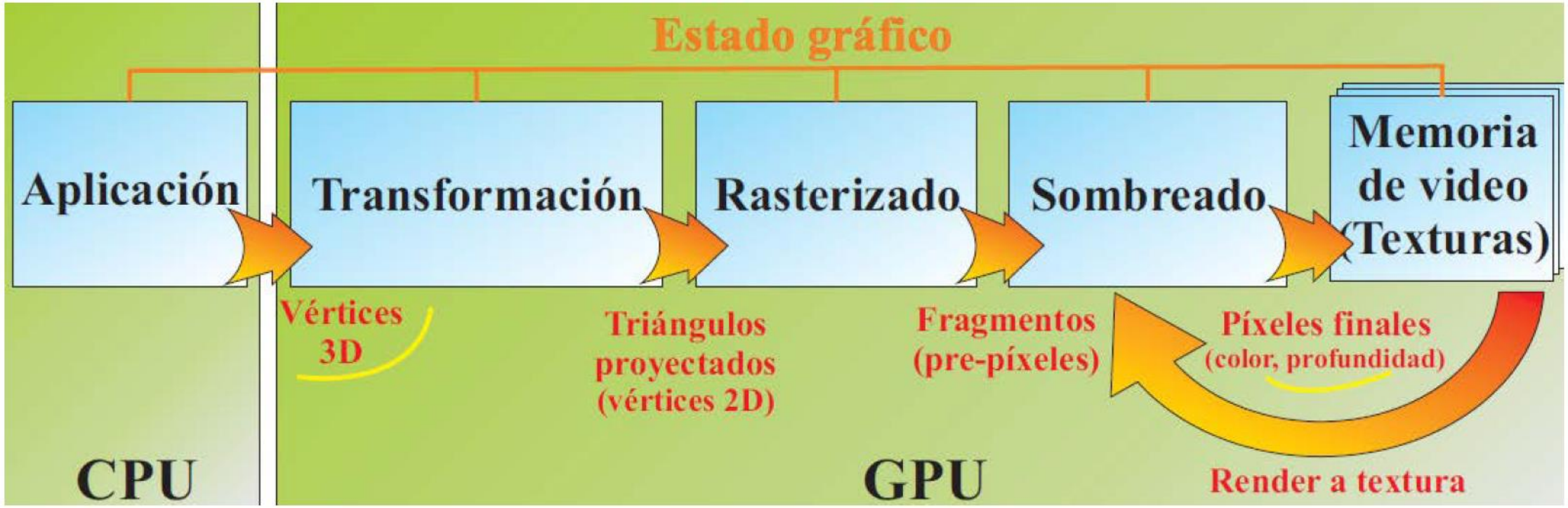
- Renderizado clásico:
  - *Iluminación local*
  - *Cálculos por vértices*
  - *Relleno de triángulos*
  - *Texturas básicas*





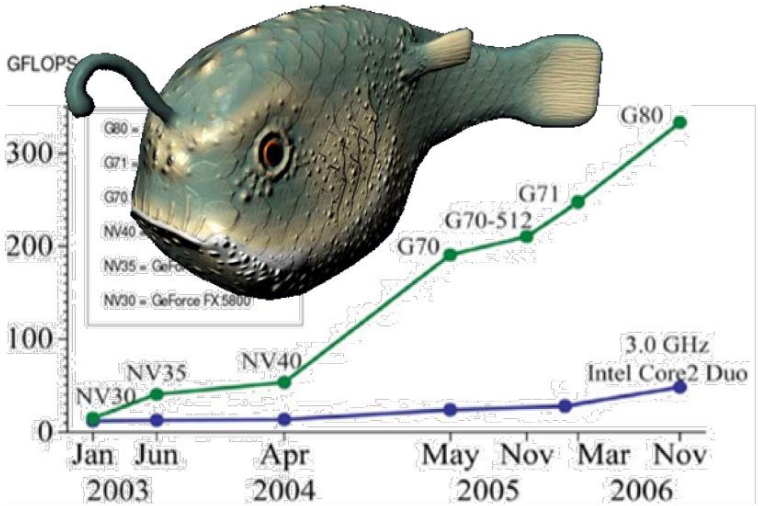
Décadas

- “**Pipeline grafico**” etapas bien definidas por las que fluye la información.



