Introducción. “The graphics pilpeline”

M.C. Hernandez
<mamen.hernandez@ehu.eus>
Joseba Makazaga
<joseba.makazaga@ehu.eus>

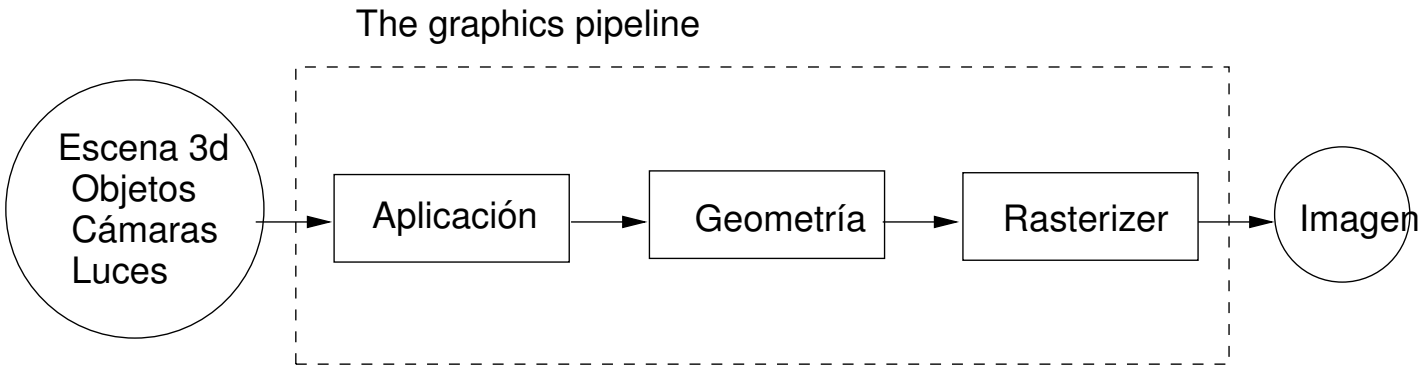
UPV/EHU

Bibliografia

- David H. Eberly. *3D Game Engine Design : A Practical Approach to Real-Time Computer Graphics (The Morgan Kaufmann Series in Interactive 3D Technology)*. Morgan Kaufmann, 2000. ISBN 1-55860-593-2.
- James D. Foley, Andries van Dam, Steven K. Feiner, John F. Hughes, and Richard L. Phillips. *Introduction to Computer Graphics*. Addison-Wesley Professional, 1993. ISBN 0-201-60921-5.
- Tomas Akenine-Moller, Eric Haines, and Naty Hoffman. *Real-Time Rendering (3rd Edition)*. AK Peters, Ltd., 2008. ISBN 978-1-56881-424-7

The graphics rendering pipeline

- Pipeline: “motor” que genera imágenes a partir de una escena 3D
- Tres pasos conceptuales
  - Aplicación (Ejecutada en el procesador principal)
  - Geometría
  - Discretización (*Rasterizer*)



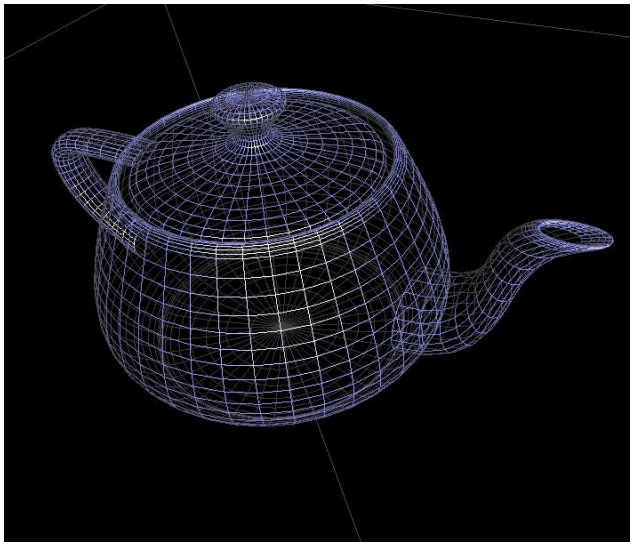
Escenas

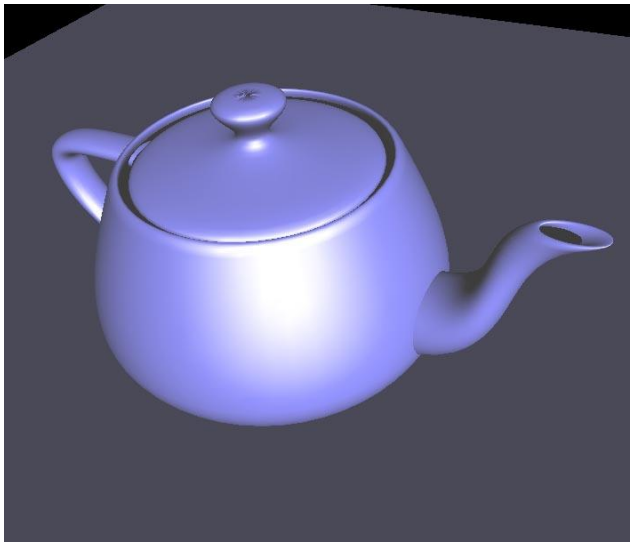
La escena 3D se compone de

1. Objetos
   - Geometría (rectas, triángulos, curvas y superficies . . .)
     - Polígonos
     - Vértices (Posición en 3D, vector normal, . . .)
   - Características de los materiales de los objetos
   - Texturas
2. Fuentes de luz
3. Cámaras
   - Cámara virtual
   - Decide qué entra en la foto

Objetos

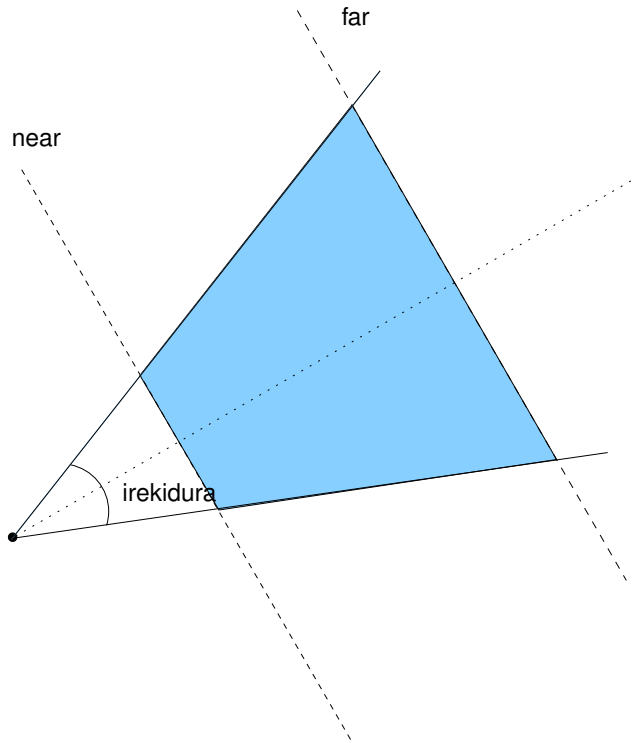
- Primitivas: puntos, rectas, triángulos, . . .
- Las superficies son aproximaciones mediante primitivas





Cámara virtual

- La escena se visualiza a traves de la cámara
- Para definirla: Posición, dirección de mira, “up vector”, apertura (“field of view”, fov), plano cercano (“near plane”) y plano lejano (“far plane”)



Aplicación gráfica interactiva

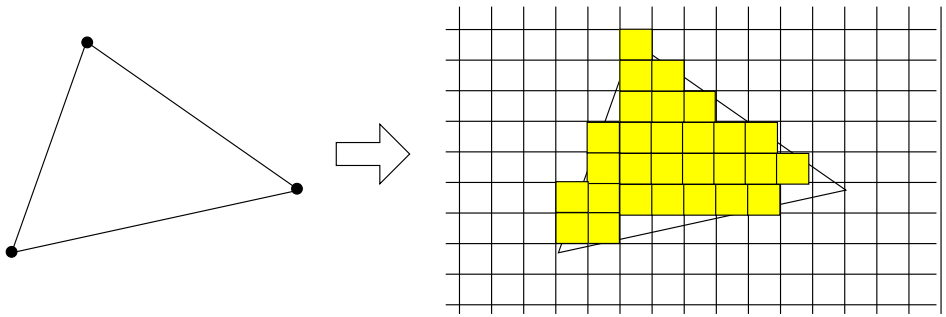
- Se ejecuta en la CPU
  - Lógica del programa
  - Grafo de la escena
  - Control de la cámara . . .
- Por ejemplo
  - Detección de colisiones
  - Técnicas de aceleracón
  - Animación
- La aplicación envía primitivas al hardware gráfico o GPU

Paso de la geometría

- En este paso se realizan operaciones geométricas sobre los datos de entrada
- Etapas usuales
  - Posicionar objetos (multiplicación matricial)
  - Paso al sistema de referencia de la cámara (multiplicación matricial)
  - Cálculo de la iluminación en los vértices
  - Proyecciones
  - Recorte (clipping)
  - Mapeo de ventana (encuadre)

Discretización (Rasterizer)

- Objetivo: Dibujar las primitivas de salida de la etapa de geometria
- A su vez, realizar operaciones a nivel de pixel y de textura
- En esta etapa se visualiza la escena



Resumiendo

- El programador envía primitivas gráficas al motor gráfico mediante una API
  - OpenGL
  - Direct 3D
  - WebGL
  - ...
- En la etapa geométrica: operaciones referentes a los vértices
- Discretización: operaciones correspondientes a los pixels

Etapa geométrica. Cambio de modelo

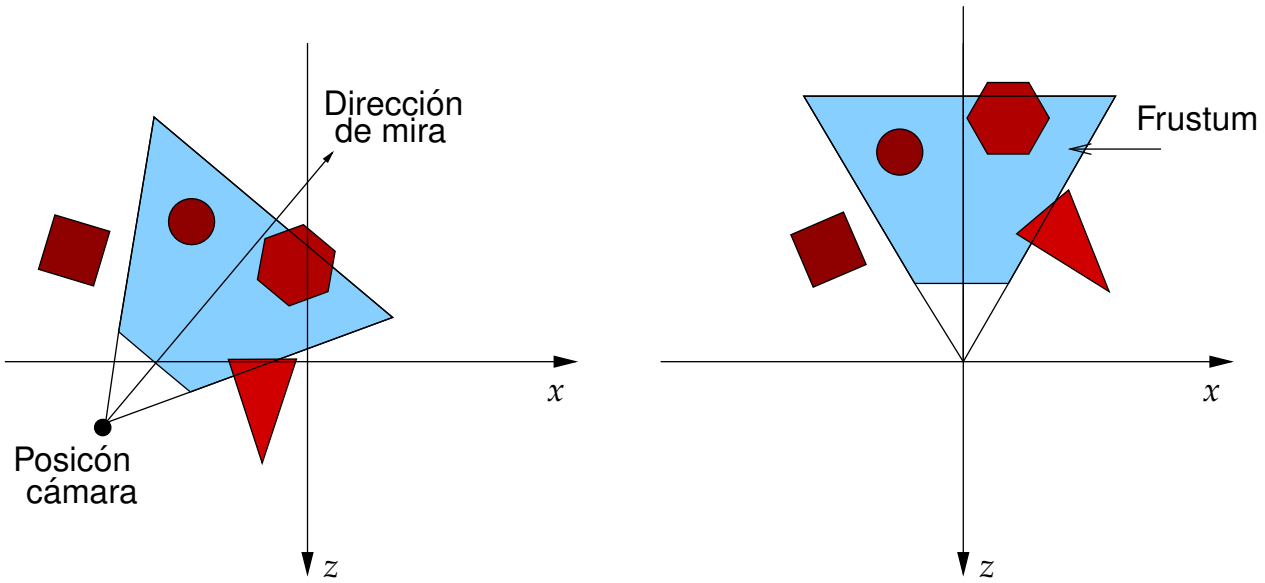
Cambio de modelo

- Los objetos geométricos se definen en su “espacio de modelo” (local coordinates)
- Los objetos se mueven, se giran . . . y se posicionan en el “espacio de la escena” (world coordinates)
- Multiplicaciónes de matrices de dimensión 4×4
- Las matrices se pueden modificar en el tiempo: animación

Etapa geométrica. Cambio de vista

Cambio de vista

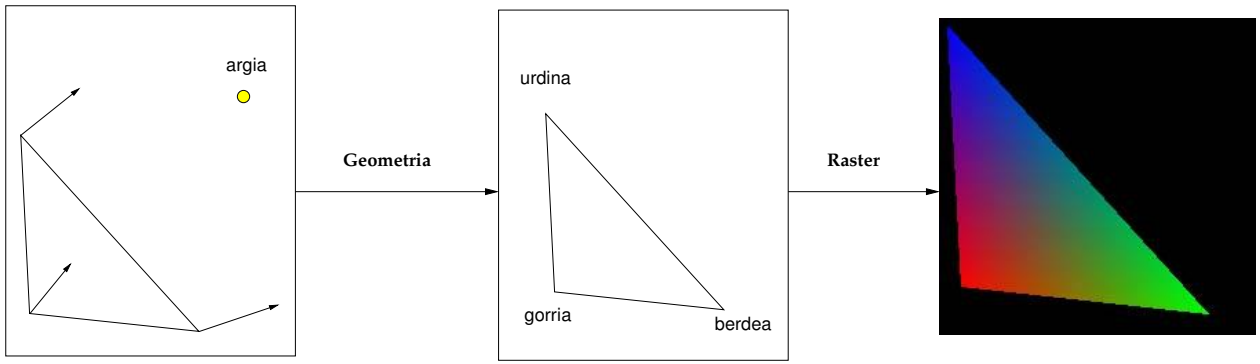
- La cámara se puede mover como cualquier otro objeto
- Cambio de vista: depende de la posición y orientación de la cámara
- Pero en este caso la transformación es la inversa, el resto de objetos se representan en función de la cámara.



Geometría. Iluminación

Iluminación

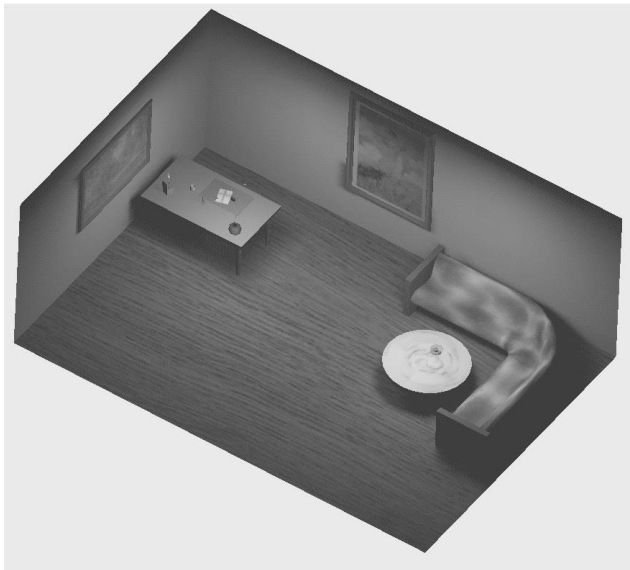
- Se calcula la “iluminación” en los vértices
- Hace falta
  - Modelo de iluminación

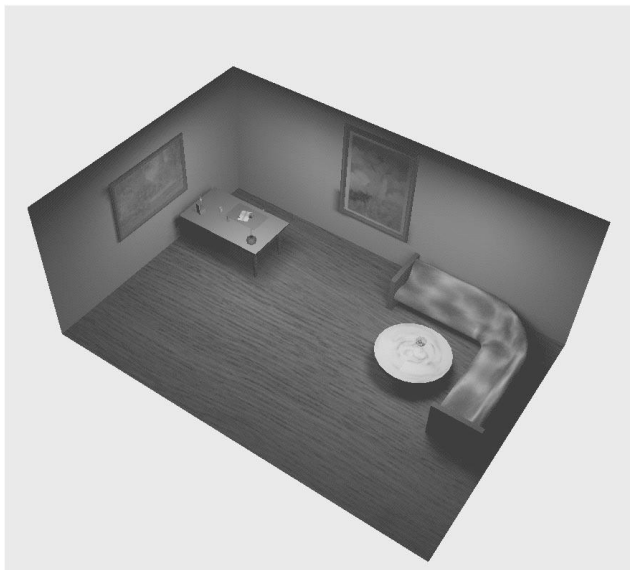


Geometría. Proyecciones

Proyecciones

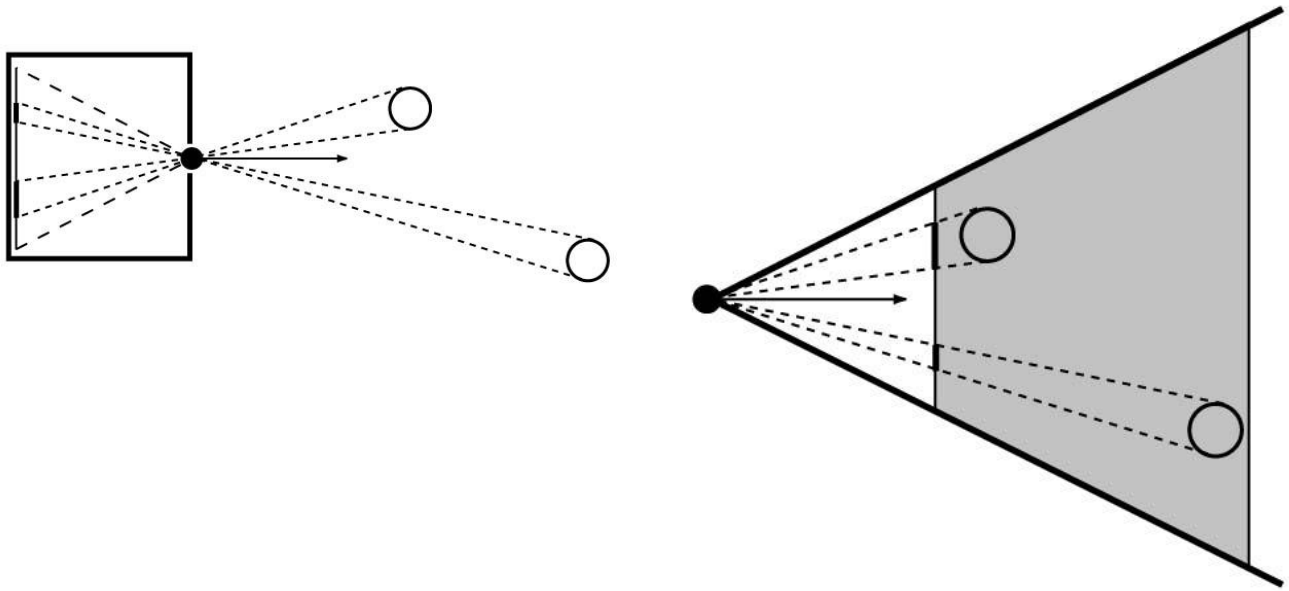
- Hay que proyectar un mundo 3D sobre una pantalla 2D
- 2 modos
  - Proyección ortogonal
    - Se utiliza en aplicaciones CAD/CAM
  - Perspectiva
    - Nuestra visión es de este tipo





Geometria. Proyecciones

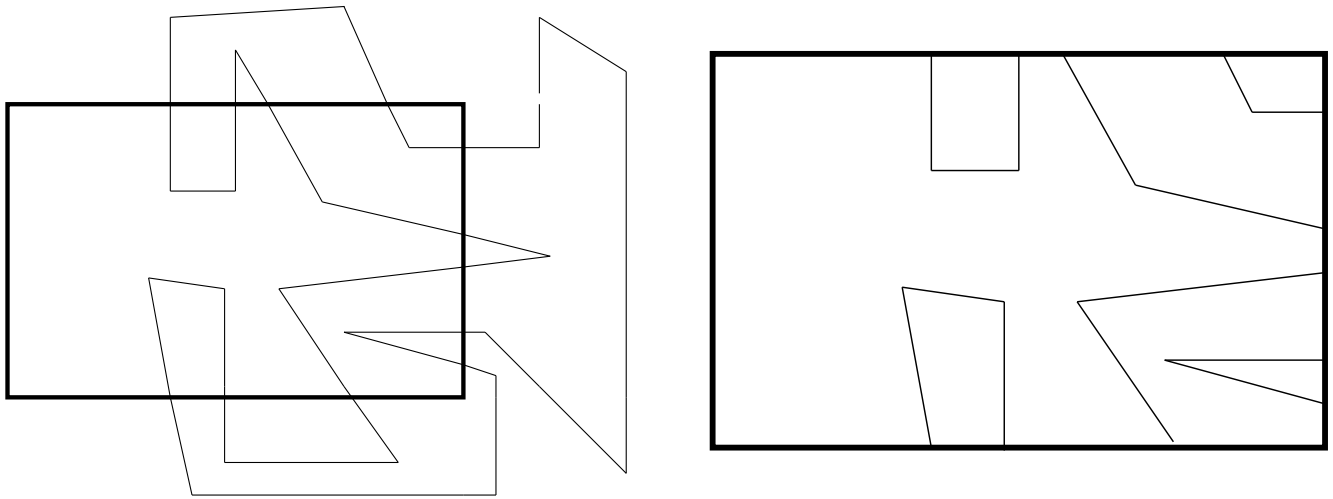
- La proyección se realiza mediante multiplicación de matrices
- Proceso parecido al realizado en el ojo



Geometría. Recorte (clipping)

Recorde

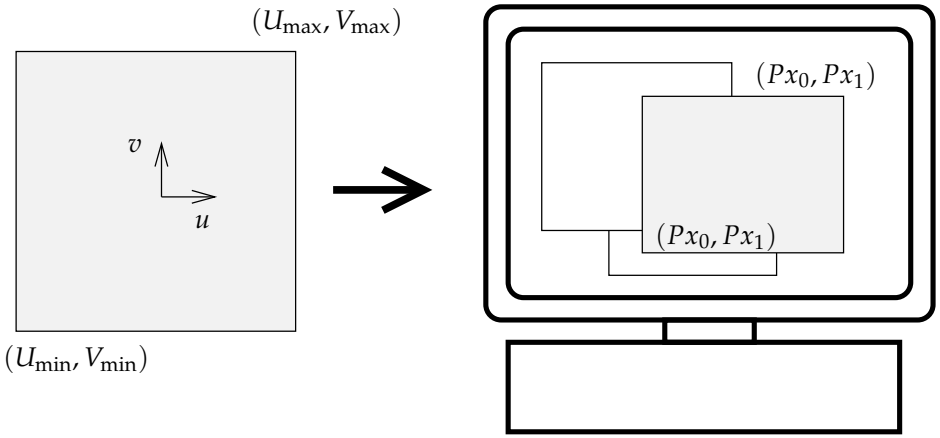
- Tras la proyección, la pantalla es rectangular (cubo)
- Las primitivas hay que limitarlas al rectángulo (recorte)



