5 Selección de visibilidad

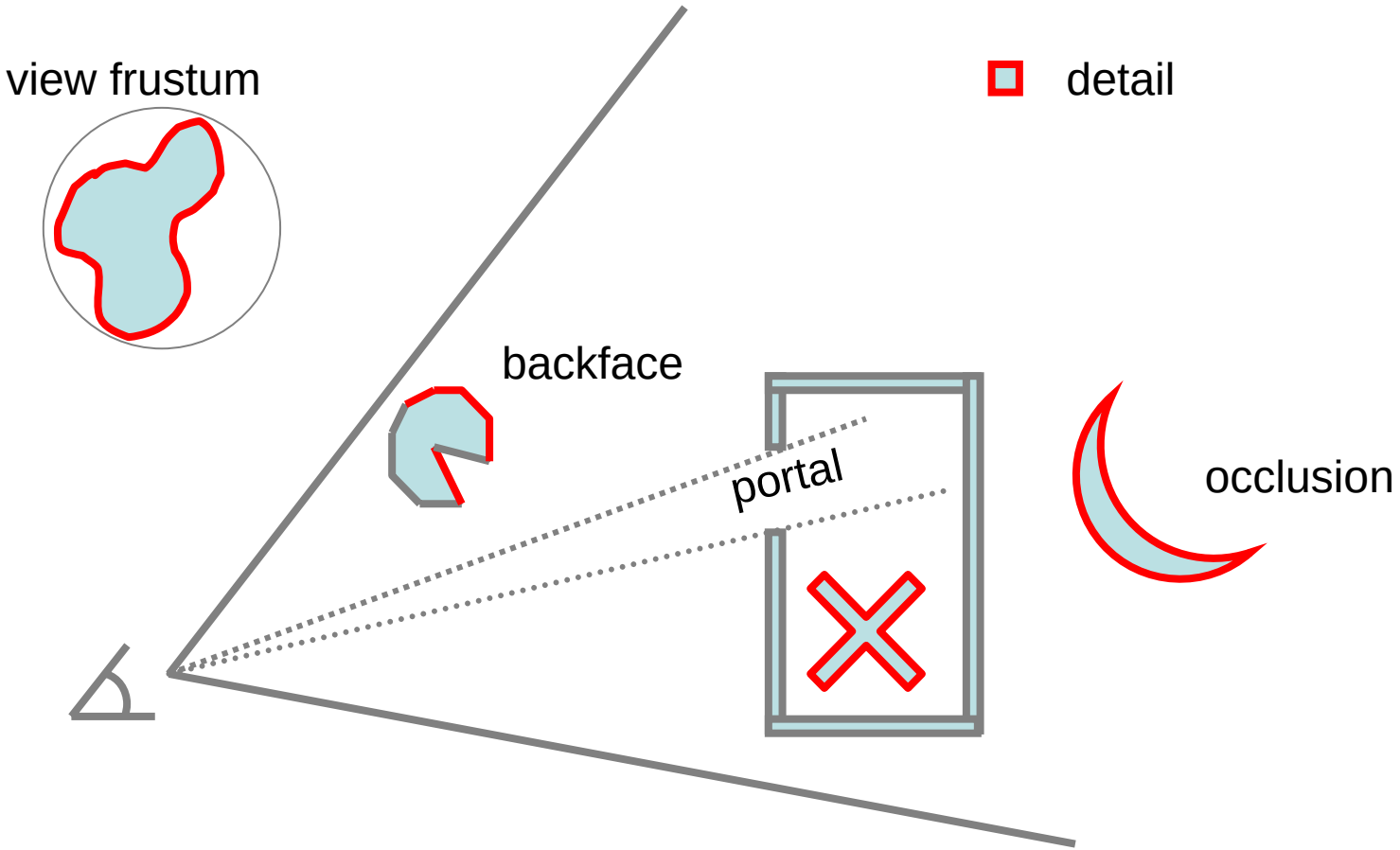
- Técnicas de selección
- Caras traseras
- Pirámide de visualización
- Pequeños detalles
- Oclusión

Técnicas de selección

- “Culling” significa seleccionar de un grupo
  - No procesar nada que no contribuya a la imagen final
  - El grupo es la escena y la selección es el subconjunto de la escena que se supone **NO** contribuye a la imagen final
- Tipos
  - Selección de caras traseras (*Backface culling*)
  - Selección contra la pirámide de visualización (*Hierarchical view-frustum culling*)
  - Selección en entornos de interior (*Portal culling*)
  - Selección según nivel de detalle (*Detail culling*)
  - Selección de objetos ocultos (*Occlusion culling*)

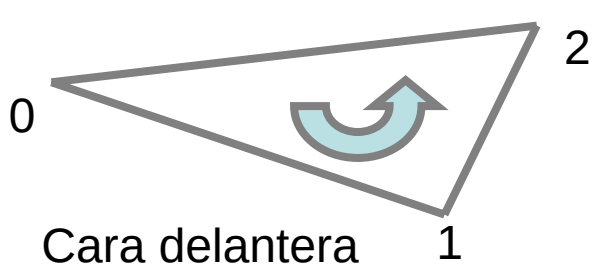
Técnicas de selección

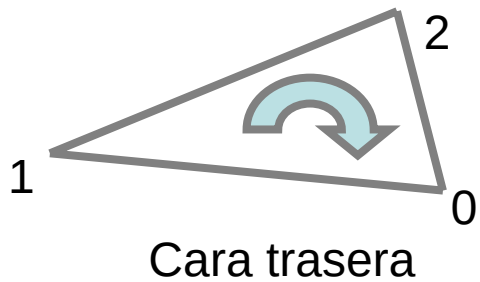
- Tipos



Caras traseras

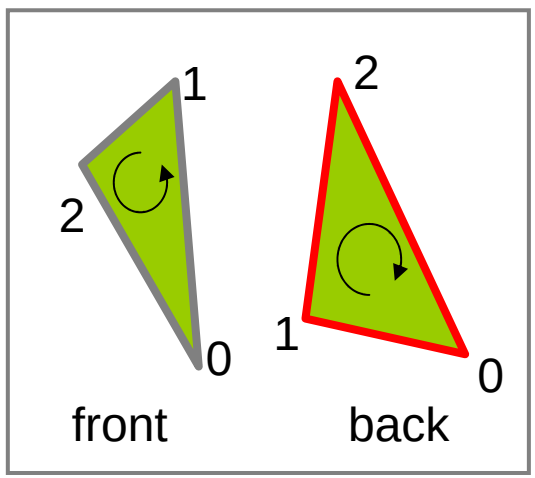
- Selección de caras traseras
  - descartar los polígonos que dan la espalda al observador
  - Puede utilizarse en:
    - Objetos cerrados (ej. Una esfera)
    - En cualquier caso en el que se sepa que las caras traseras no van a visualizarse (ej. Paredes en una habitación)
  - Implementación en el API
    - En OpenGL: `glCullFace (GL_BACK)`
    - Consistencia en la definición de los polígonos



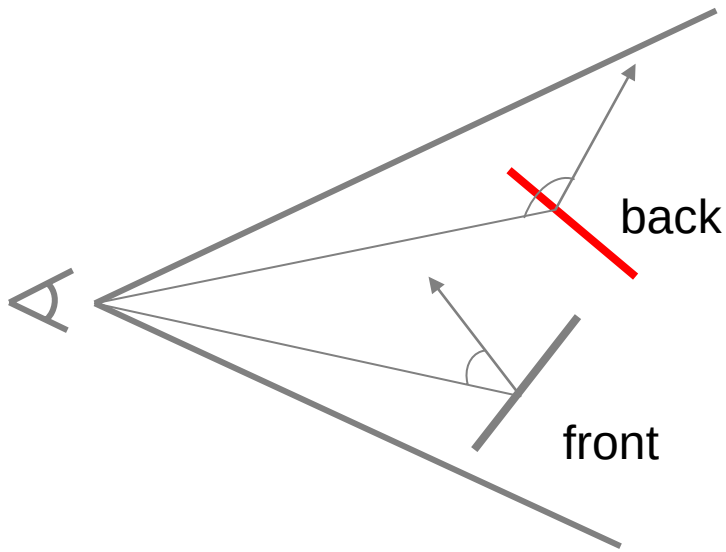


Caras traseras

- Dos métodos:
  - Espacio de la imagen
    - Según la normal del polígono proyectado
    - Según el signo del área del polígono
  - Espacio de la vista
    - Según el producto vectorial del vector de la vista y la normal
    - Según el signo de la distancia



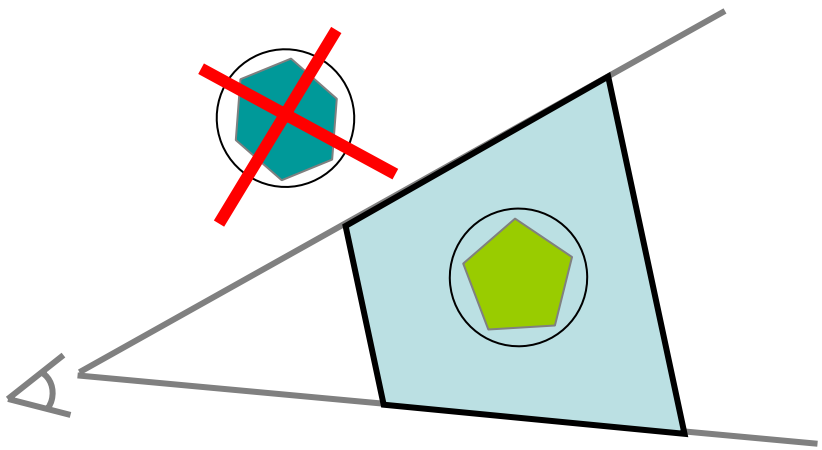
$$Area(P) = \frac{1}{2}\sum_{i=0}^{n-1}\left(x_i y_{i+1} - y_i x_{i+1}\right)$$



Se benefician la fase de geometría o/y la de conversión al raster

Pirámide de visualización

- Selección contra la pirámide de visualización
  - Organizar primitivas en grupos
  - Antes de dibujar las primitivas del grupo, probar su volumen de inclusión contra la pirámide de visualización
    - Si el grupo esta completamente fuera de la pirámide entonces no visualizar ninguna de sus primitivas
    - Si se produce intersección con la pirámide entonces añadir las primitivas al conjunto potencialmente visible y visualizarlas normalmente



Pirámide de visualización

- ¿Cuántas primitivas por grupo?
  - Elegir el tamaño mínimo de manera que:

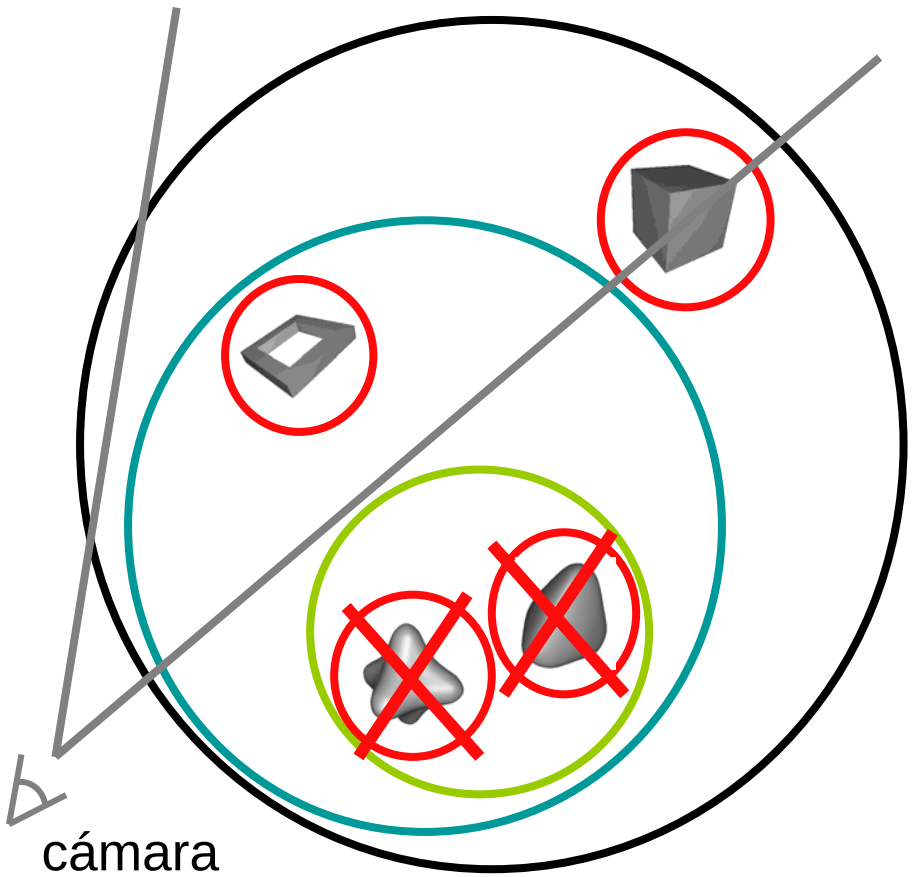
$$\text{coste}_{testBV} << \text{coste}_{clipping\_primitivas}$$

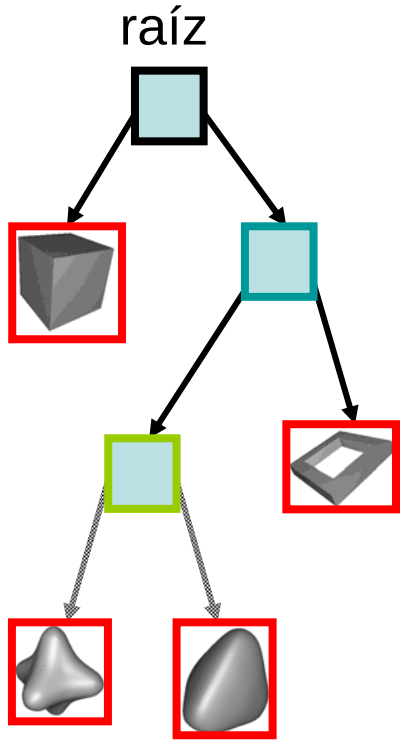
    - Al menos 500 triángulos/grupo en el HW actual
  - Organizar los grupos en una jerarquía para aumentar la eficiencia
    - Si el volumen de un objeto esta fuera de la pirámide entonces no es necesario comprobar sus hijos
    - Pueden utilizarse jerarquía de volúmenes, BSP, Octrees

Se benefician la fase de geometría, la de conversión al raster y la transferencia por el bus

Pirámide de visualización

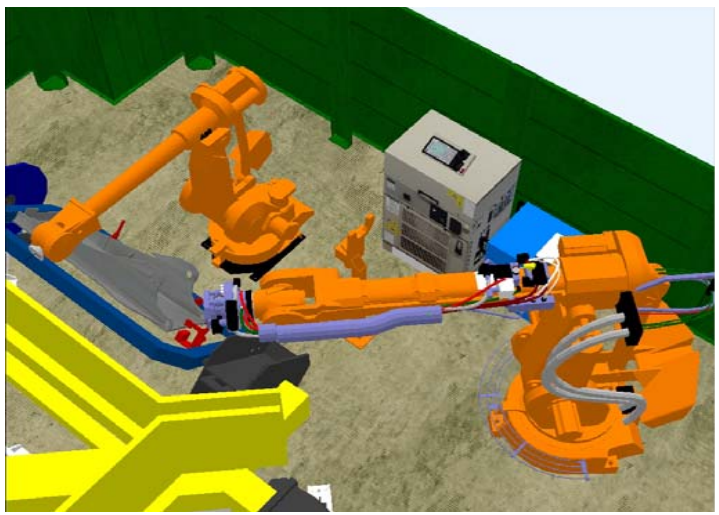
– Ejemplo con jerarquía



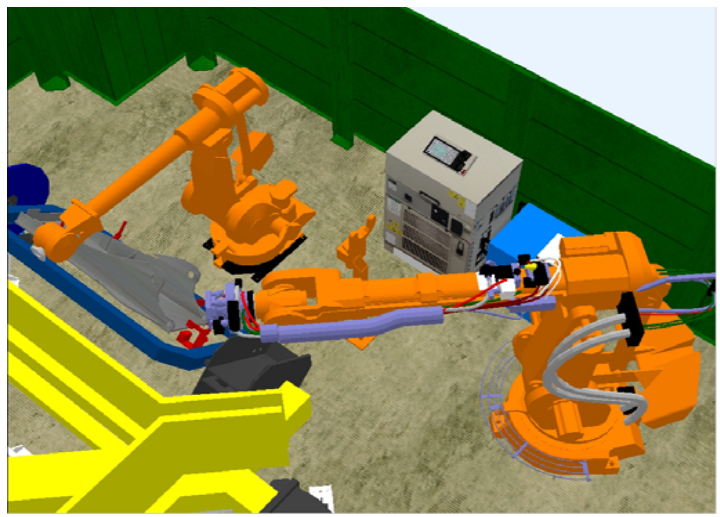


Pequeños detalles

- Selección de detalles
  - Eliminar todos los objetos con una proyección de su volumen de inclusión menor que N píxeles
  - 80-400% más rápido
  - Puede utilizarse jerarquía



Eliminación de detalles OFF

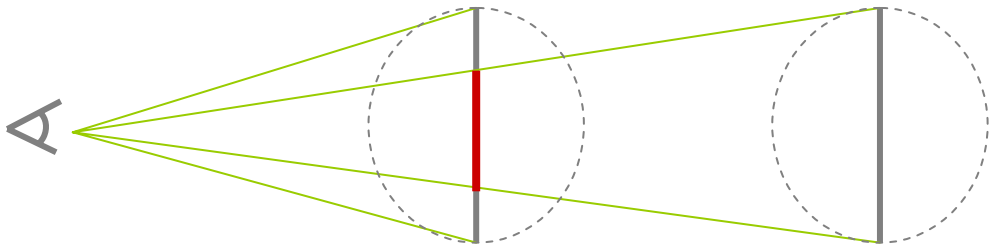


Eliminación de detalles ON

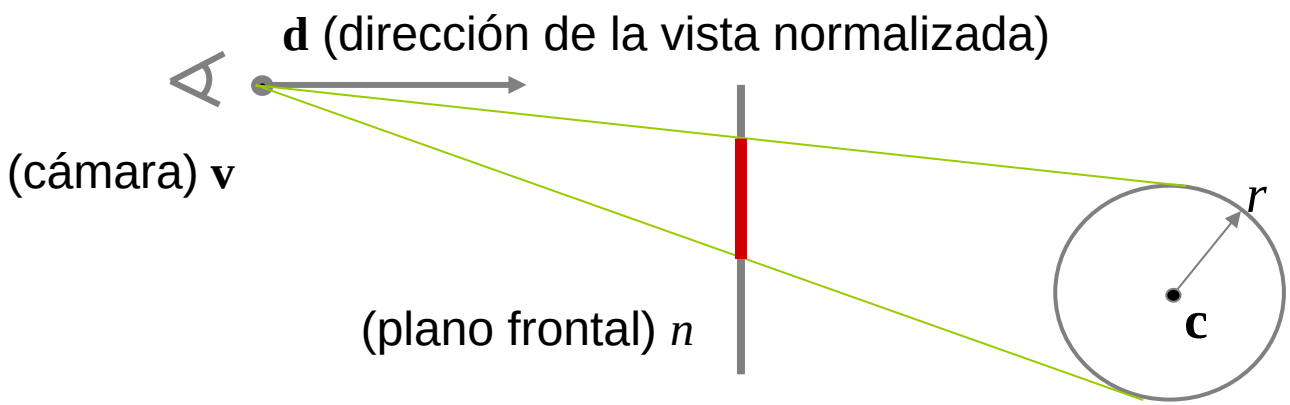
Se benefician la fase de geometría, la de conversión al raster y la transferencia por el bus

Pequeños detalles

- La proyección es la mitad cuando la distancia es el doble

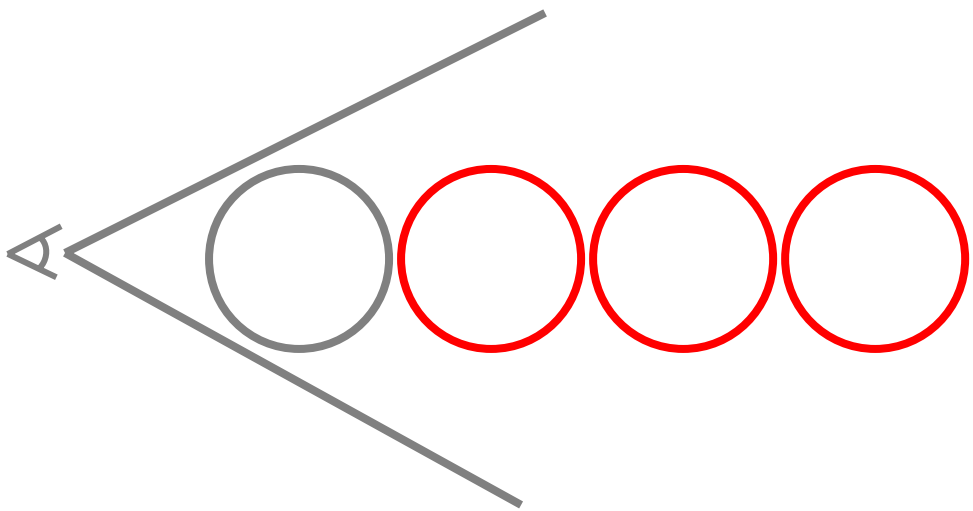


- Distancia sobre d = dot(d, (c-v))
- Estimación del la proyección del radio p = nr/dot(d, (c-v))
- Área proyectada = πp2



Oclusión

- Idea principal
  - Eliminación de objetos que están ocultos por otros
  - El Z-buffer ofrece problemas en el caso de escenas de mucha complejidad en profundidad
  - Problema de difícil solución
  - Los portales son un tipo de algoritmo de oclusión pero no sirven para todo tipo de escenas
    - p.e.: árboles en un bosque, multitud en una estación de tren



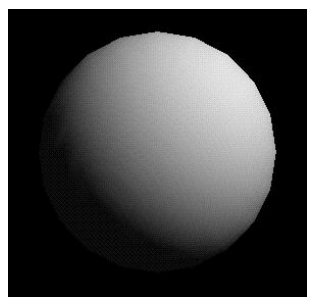
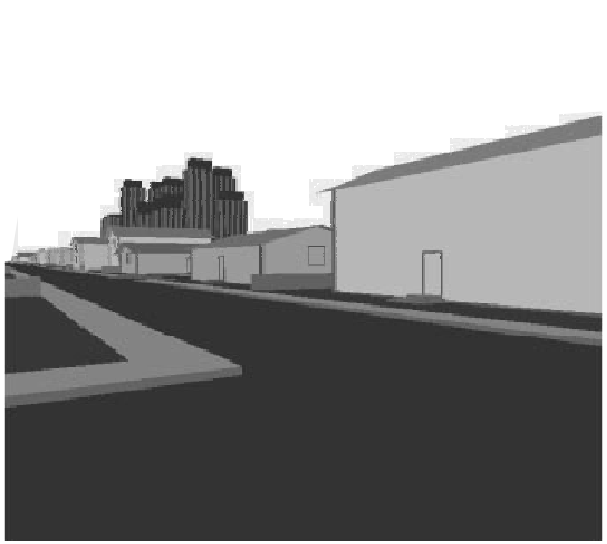


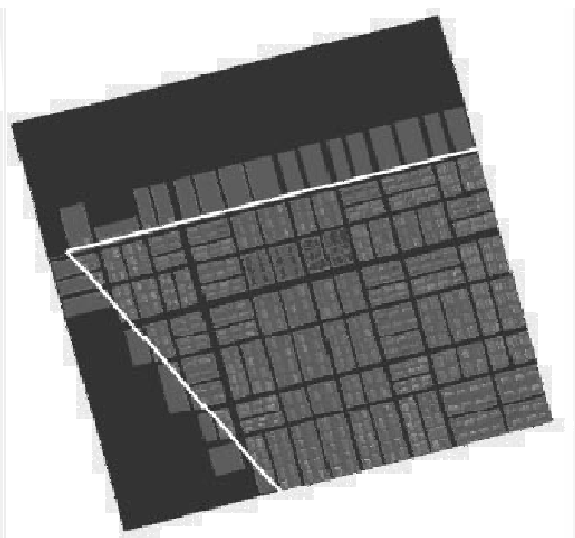
Imagen final

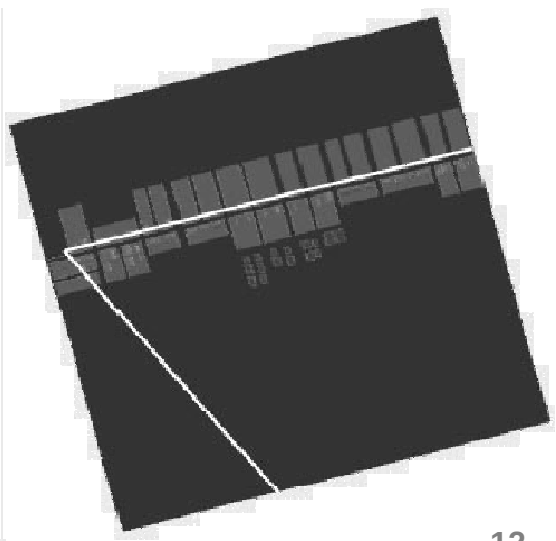
Oclusión

- Algoritmo
  - G: datos de entrada
  - O: oclusor
  - Hay que definir
    - Algoritmo para la función *estaOculto()*
    - Definición de O
    - Actualización rápida de O

```
Oclusión (G)
    O = vacío
    para cada objeto g en G
        si (estaOculto(g,O))
            eliminar g
        sino
            visualizar (g)
            actualizar (O)
```





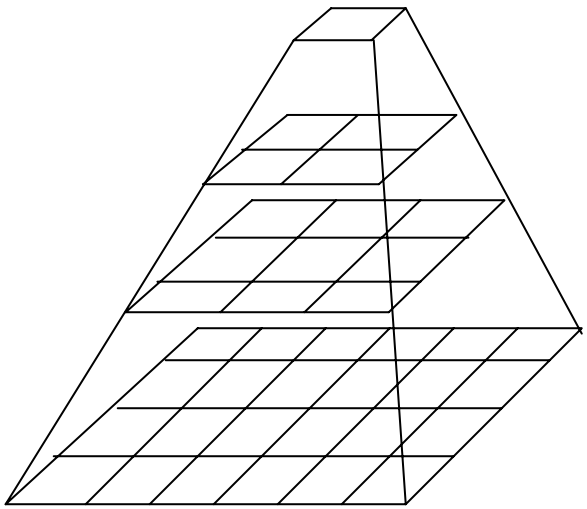


Oclusión

- Ejemplo. Octrees
  - Las primitivas de un nodo del octree están ocultas si el nodo (cubo) lo esta
  - Un cubo del octree esta oculto si sus 6 caras lo están
  - Test de visibilidad
    - Desde la raíz del octree
    - Selección contra la pirámide de visualización (View-frustum culling)
    - Conversión al raster de las 6 caras con Z-buffer
    - Si las 6 caras están ocultas, descartar el nodo y sus hijos
    - Sino, recorrer los hijos de forma recursiva
  - La conversión al raster de las caras del octree puede ser costosa.
  - Objetivo. Concluir rápidamente si un polígono grande está oculto
  - Solución: Z-buffer jerárquico

Oclusión

- Z-buffer jerárquico
  - Aproximación del espacio de la imagen

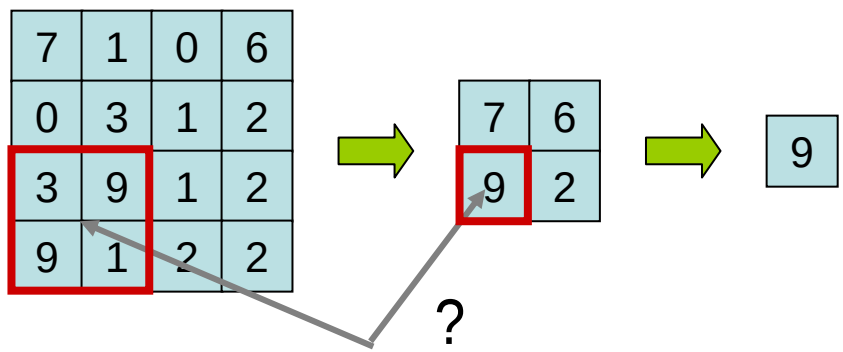


1 valor

¼ resolución

½ resolución

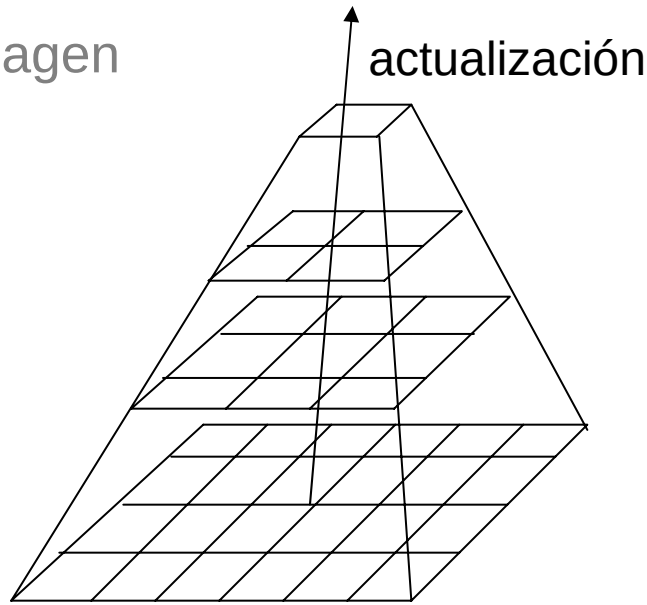
Z-buffer original



Oclusión

- Z-buffer jerárquico
  - Aproximación del espacio de la imagen

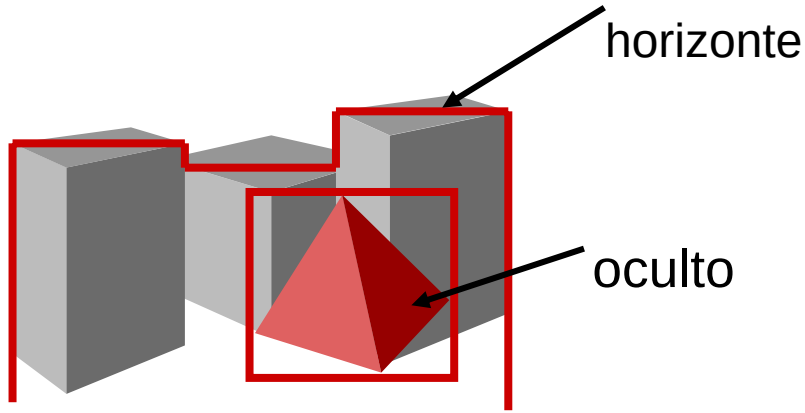
```
estaOculto(g, Zp)
   z_proxima= zProxima(BV(g))
   si (z_proxima detras Zp_raiz.z)
      return true
   sino
      return (estaOculto(g,Zp.c[0]) &&
                estaOculto(g,Zp.c[1]) &&
                estaOculto(g,Zp.c[2]) &&
                estaOculto(g,Zp.c[3]))
```

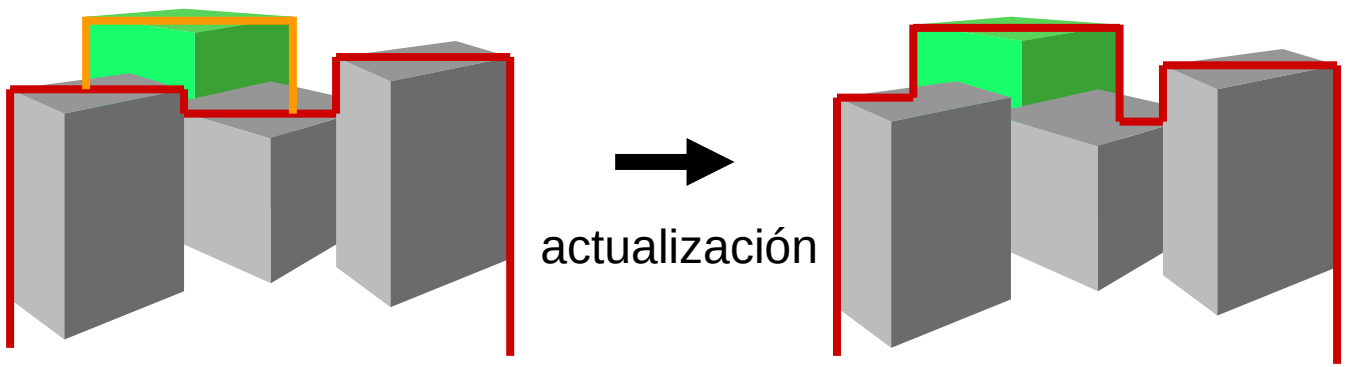



```
Elimina_o_Dibuja(NodoOctree N)
    si (estaOculto(N, Zp) then return;
    para cada primitiva p en N
       visualiza y actualiza Zp
    para cada hijo C de N en orden “front-to-back”
       Elimina_o_Dibuja ( C )
```

Oclusión

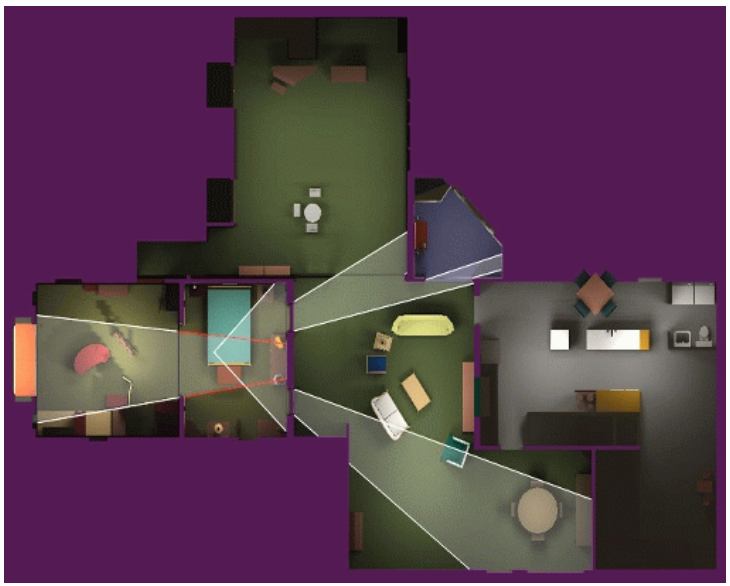
- Ejemplo. Horizontes de oclusión
  - Procesar de delante hacia atrás utilizando un quadtree
  - Mantener un horizonte de oclusión
  - Calcular objetos visibles
    - Horizonte y volumen de inclusión (o proyección)
  - Añadir objetos visibles al horizonte

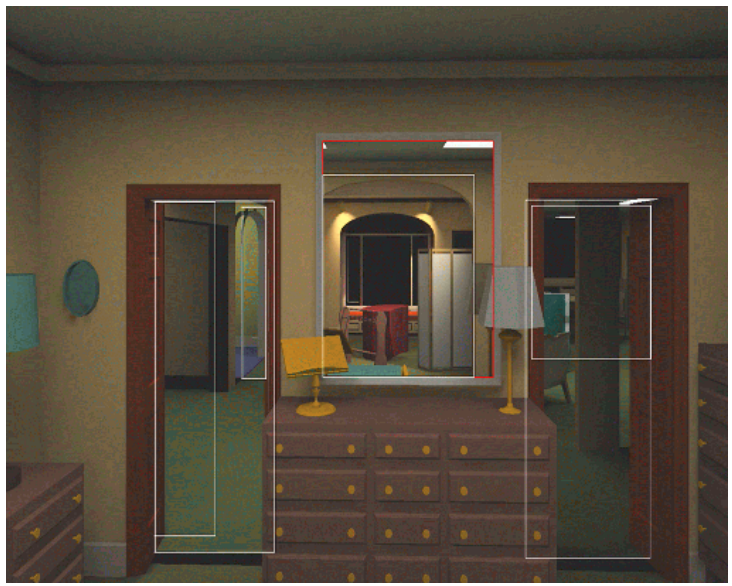




Oclusión

- Entornos de interior (*Portal culling*)
  - Dividir la escena en celdas que representan: habitaciones, pasillos, …
  - Portales transparentes conectan las celdas: entradas, ventanas, …

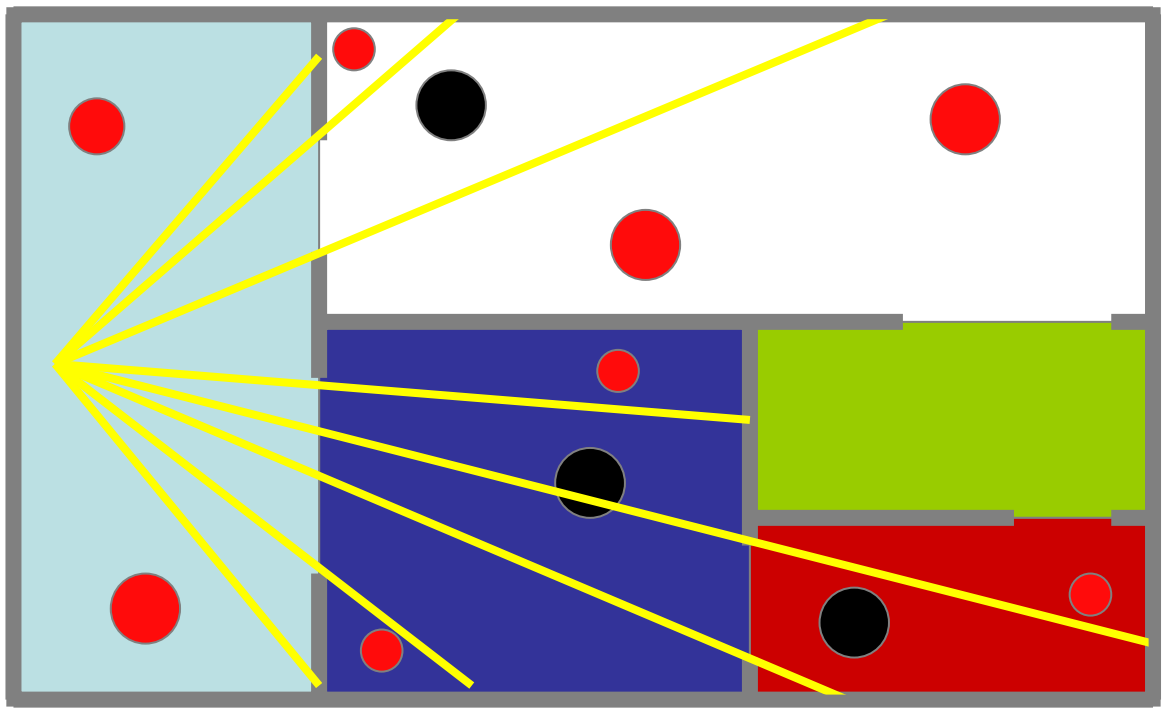




  - Eliminación de entre un 20-50% de los polígonos en la vista
  - Hasta 10 veces más rápido

Oclusión

- Ejemplo de portales
  - Los círculos son los objetos a dibujar

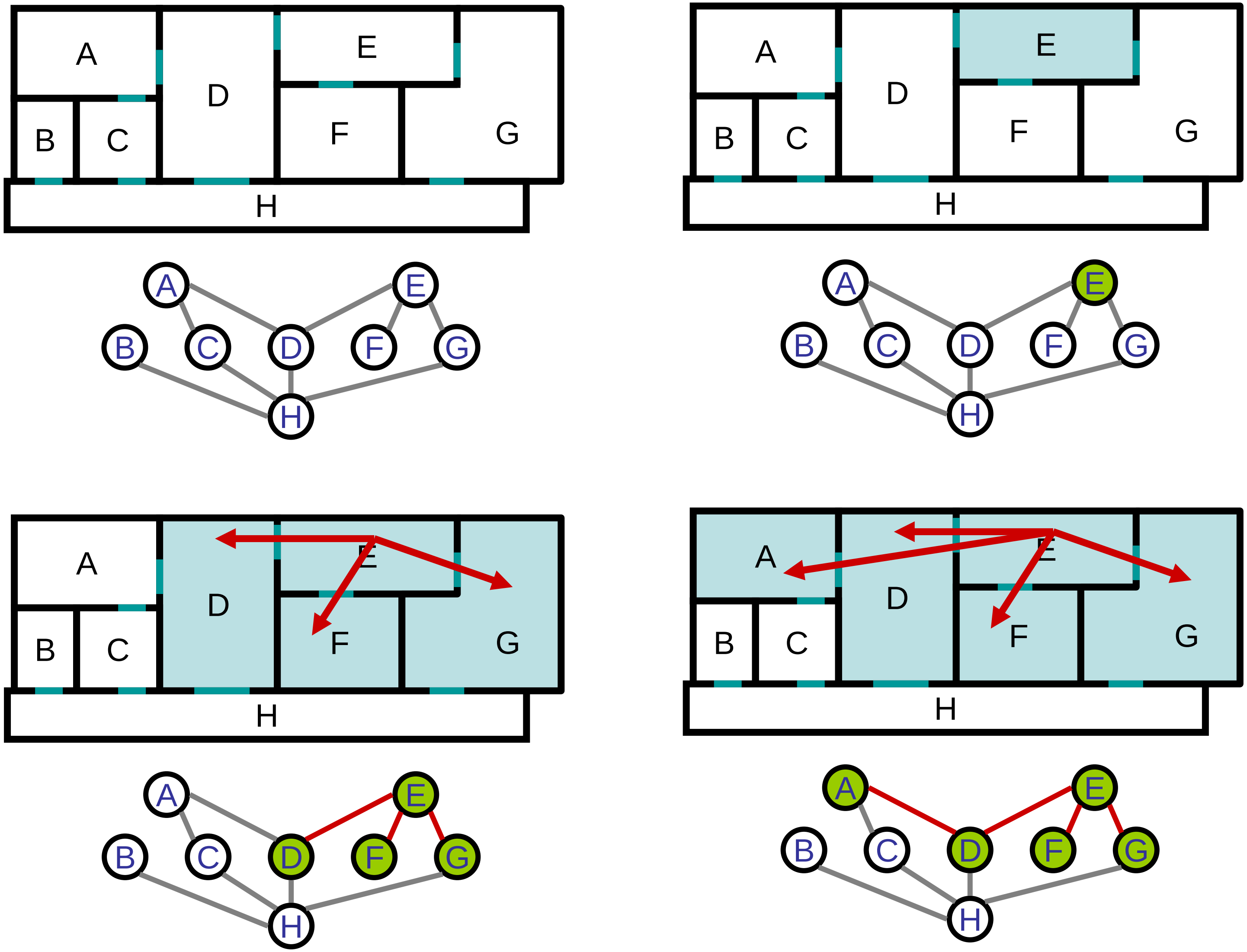


Se benefician la fase de geometría, la de conversión al raster y la transferencia por el bus

Oclusión

- Algoritmo (dependiente de la vista)
  - Dividir la escena en celdas con portales (construir un grafo)
  - Proceso
    - Localizar la celda V donde esta el observador
    - Inicializar una caja de inclusión P al rectángulo de la pantalla en 2D
    - Visualizar la geometría de la celda V sobre la pirámide de visualización (desde la cámara hasta el polígono P)
    - Recursividad sobre los portales de las celdas vecinas de V
      - Para cada portal de la celda actual calcular caja de su proyección (AABB)
      - Calcular la intersección entre P y las proyecciones anteriores
      - Para cada intersección
        - » Si es vacía, la celda no se ve desde la posición actual
        - » Sino, el contenido de la celda vecina se visualiza sobre la pirámide (desde la cámara al polígono de la intersección). Repetir recursividad

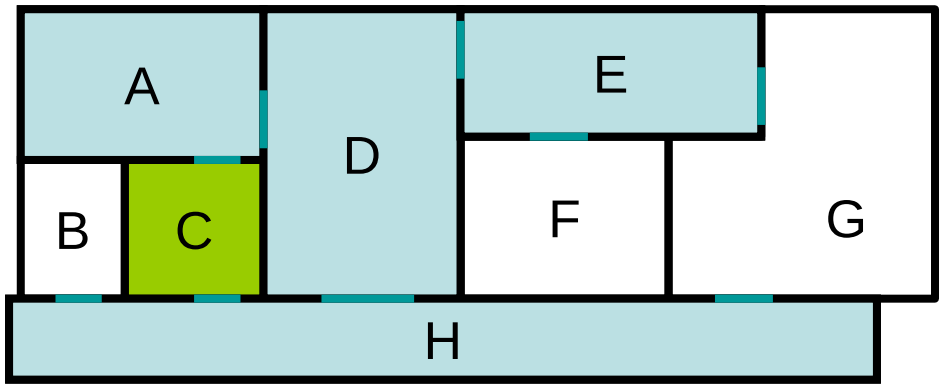
Oclusión



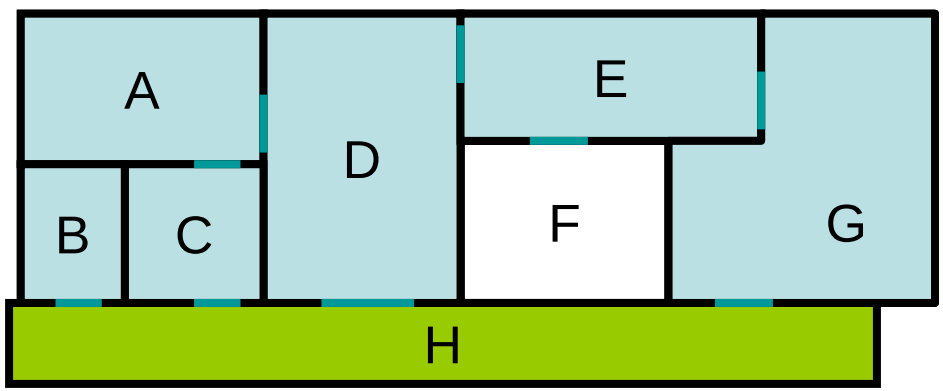
Oclusión

- Algoritmo (independiente de la vista)
  - Precálculo de la visibilidad desde el árbol BSP de la escena
  - Proceso
    - Construir el árbol BSP, almacenado todos los polígonos en los nodos hojas del árbol
    - Determinar desde que nodos hoja se ven otros nodos hoja. Creación de una matriz 2D de visibilidad
    - La matriz 2D de visibilidad, se utiliza para seleccionar las zonas del árbol BSP que se ven desde la hoja actual
  - Crear el conjunto potencialmente visible es una tarea difícil
    - De alto coste computacional
    - Existen varias formas de crearlos
      - Selección de puntos de vista aleatorios en la escena. Calculo de la visibilidad en las seis vistas ortogonales. Ej. Juego simple con 9.900 triángulos y un BSP de 1500 hojas -> 50.000 puntos de vista

Oclusión



C sólo puede verse desde A, D, E y H



Desde H nunca se verá F