Métodos para detección de superficies visibles (Hearn-Baker)

Contenido

- Clasificación de métodos
- Eliminación de caras posteriores
- Método depth buffer
- Método A-buffer
- Qué provee OpenGl.

Clasificación de métodos de detección de superficies visibles

- Métodos en el espacio de los objetos
  - Comparan objetos y partes de estos con los otros
- Métodos en el espacio de las imágenes
  - Comparan punto por punto en cada pixel sobre el plano de proyección
- La mayoría de los métodos son sobre el espacio de las imágenes

Coherencia y ordenamiento

- Para mejorar el desempeño computacional
- Ordenamiento: facilitar comparaciones en profundidad
  - Ordenar superficies de acuerdo a su distancia al view-plane
- Coherencia: sacar ventajas de regularidades de la escena
  - En un periodo de tiempo se espera que una línea tenga intensidades de pixeles constantes
  - Los frames de animaciones tienen cambios en la vecindad de los objetos que se mueven

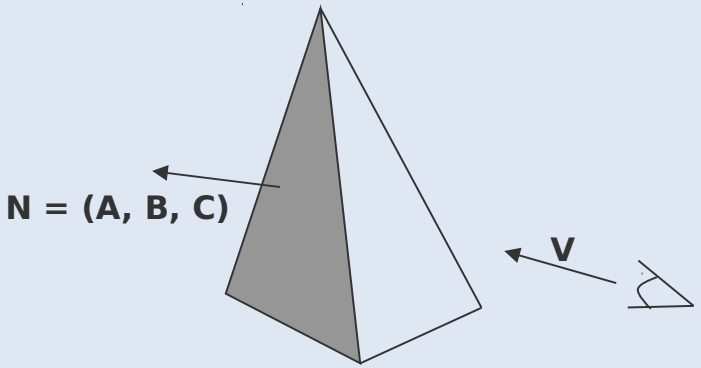
Detección de caras posteriores (back-faces)

- Algoritmo simple: detectar las caras posteriores y eliminarlas
- ¿Cómo? Un punto (x,y,z) está detras de una superficie poligonal con parámetros del plano A,B,C y D si

$$Ax + By + Cz + D < 0$$

- Un polígono es una back-face si

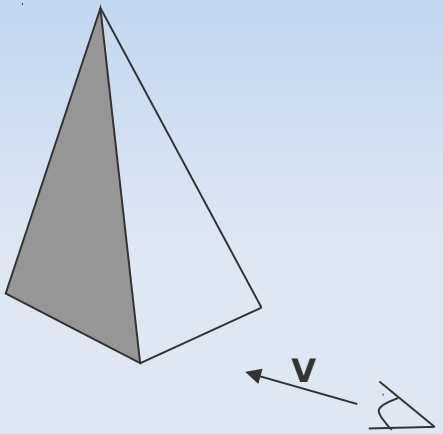
$$V \cdot N > 0$$



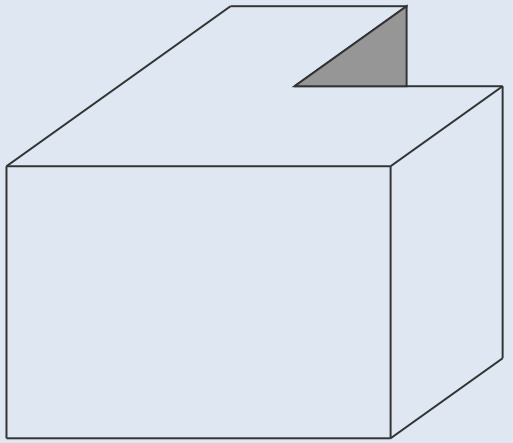
  - V es la dirección de viewing
  - N es el vector normal al polígono

Detección de caras posteriores (back-face)

- ¿En qué escenas se puede aplicar directamente?
  - Visualización de poliedros convexos que no se superponen



- En caso de poliedros concavos
  - Se necesitan más tests
  - En general, este algoritmo elimina aprox. la mitad de las caras no visibles

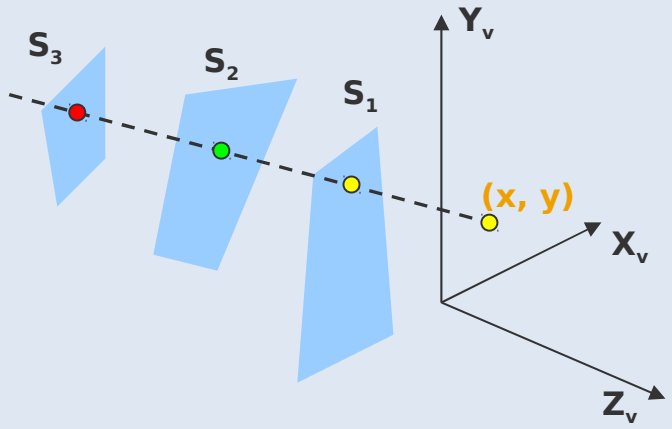


Detección y eliminación de caras posteriores (back-face) en Open GL

- La eliminación de caras posteriores (back-face) se hace con:
  - glEnable(GL_CULL_FACE)
  - glCullFace(mode)
- donde a mode de la asigna GL_BACK.
- Nota: Se podría usar esta función para borrar tambien solo las caras frontales o borrar ambas. En qué casos?
  - Ver eli nterior de los objetos. Hay dos alternativas:
    - Mode = GL_FRONT
    - Especificar que orden de los vértices GL_CW o GL_CCW se considera como cara frontal
      - glFrontFace(vertexOrder)
  - Ver el cableado (wireframe)? GL_FRONT_AND_BACK
  - glDisable(GL_CULL_FACE)

Método depth buffer

- Método que trabaja en espacio de la imágen (muy usado)
  - Compara la profundidad de cada pixel en el plano de proyección
    - Conocido como método z-buffer (comparaciones de profundidad se hacen en el eje z)
  - Generalmente se aplica a escenas de superficies poligonales
  - Valores de profundidad calculados muy rápido
  - Fácil de implementar



Depth buffer & frame buffer

- Típicamente, se aplica en coordenadas normalizadas (en 0 está el near clipping plane(view plane) y en 1 el far clipping plane)
- Se requieren dos áreas de buffer:
  - Depth buffer:
    - Almacena valores para cada posición (x,y)
    - Todas las posiciones se inicializan con la profundidad máxima
  - Frame buffer(refresh buffer):
    - Almacena los colores de cada posición
    - Todas las pocisiones son inicializadas con el color del background

Algoritmo

- Inicializar el depth buffer y el frame buffer

  depth[x, y] = 1, frame[x, y] = $I_{backgnd}$

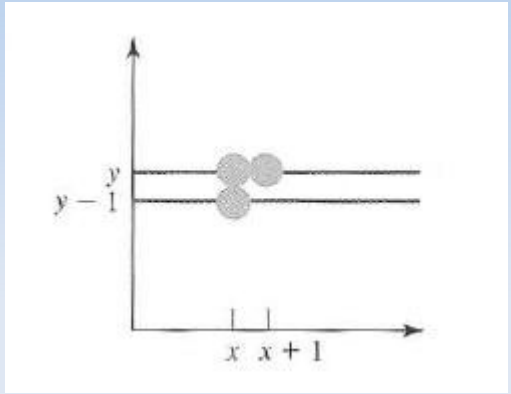
- For each posición (punto) en un polígono
  - Calcular la profundidad de cada posición (x, y) sobre le polígono
  - If z < depth[x, y] then

    depth[x, y] = z, frame[x, y] = $I_{surf}(x, y)$

Algoritmo

- En la posición (x,y) de la superficie, la profundidad z se calcula como:

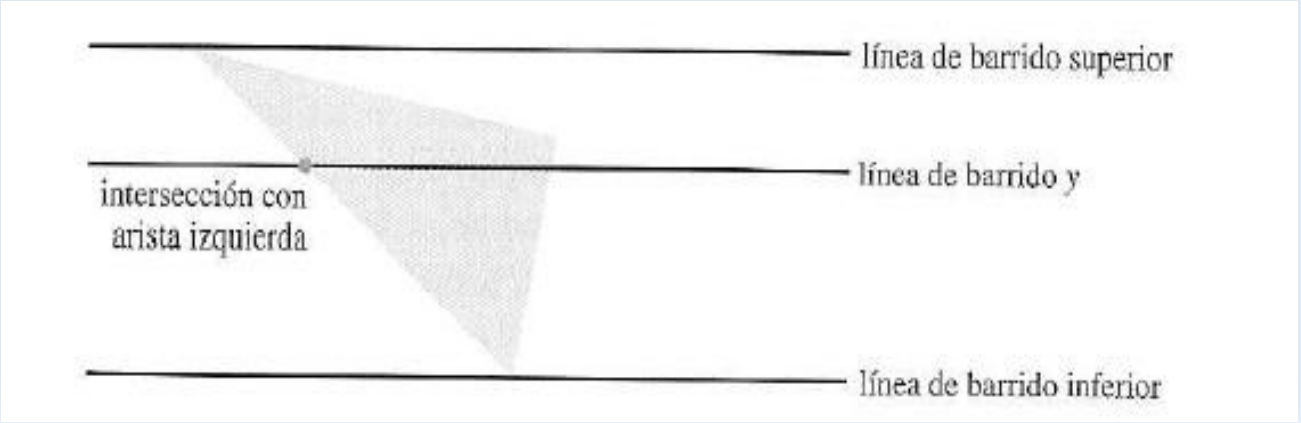
$$z = \frac{-Ax - By - D}{C}$$



- La siguiente posición z' se calcula como:

$$z' = \frac{-A(x+1) - By - D}{C} \quad o \quad z' = z - \frac{A}{C}$$

  - A/C es una costante en cada superficie
- ¿Cómo calcular los límites sobre un tríangulo?
  - Por cada scan-line se calcula la intersección con el arco izquierdo del triángulo



Algoritmo

- Para dibujar el triángulo completo:
  - Comenzar por el vértice superior
  - Calcular a cada paso la coordenada x del arco izquierdo

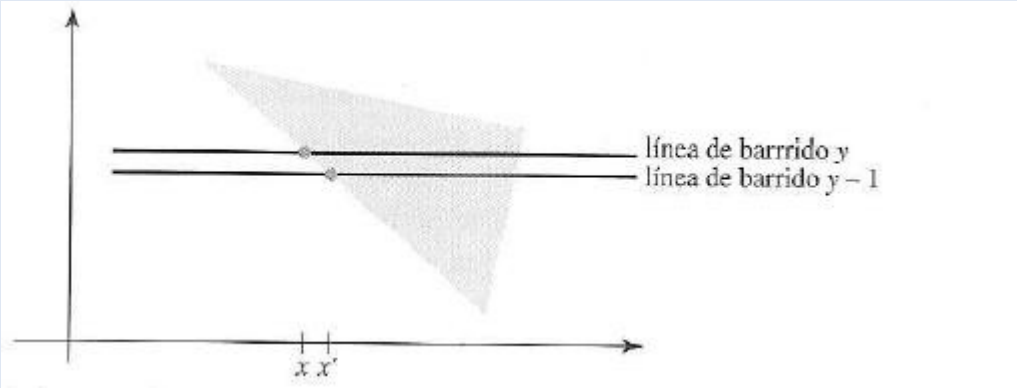
$$x' = x - \frac{1}{m}$$

donde m es la pendiente del arco

- Los valores de z bajando por este arco se obtienen usando

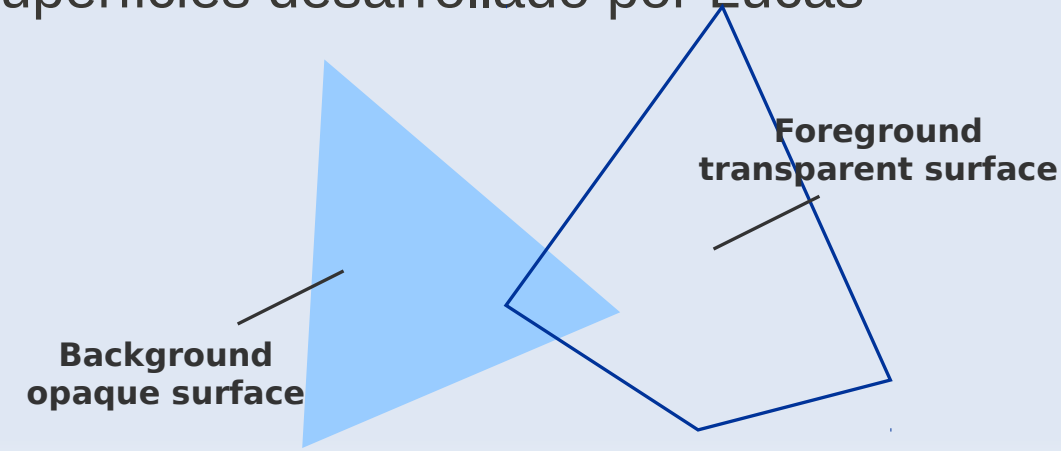
$$z' = z + \frac{A/m + B}{C}$$

Si se procesa una arco vertical se debe usar $z' = z + \frac{B}{C}$



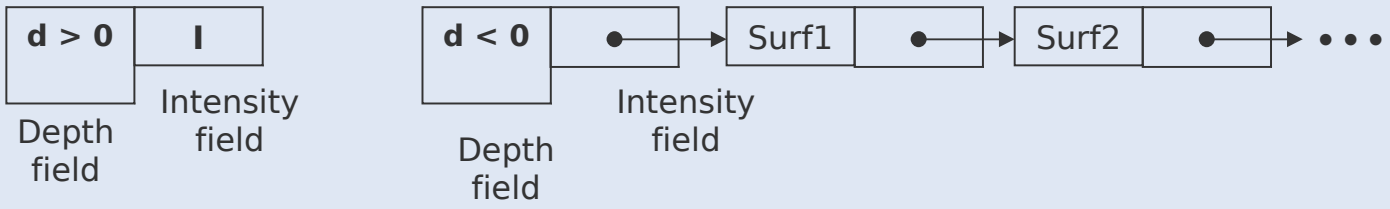
Método A-buffer

- Extiende las ideas del método z-buffer para mejorar sus desventajas
  - Maneja solo caras opacas
  - No puede acumular valores de intensidad para mas de una superficie
- A-buffer: (nombre debido a letra opuesta a Z)
  - Maneja un accumulation-buffer
  - Antialiasing, promedio de áreas, detección de visibilidad
  - Sistema de rendering para superficies desarrollado por Lucas Studios



Método A-buffer

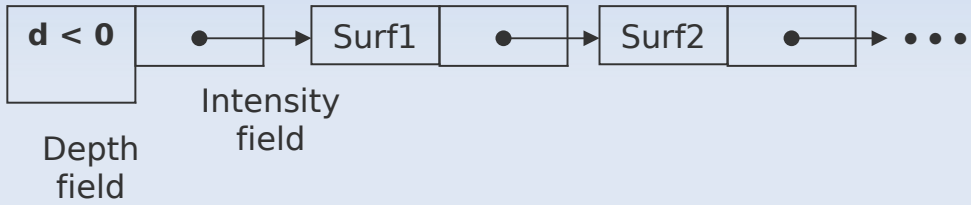
- Cada posición en el buffer accesa una lista enlazada(linked list)
  - Varios colores pueden estar asociados a cada pixel
  - Arcos de los objetos pueden ser antialiased
- Cada posición del A-buffer almacena dos tipos de información (tiene dos campos):
  - Profundidad (depth field)
  - Superficies que se superponen en el pixel (surface data field)



<Organization of an A-buffer pixel position : (a) single-surface overlap (b) multiple-surface overlap>

Método A-buffer

- Si depth es negativo, indica que hay una contribución de múltiples superficies. Por cada superficie, se almacena:
  - Componentes de intensidad RGB
  - Parámetro de opacidad (porcentaje de transparencia)
  - Profundidad
  - Porcentaje de área cubierta
  - Identificador de la superficie
  - Otros parámetros de rendering



- Nota: Las scan lines son usadas para determinar qué tanto una superficie (polígono) cubre un determinado pixel. Los polígonos (triángulos) son “recortados” (clipped) contra el borde de los pixeles, Usando los factores de opacidad y de área cubierta, los algoritmos de rendering calculan el color de cada pixel promediando la contribución de cada superficie.

Método z-buffer en openGl

- Para usar el depth buffer se debe llamar a:
  - glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH)
  - Para inicializar el depth buffer se usa:
    - glClear(GL_DEPTH_BUFFER_BIT)
    - Todos los valores a 1 (valores entre 0 y 1)
  - Las rutinas de detección de visibilidad son activadas (desactivadas) con:
    - glEnable(GL_DEPTH_TEST)
    - glDisable(GL_DEPTH_TEST)
- Otras:
  - glClearDepth(maxDepth): define cual es la profundidad máxima a considerar. Valor entre 0 y 1.
  - glDepthRange(nearNormDepth,maxNormDepth): define el rango de profundidad