TEMA 2: HARDWARE GRÁFICO

1. Dispositivos característicos

Entrada

Teclado, ratón, digitalizadores, tableta digitalizadora…

Salida

1. Terminales vectoriales (de barrido aleatorio)

- líneas componentes de una imagen se dibujan y refrescan en cualquier orden
- velocidad refresco depende nº líneas que deba mostrar, si no es suficiente -> **parpadeo**
- imagen almacenada como conjuntos de órdenes de dibujo en zona memoria (lista de visualización, archivo de refresco de visualización, archivo vectorial, programa de visualización).
- diseñados para apps de dibujo de líneas, no son capaces de mostrar escenas con matices realistas.

2. Terminales de barrido

- pantalla compuesta por **píxeles** (puntos iluminables)
- rayo recorre pantalla de izq a derecha y de arriba a abajo
- imagen almacenada en **memoria de vídeo o refresco: Frame-buffer**. Se almacena el color de cada píxel y se utiliza para refrescar la imagen en la pantalla.
- frame-buffer y píxeles se recorren simultáneamente convirtiendo la codificación digital del color en intensidades de los rayos.
- frecuencia de barrido constante: ventaja
- efecto escalera (**aliasing**): desventaja
- aliasing: es el efecto que se produce al discretizar entidades continuas, por ejemplo al dibujar una recta en una pantalla formada por píxeles, se puede disimular utilizando técnicas de antialiasing
- **Resolución pantalla:** dimensiones en píxels: altura-filas y anchura-columnas
- **Posibilidad de color:** n bits por píxel -> 2^n colores (resolución o prof color) -> máx nº colores representables en pantalla
- Cálculo **memoria de vídeo:** memoria (en Bytes) = filas x columnas x bits_por_pixel / 8

3. Terminal Gráfica: monitor + controlador + memoria
4. Tubos de rayos catódicos (CRT):
   - cañón emite electrones (cátodo)
   - intensidad del flujo controlada x rejilla
   - enfoque hace que electrones tengan trayectoria convergente (rayo)
   - deflexión hace apuntar el rayo a un punto de la pantalla
   - rayo impacta contra **fósforo** que emite luz
   - emisión fósforo decae rápidamente (refresco)
5. Monitores de Cristal Líquido (LCD):
   - cristales líquidos exhiben propiedades de líquidos y sólidos
   - luz pasa a través LC sigue alineamiento de sus moléculas
   - al aplicar voltaje -> cambio alineamiento y en la forma en la que la luz atraviesa
   - **displays** se forman como paneles de LC -entre dos filtros polarizadores- y una luz trasera -cátodo frío-
   - **TFT** (Thin Film Transistor): añade matriz de transistores -> refresco más rápido, reduce ghosting, mayor intensidad de contraste.

     - **VENTAJAS** vs CRT: menor consumo, dispositivos más pequeños.
     - **DESVENTAJAS** vs CRT: resolución óptima única -nativa-, menor velocidad de menor contraste, menor ángulo de visión, píxeles muertos.

   - Diodos LED en la actualidad, antes tubos fluorescentes
   - **LED** continuamente activa -> para color negro bloquea los tres componentes -> negro puro
   - **OLED**: no disponen de un panel iluminado encendido permanentemente como fuente de luz -> cada uno de los colores primarios RGB que forman cada píxel emite luz por sí mismo.
   - **OLED**: píxeles están formados por una película de compuestos orgánicos con propiedades electroluminiscentes -> al hacer pasar corriente reaccionan y se iluminan -> ahorro fuente de iluminación.

     VENTAJAS: pantallas son más finas y más ligeras, más baratas, un consumo menor, un mejor contraste con negros auténticos, un contraste al menos 100 veces mayor que el de una LCD, el encendido y apagado de cada píxel es casi instantáneo, esto favorece la visualización de imágenes en movimiento

6. Estéreo 3D

Visión estéreo se basa en la utilización de **dos imágenes de la misma escena**, tomadas desde dos puntos de vista ligeramente distintos. Dispositivos estéreo deben proporcionar una imagen diferente a cada ojo.

