Sección de Informática Gráfica VALENCIA | Computer Graphics Group

UNIVERSIDAD POLITECNICA DE VALENCIA

Visibilidad

Introducción

Transformación perspectiva-paralela

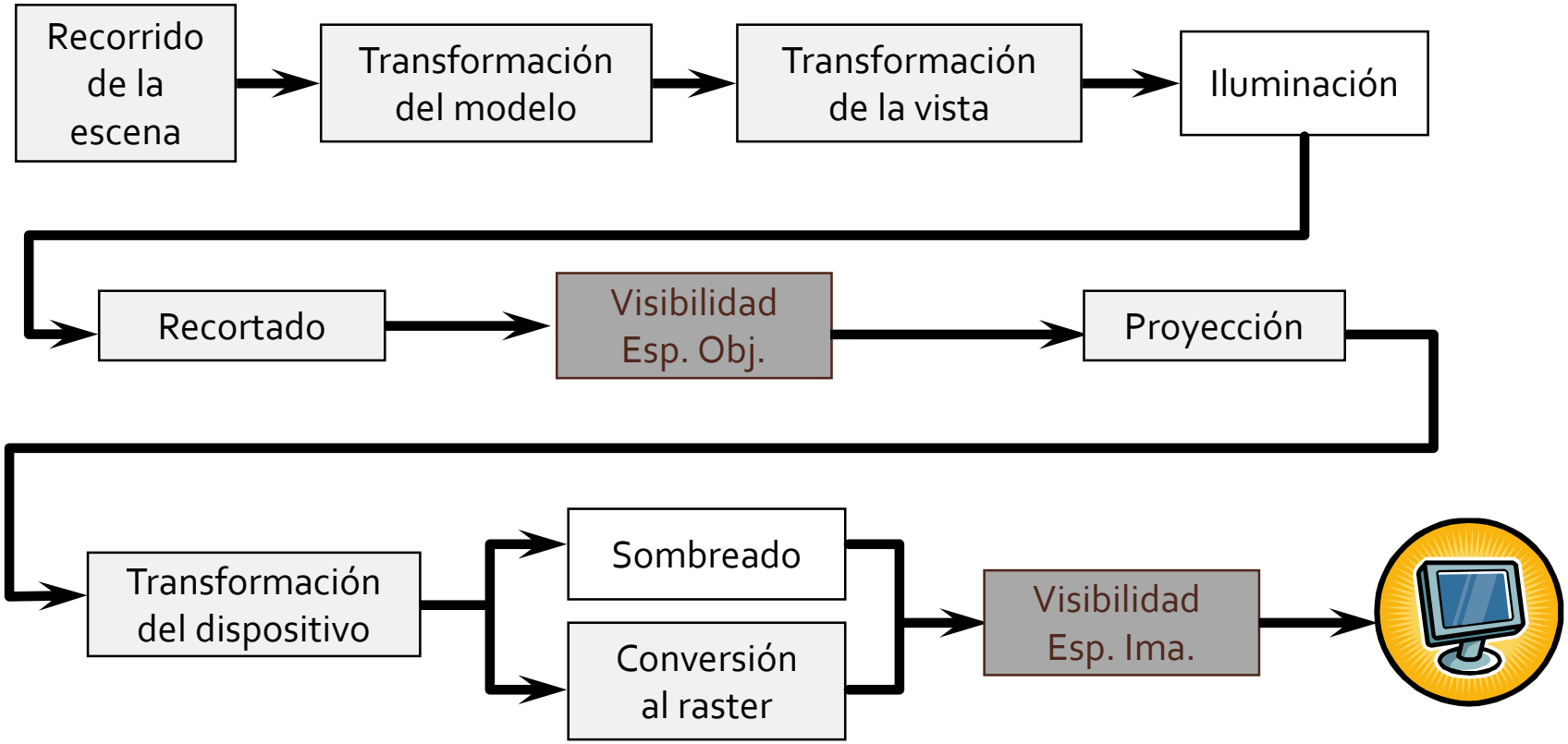
Clasificación de algoritmos

Algoritmo del pintor

Algoritmo del Z-buffer

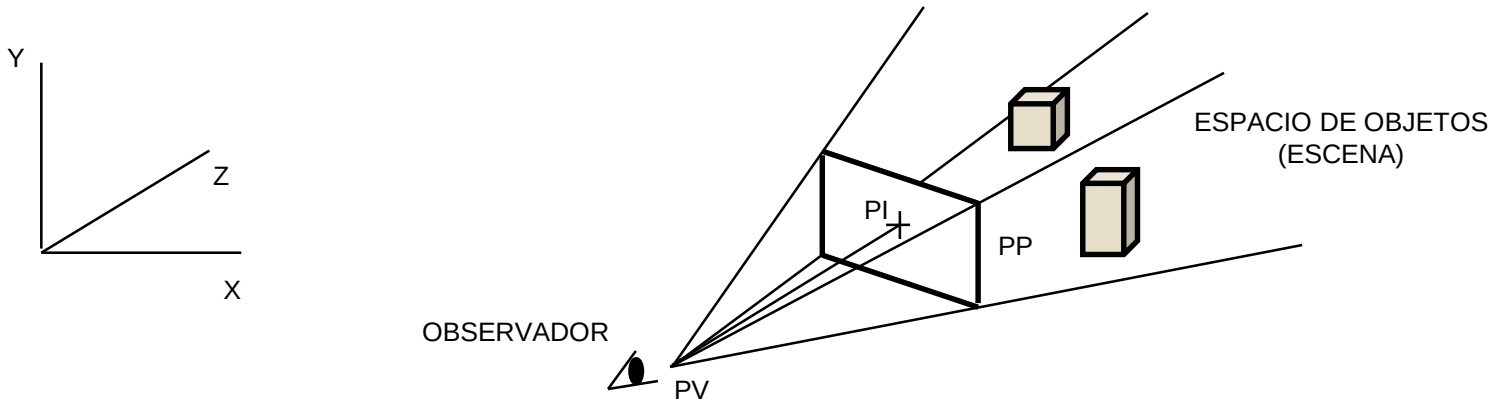
Eliminación de caras traseras

Introducción



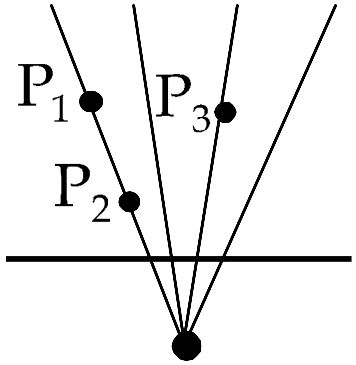
Introducción

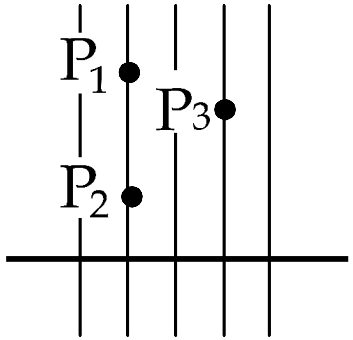
- Presentación del problema
  - Dado un conjunto de objetos 3D y un modelo de cámara sintética, determinar qué líneas y superficies son visibles
    - Desde el punto de vista (para una cámara perspectiva)
