técnicas avanzadas de gráficos

ingeniería multimedia

Seminario 7

Render básico:

clipping, culling, shading, z-buffer

Departamento de Ciencia de la Computación e Inteligencia Artificial
Universidad de Alicante

Clipping, culling, shanding, z-buffer

¿Qué es *clipping*?

¿Qué es el *culling*?

¿Qué es el *shading*?

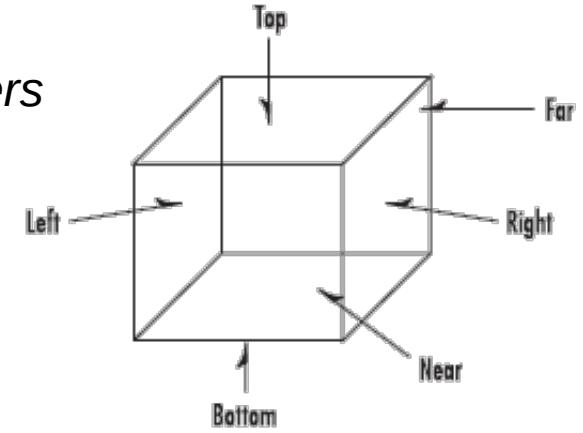
¿Qué es el *z-buffer*?

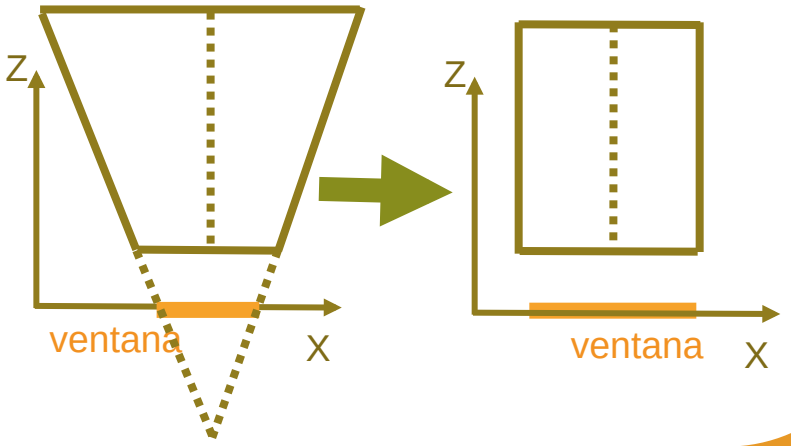
Clipping, culling, shanding, z-buffer

- *Clipping*: eliminación y recortado de los polígonos que quedan fuera del volumen de visualización. Lo realiza la librería gráfica por defecto. No se puede programar.
- *Culling*: eliminación de los polígonos no visibles. Puede programarse: antes de enviar los datos a la tubería, o en la propia tubería (en un *vertex shader*)
- *Shading*: cálculo del color concreto de cada pixel. Automático, aunque una parte puede programarse en la tubería (en un *fragment shader*)
- *Z-Buffer*: identificación de las partes de la escena que son visibles desde la posición del observador. Automático y no programable, aunque el cálculo de la profundidad puede modificarse con un *fragment shader*

Clipping

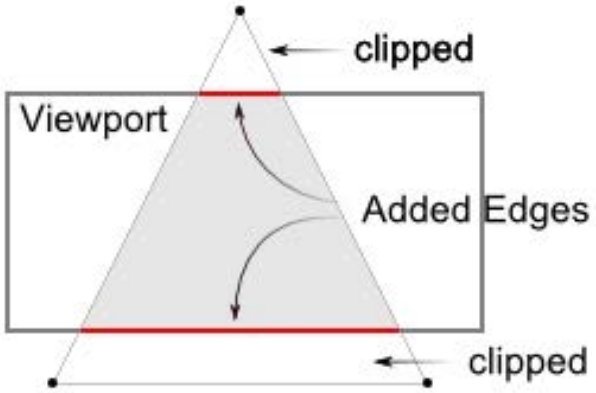
- *Clipping* es la eliminación y recortado de los objetos que quedan fuera del volumen de recorte
- Esta etapa no se puede programar con *shaders*
- El volumen de recorte tiene forma de paralelepípedo ortogonal u octoedro:
  - Si la proyección es paralela, el volumen es un octoedro
  - Si la proyección es perspectiva, la matriz de proyección, convierte el frustum en un octoedro
- El test de interioridad es muy simple: sólo es necesario comparar coordenadas

