CC-52b Computación Gráfica

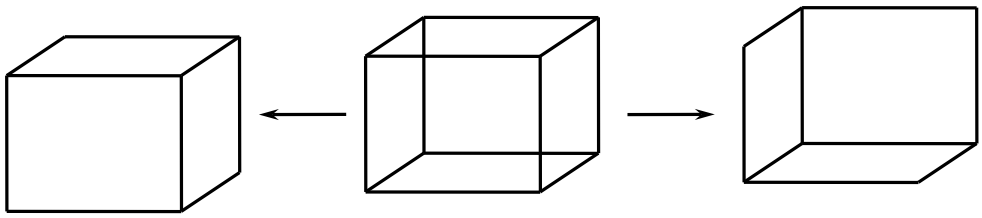
Patricio.Inostroza@*dcc.uchile.cl*

Departamento de Ciencias de la Computación

Universidad de Chile

Visibilidad

- Dado un conjunto de objetos 3D y un punto de vista :
  - ¿ Qué líneas y/o superficies son visibles ?



- Algoritmos de eliminación de superficies y líneas escondidas

Eliminación de superficies y líneas

- Los algoritmos que resuelven este problema lo enfrentan en:
  1. El espacio objeto o precisión objeto
     - En lo cálculos están involucrados polígonos
     - Requiere de punto flotante: exacta
  2. El espacio imagen o precisión imagen
     - Muestra la visibilidad de pixeles
     - Usa precisión entera

Precisión objeto

- Los algoritmos de precisión objeto realizan el proceso de calculo en el sistema de coordenadas del objeto
- Se compara directamente cada objeto con el resto de los elementos, eliminando :
  - Objetos enteros
  - Las partes no visibles del objeto

Precisión Objeto

Algoritmo:

Para cada objeto de la escena

Determinar que partes del objeto se ocultan

por otro objeto

por si mismo

Dibujar las partes visibles del color apropiado

Precisión objeto

- Dado que cada objeto se compara con el resto, su complejidad es $O(n^2)$
- La intersección entre polígonos es la operación más compleja
- Este tipo de algoritmo no 'dibuja'
  - se requiere de otro paso (algoritmo) para transformar las partes visibles en mapa de pixeles

Precisión imagen

- Los algoritmos de precisión imagen se realizan en el sistema de coordenadas de la imagen
- Aquí se determina cual de los ‘n’ objetos es visible en cada uno de los ‘p’ pixeles de la imagen

Precisión imagen

- Los algoritmos de precisión imagen se realizan en el sistema de coordenadas de la imagen
- Aquí se determina cual de los ‘n’ objetos es visible en cada uno de los ‘p’ pixeles de la imagen

Algoritmo:

Para cada pixel de la imagen

- Determinar el objeto más cercano al observador y que intersecta con el proyector que pasa por el pixel
- Dibujar el pixel del color apropiado

Precisión imagen

- Dado que debe comparar todos los objetos con cada pixel, este tipo de algoritmo tiene un complejidad O(n*p)
- La operación más completa corresponde a la intersección de la recta con el objeto

Superficies y líneas escondidas

- Algoritmos de precisión imagen
  - Zbuffer
  - A-Buffer
  - Ray-Casting
- Algoritmos de precisión objeto
  - Algoritmo del pintor
  - Árboles BSP
  - Ordenamiento en Z

Z-Buffer

- Algoritmo de espacio imagen
- Muy simple de implementar:
  - Por software
  - Por hardware

- Requiere:
  - Un buffer para los colores de los pixeles
  - Un buffer (Z-Buffer) para almacenar la coordenada Z actual