    - A lo largo de una dirección de proyección (para una cámara ortográfica)
  - Especificación de la vista: Punto de vista (PV),Punto de interés (PI), Plano de proyección (PP), Campo de visión



Transformación perspectiva-paralela

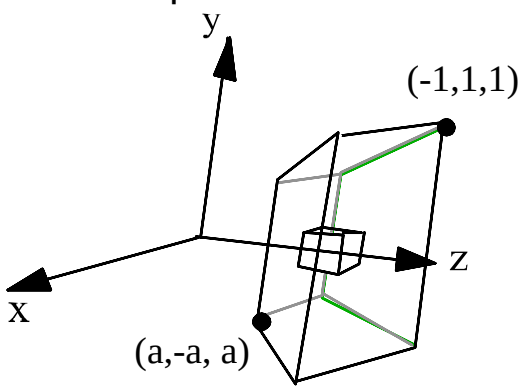
- El problema básico del cálculo de visibilidad es decidir si, dados dos puntos (P_1 y P_2), uno tapa al otro
- La cuestión se reduce a:
  - ¿Están los dos puntos en el mismo proyector?
  - Si lo están, comparar z1 y z2 para decidir cuál está más próximo a la cámara
- Comprobar si los dos puntos están en el mismo proyector
  - Proyección perspectiva
  - Proyección paralela



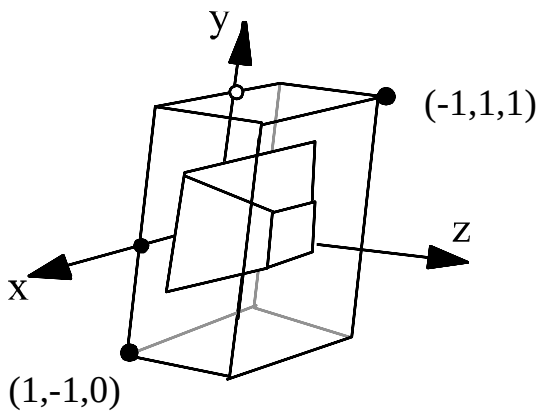


Transformación perspectiva-paralela

- Preserva profundidad relativa, líneas y planos
- En la perspectiva realiza la disminución del tamaño en función de la profundidad
- Ejemplo: Volumen de la vista de una proyección perspectiva, con un cubo en su interior

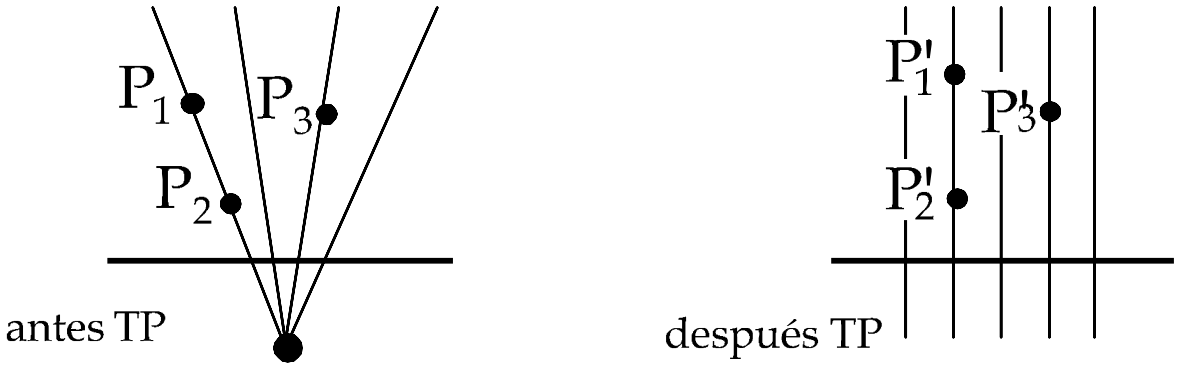


- Tras la transformación perspectiva-paralela se obtiene el volumen canónico de la proyección paralela con el cubo distorsionado en su interior



Transformación perspectiva-paralela

- El recortado tras realizar la transformación consiste simplemente en recortar contra los planos: (-1<= x <=1) (-1<= y <=1) (0<= z <=1)
- Las profundidades se comparan después de la transformación:
  - Para que un punto (P'_1) tape a otro (P'_2) se debe cumplir $x'_1 = x'_2$ e $y'_1 = y'_2$, es decir que estén en el mismo proyector: operación sencilla
  - Si los dos pares (x',y') son iguales se compara la coordenada z
- La proyección de los puntos tras la transformación consiste en descartar la componente z



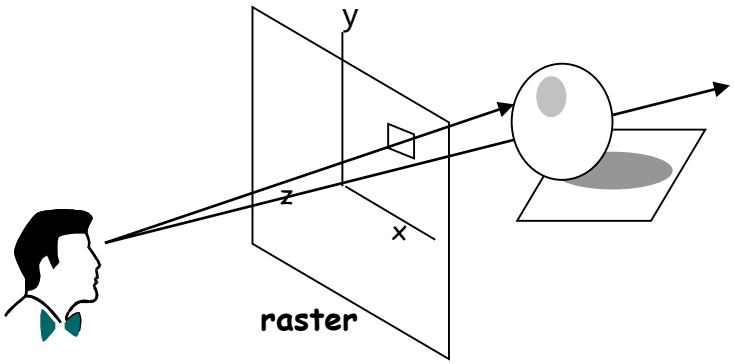
Clasificación de algoritmos

Algoritmos en el espacio de la imagen

- La visibilidad se calcula en coordenadas del espacio de la imagen
- Se calcula cuál de los *n* objetos de la escena es visible para cada uno de los *p* píxeles de la imagen

  para cada píxel hacer
  - trazar una visual que pase por PV y el píxel
  - determinar el objeto más cercano que intersecciona con la visual
  - dibujar el píxel del color del objeto

  fin_para
- Complejidad: *O(np), puesto que para cada píxel tenemos que comprobar todos los polígonos*
- La operación más compleja es el cálculo de la intersección recta-polígono



Clasificación de algoritmos

Algoritmos en el espacio de objetos

- La visibilidad se calcula en coordenadas del espacio de objetos
- Cada objeto se compara con todos los demás
- Los objetos (o partes de los objetos) que no son visibles, se eliminan de la escena

para cada objeto hacer

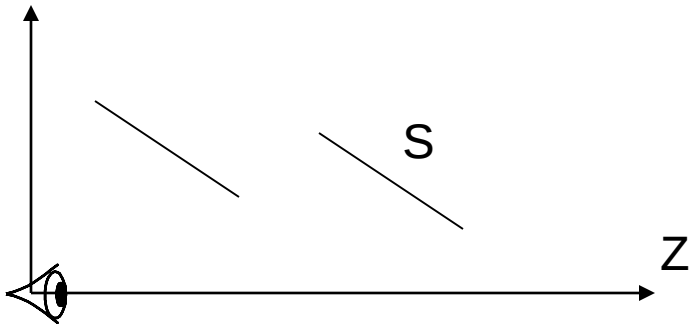
- determinar las partes del objeto que no están ocultas:
  - por otras partes de sí mismo
  - por otros objetos
- dibujar las partes no ocultas con el color apropiado

fin para

- Complejidad: *$O(n^2)$ puesto que cada objeto se tiene que comparar con todos los demás*
- La operación más compleja es el cálculo de la intersección polígono-polígono
- Después de calcular qué objetos son visibles se tienen que convertir al raster para obtener la imagen final

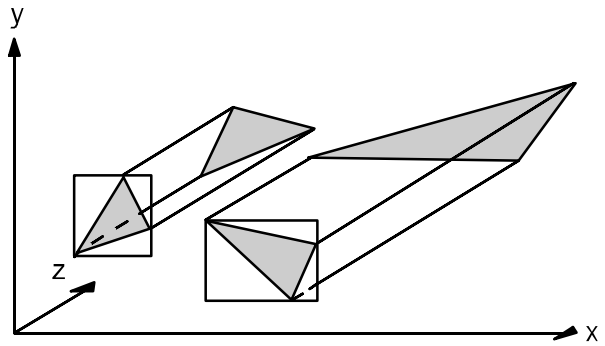
Algoritmo del pintor

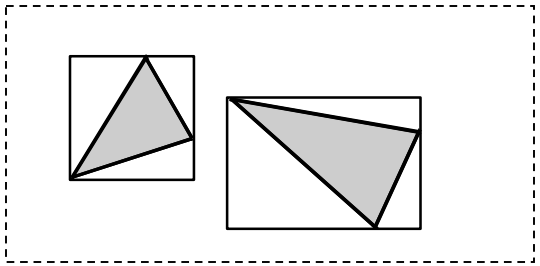
- Ordenación en Z (Newell, Newell y Sancha)
- Espacio del objeto
- Estrategia: Ordenar los polígonos en profundidad, del más lejano al más cercano y después dibujarlos en ese orden
  - Realizar una primera ordenación por la Z más alejada de cada polígono
  - Seleccionar la superficie (S) al final de la lista
  - Si no hay solape con otros polígonos de la lista
    - Convertir S al raster
  - En otro caso
    - Realizar test posteriores para determinar si se necesita reordenar

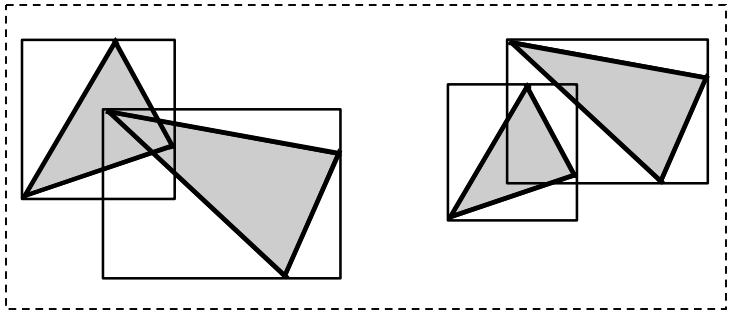


Algoritmo del pintor

- Si el polígono más alejado en la lista S solapa con otro u otros polígonos
  - Realizar los siguientes tests entre S y cada polígono que se solape con él (S'). Si algún test es CIERTO, entonces S se debería convertir al raster antes de S'.
  1. Los rectángulos de inclusión en las direcciones XY para las dos superficies no se solapan

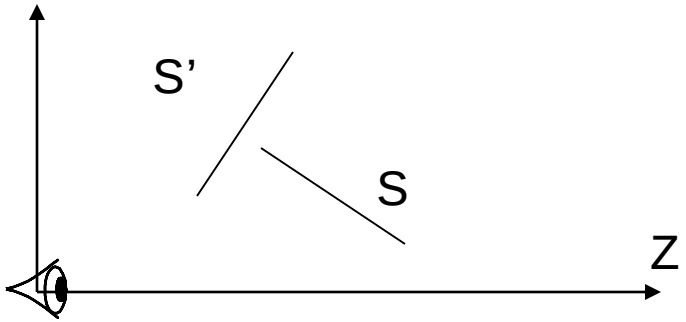




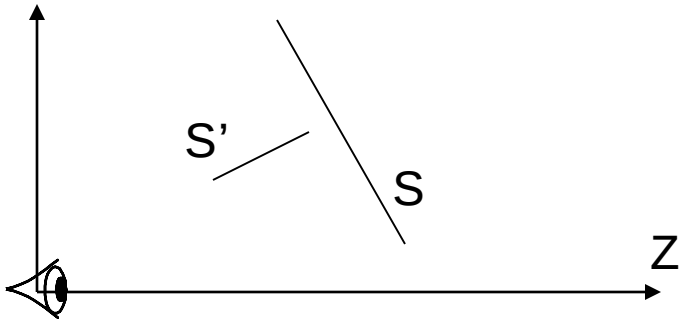


Algoritmo del pintor

2. El polígono S está completamente detrás del polígono con el que se solapa respecto a la posición de la vista

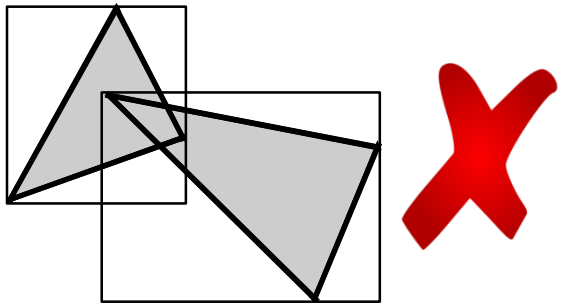


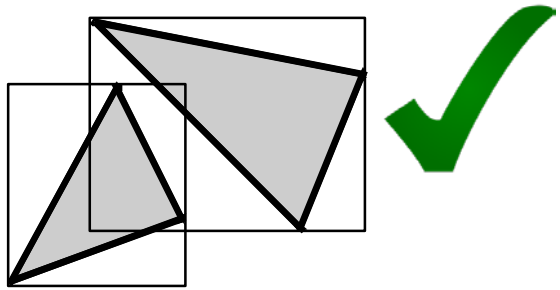
3. El polígono con el que se solapa está completamente delante de S respecto a la posición de la vista



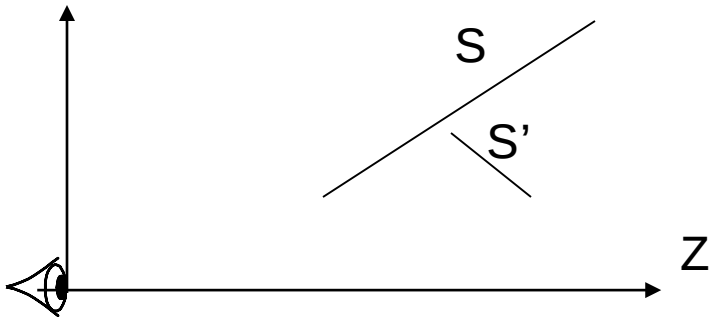
Algoritmo del pintor

4. Las proyecciones de las dos superficies sobre el plano de la vista no se solapan



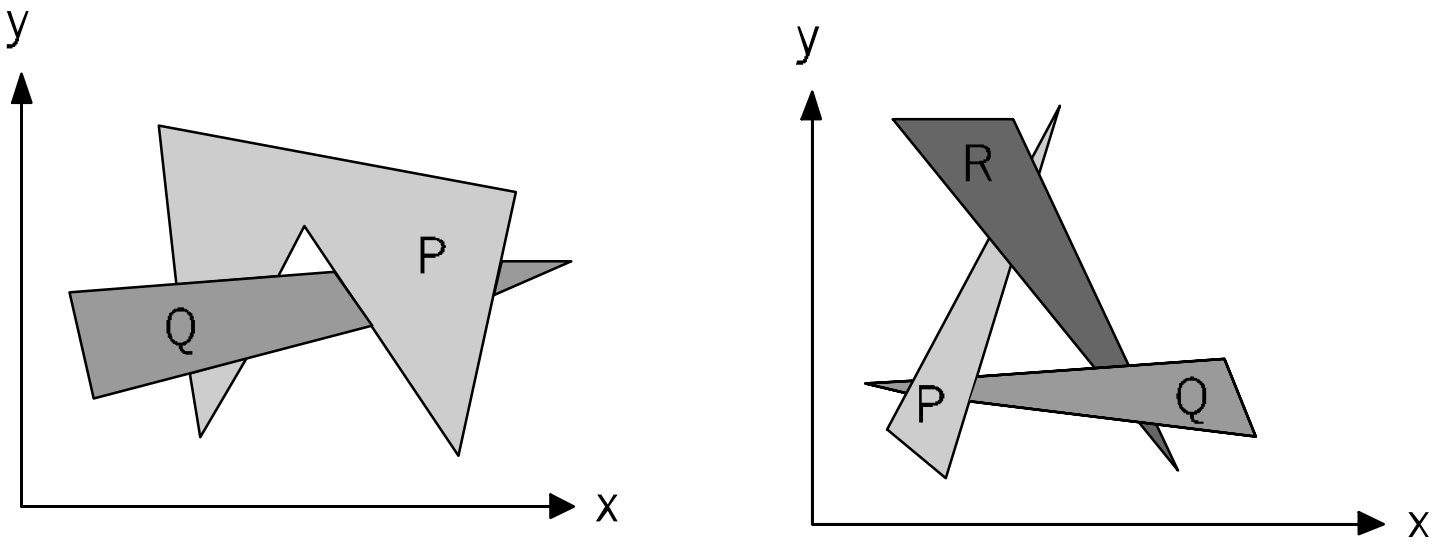


- Si cada polígono que se solapa con S pasa un test al menos
  - Entonces convertir S al raster
- Sino
  - intercambiar S y S' (polígono que falla los cuatro tests)
  - Repetir los tests para cada polígono reordenado



Algoritmo del pintor

- Este algoritmo puede entrar en un bucle infinito cuando dos o más polígonos alternativamente se tapan unos a otros:



- Una vez un polígono se ha reordenado, debería marcarse. Si se va a reordenar de nuevo, entonces se divide el polígono y se continúa el proceso.

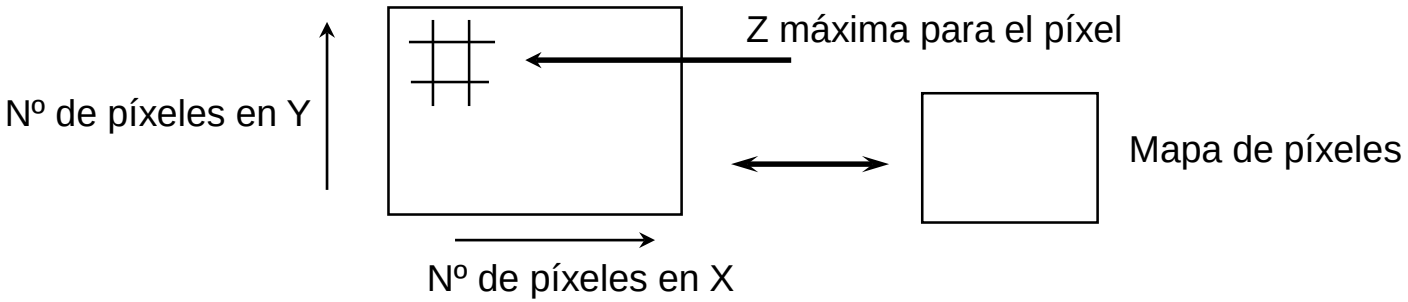
Algoritmo del pintor

- Ventajas:
  - Rápido para escenas simples
  - Es independiente de la resolución
  - La ordenación en profundidad es útil para seleccionar objetos

- Desventajas:
  - Lento para escenas de complejidad media
  - Difícil de implementar y depurar
  - Muchos casos especiales

Z-Buffer

- Características destacables
  - Precisión de Imagen
  - Gran simplicidad
  - Alto consumo de memoria
  - Almacenamiento de la profundidad máxima (Zmax) a lo largo de X, Y
  - Trabaja con escenas de cualquier complejidad
  - No necesita ningún tipo de ordenación
  - Facilidad de implementación hardware
- Estructura de datos
  - Además del mapa de píxeles se utiliza otra matriz (Z-Buffer) para almacenar la profundidad (valor de Z) para cada píxel

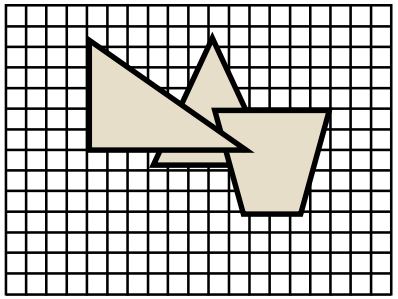


Z-Buffer

- Algoritmo
  - Preproceso
    - El raster se inicializa al color del fondo
    - El Z-Buffer se inicializa a Zmax (1 para el volumen canónico)
  - Proceso
    - Los polígonos se convierten al raster (cálculo de los píxeles que ocupa el polígono) en orden arbitrario
    - Comprobación de cercanía y actualización

Algoritmo Z-Buffer

Inicializar imagen a color del fondo
Inicializar zbuffer a la z de máximo alejamiento
para cada polígono
para cada píxel en la proyección del polígono
z:=z(x,y)
si z más cercana que zbuffer(x,y)
zbuffer(x,y):= z
escribir píxel (x,y) al color conveniente
fin Z-Buffer



Z-Buffer

- Coherencia de profundidad
  - Simplificación del cálculo del valor de *z*

$$Ax + By + Cz + D = 0 \quad \text{(Ecuación del plano)} \quad \rightarrow \quad z(x,y) = \frac{-D - Ax - By}{C}$$

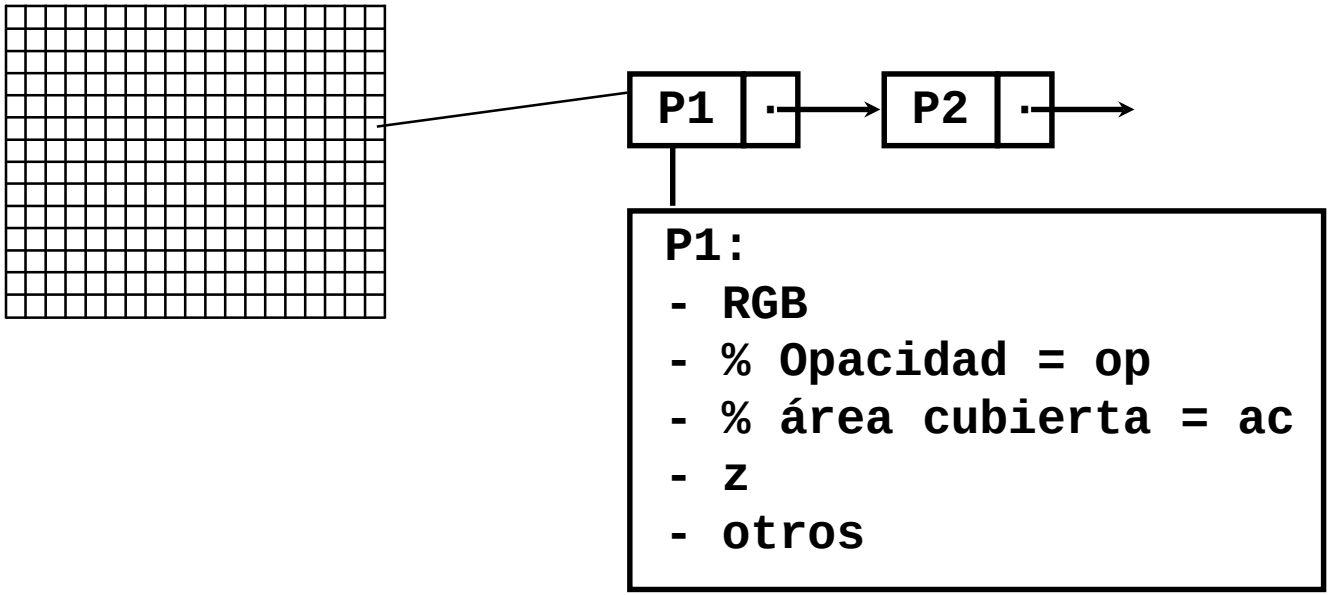
- Realizando el barrido de los píxeles del polígono de fila en fila
- Para cada fila y = cte, Dx = 1, *z* se puede calcular de forma incremental:

$$z(x+1,y) = \frac{-D - A(x+1) - By}{C} = \frac{-D - Ax - By}{C} - \frac{A}{C}$$

$$\boxed{z(x+1,y) = z(x,y) - \frac{A}{C}}$$

Z-Buffer

- Variante del Z-Buffer que contempla la concurrencia de varias superficies sobre el mismo píxel.
- Estructura del A-Buffer



- Cálculo del color del píxel de atrás hacia adelante:

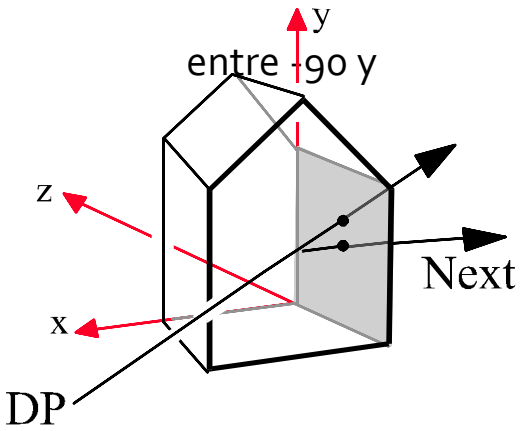
$$rgb(x,y) := (1-a)\cdot rgb(x,y) + a\cdot rgb(actual) \qquad a = op \cdot ac$$

Eliminación de caras traseras (Backface culling)

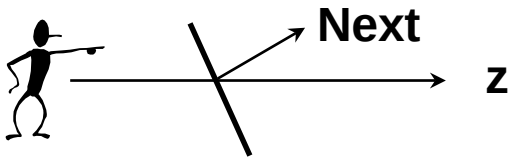
- Situaciones de aplicabilidad: Objetos poliédricos cerrados. Caso especial: poliedros convexos.
- Invarianza de signo del producto escalar entre la normal al plano y cualquier visual que lo atraviese.
  - No son visibles los polígonos cuya normal forma un ángulo entre -90 y +90 con la dirección de observación

Next = Normal externa

DP = Dirección de Proyección



- Aprovechamiento de la transformación perspectiva-paralela:
  - cara trasera = componente z de Next positiva (levógiro)
- Reducción media del 50% de polígonos



Bibliografía

- D. Hearn, M. Baker. Computer Graphics with OpenGL. Pearson Prentice Hall, 4ª edición.
  - Capítulo 16