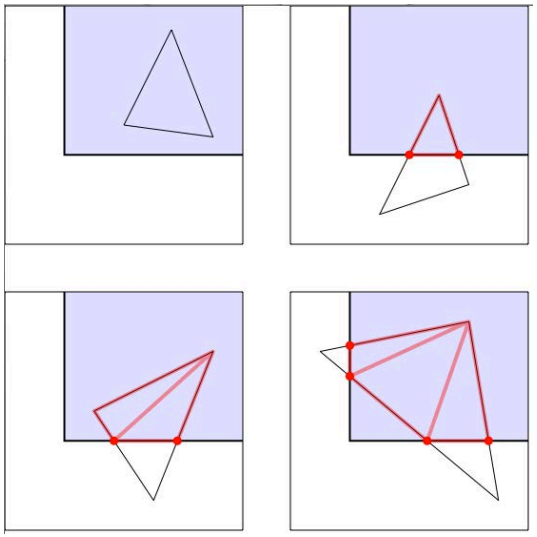




Clipping

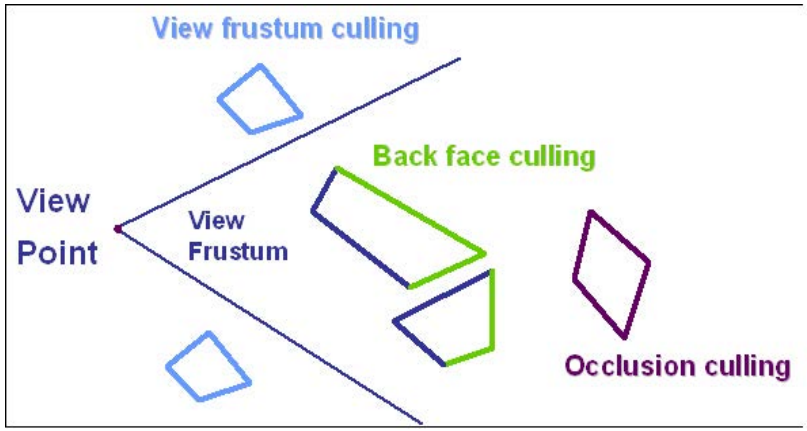
- Algoritmo básico de recorte
  - Si el polígono es totalmente interior al volumen, dibujarlo
  - Si el polígono es totalmente exterior al volumen, descartarlo
  - Si el polígono es parcialmente interior al volumen, generar nuevos vértices en la línea de corte y nuevos polígonos: algoritmos de Sutherland–Hodgman, de Weiler–Atherton, …





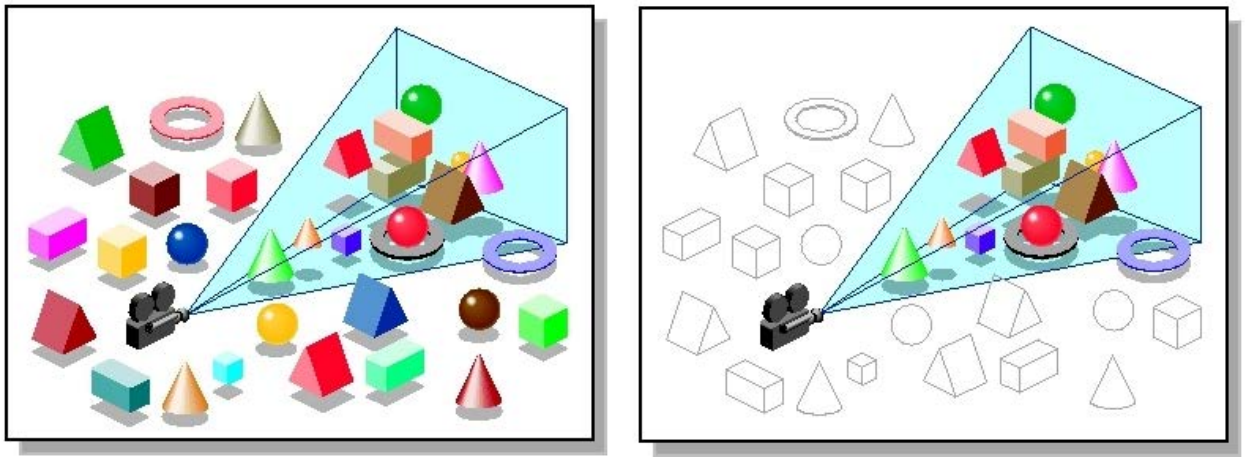
Culling

- *Culling* es la eliminación de polígonos no visibles, de forma previa a enviarlos al dibujado
- Puede programarse de forma previa al envío de los datos a la tubería 3D, o con *vertex shaders*, según el caso
- Un polígono puede ser no visible por:
  - Estar fuera del volumen de visualización: *frustum culling*
  - Ser un polígono trasero de un objeto: *back face culling*
  - Estar ocluido por otros polígonos: *occlusion culling*



Frustum culling

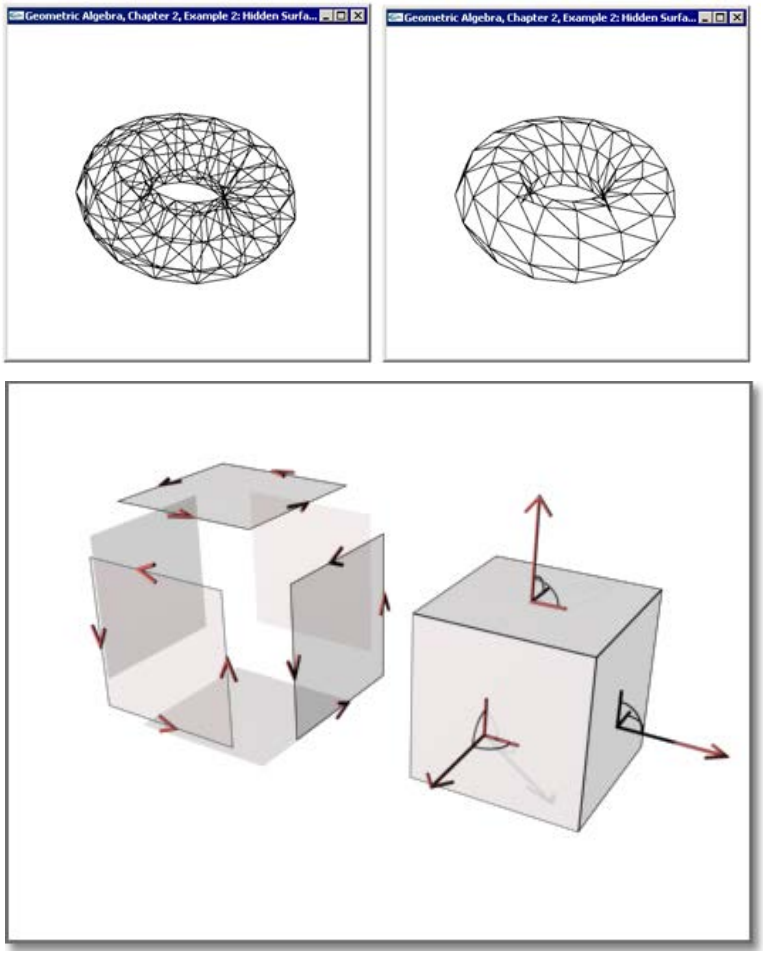
- Es en realidad el primer test del *clipping*: eliminar antes los polígonos que son completamente exteriores al frustum, pero realizado en la CPU y no en la GPU



- Puede aprovechar la estructura espacial de la escena:
  - Arboles octales
  - Escenas basadas en *tiles*
  - *Bounding volumes*

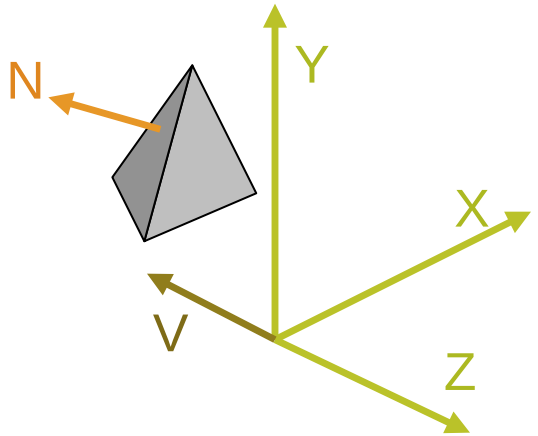
Back face culling

- Elimina las caras que se encuentran de espaldas al observador
- Para saber si una cara es trasera, utilizamos la normal, por ello:
  - Los objetos deben ser cerrados
  - Los vértices de las caras deben estar descritos en sentido antihorario vistos desde fuera del objeto



Back face culling

- Algoritmo básico: Compara la normal N de cada cara con el vector de dirección de vista V
  - Si ángulo entre N y V > 90 → Cara visible
  - Si ángulo entre N y V < 90 → Cara no visible
- Si trabajamos en coordenadas de vista: $N=N_x,N_y,N_z)$ y $V=(0,0,-V_z)$
  - Si $N_z < 0$ → Cara visible
  - Si $N_z > 0$ → Cara no visible



Occlusion culling

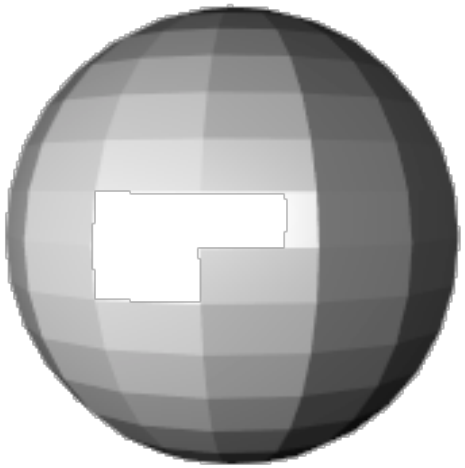
- Dos aproximaciones:
  - *Potentially visible set* (PVS): divide la escena en regiones y precalcula la visibilidad.
  - *Portal rendering*: divide la escena en sectores (habitaciones) y portales (puertas) y computa qué sectores son visibles desde cada portal, mediante un proceso similar al del frustum *culling*.
- Son algoritmos complejos
- Artículo de Cohen-Or: una revisión de los métodos

Shading

- El *shading* o sombreado de polígonos es la asignación de intensidades a cada punto de los polígonos que forman un objeto
- La etapa de *shading* se puede programar a través de *shaders* (*fragment shaders*)
- Un objeto formado por polígonos puede representar:
  - Un objeto poliédrico: al ser una representación exacta interesa marcar las aristas entre los polígonos
  - Una aproximación de un objeto curvo: interesa eliminar en los posible las aristas entre caras

Shading plano

- Se calcula un único valor de intensidad para cada polígono mediante el modelo de iluminación elegido (por ejemplo el de Phong), y se asigna a todos sus puntos
- Es exacto cuando:
  - El objeto es un poliedro
  - Las fuentes de luz se encuentran alejadas del objeto (en esos casos $\boldsymbol{N \cdot L}$ y la función de atenuación se pueden considerar constantes)
  - El observador está lejos del objeto ($\boldsymbol{V \cdot R}$ se puede considerar constante)

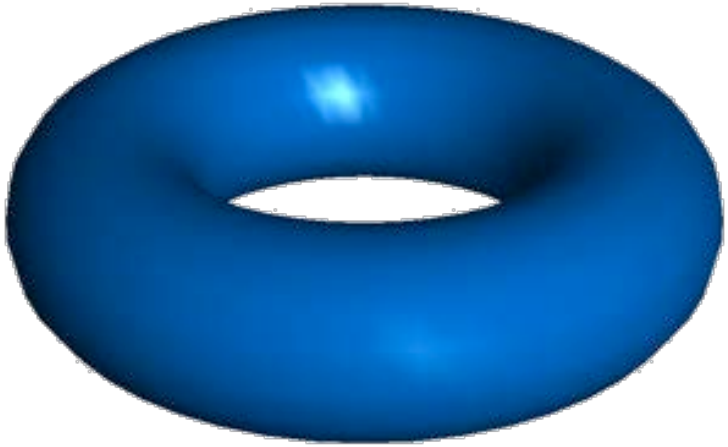


Shading plano

- Inconvenientes del sombreado plano
  - Si el objeto es una aproximación poliédrica de un objeto curvo, se visualizan las aristas entre polígonos.
  - No pueden representarse brillos interiores a los polígonos, pues todos sus puntos tienen intensidad constante.
  - Para obtener una buena representación de objetos aproximados, son necesarios muchos polígonos de pequeño tamaño.

Shading de Gouraud

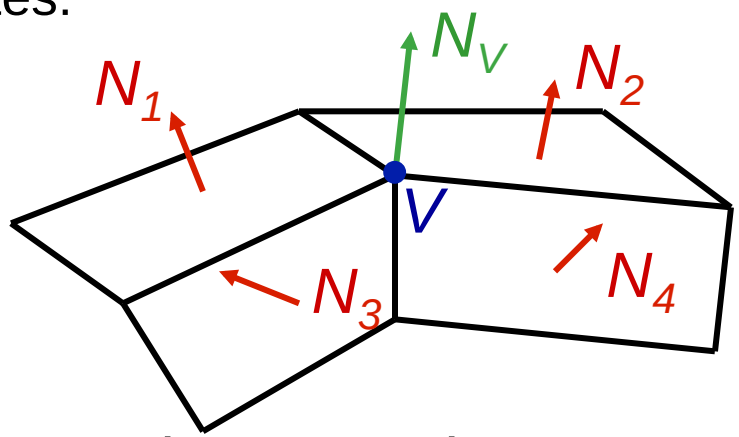
- Es un método incremental que realiza una interpolación de intensidades
- En cada vértice del polígono se calcula la intensidad según el modelo de iluminación elegido
- La intensidad de los puntos intermedios se calcula por interpolación lineal.



Shading de Gouraud

1. Calcular intensidad en los vértices
   - Calcular normal en el vértice: para evitar visualizar las aristas, la normal se calcula como la media de las normales de polígonos adyacentes:

$$N_V = \frac{\sum_{k=1}^{n} N_k}{\left| \sum_{k=1}^{n} N_k \right|}$$



   - Calcular la intensidad en el vértice I_V según el modelo de iluminación de elegido (por ejemplo Phong)

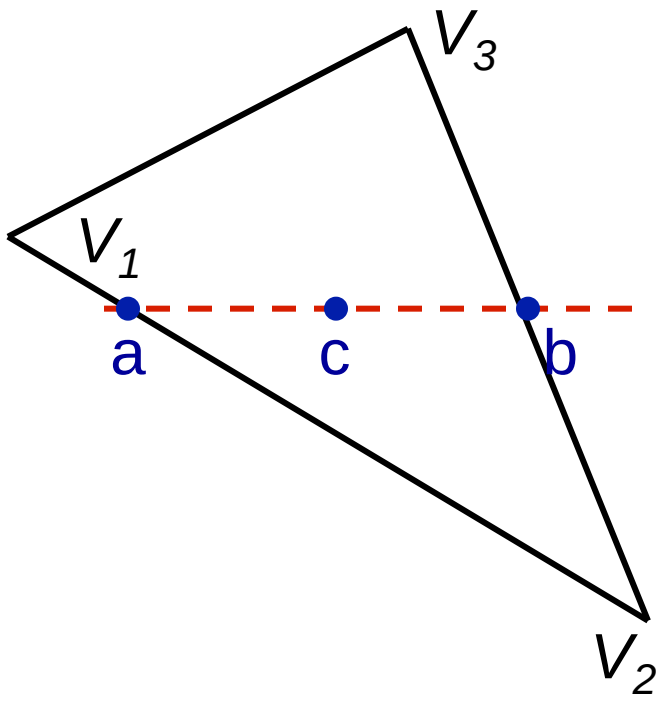
Shading de Gouraud

2. Calcular la intensidad de los puntos interiores por interpolación lineal de las de los vértices:

$$I_a = \frac{y_a - y_{V_2}}{y_{V_1} - y_{V_2}} I_{V_1} + \frac{y_{V_1} - y_a}{y_{V_1} - y_{V_2}} I_{V_2}$$

$$I_b = \frac{y_b - y_{V_2}}{y_{V_3} - y_{V_2}} I_{V_3} + \frac{y_{V_3} - y_b}{y_{V_3} - y_{V_2}} I_{V_2}$$

$$I_c = \frac{x_b - x_c}{x_b - x_a} I_a + \frac{x_c - x_a}{x_b - x_a} I_b$$

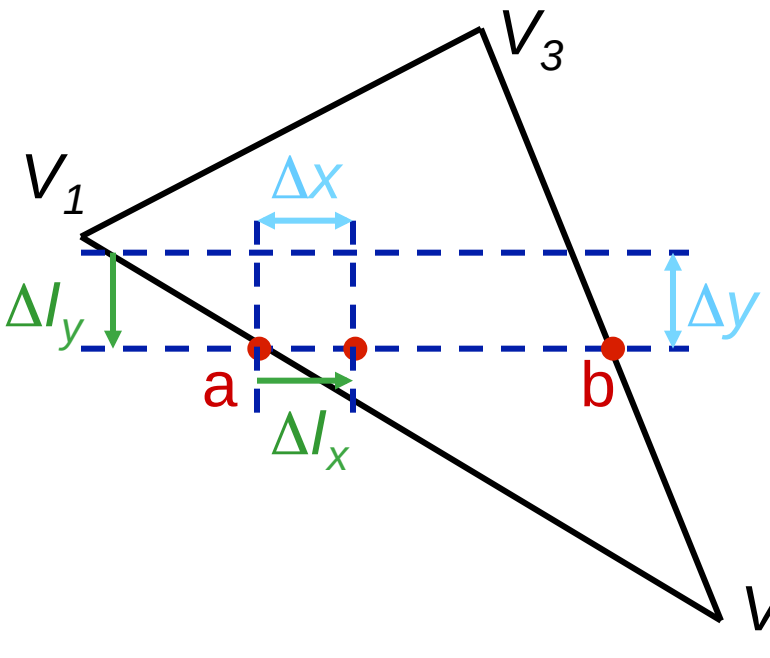


Shading de Gouraud

3. Las intensidades pueden calcularse de forma incremental, rellenando el polígono por líneas de rastreo:

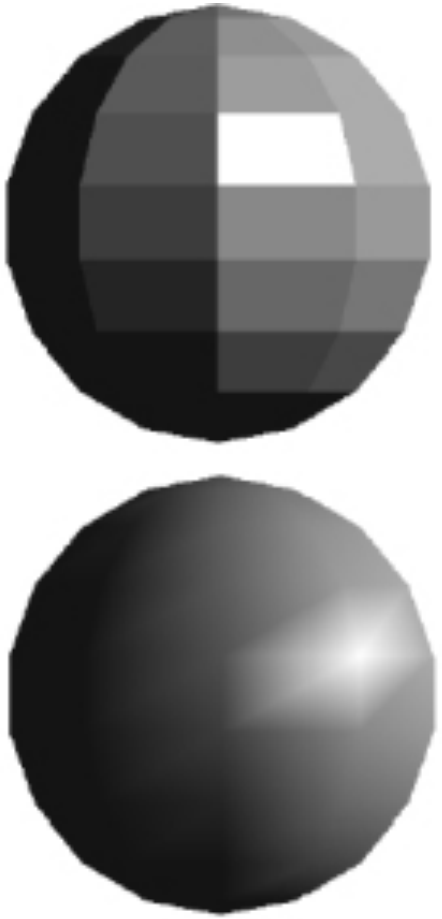
$$\Delta I_y = \Delta y \frac{I_{V_2} - I_{V_1}}{y_{V_2} - y_{V_1}}$$

$$\Delta I_x = \Delta x \frac{I_b - I_a}{x_b - x_b}$$



Shading de Gouraud

- Ventajas
  - Rápido
  - Elimina las aristas: mejora la visualización de las aproximaciones poliédricas de objetos curvos
- Inconvenientes
  - No representa bien los brillos especulares, por lo que se suele utilizar sólo para reflexión difusa



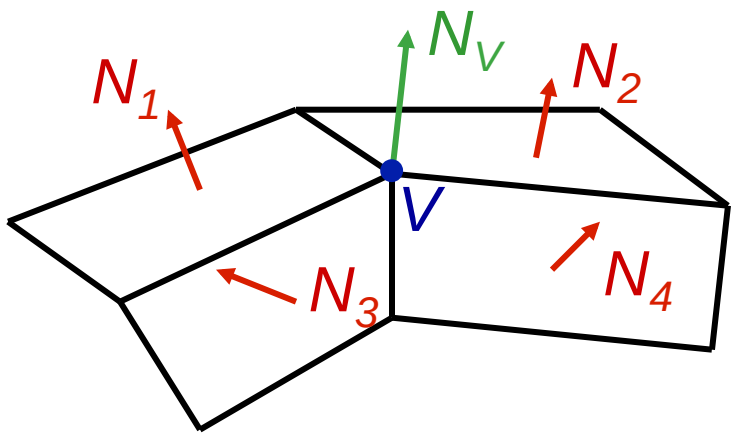
Shading de Phong

- Es un método incremental que realiza una interpolación de normales (en vez de interpolación de intensidades).
- En cada vértice del polígono se calcula la normal como media de las normales de los polígonos adyacentes. La normal de los puntos intermedios se calcula por interpolación lineal.
- En cada punto se aplica el modelo de iluminación elegido.

Shading de Phong

1. Calcular normal en los vértices
   - Como en el sombreado de Gouraud, para evitar visualizar las aristas, la normal se calcula como la media de las normales de los polígonos adyacentes:

$$N_V = \frac{\sum_{k=1}^{n} N_k}{\left| \sum_{k=1}^{n} N_k \right|}$$



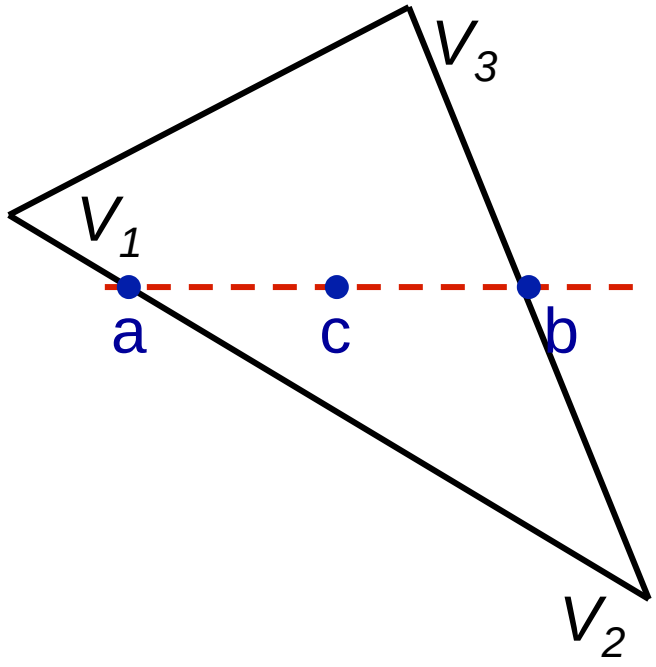
Shading de Phong

2. Calcular la normal de los puntos interiores por interpolación lineal de las de los vértices:

$$N_a = \frac{y_a - y_{V_2}}{y_{V_1} - y_{V_2}} N_{V_1} + \frac{y_{V_1} - y_a}{y_{V_1} - y_{V_2}} N_{V_2}$$

$$N_b = \frac{y_b - y_{V_2}}{y_{V_3} - y_{V_2}} N_{V_3} + \frac{y_{V_3} - y_b}{y_{V_3} - y_{V_2}} N_{V_2}$$

$$N_c = \frac{x_b - x_c}{x_b - x_a} N_a + \frac{x_c - x_a}{x_b - x_a} N_b$$

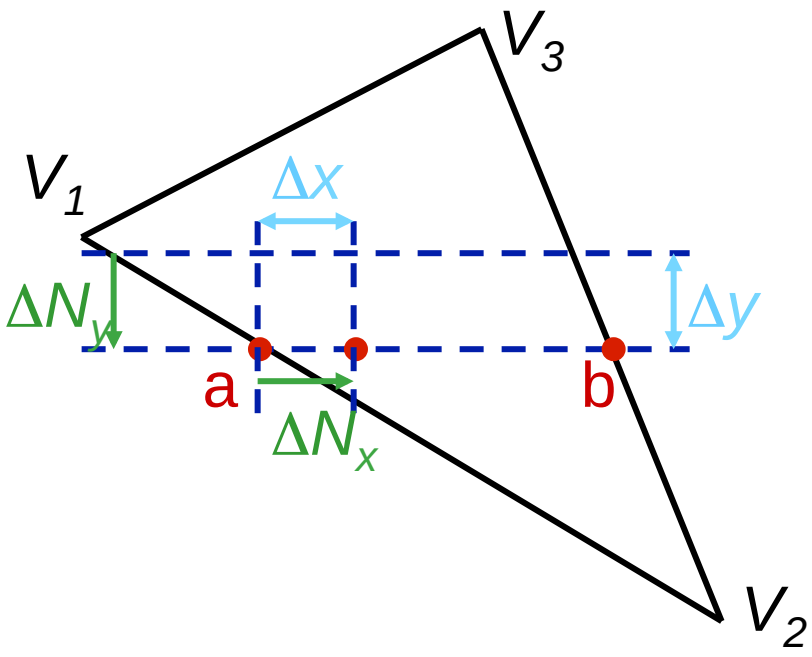


Shading de Phong

- Las normales pueden calcularse de forma incremental, rellenando el polígono por líneas de rastreo.

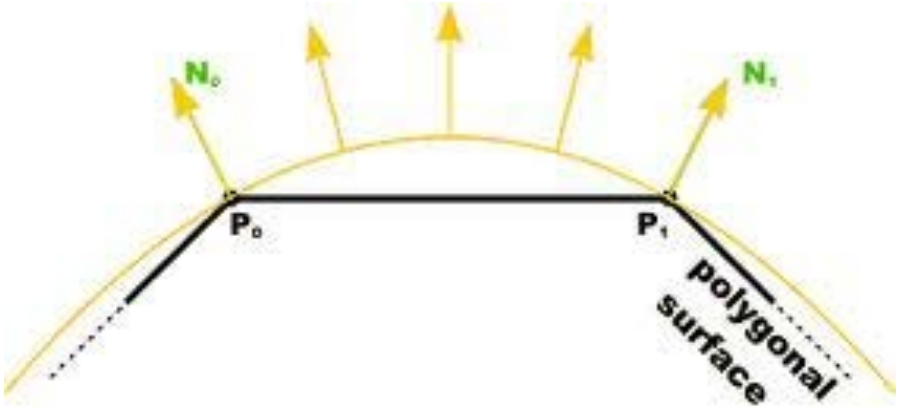
$$\Delta N_y = \Delta y \frac{N_{V_2} - N_{V_1}}{y_{V_2} - y_{V_1}}$$

$$\Delta N_x = \Delta x \frac{N_b - N_a}{x_b - x_b}$$



Shading de Phong

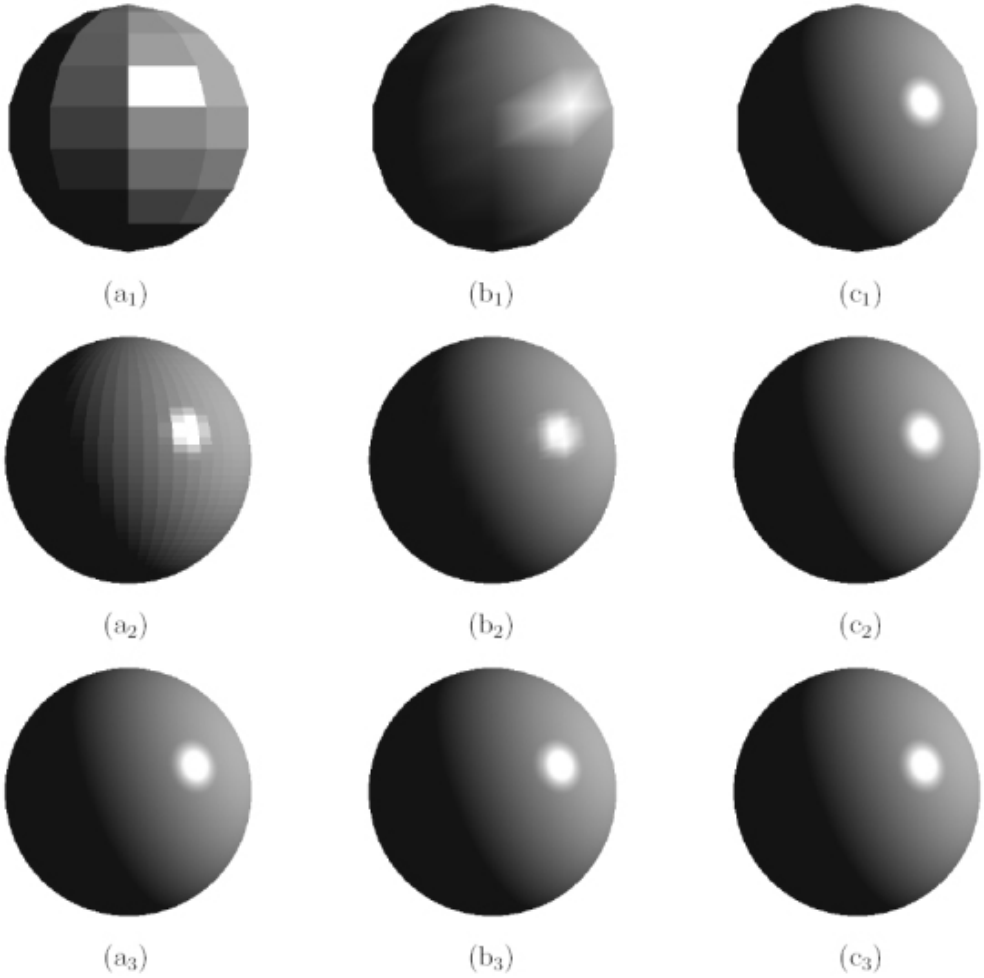
3. Calculo de la intensidad:
   - Calcularla en cada punto mediante el modelo de iluminación elegido.
   - Ahora se puede introducir también la reflexión especular



Shading de Phong

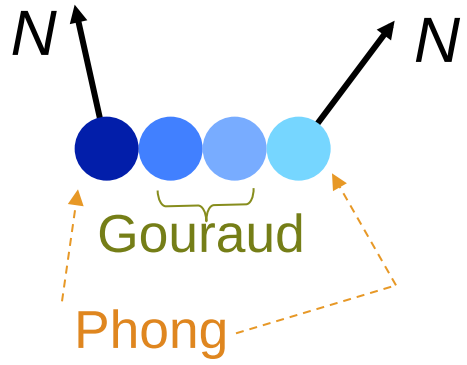
- Ventajas
  - Obtiene mejores resultados que el sombreado de Gouraud
  - Representa bien los brillos especulares
- Inconvenientes
  - Tiene mayor coste que el de Gouraud: el modelo de iluminación se aplica para cada punto en lugar de para cada vértice