Auto-estereoscópicos: no requieren gafas, pueden percibir imagen 3D varios usuarios.

- Lentes lenticulares
- Barrera de paralaje
- Dispositivos volumétricos

Estéreos: mayoría dispositivos.

Gafas activas: se comunican con el televisor y bloquean la visión de uno u otro ojo según la imagen se muestra. El televisor debe trabajar al doble de frecuencia de refresco. Se necesitan tiempos de respuesta más pequeños.

- **Mayor resolución**, cada imagen resolución máxima del panel
- Panel más brillante -> gafas oscurecen imagen
- Velocidad de refresco no buena -> ***Cristal*** (imágenes fantasmas) y ***Flickr*** (notar parpadeo)
- Cansa más, más caras, pesan más

Gafas pasivas: anaglifo, gafas polarizadas… Televisor mostrará dos imágenes a la vez: líneas pares muestran lo q ve un ojo e impares lo q ve el otro. Verticalmente u horizontalmente. Gafas tienen filtro polarizado que deja pasar solo **líneas impares o pares**. *Dual Play*: dos personas ven contenidos distintos: gafas con ambos cristales idénticos. En proyección -> dos proyectores simultáneamente.

- Más económicas, cómodas, reducen fatiga visual
- Más ángulo de visión, usar tumbado
- Dual Player
- Menos resolución

2. Dispositivos especiales

- **Hápticos**: tecnología que proporciona retroalimentación táctil o sensitiva a los usuarios a través de dispositivos electrónicos.

  Kinestética -> **retroalimentación de fuerza**: interacción con músculos y tendones
  Retroalimentación táctil -> nervios terminales de la piel, calor, presión, textura…

- **Realidad virtual y aumentada:** simulación con dispositivos de interacción especiales:
  Entrada: ratones 3D, guantes virtuales…
  Salida: sonido 3D, cascos, sistemas de proyección…

Realidad aumentada: parte de imágenes y/o vídeo del mundo real, añade info gráfica y textual en tiempo real.

Elementos:

- Navegación: **6 grados de libertad:** x,y,z, *pitch* (cabeceo), *roll* (balanceo), *yaw* (guiñada o arfada).
- Detectores de posición y orientación: tasa de actualización, resolución, exactitud. Tipos: electromagnéticos, ópticos, inerciales… **GPS**.
- Dispositivos de interacción y manipulación: ratón 3D, palanca de mando, guante, volantes…
- Cascos: **HMD** (*Head Mounted Display*) -> visualizan estéreo; tipos: pantallas LCD, tubos de rayos catódicos, columna de LEDs y espejo. **BOOM** (*Binocular Omni-Orientation Monitor*).
- Gafas o cascos *optical see-through*: combinan la vista del mundo real con imagen sobre un LCD.
- *CAVE*: *Cave Automatic Virtual Environment*.

TEMA 3: PRIMITIVAS GRÁFICAS

1. Primitivas Gráficas

Una librería gráfica (*Computer Graphics Application Programming Interface CG API*) proporciona las funciones necesarias para dibujar.

Permiten describir la forma de los componentes de la escena (geometría: puntos, líneas, círculos, cónicas, superficies cuádricas, curvas y superficies Splines, áreas y polígonos).

El **aspecto** de las primitivas se define mediante sus **atributos**.

Se referencian utilizando un **sistema de coordenadas:** en monitor, coordenadas de pantalla; un píxel se referencia por su esquina inferior izquierda.

- líneas: referenciadas por sus puntos extremos.
- conjunto de líneas: líneas no conectadas
- polilíneas: líneas conectadas pero no cierran
- polilínea cerrada: cierran
- curvas: se aproximan mediant polilíneas
- áreas rellenas: conjuntos de píxeles conectados con un mismo color o patrón. Para definir superficies y objetos sólidos, se pueden especificar mediante polígonos.
- texto: mediante patrón *bitmap* vectorial

2. Atributos

Un atributo es un parámetro que afecta al aspecto que va a tener una primitiva al dibujarse (color, tamaño).