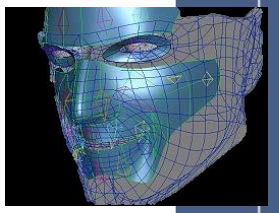
Décadas

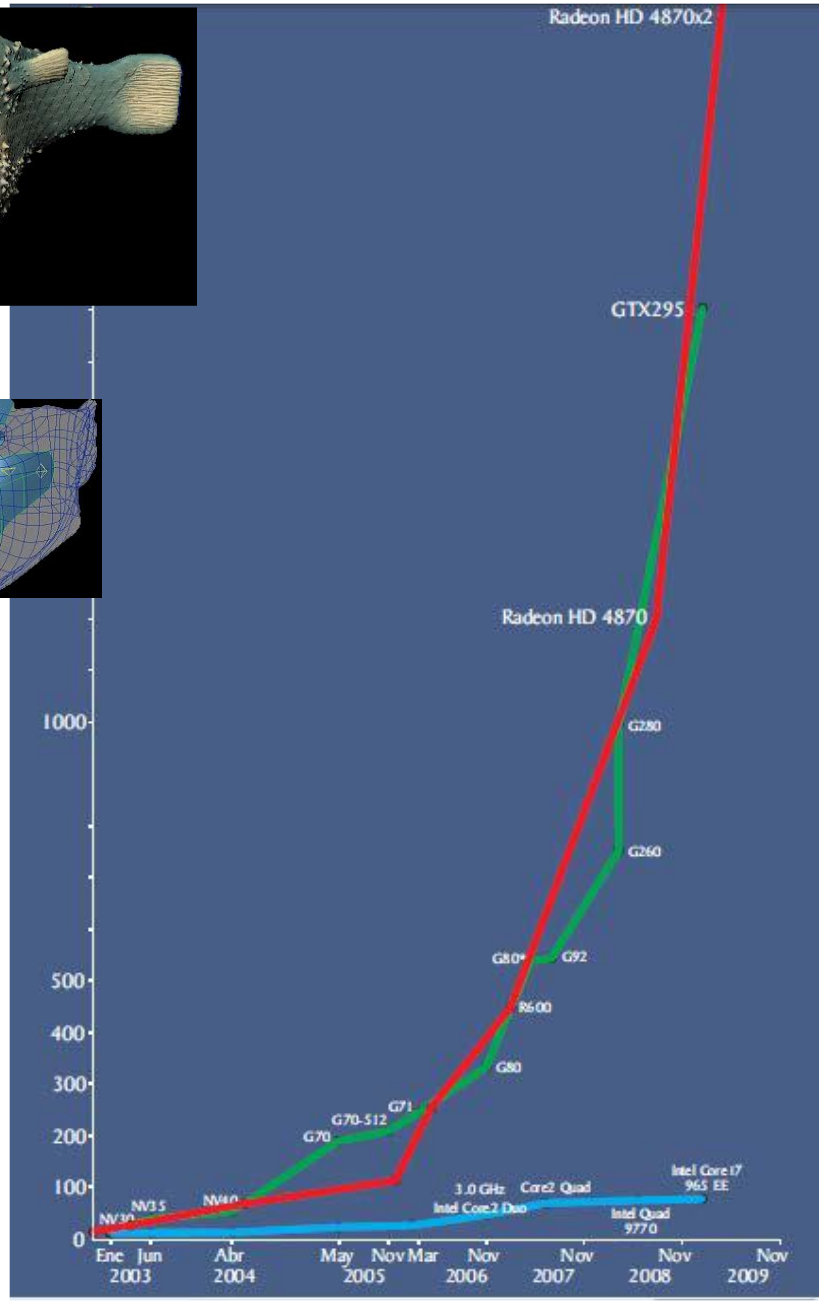
- 4ª Generación 3D Programabilidad (Shaders) años 2000-2007
  - ***Procesador de Vértices***
  - ***Procesador de Fragmentos***
  - *Image-based rendering*
  - *Curved surfaces*
  - *Dates:2000 -2007*



Décadas

- **5ª Generation–GPGPU 2007…**
  - ***GPU multi-core***
  - *Iluminación Global*
  - *Geometría*
    - “Tesselation”
    - “Displacement maps”
    - “Physics”
  - *Multimedia*
  - *Dates:2007 –Actualidad*
- Dispositivos portatiles con GPU(GFLOP/W)