Gouraud vs Phong



Optimizaciones
Combinación Gouraud-Phong

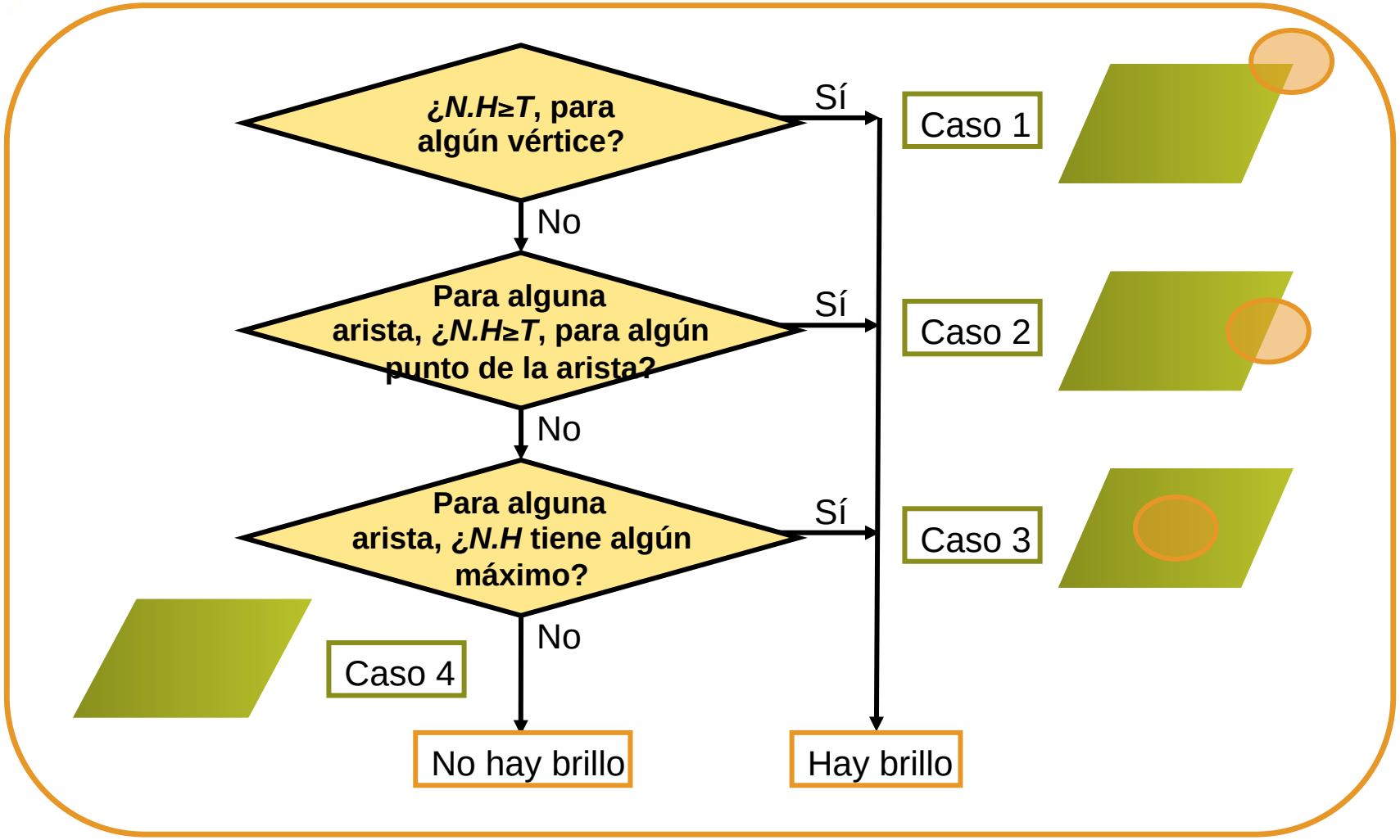
- Consiste en calcular normales en pixeles alternos
- La intensidad en esos puntos se calcula utilizando la media de los adyacentes
- También pueden saltarse más de un píxel y utilizar interpolación lineal
- No existe pérdida de calidad apreciable



Optimizaciones
Detección de brillos

- Consiste en utilizar el *shading* de Phong en polígonos con brillos, y el de Gouraud cuando no hay brillos
- Definimos un test de brillos o *H-Test*
- La componente especular depende del producto ***N.H***
- Diremos que existe brillo cuando $\boldsymbol{N}\cdot\boldsymbol{H} \geq T$, donde T es un umbral adecuado

Optimizaciones
Detección de brillos

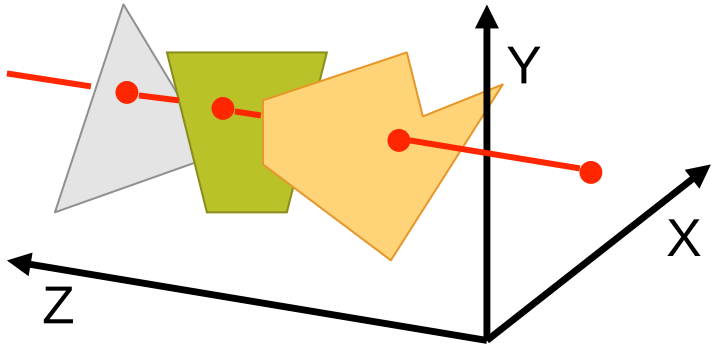


Optimizaciones
Shading de Phong rápido

- Aproxima los cálculos de la intensidad utilizando una serie de Taylor
- Desarrollado para poliedros formados por triángulos
- También se denomina método de las diferencias hacia delante
- Cada intensidad se calcula en función de las anteriores con sólo dos adiciones
- Reduce los cálculos en un tercio