Z-Buffer

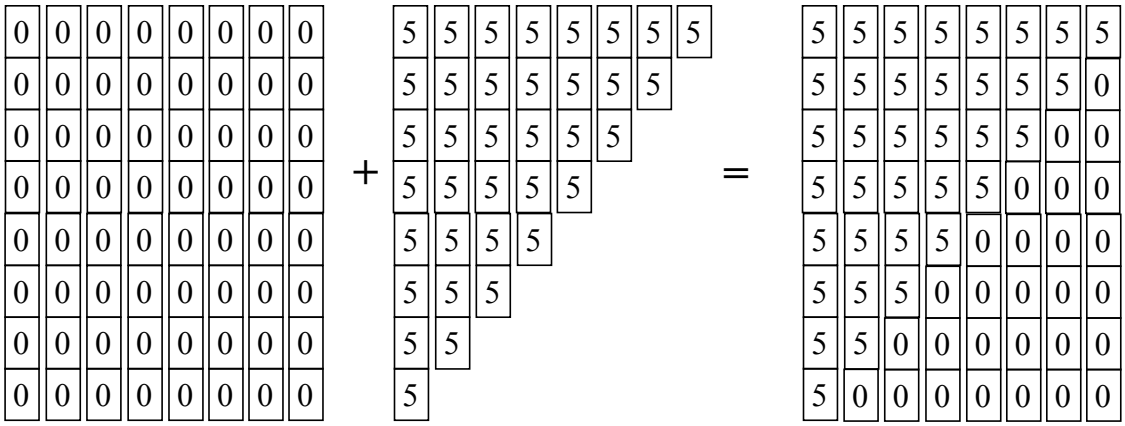
- El buffer de colores es inicializado con el color de fondo
- El Z-buffer es inicializado con valor cero
  - Valor del plano posterior en el clipping 3D

Algoritmo:

```
Inicializar el buffer con el color de fondo y el Z-Buffer
Para cada polígono
    Para cada pixel en la proyección del polígono
        z = z(x,y)
        Si z es más cercano que zbuffer(x, y)
            zbuffer(x, y) = z
            Dibujar pixel(x,y) del color adecuado
```

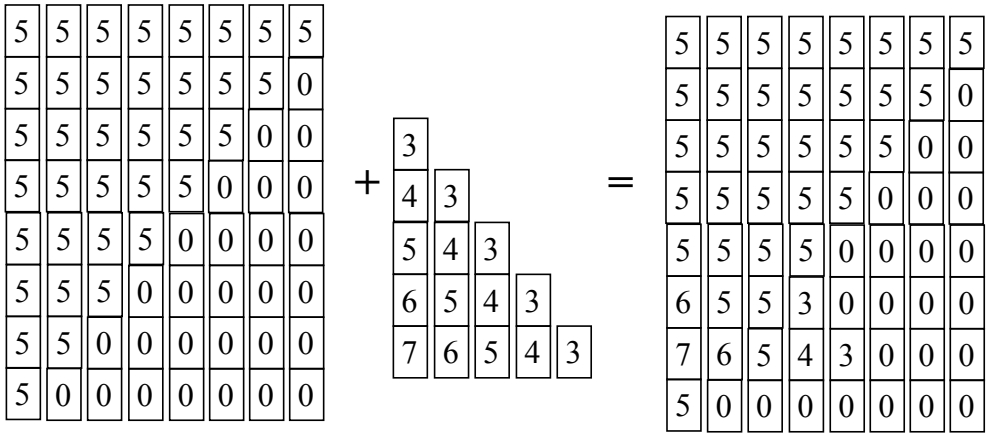
Z-Buffer

- Ejemplo:



Z-Buffer

- Ejemplo:



Z-Buffer

- Ventajas:
  - Simple de implementar
  - No requiere ordenamiento
  - Complejidad ilimitada de la escena
- Desventajas
  - Requiere de buffers (memoria)
  - Gasta tiempo dibujando objetos ocultos
  - Problemas de precisión en Z
  - No maneja transparencias

A-Buffer

- Z-buffer no puede manejar la transparencia de la superficie
- El algoritmo A-buffer es una variación de Z-buffer
- Para implementar este algoritmo se adiciona un estructura de datos (lista) que mantiene un enlace con todas las superficies que contienen el punto
- Propuesto:
  - Investigue el funcionamiento de esta técnica

