



técnicas avanzadas de gráficos  
ingeniería multimedia

# Seminario 9

## *Sombras (shadowing)*

# Sombras, sombreado...

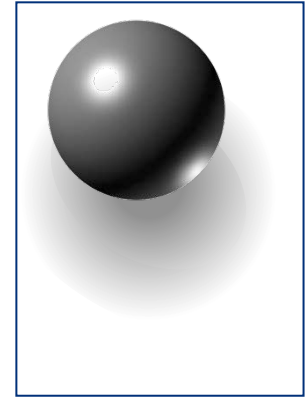
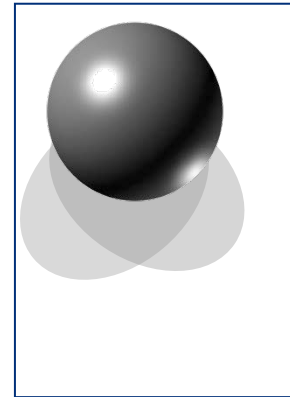
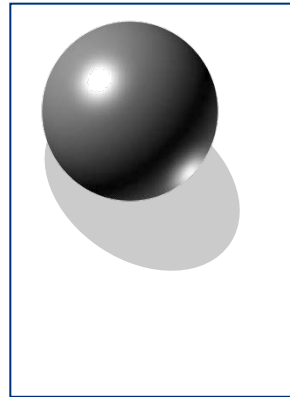
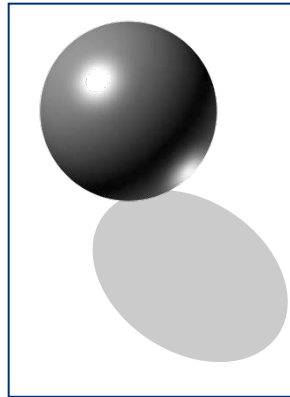
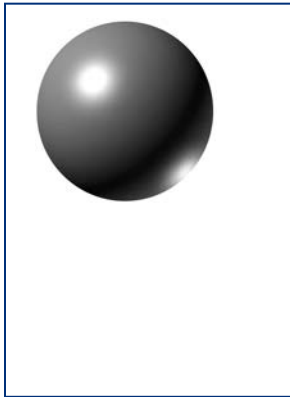


¿Cuál es la diferencia entre *shade* y *shadow*?

¿Qué son zonas de sombra y zonas de penumbra? ¿A qué se deben?

# ¿Por qué son necesarias las sombras?

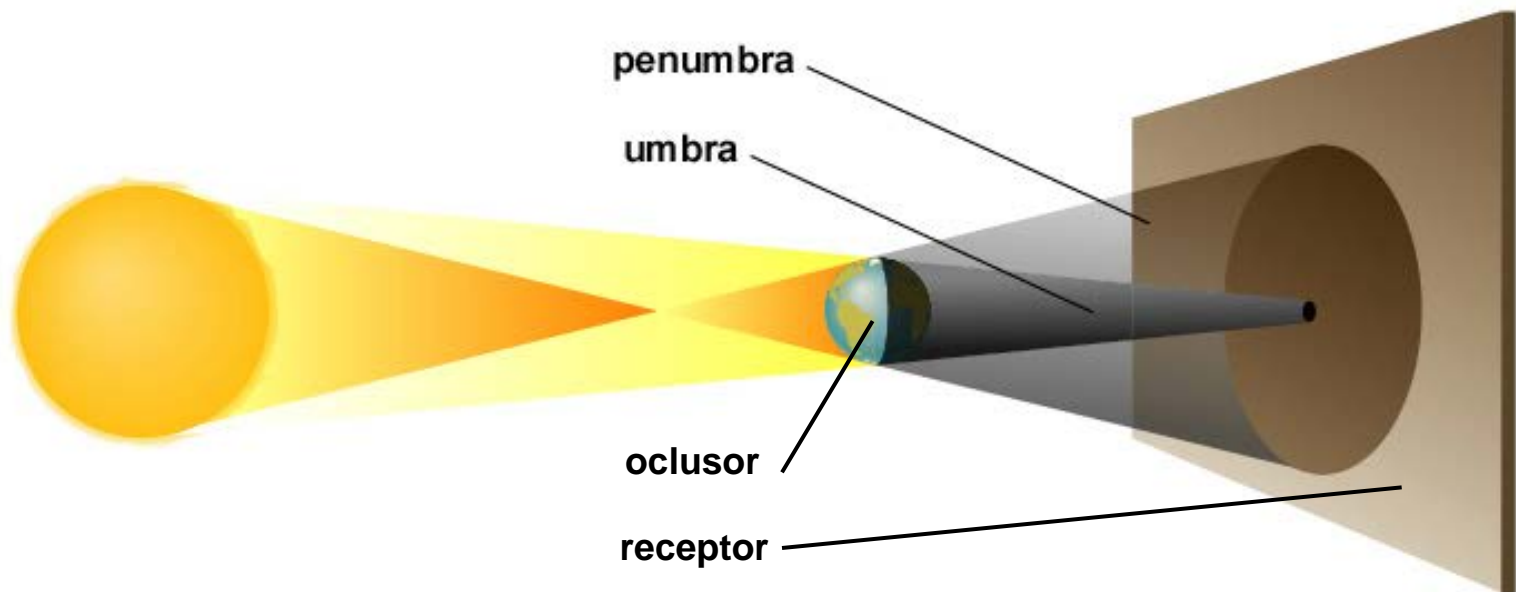
- Ayudan a establecer la relación espacial entre los objetos
- Dan información sobre el tipo de iluminación



# Conceptos sobre sombras

- Problemas principales de la generación de sombras:
  - ¿Qué forma tienen las sombras?
  - ¿Cuál es la intensidad de la luz en la sombra?
- Conceptos
  - Ocluser: elemento que produce la sombra
  - Receptor: elemento sobre el que se proyecta la sombra
  - Sombra: zona del receptor que no es iluminada directamente por la luz. Dentro de la sombra distinguimos dos zonas:
    - Umбра: zona a la que no llega ninguna luz, ni directa, ni indirectamente
    - Penumbra: zona a la que no llega luz directa, pero sí indirecta

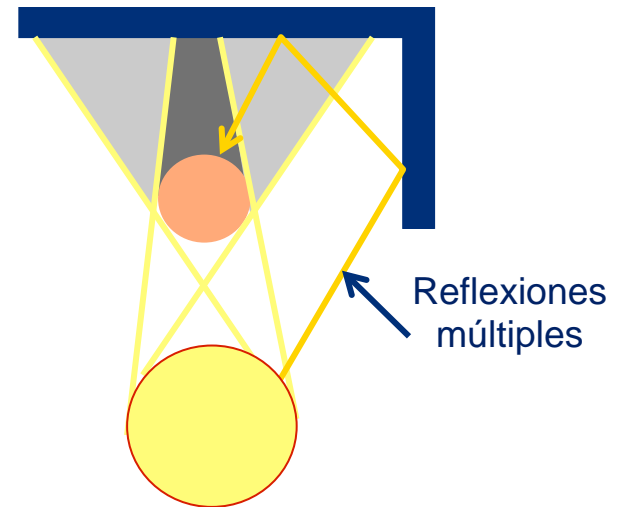
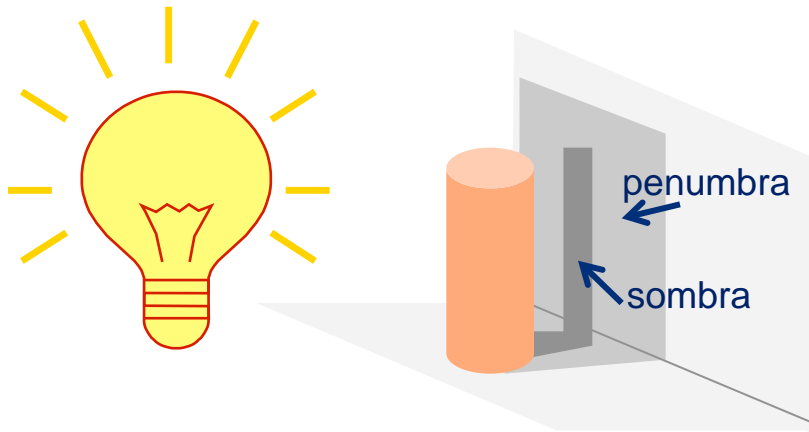
# Conceptos sobre sombras




Copyright © Addison Wesley

# Factores que afectan a las sombras

- Luces distribuidas:
  - Generan zonas de umbra y de penumbra
- Reflexiones múltiples
  - La luz puede llegar a la zona de sombra indirectamente



# Métodos de generación de sombras

- Métodos empíricos 
  - Tratan el problema geoméricamente
  - Adecuados para los métodos de reflexión local: Phong, Blinn, Cook&Torrance...
- Métodos de iluminación global
  - Los propios algoritmos de iluminación global incluyen la generación de sombras de forma implícita
  - Es el caso de Trazado de Rayos y de Radiosidad



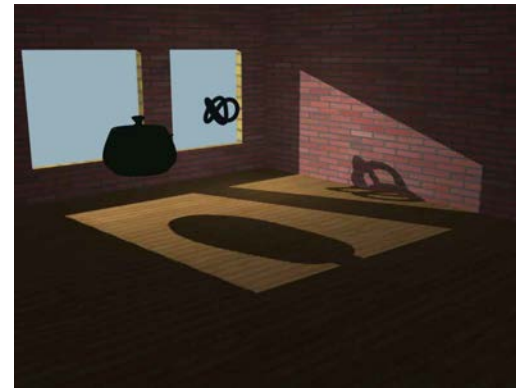
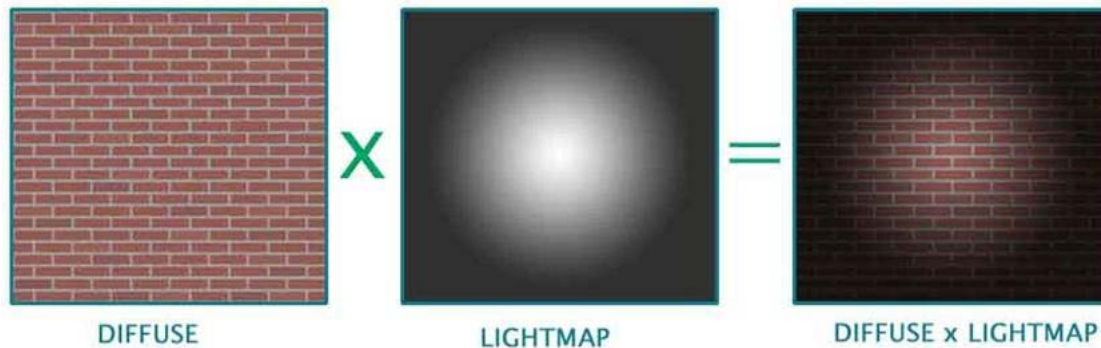
# Métodos empíricos de generación de sombras

- Mapas de luz (*Light Maps*)
- Sombras planas proyectadas (*Projected Shadows*)
- Volúmenes de sombra (*Volume Shadows*)
- Z-buffer de sombra o mapas de sombra (*Shadow Maps*)



# Mapas de luz

- Consiste en precalcular luces y sombras y sumarlas a las texturas



# Mapas de luz

- **Ventajas**
  - Pueden precalcularse de forma muy precisa, con resultados muy realistas, utilizando técnicas muy sofisticadas: radiosity, raytracing...
  - El uso de mapas es muy eficiente una vez precalculados
  - Pueden utilizarse mapas para muchos aspectos: mapas de luz, de normales...
- **Inconvenientes**
  - No se pueden calcular en tiempo real, por lo que sólo son aptos para luces y objetos estáticos
  - Los objetos dinámicos deben utilizar otras técnicas



# Mapas de luz

Deep Creator - multistage textures and light map support

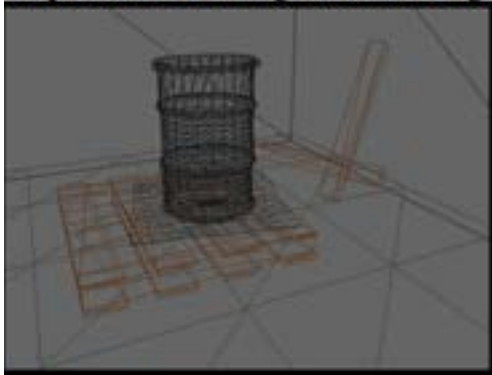


Figure 1. Wireframe



Figure 2. Default lighting no textures



Figure 3. Simple textures applied



Figure 4. Light Map texture only



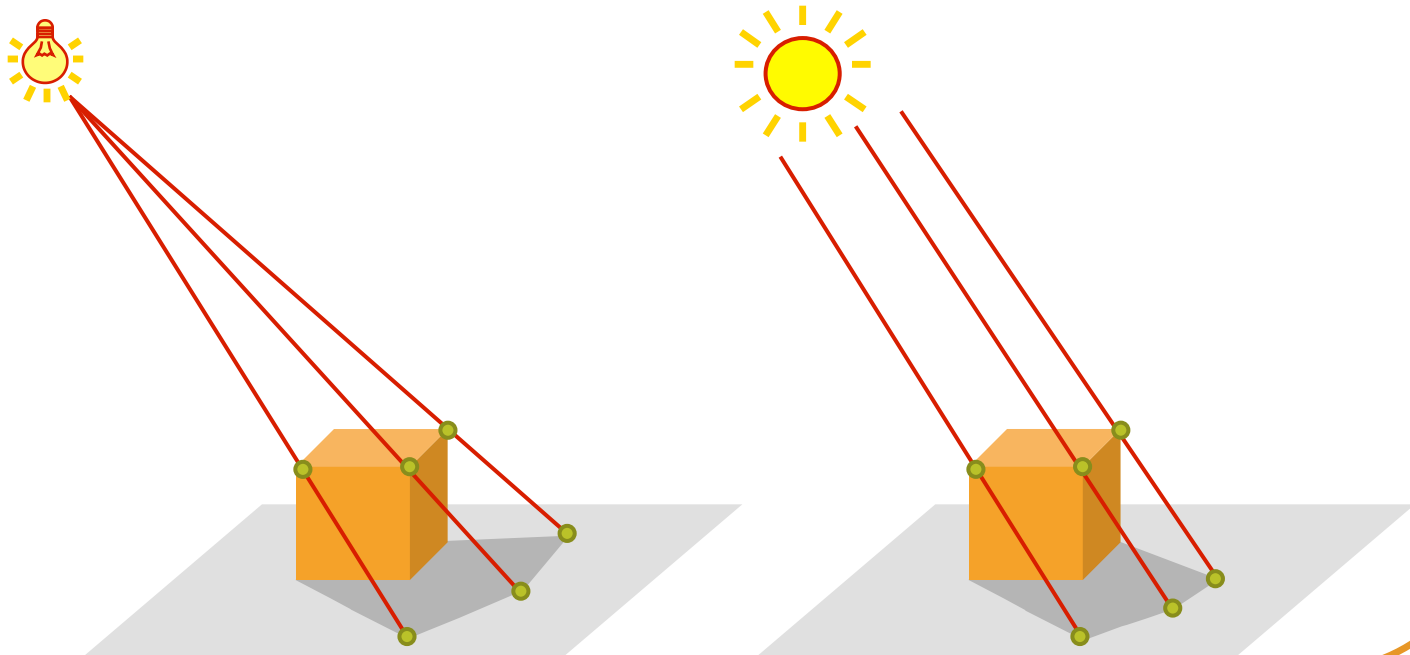
Figure 5. Light map and additional "dirt map" and detail texture stages applied



Figure 6. Final scene – with all textures applied and emissive set to half.