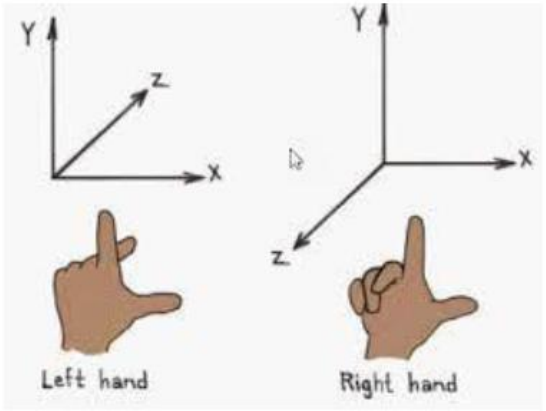




Introducción

Gráficos tridimensionales

- Los gráficos tridimensionales se representan utilizando gráficos vectoriales.
- Las figuras geométricas pasan de representarse sobre el **plano** euclídeo a representarse sobre el **espacio** euclídeo, lo que añade una nueva **coordenada Z**.
- Utilizando primitivas gráficas (principalmente triángulos), podemos crear figuras con volumen.
- El sistema de coordenadas puede ser de mano izquierda o de mano derecha.



Primitivas gráficas

- Llamamos “primitivas gráficas” a los elementos más simples que se usarán para almacenar la información (vertice, arista …)

- Con grupos de primitivas gráficas más sencillas podremos crear primitivas más complejas, y éstas a su vez podrán representar gráficos más complejos.

- Cuanto mayor sea la complejidad, mayor será la información presentada al usuario.

Primitivas gráficas 3D

- Vértice
  - Punto en un espacio tridimensional , que almacena las coordenadas (X,Y,Z), el color y opcionalmente una coordenada de textura.

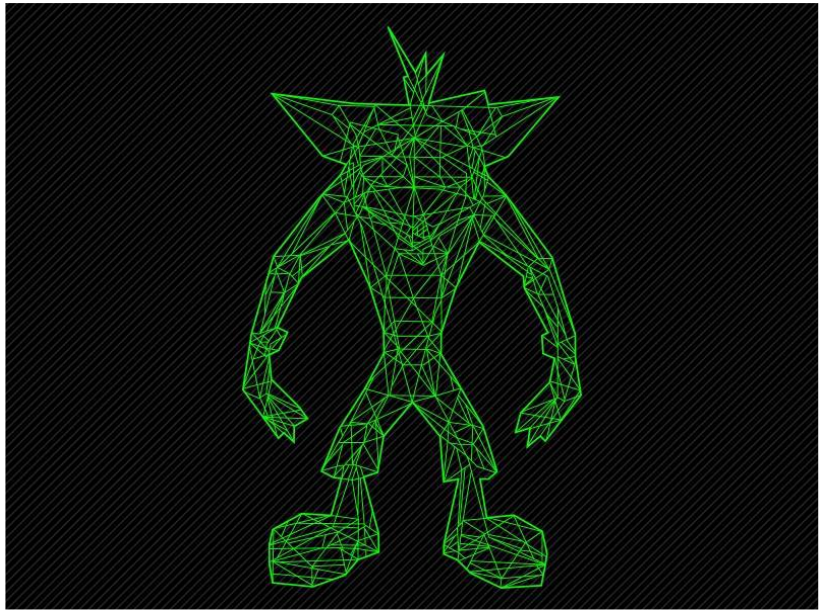
- Arista
  - Línea en un entorno 3D que se encuentra delimitada por dos vértices, cada uno en uno de sus extremos.

- Polígono
  - La unión de varias aristas de forma circular cerrada permite formar polígonos.

- Malla3D
  - La unión de varios polígonos permite crear mallas 3D.

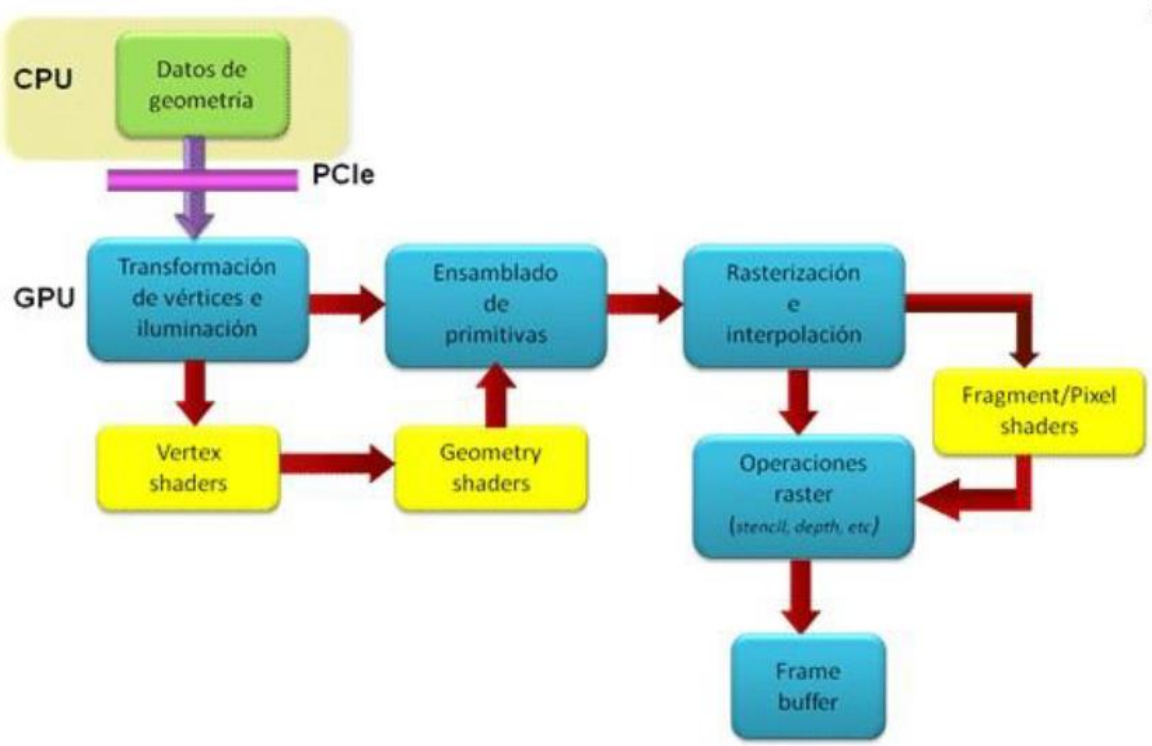
Primitivas gráficas 3D

- QUADS (SATURN) https://youtu.be/FM3cQt-3kGM
- TRIANGULOS (PSX) https://youtu.be/VFQjE0n2e_k



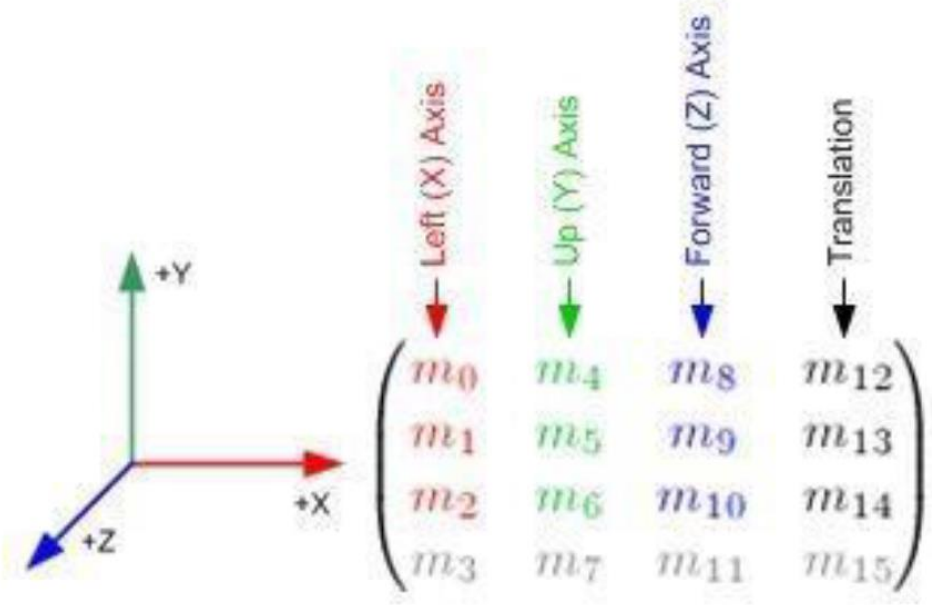
Pipeline gráfico: Fases

- Emisión de primitivas gráficas desde la CPU
- Transformaciones
- Iluminación de vértices
- Proyección
- Clipping
- Rasterización
- Texturizado y sombreado