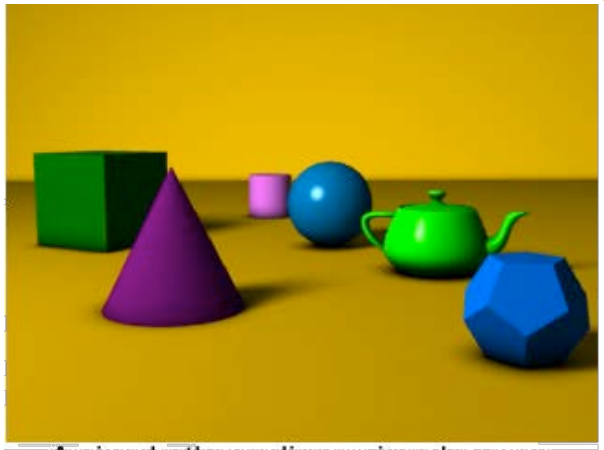
Z-buffer

- También se le llama depth-buffer o buffer de profundidad
- Algoritmo que opera pixel a pixel (espacio de la imagen)
- Para cada pixel, compara la profundidad de todas las superficies que se proyectan en él y toma el color de la más cercana



Z-buffer

- Se utilizan dos buffers o matrices del tamaño de la resolución en pixels
  - Buffer de Intensidad o I-Buffer, que almacena la intensidad de cada pixel
  - Buffer de Profundidad o Z-Buffer, que almacena la profundidad de cada pixel al procesar cada polígono
- El I-Buffer se inicializa al color del fondo y el Z-Buffer a la profundidad máxima
- Se procesa cada polígono, calculando la profundidad de cada punto asociado a cada pixel. Si esa profundidad es menor que la almacenada, se sustituye por ella y se actualiza la intensidad



A simple three dimensional scene

Z-buffer representation

Z-buffer

Para cada pixel (x, y)

Z-Buffer (x, y) ← Profundidad_máxima

I-Buffer (x, y) ← $\text{Intensidad}_{\text{Fondo}}$

Para cada polígono P

Para cada pixel (x, y) en la proyección de P

$Z \leftarrow \text{Profundidad}_{P}$ (x, y)

Si Z < Z-Buffer (x, y) entonces

Z-Buffer (x, y) ← Z

I-Buffer (x, y) ← Intensidad_{P} (x, y)

Z-buffer

- Despejando en la ecuación del plano

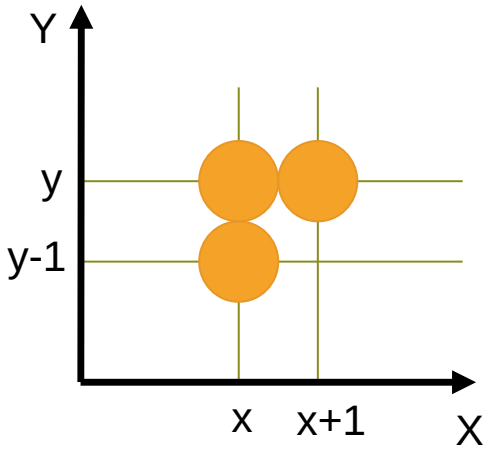
$$Ax + By + Cz + D = 0 \rightarrow z = \frac{-Ax - By - D}{C}$$

- Se pueden aprovechar las líneas de rastreo

$$P_1 = (x, y) \qquad z_1 = \frac{-Ax - By - D}{C}$$

$$P_2 = (x+1, y) \qquad z_2 = \frac{-A(x+1) - By - D}{C} \qquad z_2 = z_1 - \frac{A}{C}$$

$$P_3 = (x, y-1) \qquad z_3 = \frac{-Ax - B(y-1) - D}{C} \qquad z_3 = z_1 + \frac{B}{C}$$



Z-buffer

- Ventajas
  - Muy sencillo de implementar
  - No requiere clasificar las superficies
  - Fácilmente paralelizable e implementable en Hardware
- Inconvenientes
  - Altos requerimientos de espacio (poco importante hoy en día)→ Para solucionarlo podemos dividir la imagen en varios trozos y tratarlos por separado

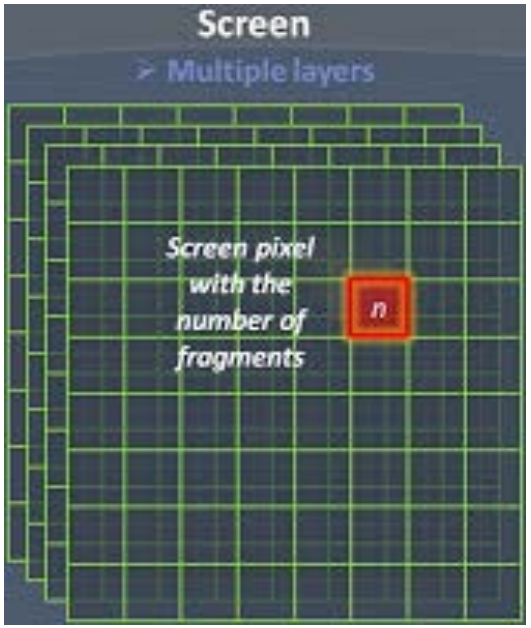
A-buffer

- Buffer de acumulación
- Es una variación del método del Z-Buffer
- En lugar de un I-Buffer y un Z-Buffer utiliza un A-Buffer (buffer de acumulación)
- Cada posición del buffer almacena una lista de profundidades e intensidades para varias superficies, en vez de una única posición e intensidad
- Este método permite visualizar objetos transparentes y translúcidos

A-buffer

- Cada posición del buffer contiene dos campos: valor de profundidad y una intensidad o puntero
  - Si profundidad>0 → Intensidad para ese pixel
  - Si profundidad<0 → Puntero a una lista de superficies. Cada superficie contiene:
    - Su intensidad en ese punto
    - Su profundidad en ese punto
    - Su opacidad
    - Otros atributos
- La intensidad del pixel se calcula combinando las de las superficies implicadas

A-buffer


Screen
> Multiple layers
Screen pixel with the number of fragments
n

A-buffer

- Ventajas
  - Las mismas del Z-Buffer
  - Permite visualizar superficies transparentes totalmente o en parte
- Inconvenientes
  - Los mismos del Z-Buffer
  - Tiene una estructura de datos más compleja