Ray-Casting

- Técnica que permite encontrar los objetos visibles utilizando 'rayos de luz' que van desde el observador hacia la escena
  - Sentido inverso del rayo de luz proveniente de la escena
- Los rayos pasa a través de cada píxel de la pantalla
- De las intersecciones resultantes, su elige aquella que sea la más próxima al observador
- El píxel se colorea con el color del punto intersectado

Ray-Casting Seudo-Algoritmo

```
Image RayCasting(camara, scena, ancho, alto) {
    Imagen imagen = new Imagen(ancho, alto);
    for (int i=0; i< ancho; i++)
        for (int j=0; j< alto; j++) {
            Rayo rayo = ConstruirRayoAtravesPixel(camara, i, j);
            Intersection punto = EncontrarInterseccion(rayo,
              escena);
            imagen[i][j] = getColor(punto);
        }
    return imagen;
}
```

Ray-Casting

- Es necesario determinar la intersección de un rayo con los objetos de la escena:
  - Modelo sólidos
    - Esfera
    - Cilindro
  - Superficies
    - Plano
    - Triángulo
    - polígono

Rayos

- Los rayos
  - Son líneas paramétricas
  - Definidas por
    - Un origen (p0)
    - Una dirección (d)
- Ecuación del rayo:

$$p(t) = p0 + t * d$$

Intersección con la esfera

- Esfera:
  - Centrada en C(cx, cy, cz)
  - Un vector Q que nace desde el centro
- Cada punto de la esfera satisface la ecuación:

$$| Q - C |^2 - r^2 = 0$$

- Donde, r es el radio de la esfera
- Recordemos que la ecuación del rayo es:

$$p(t) = p0 + t * d$$

- Sustituyendo...

Intersección con la esfera

- Sustituyendo...

$$| p_0 + t * d - C |^2 - r^2 = 0$$

- Llamando

$$Dp = p_0 - C$$

- Desarrollando, se obtiene la ecuación cuadrática:

$$t^2 + 2t(d \bullet Dp) + | Dp |^2 - r^2 = 0$$

Intersección con la esfera

- Desarrollando, se obtiene la ecuación cuadrática:

$$t^2 + 2t(d \bullet Dp) + | Dp |^2 – r^2 = 0$$

- Sin solución para t $\Rightarrow$ no hay intersección

- Con dos soluciones (t1 < t2)
  - t1 corresponde a punto donde el rayo entra de la esfera
  - t2 corresponde al punto donde el rayo sale de la esfera

Intersección con otros elementos

- Propuesto:
  - Investigue y desarrolle la intersección del rayo (descrito en forma paramétrica) con:
    - Un cilindro
    - Un plano
    - Un triángulo

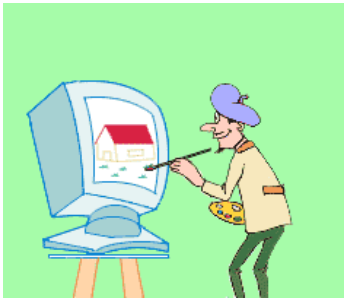
Ray-Casting

- Intersección rayo-polígono:
  - Encontrar P punto de intersección con el plano,
  - Chequear si el polígono contiene P

- Eficiencia: entre un 75% y un 95% del tiempo es dedicado a intersecciones
  - Una pantalla de 320x240 pixeles y una escena de 20 objetos
  - 320x240x20 = 1.536.000 test de intersecciones

