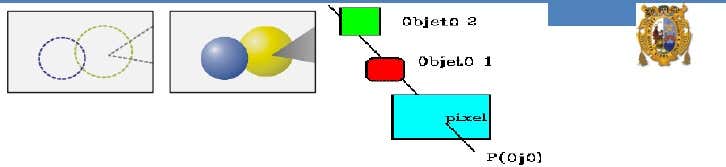
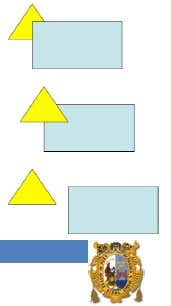
OCULTAMIENTO

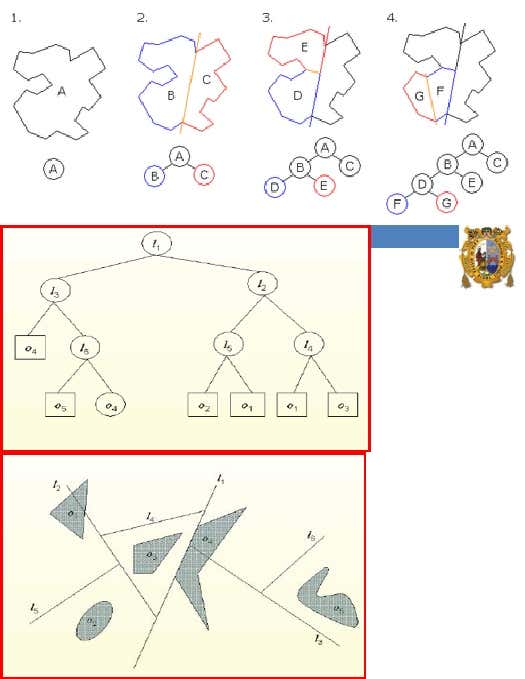
* Clasificación de algoritmos de ocultamiento.
  + Precisión a la imagen:
    - Determinar cuál de los objetos “n” en la imagen es visible en cada pixel.
    - Se realiza en el sistema de coordenadas de la imagen.
    - Complejidad: p\*n, “p” es el número de pixeles.
    - Depende del dispositivo utilizado y el tamaño de la ventana, al cambiar la cantidad de pixeles se rehacen los cálculos.
    - Algoritmos: Z Buffer, Warnock, Scan-Line, Back-Face Culling.



* + Precisión al objeto:
    - Comparar los objetos con los demás elementos y eliminar las partes no visibles.
    - Se realizan los cálculos en el sistema de coordenadas del objeto.
    - Complejidad: .
    - Los algoritmos son más difíciles de implementar.
    - No se considera la resolución de la pantalla para los cálculos.
    - Algoritmos: Algoritmo del pintor, Algoritmo utilizando el arbolo BSP.



* Algoritmos de ocultamiento de superficies: Estos mejoran el rendimiento al eliminar u ocultar partes de objetos no visibles.
  + Back-Face Culling: Eliminar superficies no visibles de un objeto.
    - Se usa como pre filtrado.
    - Cada cara define un plano que se divide en el semi-espacio “in” (Parte del objeto en contacto con el exterior, cara detrás del centro de proyección) y el semi-espacio “out” (Cara delante del centro de proyección).
    - Si se indica que la cara está detrás no será visible, pero si la cara esta adelante, puede que este oculta si el objeto no es convexo o si tiene otro objeto por delante.
  + Warnock: Busca áreas triangulares que tengan color uniforme.
    - Es más complejo que el Z Buffer, pero no requiere tanta memoria.
    - Se usa mayormente con caras triangulares.
  + Z Buffer: Cada vez que se realiza el renderizado, verifica que no se haya dibujado antes en la posición del pixel más cercano respecto a la cámara.
    - Se debe especificar los puntos que van a ser dibujados.
    - Se registra que se dibuja en cada pixel del puerto de visualización.
    - Presenta inconvenientes de tiempo y memoria, se requieren 2 áreas de memoria para implementar la profundidad (Valores de z en cada x, y) y la renovación (Valores de intensidad en cada posición).
    - A pesar de ser un sistema que siempre demuestra los mejores resultados, la gran cantidad de memoria y la dificultad de programación generan lentitud en el sistema (hablando de tiempos de procesamiento de minutos hasta horas en el peor caso).
  + Algoritmo del pintor: Hace referencia a un pintor que primero dibuja los objetos a lo lejos y después los cubre con lo más cercano.
    - Al proyectar una escena de tres dimensiones es necesario determinar que polígonos son visibles y cuáles no.
    - Este algoritmo ordena todos los polígonos de la escena en función de la profundidad y después los pinta en el mismo orden, pintando encima de las partes no visibles.
    - Los polígonos superpuestos pueden provocar que el algoritmo falle.
    - En las implementaciones básicas este algoritmo resulta poco eficiente ya que se renderiza cada punto de todos lo polígonos visibles, incluyendo los ocultos, al final se consumen más recursos de los necesarios.
  + Algoritmo utilizando el arbolo BSP (Partición binaria del espacio): Método para subdividir recursivamente un espacio en elementos convexos con hiperplanos.
    - Sirve como base para otros algoritmos, como listas de visibilidad.
    - Necesita un pre-procesamiento de la escena, lo que hace ineficiente insertar objetos móviles.
    - Se emplean normalmente en videojuegos.
    - Divide la escena en dos hasta satisfacer uno o más requisitos.



* + - Mantiene la información necesaria en la memoria para poder retroceder en caso de que una rama no produzca un resultado satisfactorio.
    - Toma mucho tiempo de computación.