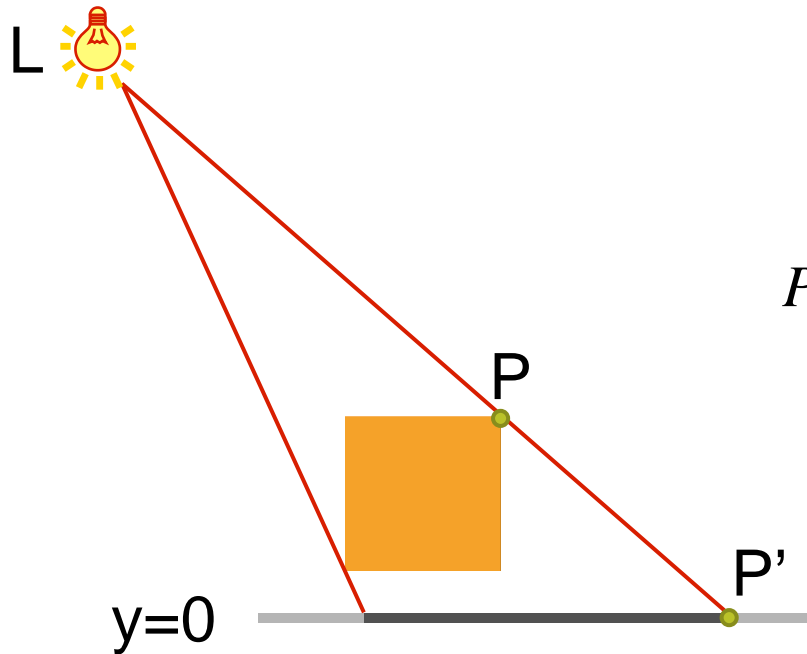
# Sombras planas proyectadas

- Simplificación adecuada para calcular sombras proyectadas sobre un plano
- La forma de la sombra se calcula proyectando el *occludor* sobre el *receptor* de la sombra utilizando una matriz de proyección



# Sombras planas proyectadas

- Caso general: utilizamos una matriz de proyección, que puede ser perspectiva (ver ejemplo) o paralela



$$P' = \begin{pmatrix} L_y & -L_z & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & -L_z & L_y & 0 \\ 0 & -1 & 0 & L_y \end{pmatrix} \cdot P$$

# Sombras planas proyectadas

- **Algoritmo en OpenGL**

`proyecluz = matriz proyección de la luz`

`Dibujar la escena sin sombras`

`Desactivar la iluminación`

`Apilar la matriz modelview`

`modelview = modelview * proyecluz`

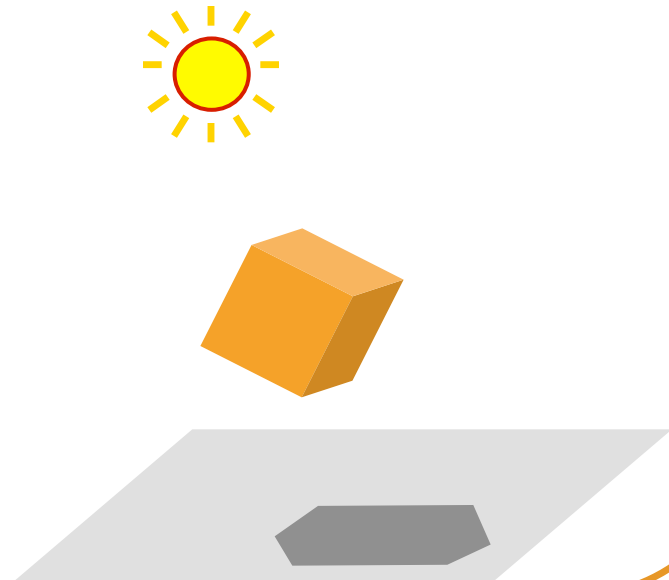
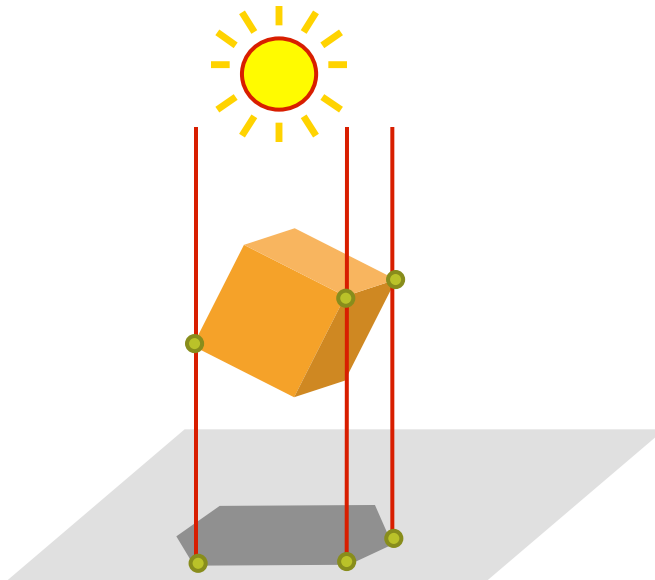
`Seleccionar color de las sombras`

`Dibujar sólo los objetos que proyecten sombras`

`Desapilar modelview`

# Sombras planas proyectadas

- Simplificación para iluminación del sol al mediodía
  - La matriz es paralela y el plano es el suelo ( $y=0$ ): basta con eliminar la coordenada  $y$  de todos los vértices del ocluser
  - Para que la sombra no está totalmente debajo del objeto, puede desplazarse un poco



# Sombras planas proyectadas

- A partir de los puntos proyectados se genera un nuevo objeto que se dibuja en un color oscuro con una cierta transparencia (alfa)
- El polígono se coloca a una cierta distancia del plano del suelo:
  - Si está muy lejos, se verá separado
  - Si está muy cerca, puede haber problemas de precisión
  - Solución: utilizar un *offset* (`glPolygonOffset`)