Algoritmo del pintor

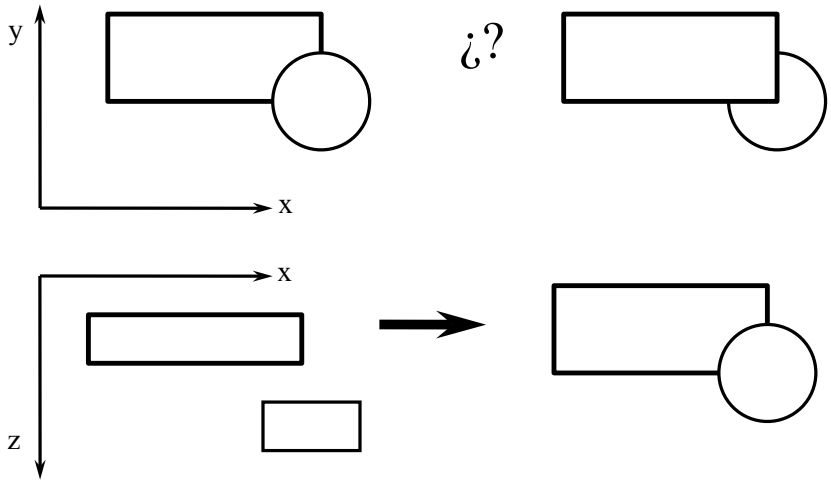
- Un pintor dibuja sus cuadros partiendo desde el fondo hasta llegar a los objetos más próximos al observador.



- El algoritmo del pintor utiliza este mismo razonamiento:
  - Dibujar las superficies desde atrás hacia delante
    - Las superficies más cercanas son dibujadas sobre las superficies de fondo

Algoritmo del pintor

- Ejemplo:



Algoritmo del pintor

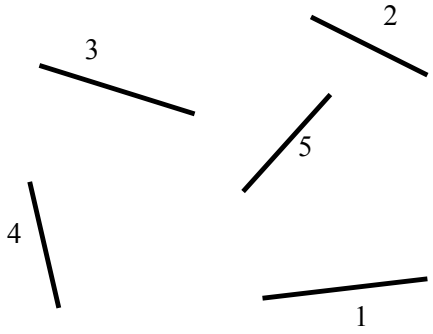
- Ventajas:
  - Permite el manejo de transparencia
- Problema:
  - La determinación del orden entre los elementos

Árboles BSP

- BSP: *Binary partition algorithm*
- Esta técnica subdivide recursivamente los semiplanos en frontales y posteriores
- La generación de un árbol BSP es un pre-proceso
- Este tipo de árbol permite generar imágenes desde cualquier punto de vista

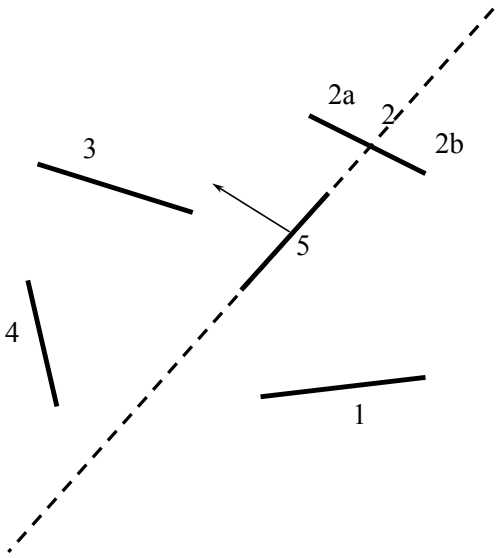
Árboles BSP: Algoritmo

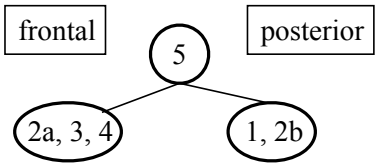
- Elija un polígono (plano) como raíz
- Realice una separación recursiva de planos frontales y posteriores
  - Definido por la normal



Árboles BSP

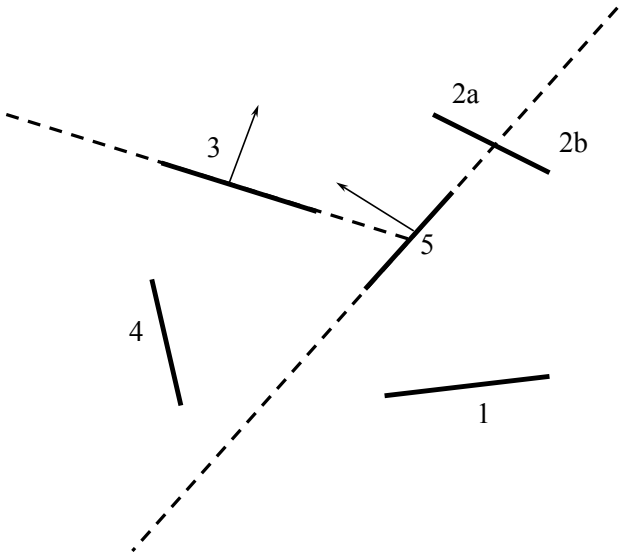
- Ejemplo:

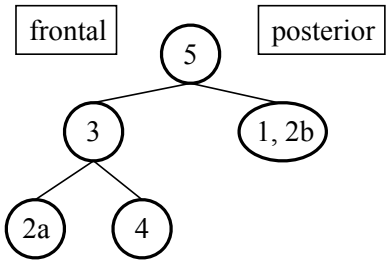




Árboles BSP

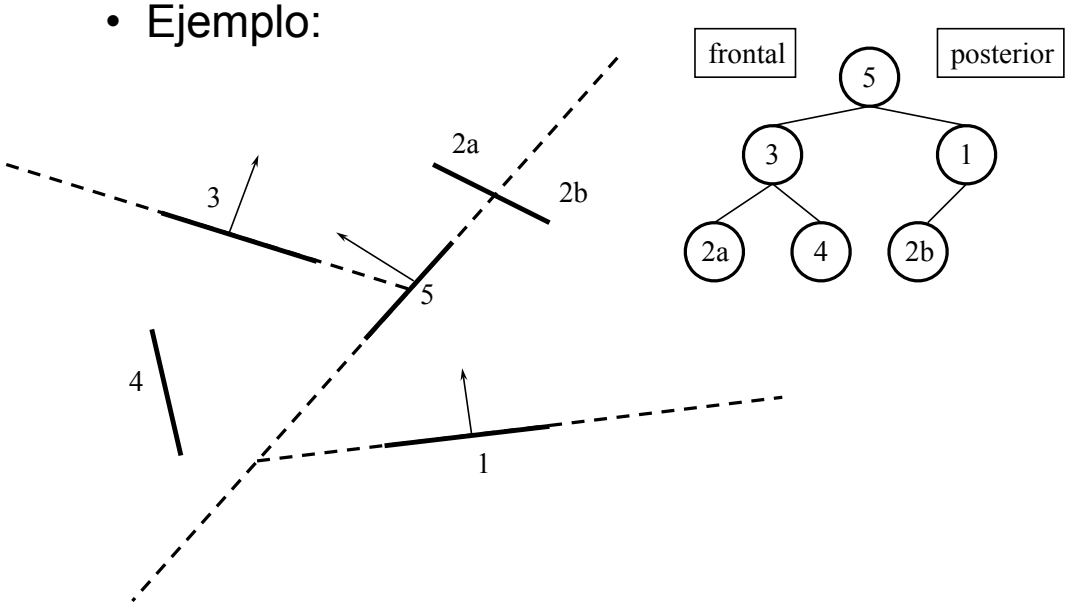
- Ejemplo:





Árboles BSP

- Ejemplo:



Árboles BSP: Rendering

- Se usa el siguiente algoritmo recursivo:
  - Despliegue los polígonos que están detrás de la raíz
    - No pueden ocultar la raíz
  - Despliegue la raíz
  - Despliegue los polígonos que están al frente de la raíz
    - Pueden ocultar la raíz

- Obs: *detrás* y *frente* se determina c/r a la posición del observador
- Applet BSP:

http://symbolcraft.com/graphics/bsp/bsptreedemo_spanish.html

Propuestos

- Propuestos:
  - Investigue el funcionamiento del ordenamiento en Z (generalización del algoritmo del pintor)
  - Investigue otros algoritmos de superficies y líneas escondidas
  - Qué modificación haría al algoritmo del pintor para dibujar, por ejemplo, la siguiente figura: