6 Sombras arrojadas

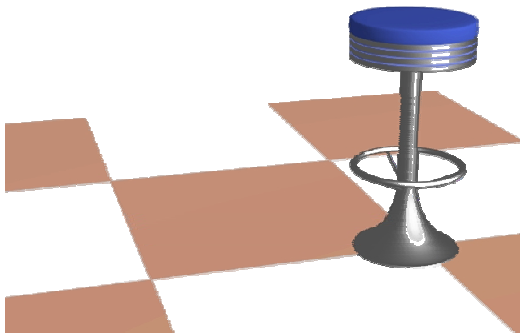
Introducción
Sombras de proyección
Sombras con texturas
Volúmenes de sombra
Mapas de sombras
Comparación de técnicas

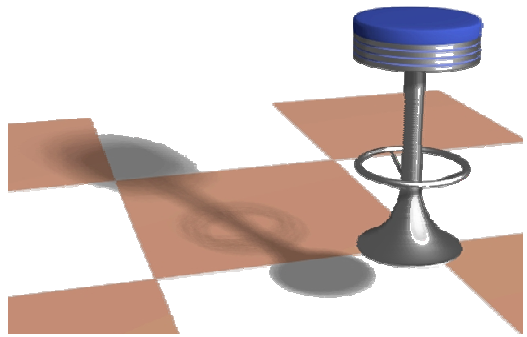
Introducción

- Característica principal
  - Las sombras mejoran el realismo y crean efectos de atmósfera

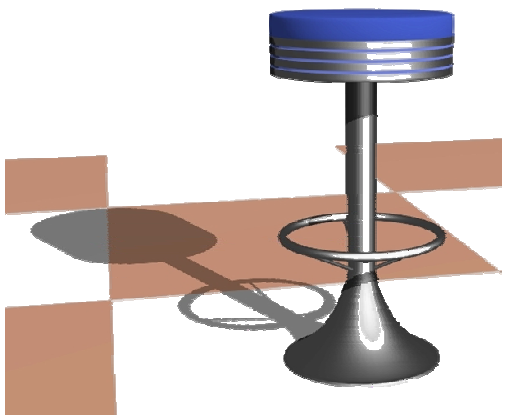
Introducción

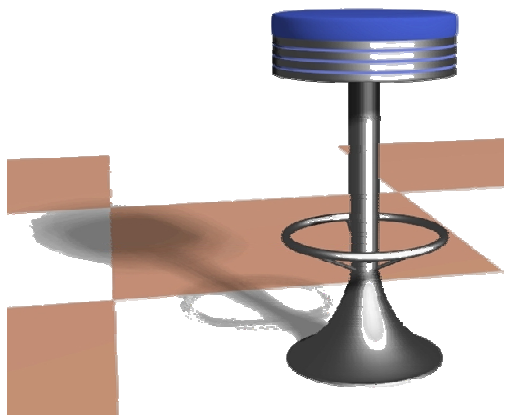
- Las sombras arrojadas dan sensación tridimensional





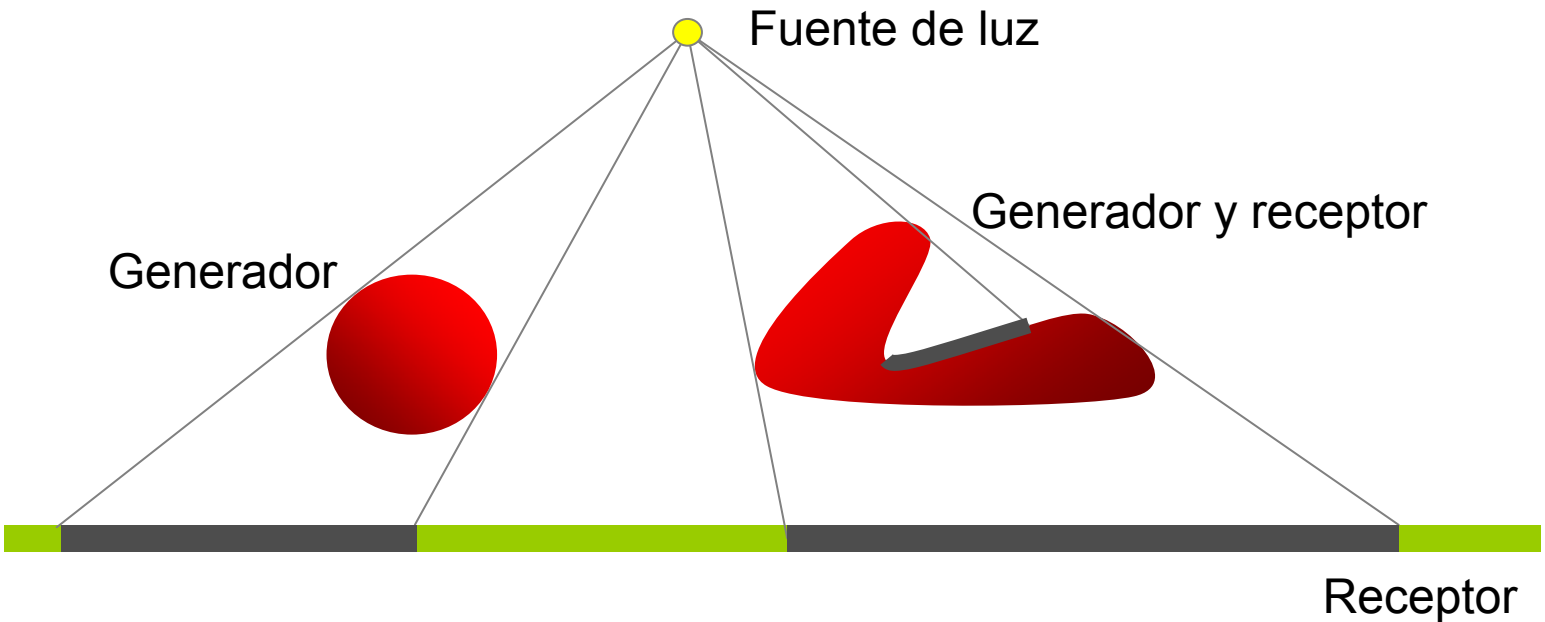
- Pueden ser marcadas o suaves





Introducción

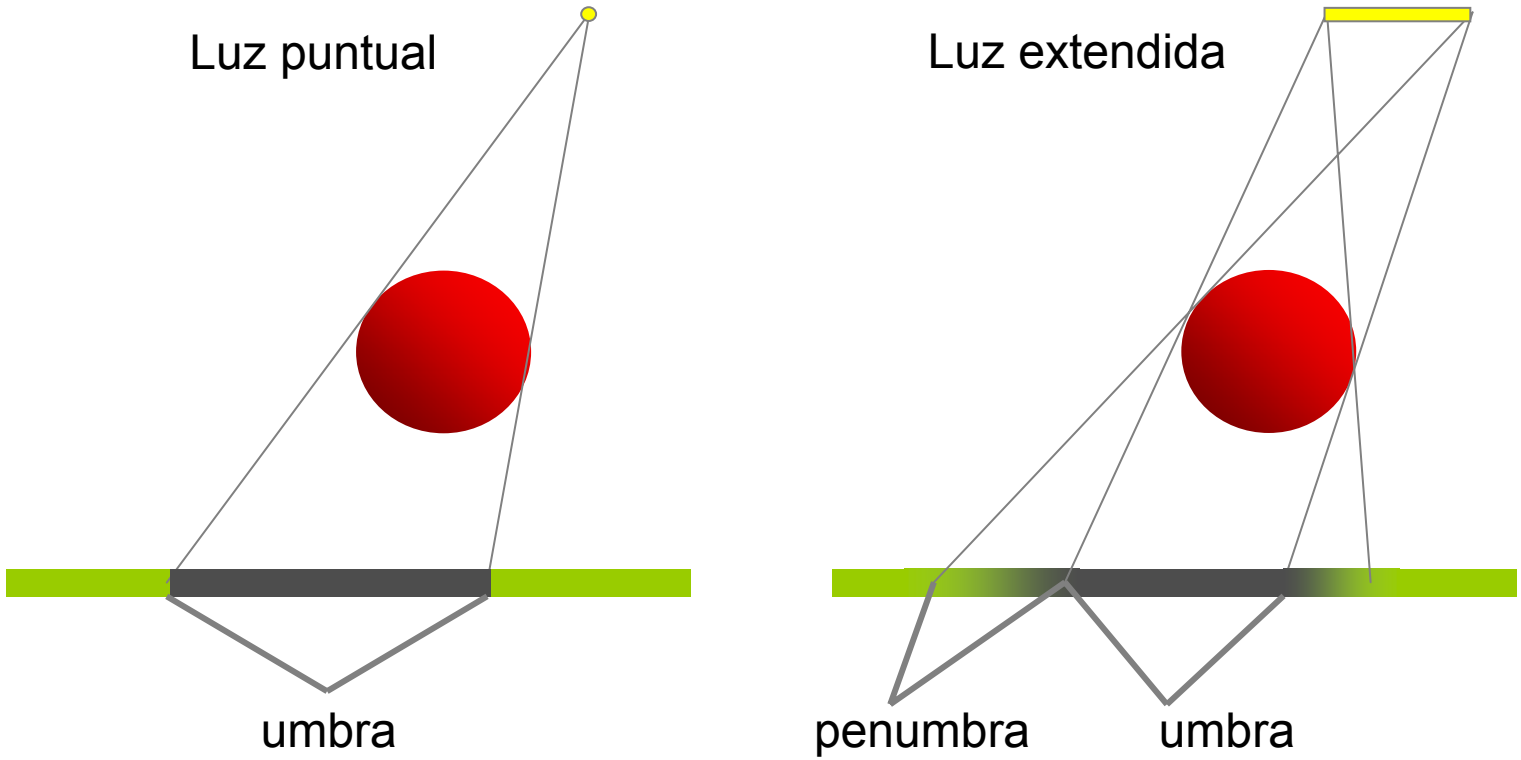
- Elementos involucrados



- ¿Cómo ver las sombras?
  - Como objetos independientes
  - Como volúmenes de oscuridad
  - Como zonas que no se ven desde las fuentes de luz

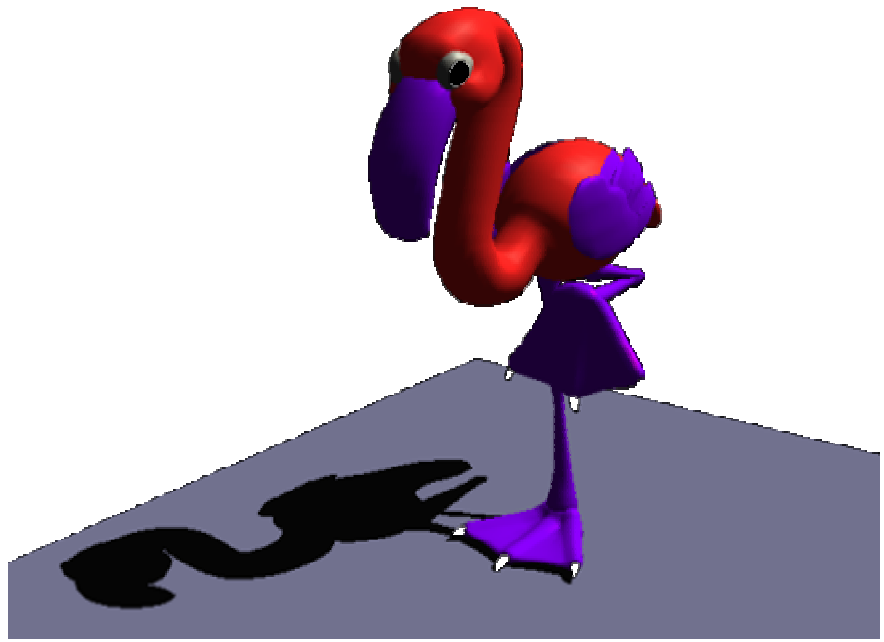
Introducción

- Tipos de fuentes de luz



Sombras de proyección

- Descripción
  - Proyectar los vértices del objeto generador de la sombra sobre el plano del suelo de acuerdo con la posición de la fuente de luz
  - El *convex hull* de los vértices proyectados forman el polígono de sombra
  - El polígono de sombra se dibuja del color de la sombra



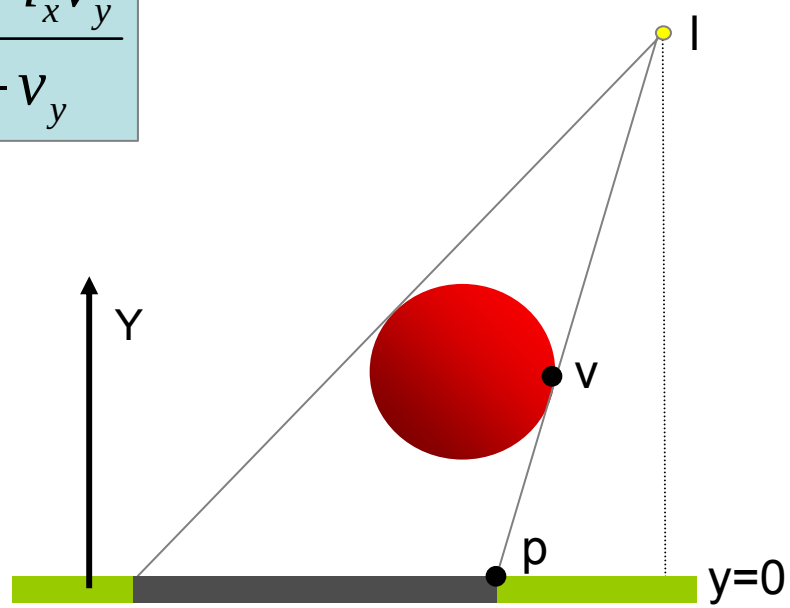
Sombras de proyección

- Cálculo de la matriz de proyección de la sombra (y=0)

$$p = Mv$$

$$\frac{p_x - l_x}{v_x - l_x} = \frac{l_y}{l_y - v_y} \Leftrightarrow p_x = \frac{l_y v_x - l_x v_y}{l_y - v_y}$$

$$M = \begin{pmatrix} l_y & -l_x & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & -l_z & l_y & 0 \\ 0 & -1 & 0 & l_y \end{pmatrix}$$

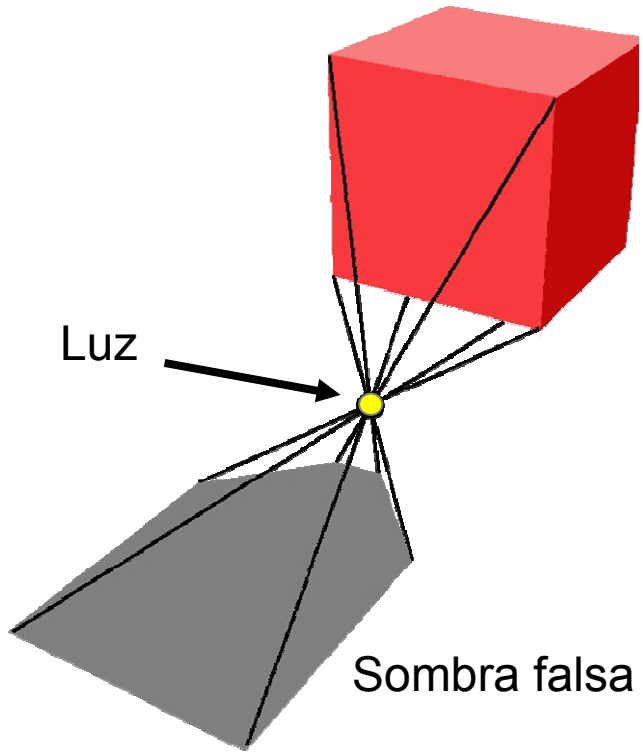


Sombras de proyección

- Funcionamiento
  - Aplicar la matriz de proyección de sombra al objeto que debe arrojar sombra
  - Dibujar con el color de la sombra
  - Evitar solapamientos (Z-buffer)
    - `glPolygonOffset()`
    - Dibujar el plano, deshabilitar Z-buffer, dibujar sombra
  - Utilizar el *stencil buffer* para evitar las sombras fuera del receptor
- Sombras semitransparentes
  - Si se trata de objetos convexos basta con dibujar los polígonos de la sombra semitransparentes
  - En otro caso hay que utilizar el *stencil buffer* contando los píxeles que se dibujan

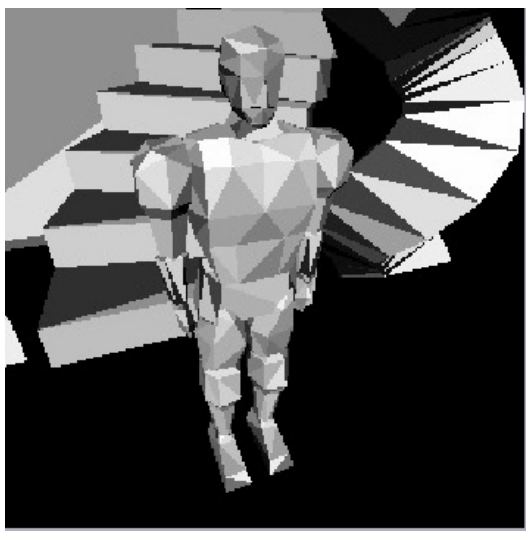
Sombras de proyección

- Problemas
  - Sombras falsas:
    - objeto detrás de la luz,
    - objeto debajo del plano
- Desventajas
  - Sólo sombras planas
  - Sólo sombras marcadas
  - Si las sombras son transparentes pueden aparecer zonas claras
  - Muchos elementos a tener en cuenta: transparencia, sombras falsas, sombras fuera del receptor, …
- Ventaja
  - Mejor que no tener sombras



Sombras con texturas

- Descripción
  - Dibujar los polígonos proyectados en una textura (*render-to-texture*)
  - Aplicar la textura sobre los receptores
    - Calculo de las coordenadas de textura en receptores
    - Uso de las texturas de proyección



Vista desde la luz

Textura de sombra

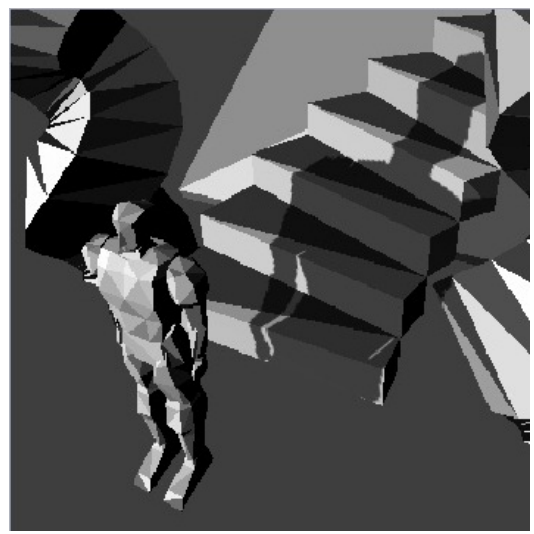
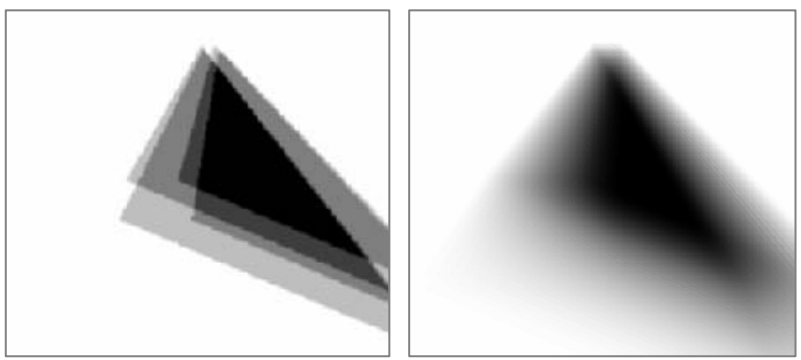


Imagen final

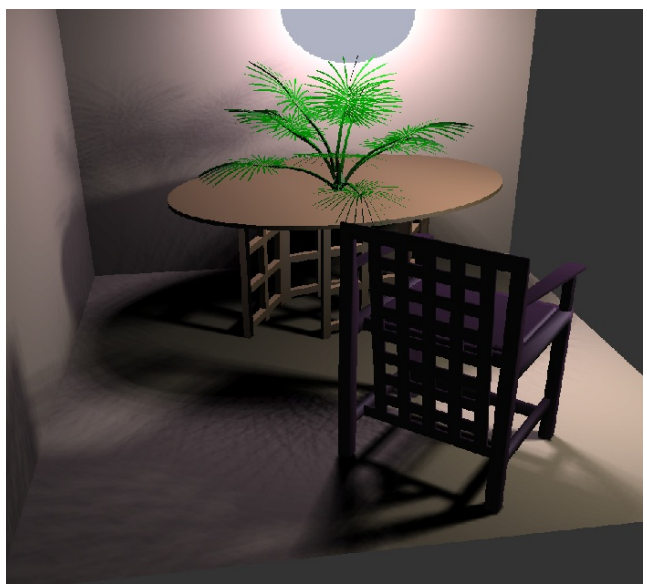
Sombras con texturas

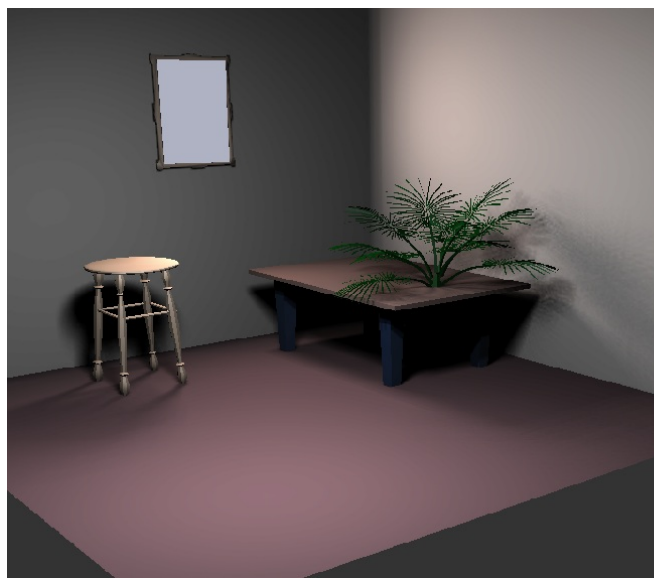
- Sombras suaves
  - Muestreo del área de la fuente en m x n puntos de la fuente (extendida)
  - Dibujar la sombra acumulando muestreos
  - Suavizado en la textura
  - Preproceso: se necesitan muchos muestreos (64-256)



Sombras con texturas

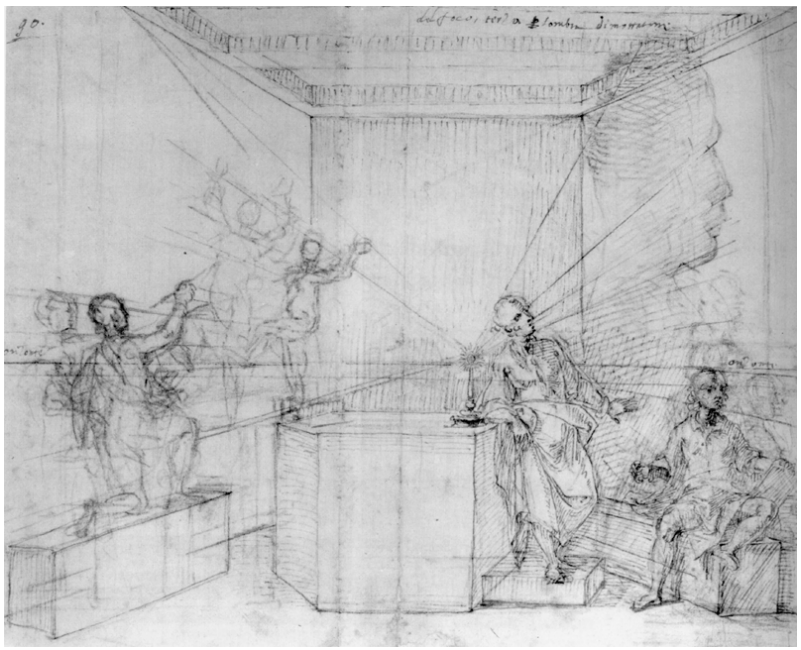
- Ventaja
  - La textura puede proyectarse sobre varios receptores
  - No es necesario regenerar la textura si la escena es estática
  - Las sombras pueden proyectarse sobre receptores curvos
- Desventajas
  - El diseñador debe identificar que objetos reciben y arrojan sombras
  - La superposición de sombras no es muy convincente





Volúmenes de sombra

- Los volúmenes de sombra definen el volumen del espacio en sombra

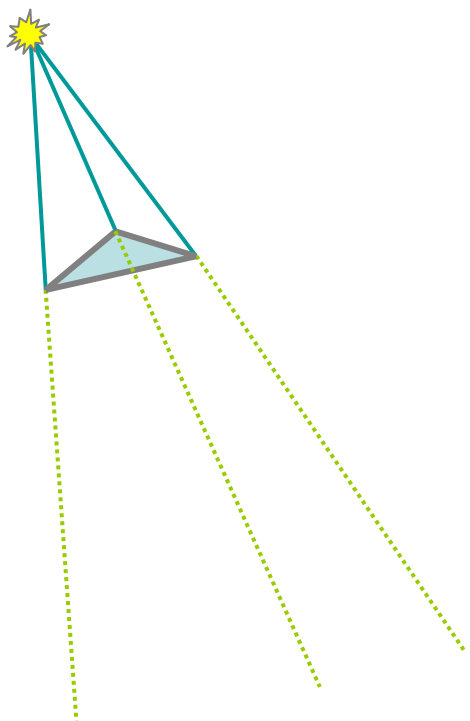


Leonardo Da Vinci

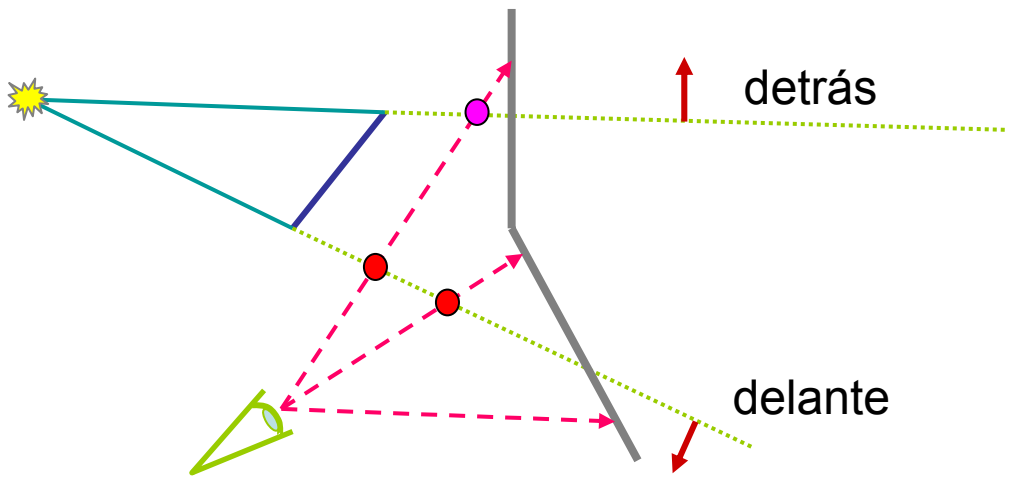
Motor de juegos

Volúmenes de sombra

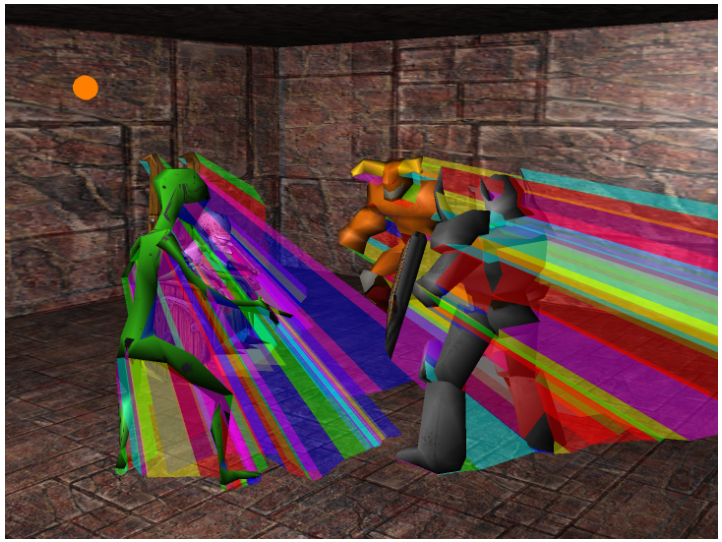
- Conceptos básico
  - Crear volúmenes de sombra para cada polígono iluminado
  - El volumen de cada triángulo se define por 3 cuadriláteros



- Funcionamiento
  - Contar el número de puntos intersecciones entre la vista y el punto a sombrear
  - Si se han atravesado más polígonos visibles que traseros, entonces el punto esta en sombra



Volúmenes de sombra

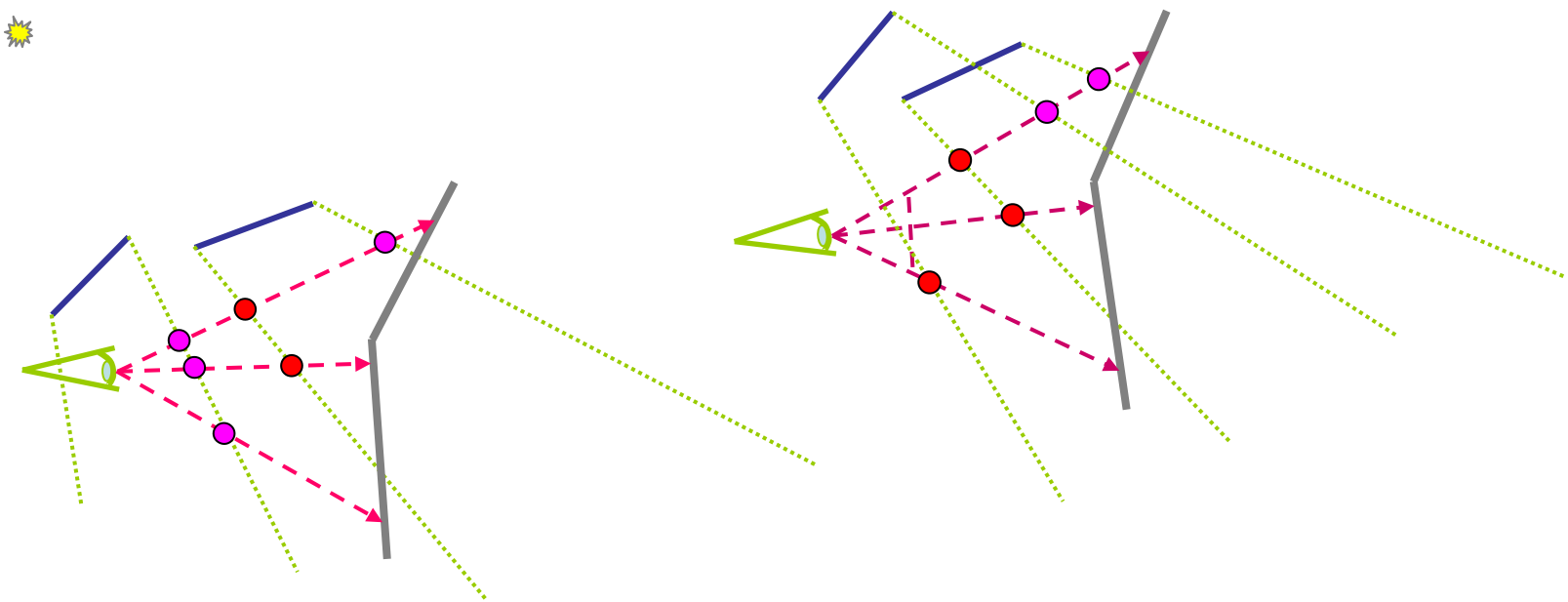


1er paso: dibujar escena únicamente con luz ambiente
Deshabilitar actualización del Z-buffer y no dibujar en el buffer de color (sólo comparar con Z-buffer, sólo dibujar en el stencil buffer)
2º paso: dibujar los polígonos visibles del volumen de sombreado, incrementando un contador cada vez que dibujamos un polígono
3er paso: dibujar polígonos traseros del volumen de sombra de igual forma decrementando el stencil buffer
4º paso: dibujar con iluminación difusa y especular donde el stencil buffer vale 0

Algoritmo volúmenes de sobra

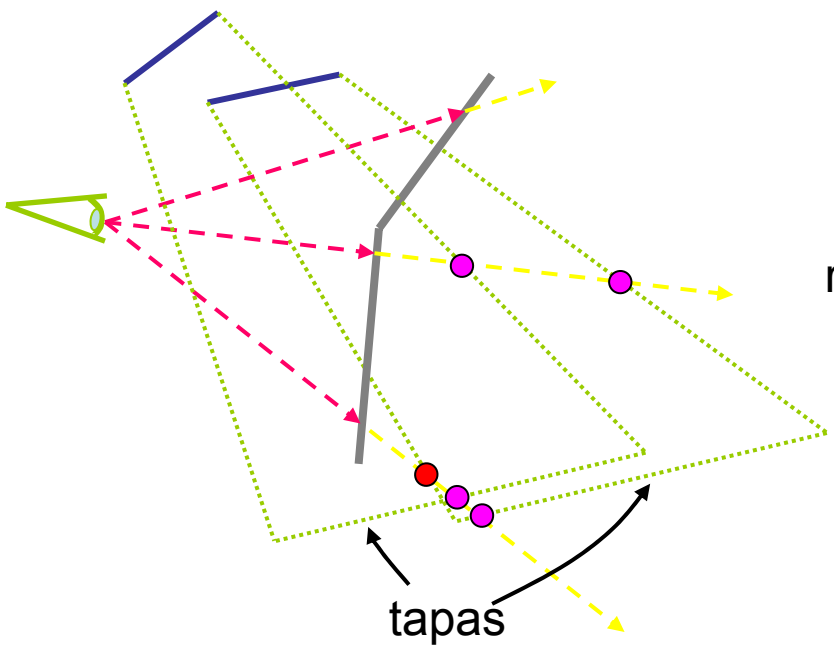
Volúmenes de sombra

- Problemas
  - Punto de vista en el interior de uno o más volúmenes de sombra (fallo en el contador).
  - **Solución**: inicializar stencil con el número de volúmenes en los que esta la vista
  - La pirámide de visualización intersecta con el volumen de sombra (fallo en el contador)
  - **Solución**: detectar la intersección del plano frontal

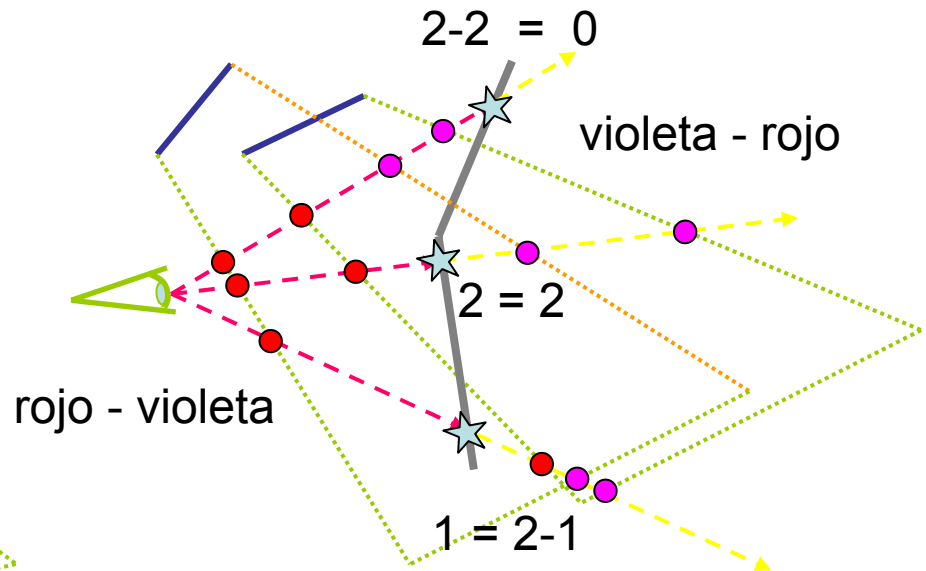


Volúmenes de sombra

- Dibujar sólo el volumen con z>= que el Z-buffer evita colocar tapas al volumen



- ¿Por qué funciona?: (f-b) delante = (b-f) detrás

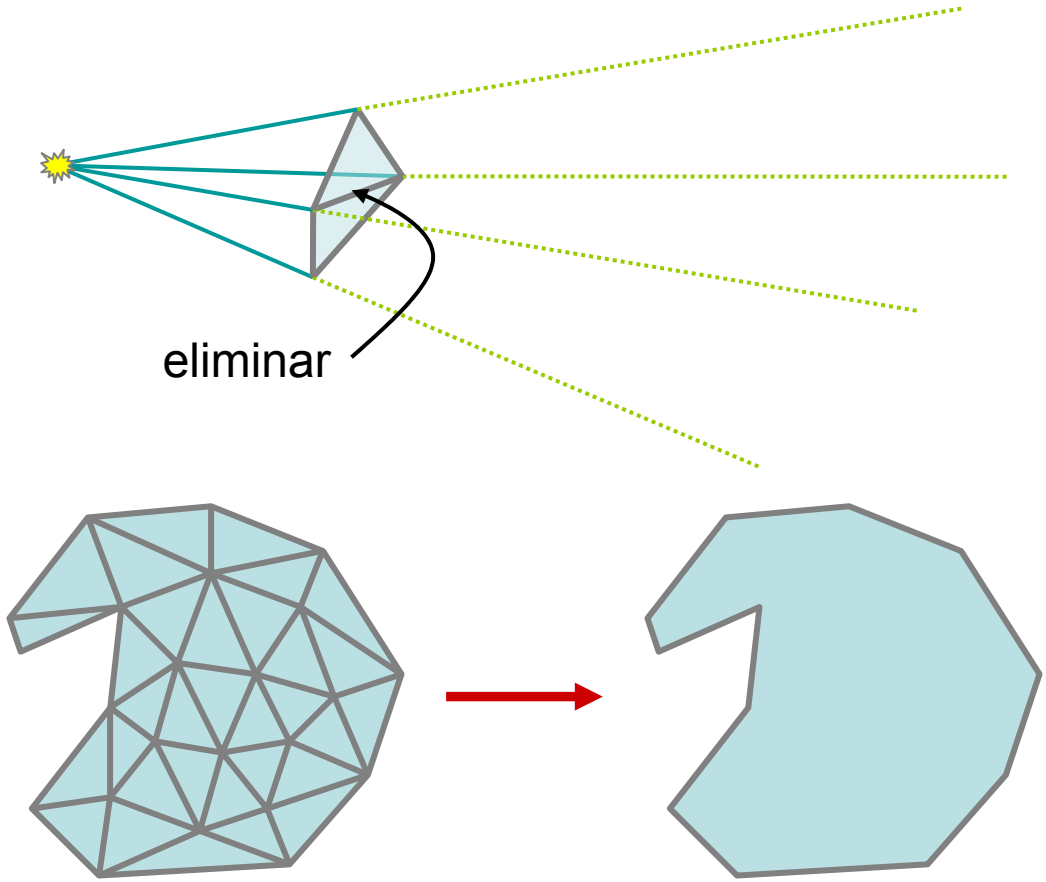


Volúmenes de sombra

- Ejemplo: Neverwinter Nights
  - En lugar de redibujar la escena en el último paso, dibujarla con todo detalle en el primer paso. Después los polígonos de la sombra con transparencia utilizando el *stencil buffer*
  - La iluminación difusa y especular es incorrecta, pero aceptable
  - No se produce auto sombreado (se pueden usar volúmenes más simples)
  - Los volúmenes de sombra para luces/objetos estáticos se almacenan en *display lists*

Volúmenes de sombra

- Unir los volúmenes
  - Encontrar la silueta de los polígonos visibles simplifica el volumen de sombra

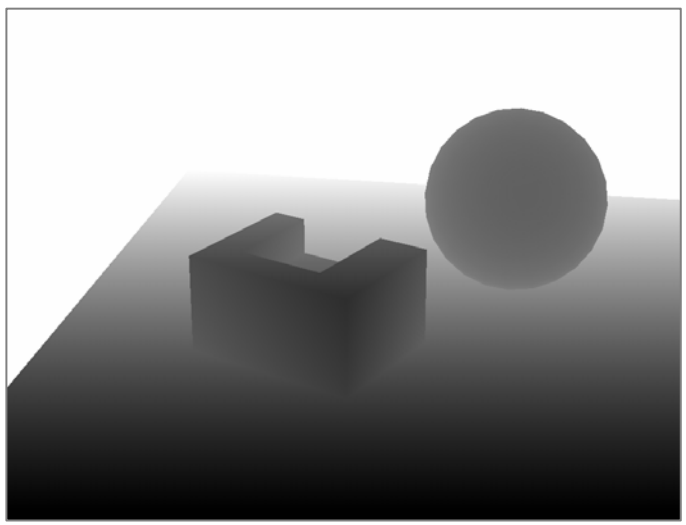


Mapas de sombras

- Descripción
  - El cálculo de la sombra se integra con el Z-buffer
  - Es necesario un nuevo buffer llamado Z-buffer de sombras
  - Las sombras se generan en dos pasos
    - Cálculo de la información de profundidad (Z-buffer de sombras)
    - Visualización con el algoritmo del Z-buffer modificado

Mapas de sombras

- Cálculo del Z-buffer de sombras
  - La escena se dibuja desde el punto de vista de la luz
  - No se almacena información de color, sino información de profundidad para crear el Z-buffer de sombras. La distancia desde la fuente de luz a las superficies más próximas



Mapa de sombras

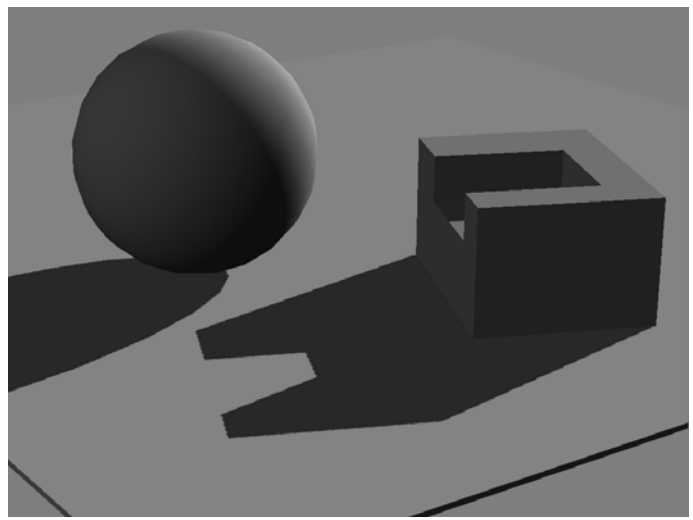
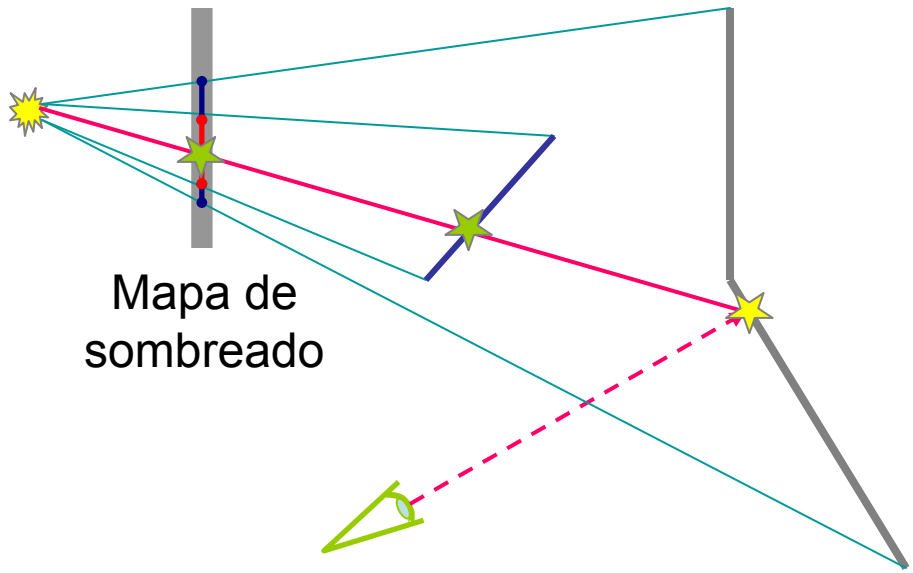


Imagen final

Mapas de sombras

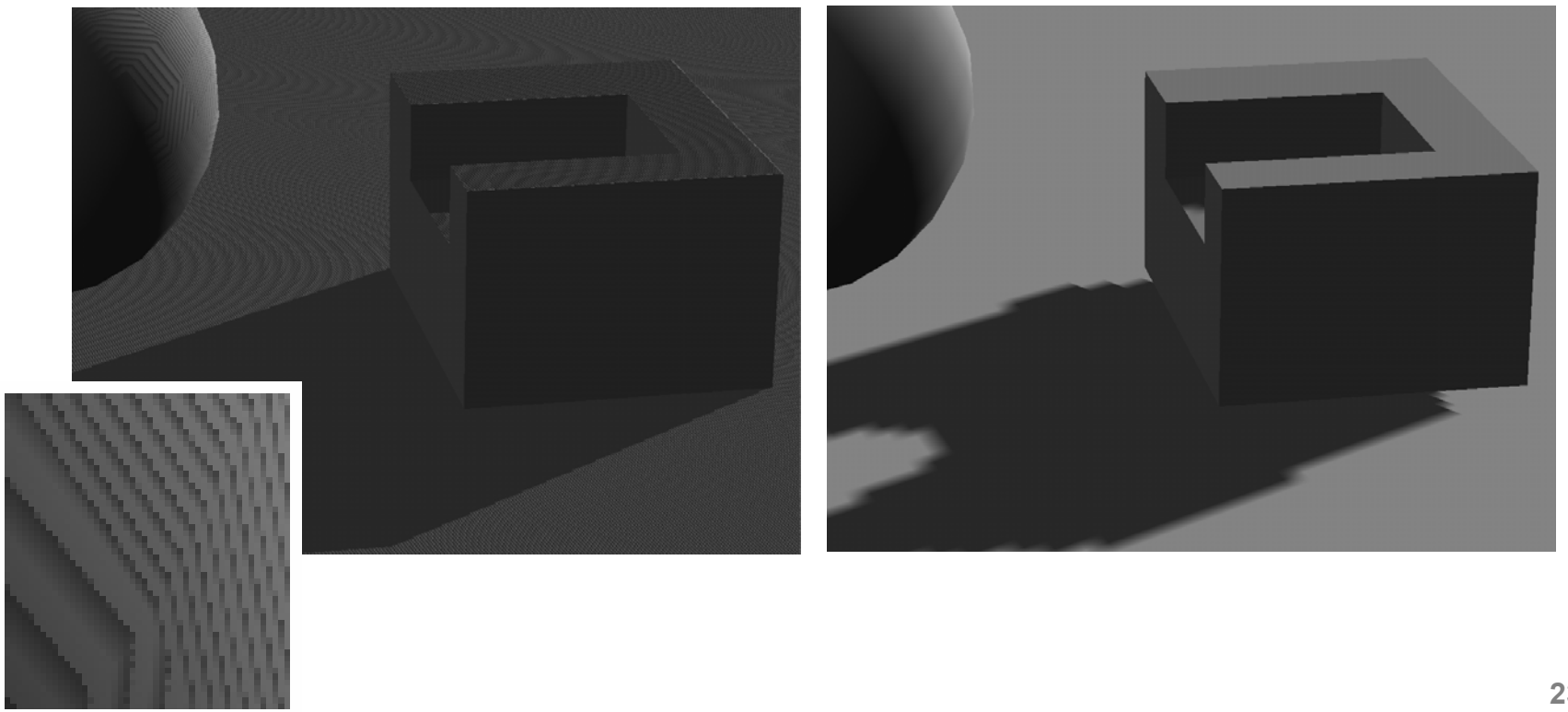
- Algoritmo de dibujado
  - Dibujar la escena desde el punto de vista de la manera habitual
  - Si el punto es visible, transformarlo al sistema de coordenadas de la luz
  - La (x,y) del punto transformado se utiliza para indexar en el Z-buffer de sombras
  - Si la z del punto transformado es mayor que el valor almacenado en el Z-buffer entonces el punto esta en sombra y debe dibujarse utilizando la intensidad de la sombra



Para cada píxel comparar la distancia a la luz con la profundidad almacenada en el mapa de sombras

Mapas de sombras

- Problemas
  - Auto-sombreado
  - Excesivo suavizado (filtrado de la textura)



Comparación de técnicas

- Sombras de proyección
  - ↑: Simple, rápido, implementadas en cualquier hardware
  - ↓: Sólo se arrojan sombras en un plano
  - ↕: La sombras transparentes necesitan usar el *stencil buffer*
- Sombras con texturas
  - ↑: Transparentes, las texturas pueden reutilizarse, sobras sobre cualquier superficie
  - ↓: Los objetos sólo pueden arrojar o recibir sombras
  - ↕: Necesita *render-to- texture*
- Volúmenes de sombra
  - ↑: Cualquier objeto puede arrojar sombras, permiten auto-sombreado
  - ↓: 3 o 4 fases, hay que generar y usar muchos polígonos
  - ↕: Problemas cuando el *fustrum* intersecta los volúmenes
- Mapas de sombras
  - ↑: Cualquier tipo de escena, coste constante independiente de la complejidad
  - ↓: Sólo posible en algún hardware, luz distante
  - ↕: Problemas de precisión