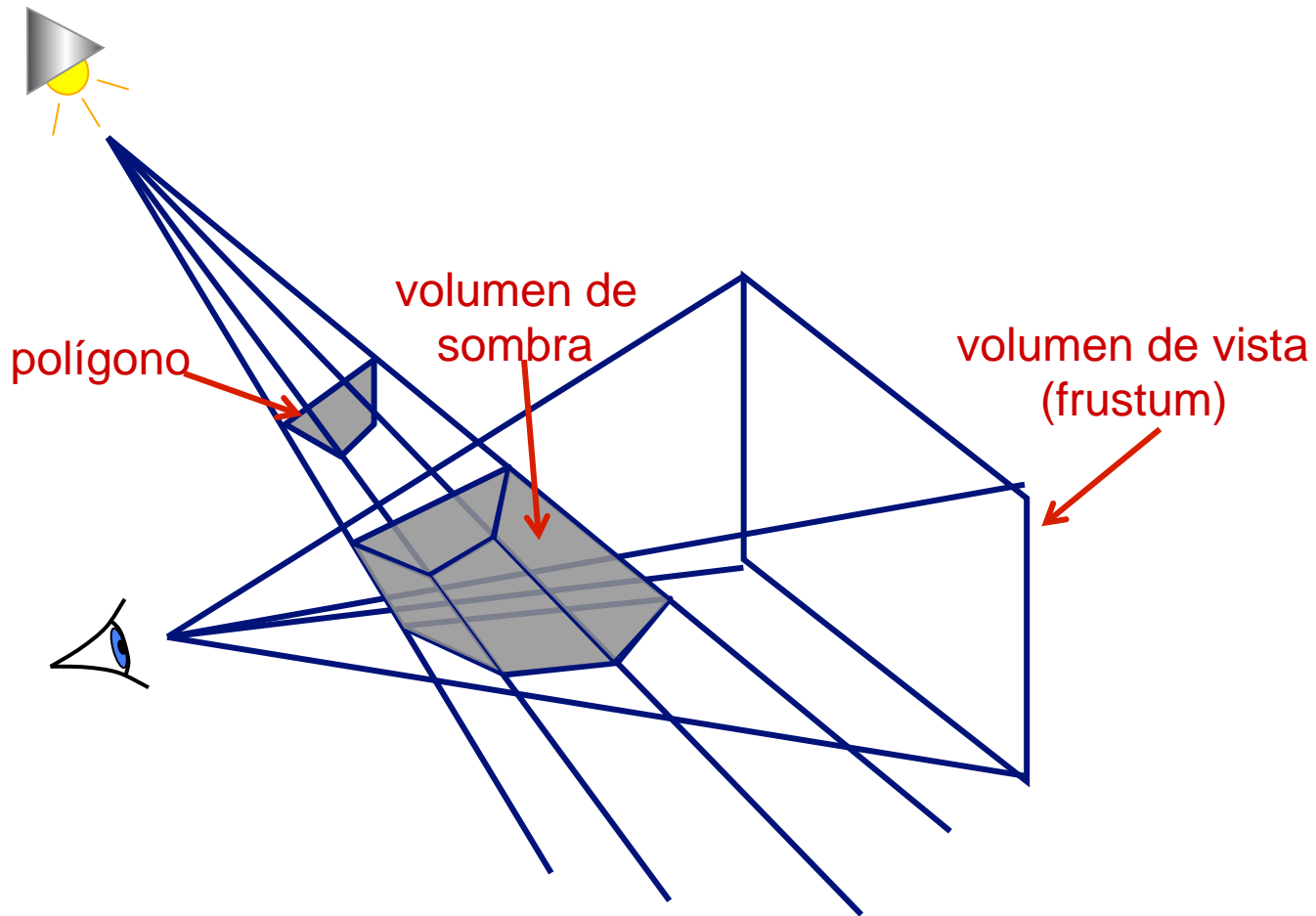
# Sombras planas proyectadas

- Limitaciones:
  - Es sencillo sobre planos, pero complejo extenderlo a figuras genéricas
  - Las sombras tienen los bordes muy marcados, son poco realistas
  - La intensidad de la luz dentro del perfil de la sombra se calcula de forma aproximada → no siempre es adecuada
  - Sólo aceptable cuando la luz es puntual y de alta intensidad → sol directo
- Este método se puede mejorar para generar sombras más suaves, mediante varias pasadas de dibujado, alterando levemente la posición de la luz

# Volúmenes de sombra

- Para objetos poligonales. No se puede extender fácilmente.
- Idea básica: dada una luz, generar un volumen de sombra proyectada por cada objeto, de manera que los objetos interiores a ese volumen se encuentran dentro de la sombra → un volumen por objeto y por luz
- Cada volumen de sombra es una pirámide truncada semi-infinita. Al intersectar con el volumen de vista (frustum) forma un objeto poligonal cerrado. **Problema 1: Formar los volúmenes**
- Los polígonos que forman el volumen de sombra se incorporan a la lista de polígonos de la escena como un objeto más.
- Estos volúmenes no se visualizan sino que se utilizan para determinar si cada punto de la escena es interior o no a ese volumen de sombra. **Problema 2: Test de interioridad**

# Volúmenes de sombra



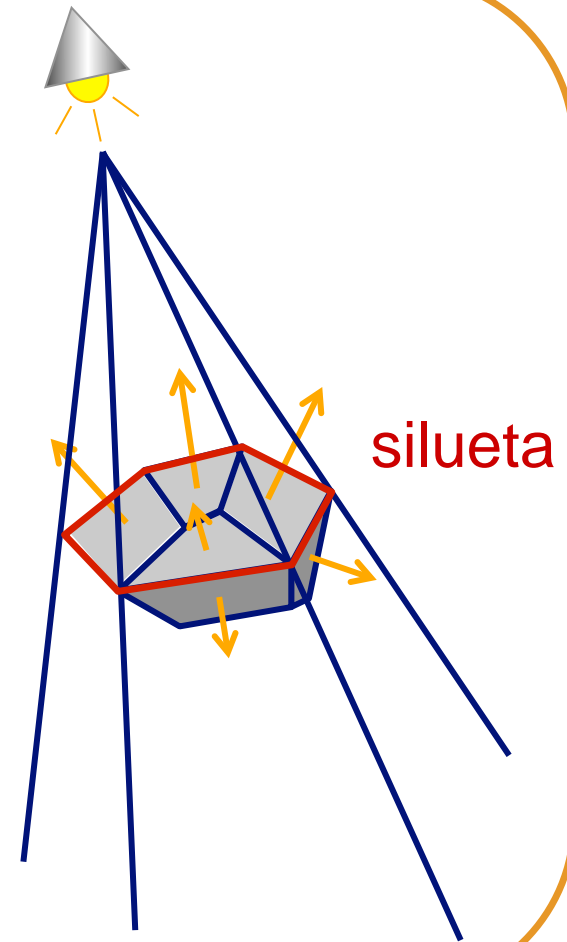


# Volúmenes de sombra: Formar los volúmenes

- Problema 1: Formar los volúmenes:
  - Se puede generar un volumen por cada polígono de la escena.
  - Para disminuir el número de volúmenes:
    - Se generan volúmenes asociados sólo a aquellos polígonos que son visibles desde la luz (su normal apunta hacia la luz)
    - Se unen en un único volumen los asociados a un mismo objeto y que sean adyacentes: formados por los bordes de silueta del objeto

# Volúmenes de sombra: Formar los volúmenes

- Generación de los volúmenes de siluetas:
  - Partir de un polígono orientado hacia la luz
  - Si el polígono no tiene otros polígonos adyacentes, él solo forma una silueta
  - Si el polígono tiene otros polígonos adyacentes, añadirlos a la silueta si también están orientados hacia la luz
  - Formar el volumen de sombra extruyendo las aristas de la silueta una cantidad muy grande





# Volúmenes de sombra: Formar los volúmenes

- Algoritmo para formar la silueta

Para cada objeto y cada luz

Para cada polígono del objeto

Sea  $L$  el vector de luz y  $N$  la normal al polígono

Si  $L \cdot N \geq 0$  (el polígono es visible)

Para cada arista del polígono

Si la arista ya está en la silueta

Eliminarla de la silueta (si está dos veces es porque es compartida por dos polígonos adyacentes iluminados)

Si no

Añadir la arista al contorno

Fin Si

Fin Para

Fin Si

Fin Para

Fin Para



# Volúmenes de sombra: Formar los volúmenes

- Algoritmo para formar el volumen

Extrusión = NUMERO\_MUY\_GRANDE

Para cada silueta de un objeto

Para cada arista de la silueta V1-V2

Quad\_Sombra.V1 = V1

Quad\_Sombra.V2 = V2

Quad\_Sombra.V3 =  $V1 + Extrusión * (V2 - L)$

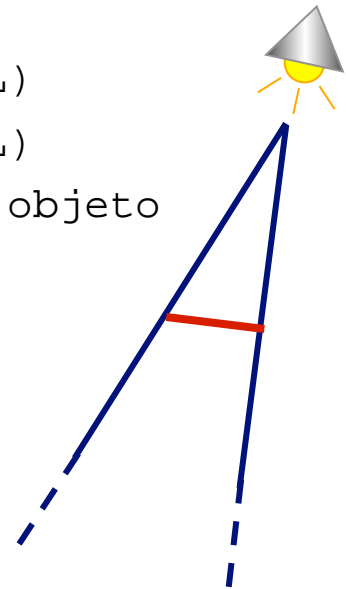
Quad\_Sombra.V4 =  $V2 + Extrusión * (V1 - L)$

Añadir Quad\_Sombra al volumen de ese objeto

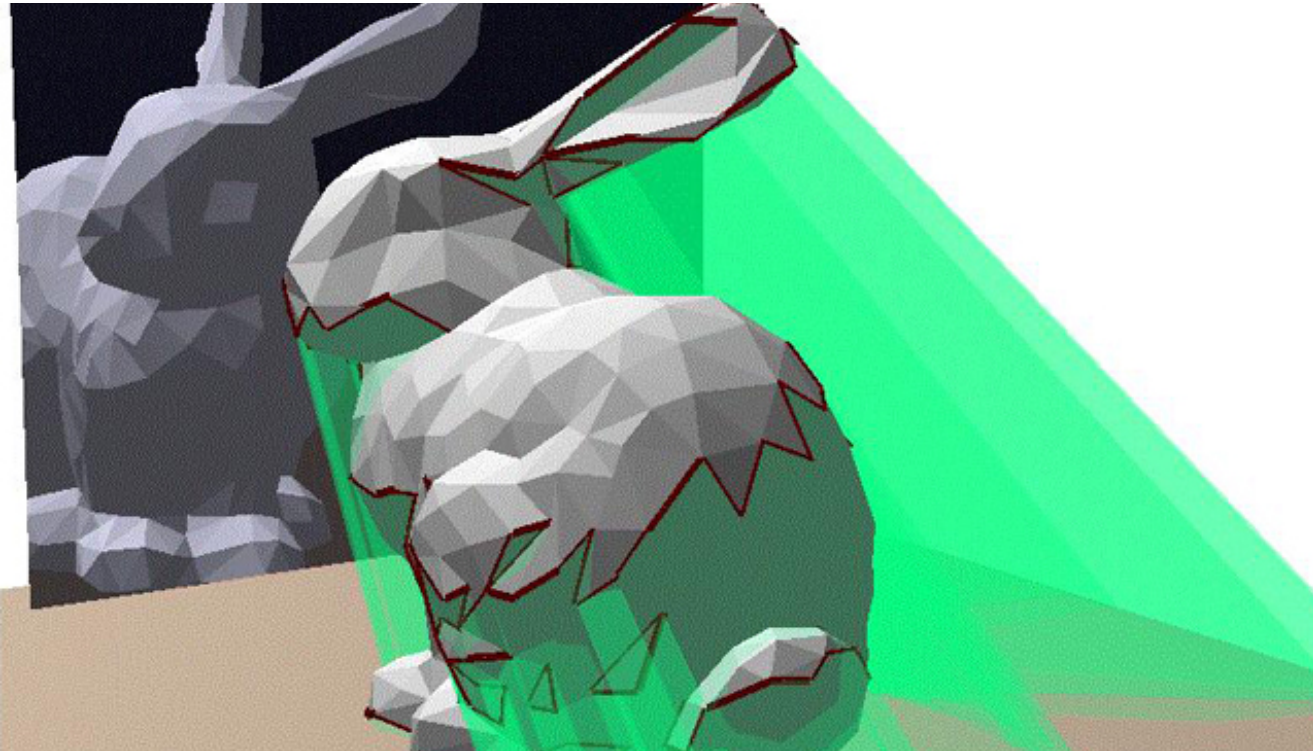
Fin Para

Cerrar el volumen por arriba y abajo

Fin Para



# Volúmenes de sombra: Formar los volúmenes





# Volúmenes de sombra:

## Test de interioridad

- Una vez calculados los volúmenes, debemos dibujar la escena, y saber si cada fragmento es interior o no a los volúmenes de sombra
- Test de interioridad básico: trazar un rayo entre el punto de vista y el fragmento y contar cuántas veces se entra y se sale de cada volumen de sombra

# Volúmenes de sombra: Test de interioridad

- Algoritmo para el test de interioridad básico

Para cada fragmento a dibujar

    contador = 0

    rayo = rayo desde posición del fragmento al observador

    Para cada intersección rayo-cara de un volumen

        Si cara orientada hacia observador

            contador++

        Si no

            contador--

        Fin Si

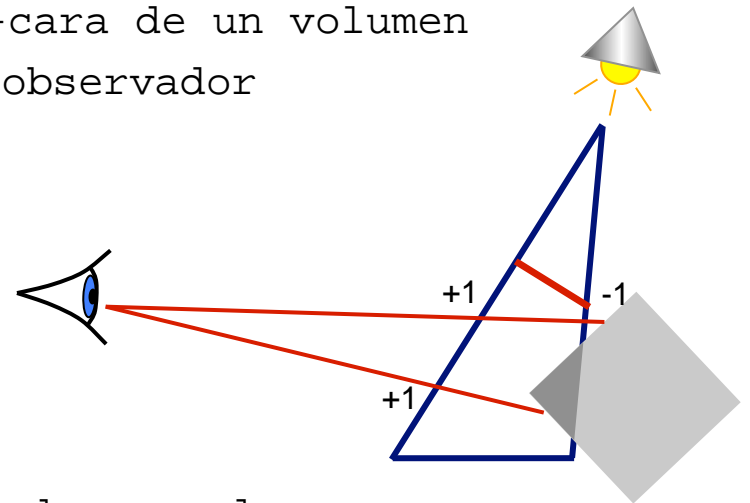
    Fin Para

    Si contador > 0

        El fragmento está dentro de un volumen

    Fin Si

Fin Para



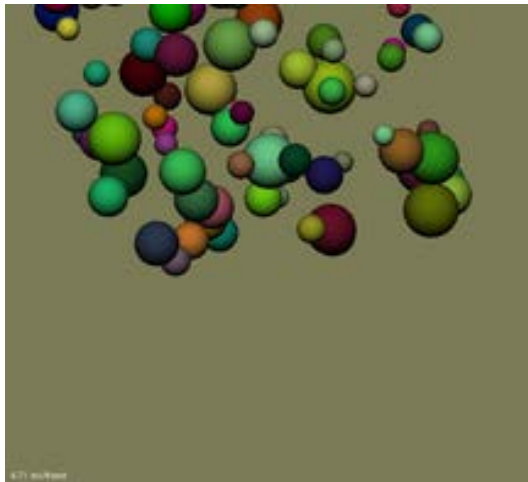
# Volúmenes de sombra:

## Test de interioridad

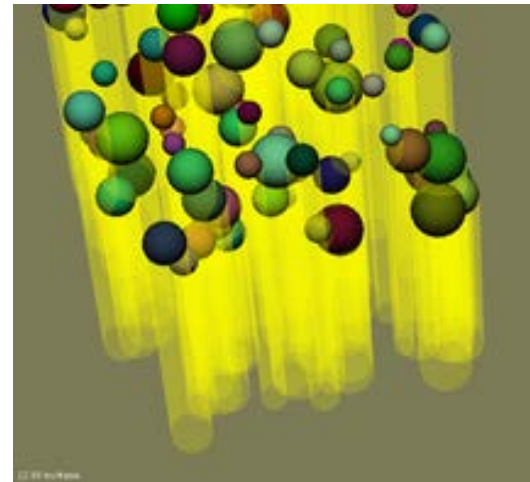
- Test de interioridad utilizando el stencil buffer
  - Limpiar el stencil buffer y el Z-buffer. Activar el Z-buffer y desactivar el stencil buffer
  - Dibujar la escena sólo con la iluminación ambiental (y emisiva, si existe) → se actualiza color buffer y Z-buffer
  - Desactivar el color buffer y el Z-buffer
  - Dibujar las caras frontales de los VS, incrementando el stencil buffer para los polígonos dibujados (pasan el test Z)
  - Dibujar las caras posteriores de los VS, decrementando el stencil buffer para los polígonos dibujados (pasan el test Z)
  - Limpiar el Z-buffer y activar todos los buffers
  - Dibujar la escena, añadiendo las componentes especular y difusa en los fragmentos con el stencil buffer = 0

# Volúmenes de sombra

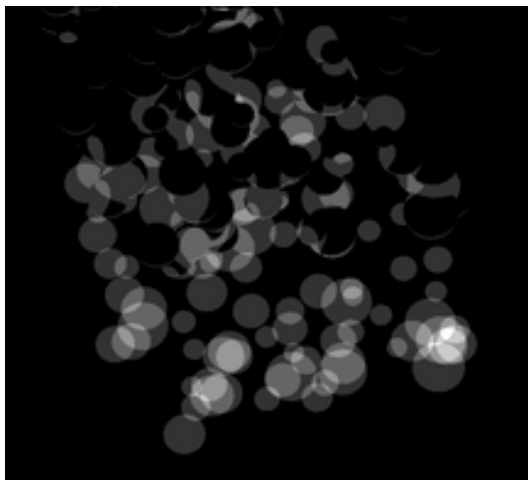
sin  
sombras



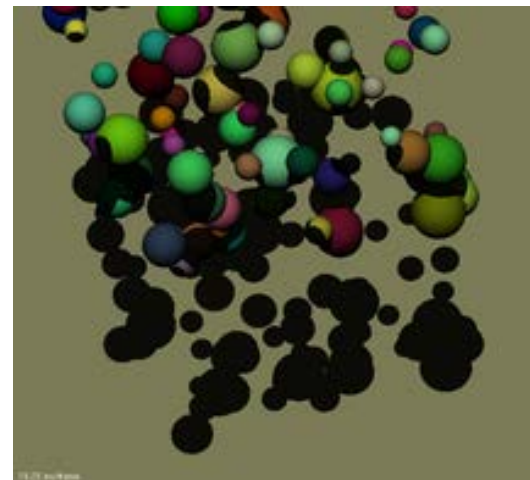
volúmenes



stencil  
buffer



con  
sombras

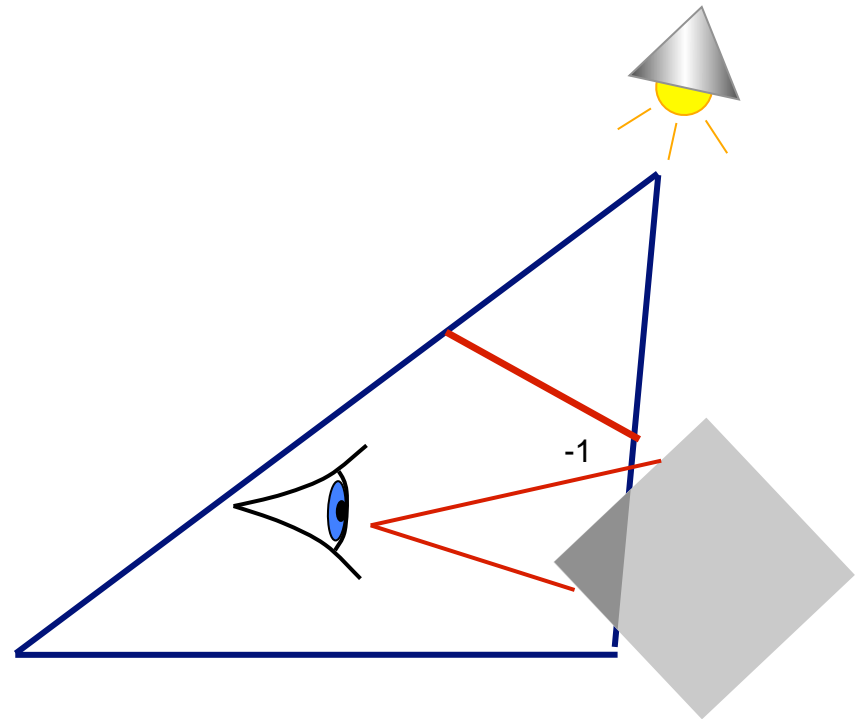


# Volúmenes de sombra

- Ventajas
  - Todo puede generar sombras, incluso un objeto sobre sí mismo
  - Sombras precisas
- Inconvenientes
  - Sólo para luces puntuales con sombras duras
  - Poco eficiente
  - Los objetos translúcidos no producen sombras correctas
  - Falla en algunos casos especiales

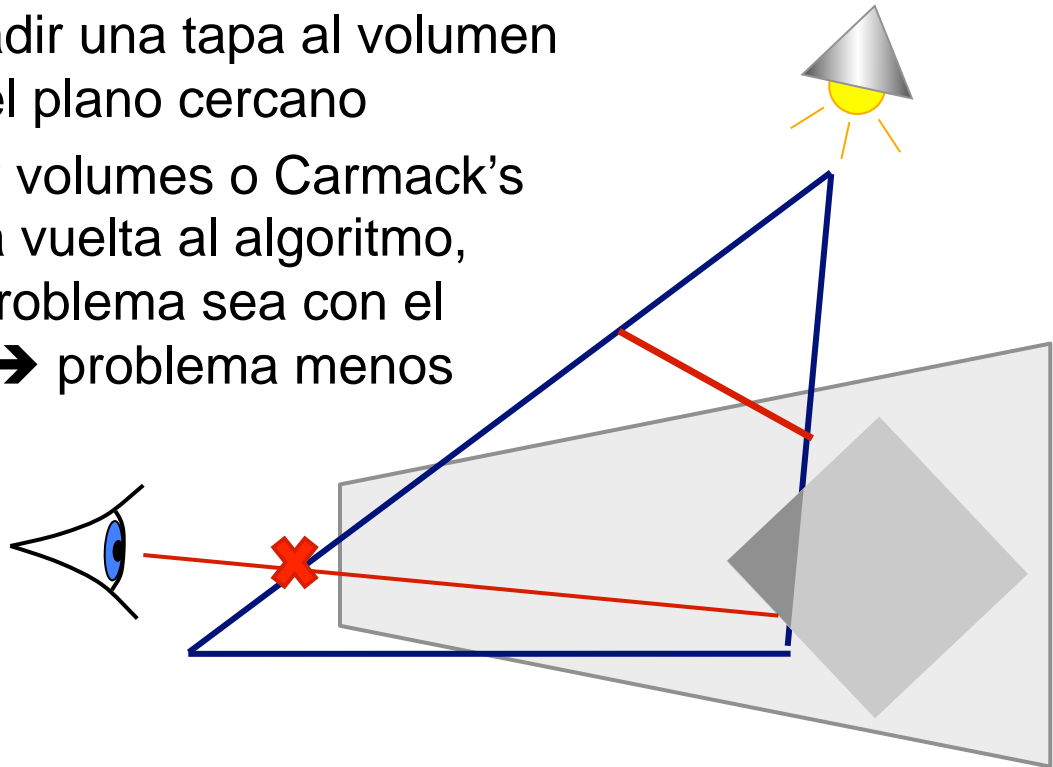
# Volúmenes de sombra: Casos especiales

- El observador está dentro de algún volumen de sombra:
  - Detectar dentro de cuántos volúmenes se encuentra el observador
  - Inicializar el stencil buffer a ese número, en lugar de a 0



# Volúmenes de sombra: Casos especiales

- Pueden perderse intersecciones debido al recorte de algún volumen de sombra con el plano cercano:
  - Capping: Añadir una tapa al volumen por el lado del plano cercano
  - Z-fail shadow volumes o Carmack's reverse: da la vuelta al algoritmo, para que el problema sea con el plano lejano → problema menos molesto



# Volúmenes de sombra: Casos especiales

- Z-fail shadow volumes: Algoritmo idéntico al original pero con cambios en el incremento/decremento del stencil buffer:
  - Dibujar las caras **posteriores** de los VS, incrementando el stencil buffer para los polígonos **no dibujados** (falla el test Z)
  - Dibujar las caras **frontales** de los VS, decrementando el stencil buffer para los polígonos **no dibujados** (falla el test Z)
- Ahora el fragmento está dentro de un volumen de sombra si el stencil buffer es 0
- El problema está ahora con los volúmenes de sombra que intersectan con el plano lejano

¡el plano lejano siempre se puede poner más lejos!

¿Soluciona esto el problema de verdad?

Soluciones hardware (depth clamp) y software (proyección con plano lejano infinito)



# Volúmenes de sombra

- Mejoras sobre el método:
  - Se puede generalizar para luces distribuidas
  - Se pueden conseguir zonas de sombra y penumbra, y sombras difuminadas. La forma más sencilla es utilizando filtros (poco realista) o técnicas de supermuestreo con jittering
  - Alternativamente, se pueden construir volúmenes de luz (los complementarios a los volúmenes de sombra). Los volúmenes de luz pueden visualizarse para conseguir efectos como polvo en suspensión

# Z-Buffer de sombras

- Utiliza un Z-Buffer para generar las sombras
- Se puede incorporar fácilmente al algoritmo conocido del Z-Buffer para detección de superficies visibles
- Se compone de dos pasos:
  1. Se toma como punto de vista la posición de la luz y se calcula el Z-Buffer de sombra
  2. Se aplica el algoritmo conocido del Z-Buffer, modificado para manejar las sombras (Z-Buffer de visualización)

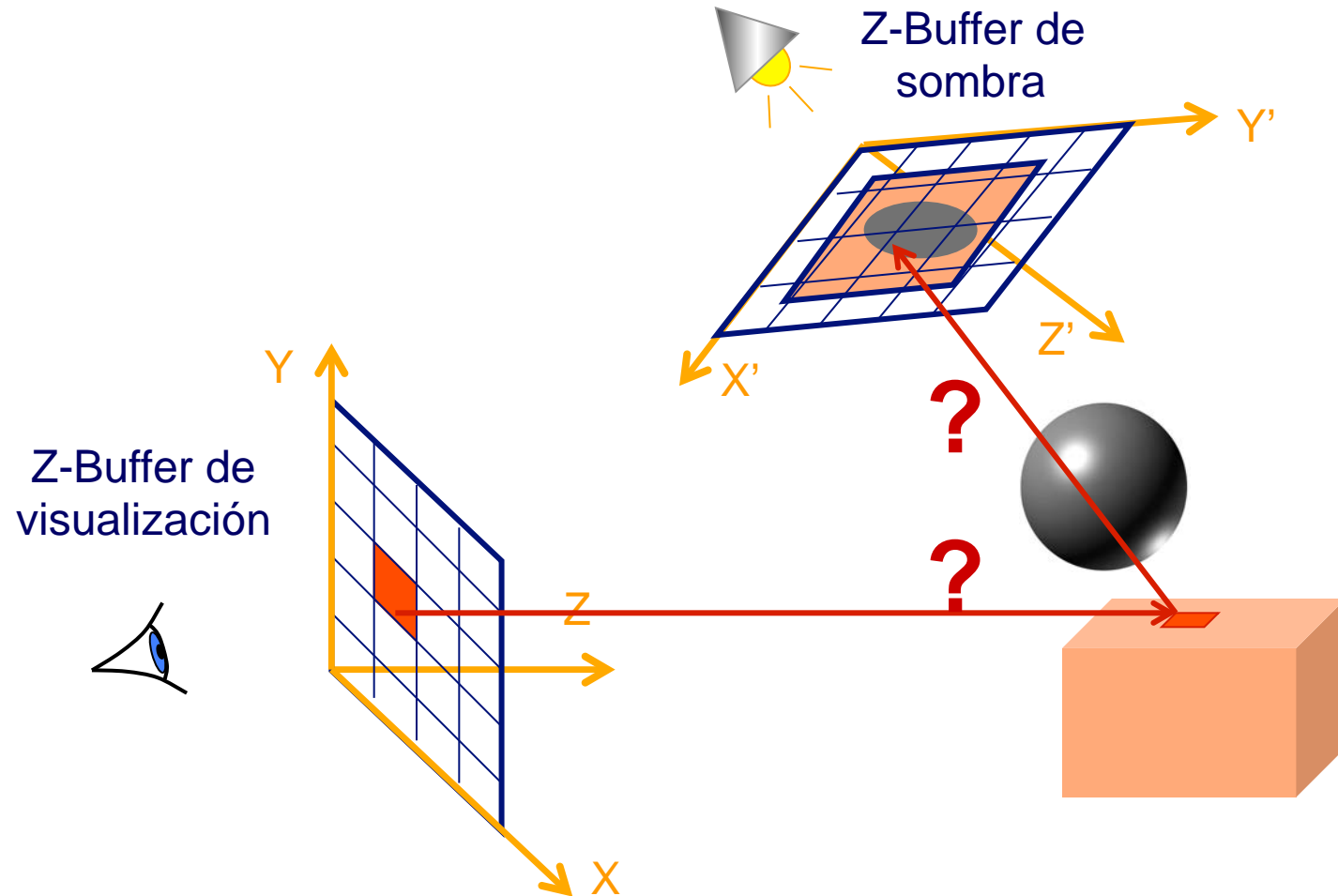
# Z-Buffer de sombras

- Paso 1: Z-Buffer de sombra
  - Se toma como punto de vista la posición de la luz. Esta posición constituye el sistema de coordenadas de la luz
  - Se aplica el algoritmo conocido del Z-Buffer pero sólo para calcular profundidad
  - A éste buffer se le llama Z-Buffer de sombra, y contiene qué objetos son visibles desde la luz (es decir, son directamente iluminados por la luz)
  - Si hay varias luces, se construye un Z-Buffer de sombra para cada una de ellas

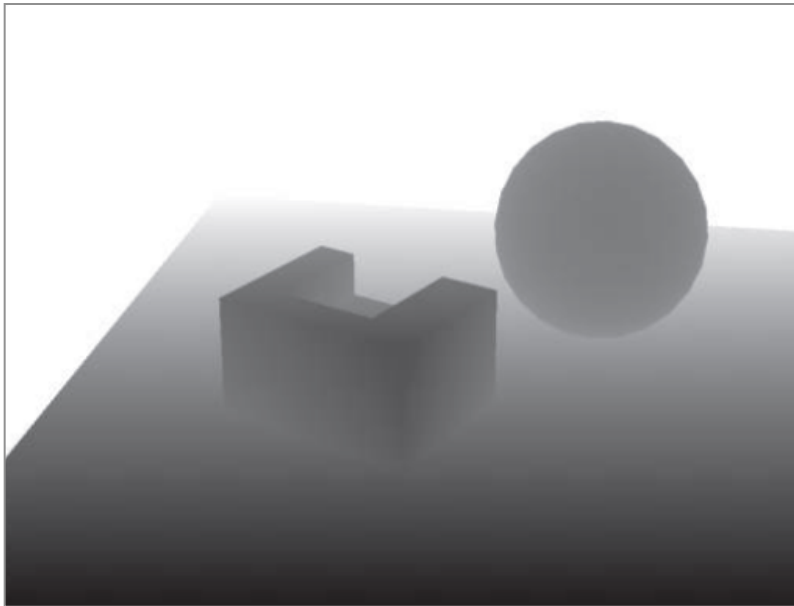
# Z-Buffer de sombras

- Paso 2: Z-Buffer de visualización (algoritmo tradicional del Z-Buffer, con alguna modificación)
  - Sea un píxel que es visible para el algoritmo del Z-Buffer. Sus coordenadas son  $(x, y, z)$
  - Transformar las coordenadas al sistema de coordenadas de la luz. Se obtiene  $(x', y', z')$
  - Comparar  $z'$  con el valor de profundidad almacenado para ese píxel en el Z-Buffer de sombra de la luz correspondiente
  - Si  $z'$  es mayor que esa profundidad, hay un objeto más cercano a la luz que ese píxel, por lo que no debemos iluminar el punto con esa luz

# Z-Buffer de sombras



# Z-Buffer de sombras



Z-Buffer desde la posición de la luz

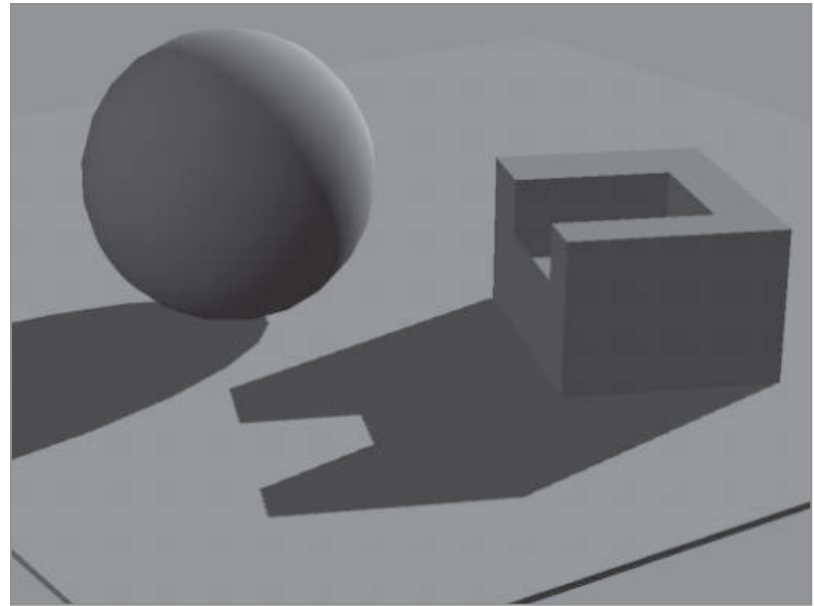
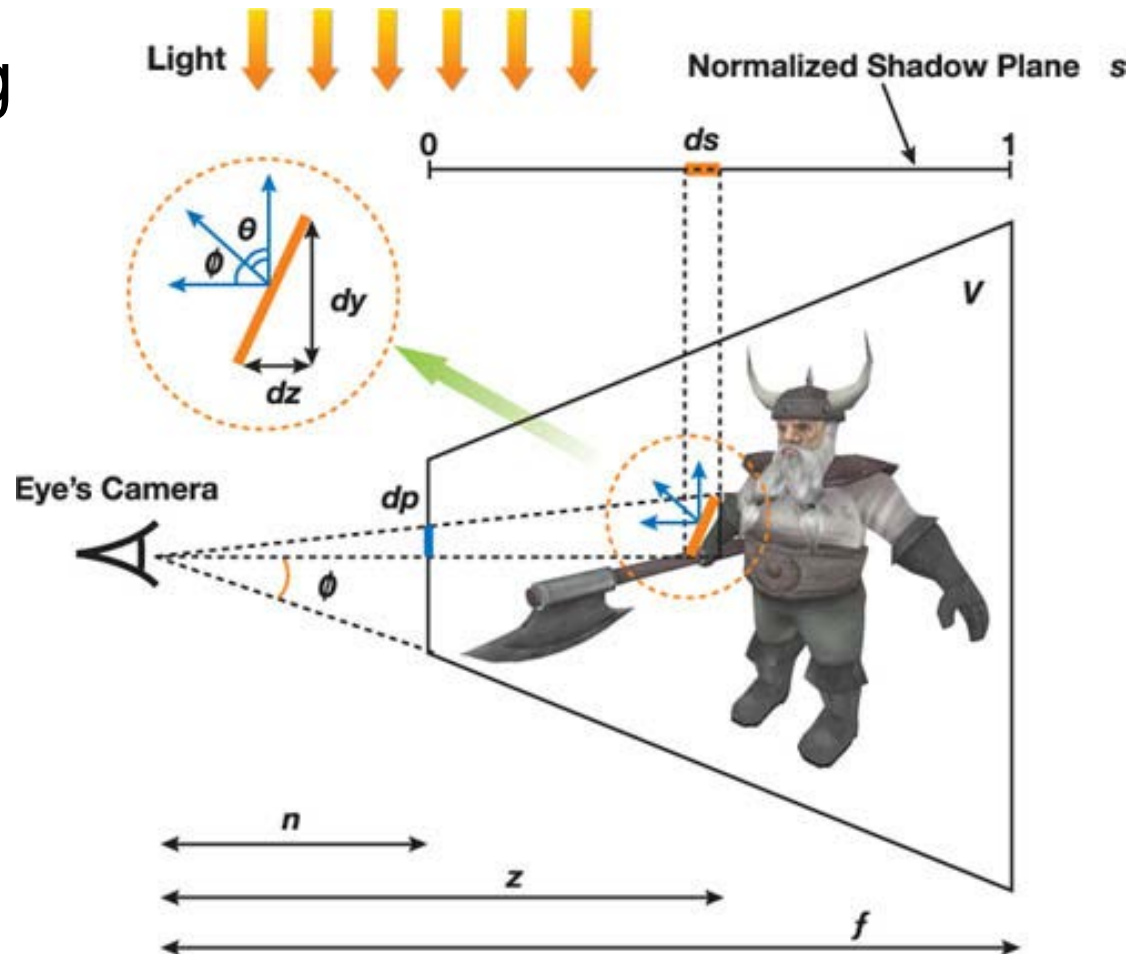


Imagen final

# Z-Buffer de sombras

- Aliasing



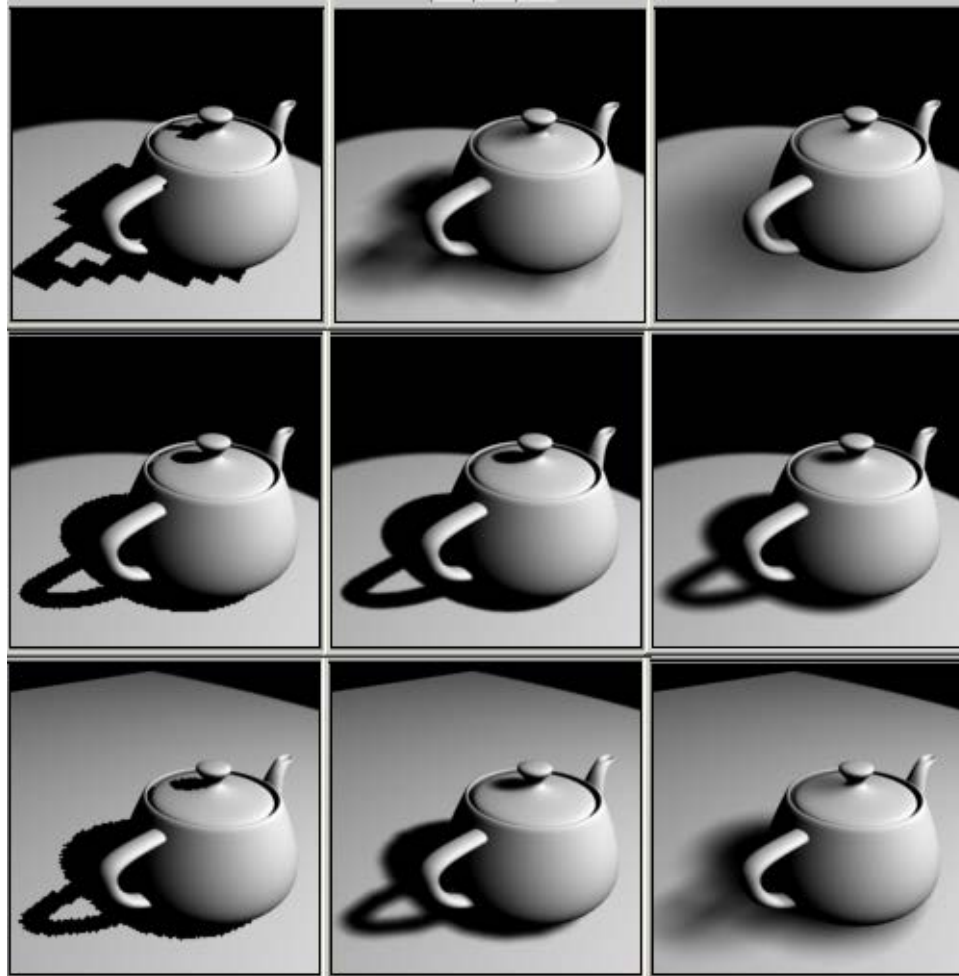
# Z-Buffer de sombras

- Los problemas de aliasing pueden paliarse mediante supermuestreo o filtros





# Z-Buffer de sombras





# Z-Buffer de sombras

- Ventajas:
  - Es muy fácil de implementar
  - Coste temporal asumible
- Inconvenientes:
  - Problemas de aliasing
  - Coste espacial alto, sobre todo si queremos hacer supermuestreo