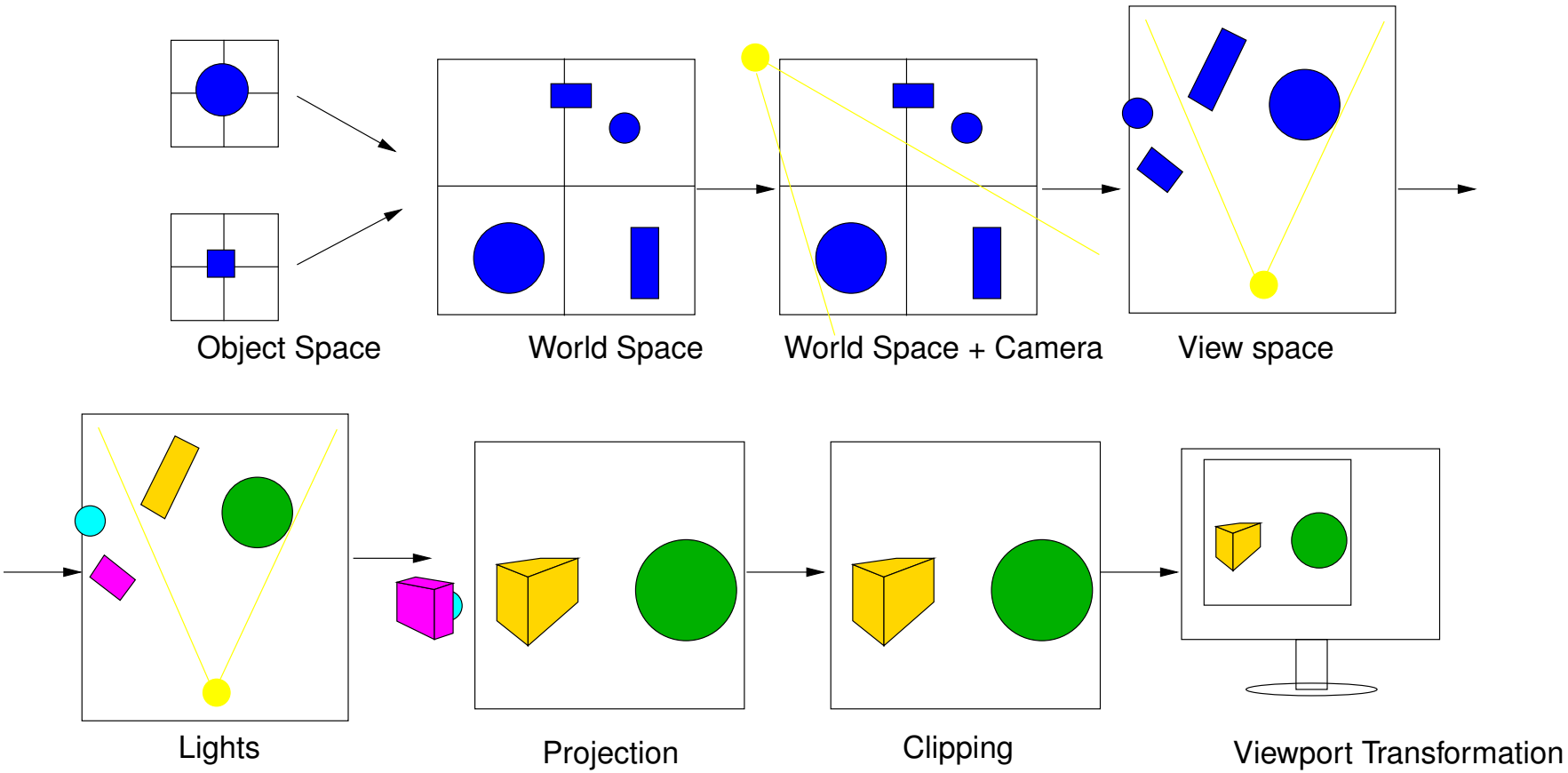
Geometría. Encuadre

Encuadre

- Tras el recorte, hay que pasar las primitivas al “espacio de la pantalla” (no tiene por qué ser rectangular)
- Las coordenadas pantalla de las primitivas (más profundidad z) llegan a la etapa de “discretización”
- Traslación y escalado



Geometría. Resumen



Discretización (rasterizer)

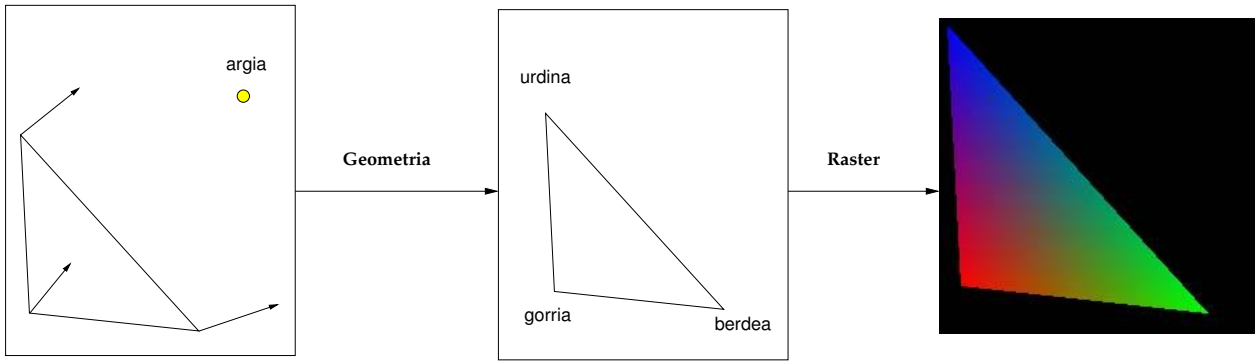
- Conversiones a fragmentos lineales
  - Decisón de los pixels que toman parte en una primitiva
- Asignación de texturas
  - Pegar imágenes en los polígonos
- Interpolación en los polígonos
- Visualización: Z-buffer
- “Double buffer”
- . . .

Discretización. Conversión a fragmentos

- Los vértices de los plígonos vienen de la etapa geométrica
- Hay que decidir los pixels que corresponden a la primitiva
  - A una recta, ó a un punto
- Realizar operaciones a nivel pixel
  - Interpolación
  - Mapeo de texturas
  - Z-buffer
  - . . .

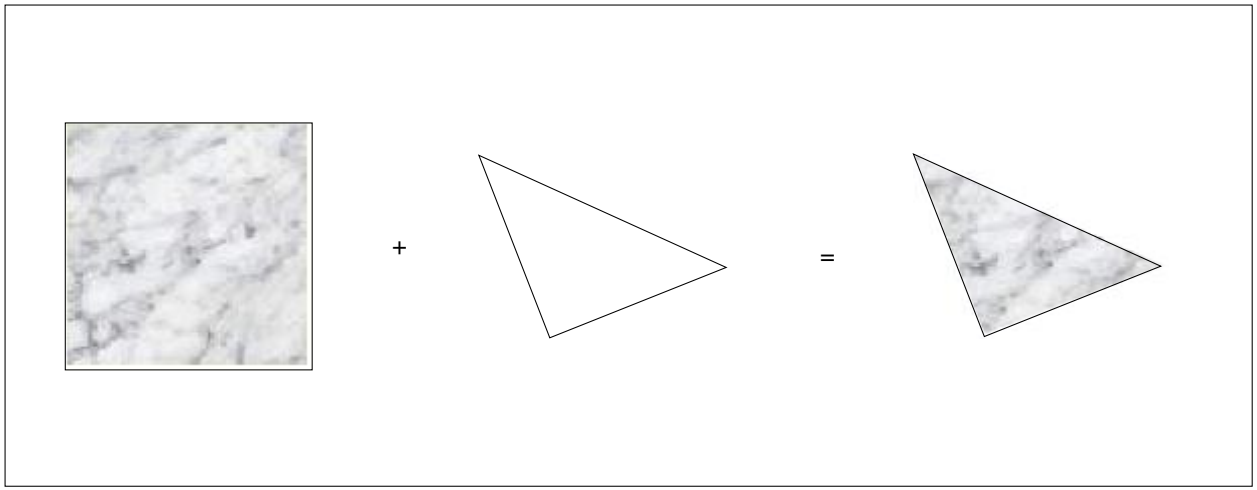
Discretización. Interpolación

- Interpolación del color en los pixels internos al polígono
  - modelo de sombreado (shading): *gouraud*, *phong*



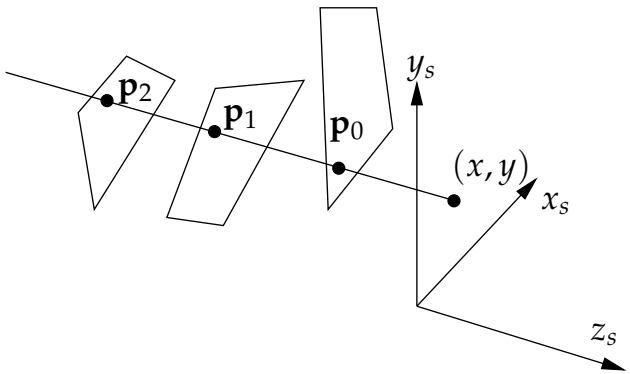
Discretización. Texturas

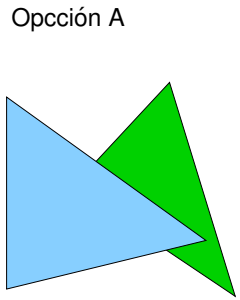
- “Pegar” imágenes en los polígonos
- Muchos usos
  - Incremento de realismo
  - Bump mapping
  - Simulación de reflejos
  - Guardar la iluminación
  - . . .

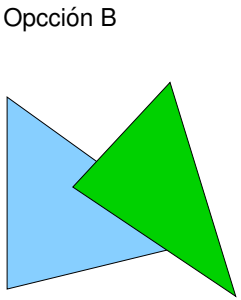


Discretización. Visualización

- En un pixel de la pantalla se pueden proyectar varios objetos
- Tras la proyección, Hay que tener en cuenta la coordenada z de los objetos
- En la figura el objeto más cercano es $\mathbf{p}_0$







Discretización. Z-buffer

- El *Z-buffer* ayuda a resolver los problemas
  - Se guarda la profundidad (z) de cada pixel
  - Al dibujar un triángulo, se calcula la profundidad de cada pixel interno
  - Compara la componente z del pixel con el contenido del *Z-buffer* del pixel
  - Si la del triángulo es menor, se cambia el pixel (asi como el *Z-buffer*)
  - En otro caso, no hacer nada!

  Permite el tratamiento de las primitivas en cualquier orden

Discretización. Double-buffering

- El monitor muestra la imagen de golpe
  - Frecuencia de refresco (70-100 Hz)
- Para mostrar las primitivas del la siguiente imagen
  - flickering
- Es preferible trabajar con dos buffers
  - Uno contiene la imagen actual
  - En el otro se dibujan los datos de la siguiente imagen
  - Tras la terminación de la nueva imagen, se intercambian los buffers de golpe