Matrices

- Las librerías de gráficos 3D utilizan la representación matricial de un sistema de coordenadas homogéneas.
- Se utiliza una matriz cuadrada de orden 4para representar las transformaciones de los elementos de la escena.
- En OpenGL, las transformaciones se representan mediante la matriz:



Matriz de traslación

$$\begin{bmatrix} 1 & 0 & 0 & X \\ 0 & 1 & 0 & Y \\ 0 & 0 & 1 & Z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Matriz de escalado

$$\begin{bmatrix} x & 0 & 0 & 0 \\ 0 & y & 0 & 0 \\ 0 & 0 & z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Matriz de rotación

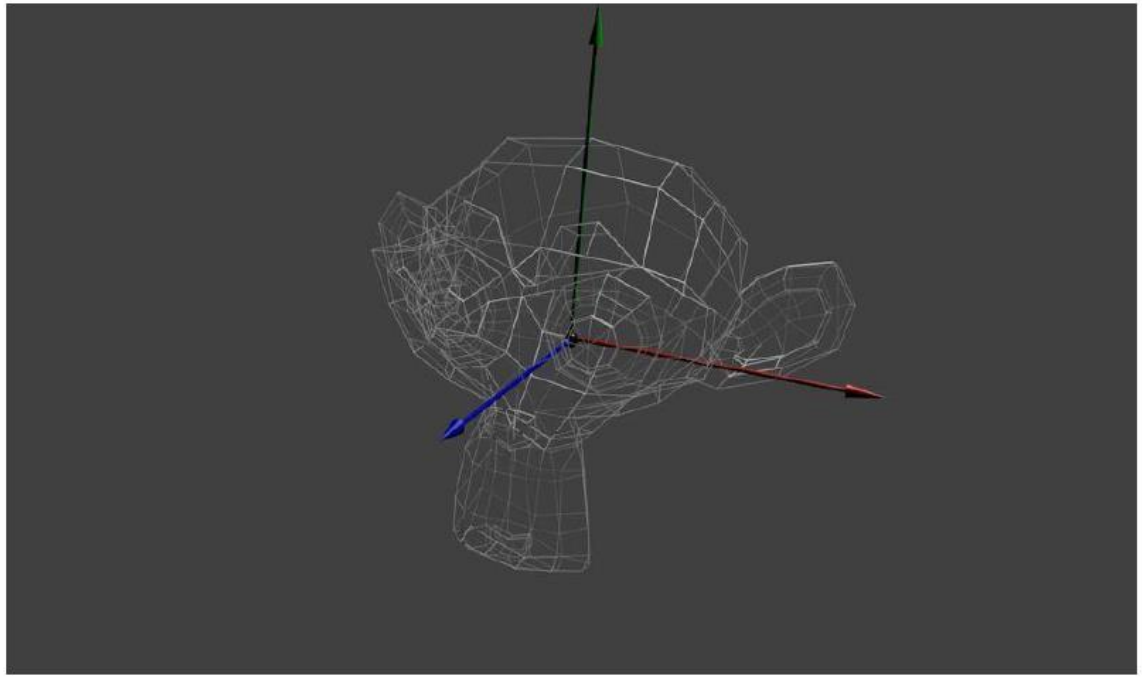
$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$ **Rotación en x**

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$ **Rotación en y**

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$ **Rotación en z**

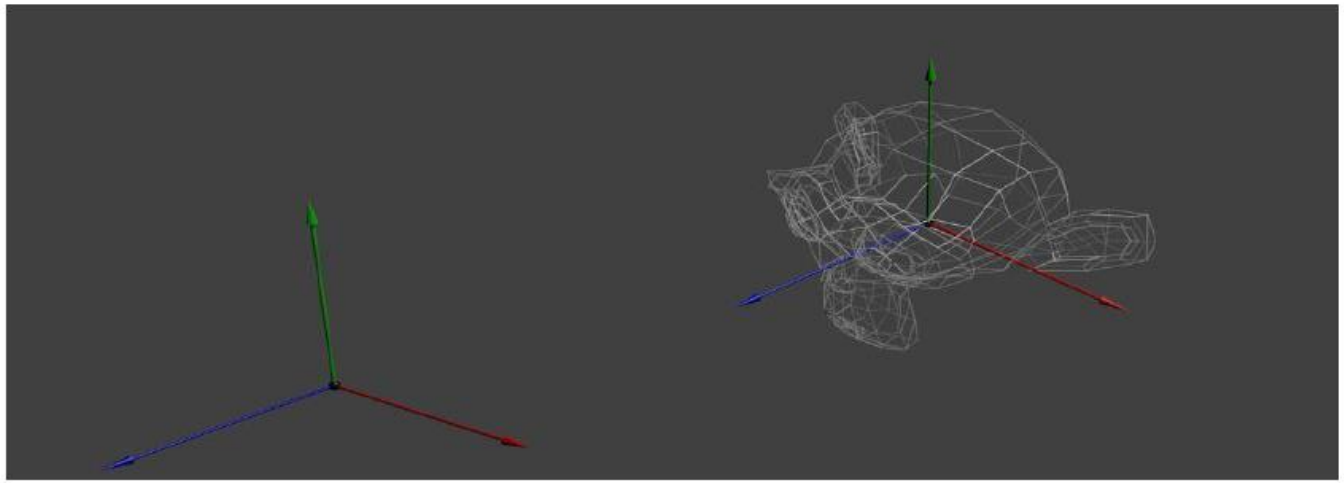
Modelo, Vista, Proyección

- Cuando definimos los vértices de un modelo, sus coordenadas son relativas al origen del modelo.
- Decimos que dichas coordenadas están en el espacio del modelo.