Siempre tienen que tener un **valor por defecto.**

En una llibrería se activan mediante **variables de estado.**

3. Algoritmos Líneas

Calcular las coordenadas de los píxeles que representan una recta infinitamente delgada colocada sobre la malla de un raster 2D. Se asume que las rectas son continuas, color constante, independientes de orientación y longitud, se han de dibujar tan rápido como sea posible y no se consideran rectas con atributos.

Ambos algoritmos dibujan, generalmente, los mismos píxeles.

1. Fuerza bruta

Para convertir rectas al raster: **y = m*x + b** donde **m = dy / dx**

Restricción: m debe estar entre -1 y 1.
Soluciones a la restricción: calcular x en función de incrementos unitarios de y para valores de m que no se encuentren entre -1 y 1 (pendiente 1/m).

```
m=(y1-y0)/(x1-x0)
b=y1-x1·m
Para x desde x0 hasta x1
(increm/decrem unitarios)
  y = x·m + b
  dibuja_pixel(x, round(y))
finPara
```

Desventajas: ineficiente, cada iteración requiere una multiplicación en coma flotante y una invocación a round().

2. Algoritmo del Punto Medio (Bresenham)

Utiliza únicamente operaciones con **enteros**.
Ventaja: fácilmente aplicable a cuaqluier tipo de elemento geométrico.
Restricción: inclinación de la recta m se ha de encontrar dentro del intervalo [0,1]. Esto permite afirmar que si (Xk, Yk) es el píxel dibujado en la etapa k, el siguiente se encontrará más cerca de la recta será (Xk + 1, Yk) o (Xk+1, Yk+1). Al parecer esta restricción es un supuesto, el algoritmo es generalizable para cualquier tipo de recta.

F(x, y) = dy * x - dx * y + b * dx

Si F(x,y) = 0 -> punto (x,y) está en la recta
Si F(x,y) > 0 -> punto debajo de la recta y
Si F(x,y) < 0 -> punto encima de la recta real

Para elegir el punto hay que calcular F(Xk + 1, Yk + 1/2) y analizar el signo de la función.

El algoritmo tendría esta forma:

```
Introducir los puntos extremos de una recta
(x0, y0) = punto más a la izquierda de la recta
y=y0
d=2dy - dx
Para cada x entre x0 y x1 (incrementos unitarios)
    dibuja_pixel(x, y)
    Si d < 0
        d= d + 2dy
    sino
        y=y + 1
        d=d + 2dy - 2dx
    finSi
finPara
```

4. Algoritmos Circunferencias

1. Fuerza bruta

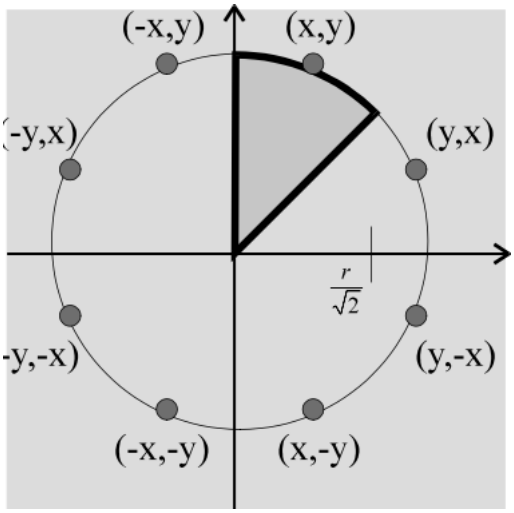
Se utiliza la ecuación de la circunferencia: **(X - Xc)^2 + (Y - Yc)^2 = r^2**
Círculo sobre el origen de coordenadas y es de radio r: **y = √ r^2 - x^2**
Desventajas: operaciones en coma flotante, distancia entre puntos de la circunferencia no es homogénea, se pueden utilizar coordenadas polares y trazar líneas entre la secuencia de puntos generados.

```
Para cada x desde -r hasta r
(incrementos unitarios)
    y1 = +√(r^2 - x^2)
    y2 = -√(r^2 - x^2)
    dibuja_pixel(x, round(y1))
    dibuja_pixel(x, round(y2))
finPara
```

2. Simetría de ocho puntos

Mejora el proceso de dibujo anterior mediante la utilización de la simetría del círculo.

Dado un punto (x,y) de un círculo centrado en el origen de coordenadas, a partir de dicho punto se pueden calcular los otros siete restantes (basta con los 45° sombreados).



Incorporación en el algoritmo de fuerza bruta: hacer que x varíe entre 0 y r/√2.

Solo se calculan las y positivas y se llama a puntos_circulo() en lugar de a dibuja_pixel() -> se eliminan las discontinuidades del algoritmo de fuerza bruta pero se siguen haciendo cálculos en coma flotante.

3. Algoritmo del Punto Medio

Aritmética entera, basada en la simetría del círculo (8 puntos)

Dadas las características del segundo octante, si el último punto que se ha convertido al raster es (Xk, Yk), el siguiente será (Xk+1,Yk) o (Xk + 1, Yk - 1).

Para determinar cuál de los dos puntos se encuentra más cerca del círculo real -> mira**r punto medio** entre ambos -> si está por encima se elige (Xk + 1, Yk - 1), si está por debajo, (Xk+1,Yk).

Posición punto medio -> F(x , y) = x^2 + y^2 - r^2

Se aplica al segundo octante y por cada píxel calculado se dibujan 8, por simetría .

5. Algoritmos Regiones

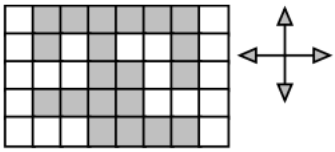
Una región es un **conjunto de píxeles conectados**.

Los píxeles de una misma región se definen por:

- tipo de conectividad:

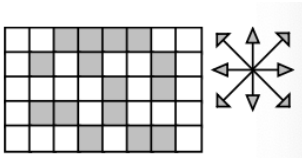
  Regiones 4-Conectadas: dos píxeles se consideran conectados si son adyacentes en dirección horizontal o vertical, pero no en diagonal.

Esto significa que la conectividad se limita a los **píxeles vecinos directos**, creando una cuadrícula en la que la conexión se da en cuatro direcciones posibles: arriba, abajo, izquierda y derecha.



4-CONECTADA

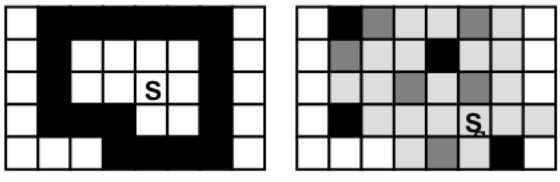
Regiones 8-Conectadas: dos píxeles se consideran conectados si son adyacentes en cualquier dirección: horizontal, vertical o diagonal.



8-CONECTADA

- color:

Definida por su interior: píxeles de un mismo color.
Definida por su frontera: píxeles de un collor distinto al de la frontera.



FRONTERA INTERIOR

Los algoritmos precisan de un punto inicial de la región denominado semilla.
La frontera de una región 4-conectada puede ser 4-conectada u 8-conectada.
La frontera de una región 8-conectada es obligatoriamente 4-conectada.

```
Procedimiento rellenadoInterior4(x, y, : entero; viejo, nuevo: color);
Empezar
    Si obtener_pixel(x, y) = viejo entonces
            dibujar_pixel(x, y, nuevo);
            rellenadoInterior4(x, y-1, viejo, nuevo);
            rellenadoInterior4(x, y+1, viejo, nuevo);
            rellenadoInterior4(x-1, y, viejo, nuevo);
            rellenadoInterior4(x+1, y, viejo, nuevo);
    finSi
finProcedimiento
```

```
Procedimiento rellenadoFrontera4(x, y, : entero; frontera, nuevo: color);
Var   c: color;
Empezar
        c ← obtener_pixel(x, y);
        Si (c<>frontera) y (c<>nuevo) entonces
                dibujar_pixel(x, y, nuevo);
                rellenadoFrontera4(x, y-1, frontera, nuevo);
                rellenadoFrontera4(x, y+1, frontera, nuevo);
                rellenadoFrontera4(x-1, y, frontera, nuevo);
                rellenadoFrontera4(x+1, y, frontera, nuevo);
        finSi
finProcedimiento
```