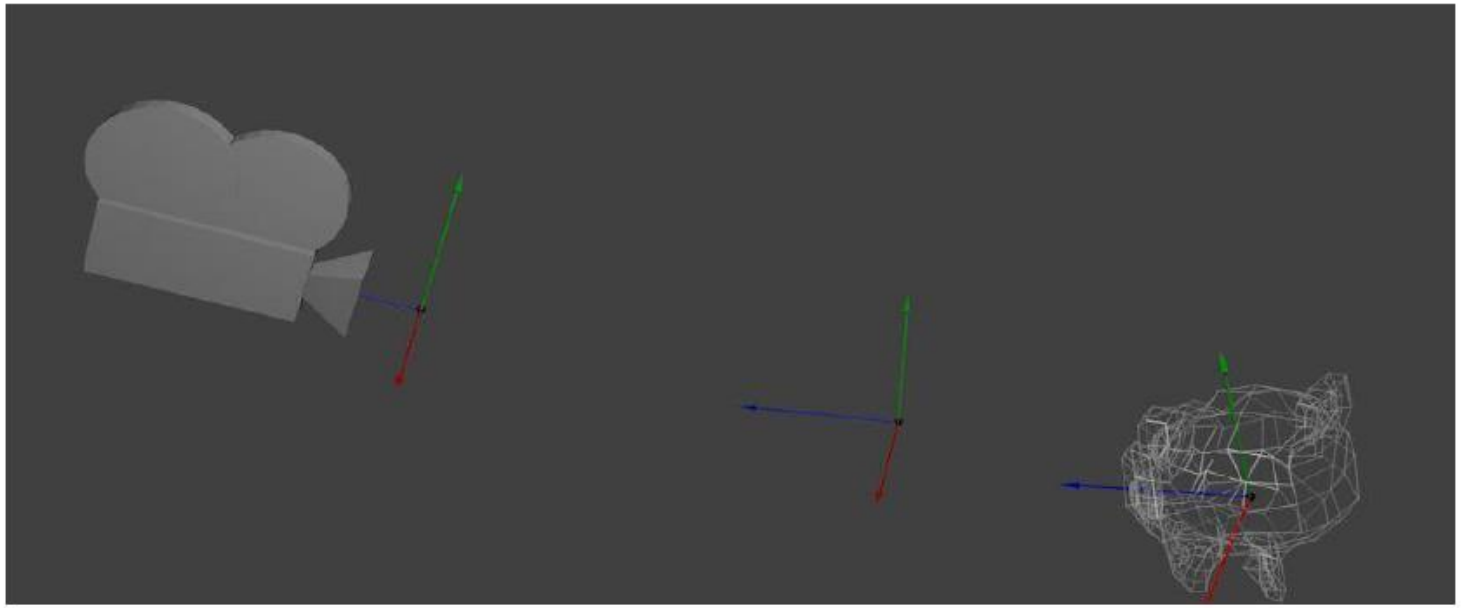
Modelo, Vista, Proyección

- **Matriz modelo**: Queremos aplicar una serie de transformaciones a este modelo para situarlo en la escena.
- Las transformaciones se aplican en el orden:
  - escala, rotación, traslación.
- Generamos una matriz con estas transformaciones y la aplicamos a todos los vértices del modelo.
- Ahora tenemos las coordenadas en el espacio de la escena.



Modelo, Vista, Proyección

- Matriz Vista : Cuando queremos visualizar esta escena, lo haremos desde la transformación relativa de una cámara u observador.
- Se debe definir una matriz con la transformación de dicho observador.
- Al multiplicar todos los vértices por esta Matriz Vista, pasamos al espacio del observador.



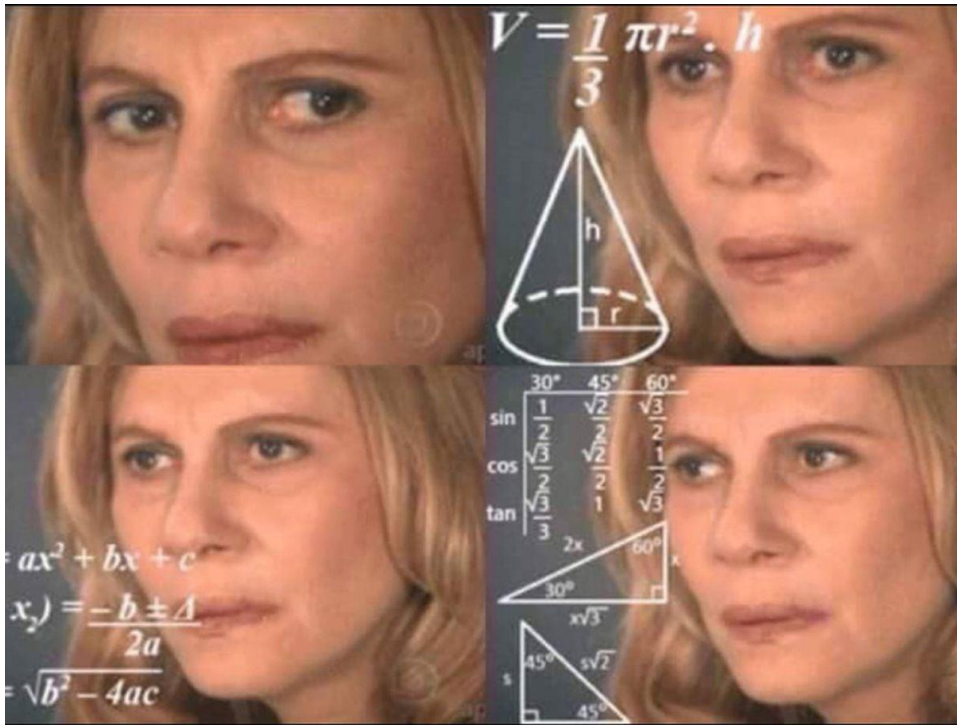
Modelo, Vista, Proyección

- Matriz Proyección : Una vez que tenemos nuestros vértices en el espacio del observador, aún es necesario proyectarlos sobre un plano para poder representarlos en la pantalla.
- Esta última transformación se realiza mediante la Matriz Proyección.
- Puede ser en perspectiva u ortogonal.

Modelo, Vista, Proyección

- La secuencia de transformaciones necesaria para proyectar gráficos 3D en la pantalla es la siguiente:
  - Coordenadas de modelo
  - Coordenadas del mundo 3D
  - Coordenadas de la cámara
  - Coordenadas homogéneas(proyección)
- A la combinación de estas tres matrices la llamamos matriz MVP.

¿Dudas?


$V = \frac{1}{3}\pi r^2 . h$
h
r
$ax^2 + bx + c$
$x_2) = \frac{-b \pm \Delta}{2a}$
$\sqrt{b^2 - 4ac}$
30° 45° 60°
sin
cos
tan
2x
60°
30°
x√3
45°
s√2
s
45°