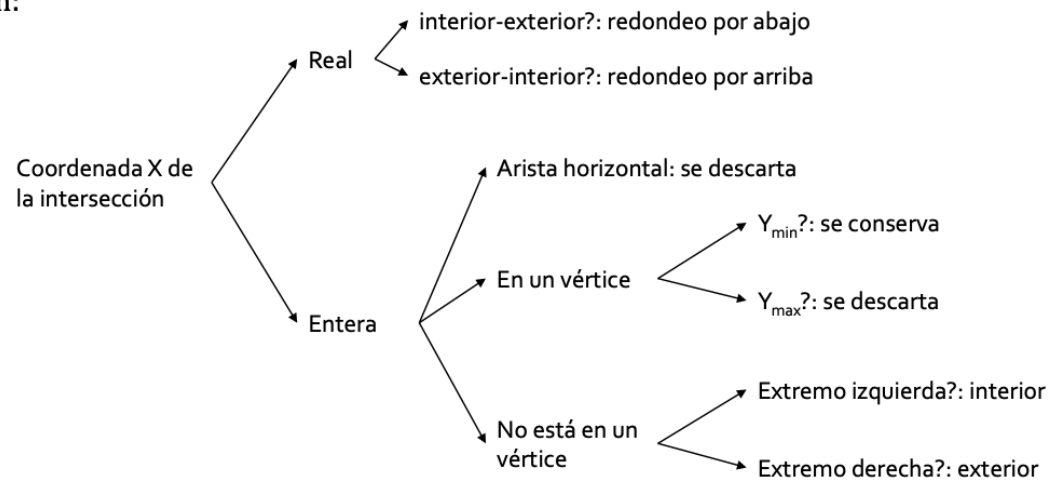
Algoritmo iterativo (área 4-conexa) por frontera

```
P = pixel semilla
Apilar P en la pila de semillas
Mientras la pila de semillas no esté vacía
    S = cabeza pila semillas (se desapila)
    Se rellena el tramo de píxeles de S
    Se apilan los extremos izquierdos de los tramos
    conectados por abajo con los píxeles del tramo actual
    que no estén al nuevo color
     Se apilan los extremos izquierdos de los tramos
    conectados por arriba con los píxeles del tramo actual
    que no estén al nuevo color
finMientras
```

6. Algoritmos Polígonos

Dadas las coordenadas y atributos dle polígono: determinar qué píxeles debemos rellenar, decidir color de los píxeles.

Resumen:



1. Algoritmo de línea de rastreo

Rellena áreas cuyo límite se define mediante un polígono.

1. Trazar líneas de rastreo y calcular las intersecciones por orden creciente de las x
2. Ordenar las intersecciones por orden creciente de las x
3. Agrupar las intersecciones por pares consecutivos
4. Rellenar los píxelles que caen entra las intersecciones

2. Algoritmo de lista de aristas activas

Por cada arista: Y máximom de la arista, X correspondiente a la Y mínima de la arista, incremento de X entre dos LDR (cada una de las líneas horizontales de píxeles del raster)(1/m).
Se almacena en la lista asociada a la primera LDR que corta la arista.
Y = primera LDR significativa
LAA vacía
Repetir hasta que LA y la LAA quedan vacías
(…) //ACABAR Y ENTENDER

TEMA 4: TRANSFORMACIONES Y VISUALIZACIÓN 2D

1. Fundamentos matemáticos

Geometría -> estudio de las relaciones entre objetos en un espacio n-dimensional.
Operadores principales -> producto escalar y producto vectorial.
Terminología:

Normal -> perpendicular
Norma o módulo de un vector -> tamaño
Ser coplanares -> un plano los contiene
Trasladar -> cambiar algo sin cambiar su orientación
Dependencia lineal -> dos vectores son linealmente dependientes si uno es múltiplo de otro
Convexidad -> un objeto es convexo si y solo si para cualquier dos puntos dentro del objeto, todos los puntos del segmento de recta entre ellos, también están dentro del objeto

Conjunto mínimo de primitivas con las que construir objetos:

- Puntos (ubicaciones en el espacio) -> su traslación resulta en un punto diferente. Si se restan dos puntos se obtiene un vector. Un punto es equivalente a sumar un punto a un vector. Se utilizan coordenadas homogéneas para representar los puntos para poder utilizar matrices cuadradas para representar las transformaciones.

- Escalares -> miembros de conjuntos que se pueden combinar mediante sumas y productos -> estas operaciones son asociativas, conmutativas e inversas -> nº reales, enteros, complejos -> por sí mismos no tienen propiedades geométricas.

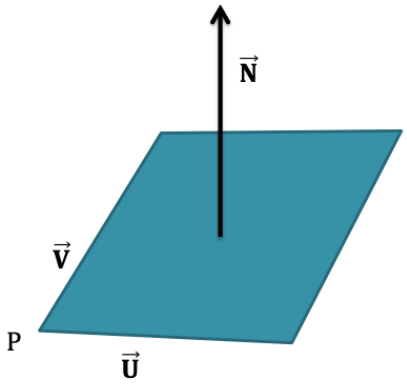
- Vectores -> representan desplazamientos entre puntos / direcciones-orientaciones / localizaciones-puntos en el espacio -> cantidad que tiene dirección, sentido y magnitud (definición física) -> se mantiene igual al trasladarse -> en GRÁFICOS se utilizan para representar las posiciones de los vértices, determinar la orientación de una superficie (***vector normal a la superifice***) y modelar la interacción de la luz (***vector de incidencia de la luz***) -> propiedades: conmutativa, asociativa, identidad (suma y producto), inverso, distributiva (suma vector y suma escalar), asociativa (producto escalar).

Ecuaciones de la recta		Interesante
Ecuación vectorial	$(x,y)=(P_x,P_y)+t(v_x,v_y)$	Siendo el punto $P=(P_x,P_y)$ y el vector director $v=(v_x,v_y)$
Ecuación paramétrica	$\begin{cases} x = P_x + v_x t \\ y = P_y + v_y t \end{cases}$	
Ecuación continua	$\frac{x - P_x}{v_x} = \frac{y - P_y}{v_y}$	
Ecuación general o implícita	$Ax+By+C=0$	Podemos calcular el vector director con la fórmula $v=(-B,A)$
Ecuación explícita	$y=mx+n$	Siendo m la pendiente y n la ordenada en el origen. Podemos comprobar la pendiente con la fórmula $m= v_y/v_x$
Ecuación punto pendiente	$y- P_y=m(x- P_x)$	

Recta -> definida por todos sus puntos -> conjunto de todos los puntos que pasan por P en la dirección del vector V.

//rayos ??

Plano -> se define mediante un punto y dos vectores o mediante tres puntos.



Cada plano tiene un vector normal a él.

Vector normal se calcula como el producto vectorial de los vectores en la direccón de dos aristas adyacentes: N = U x v

Producto de matrices no conmutativo. Solo se pueden multiplicar cuando

$$M\ a \times b = M'\ b \times c$$

2. Transformaciones 2D

Puntos 2D -> vectores columna
Transformaciones -> matrices cuadradas que pre multiplican all vector

1. Traslación

Desplazar un objeto desde una posición a otra diferente:
- Borrar primitiva posición actual
- Sumar desplazamiento a los puntos de la primitiva
- Redibujar primitiva

No depende de un punto de referencia.

2. Escalado

Modificación del tamaño de un objeto.

$$P' = \mathbf{S} * P$$

Si un factor de escala es menor que 1 el objeto se reduce en esa dimensión.
Si un factor de escala es mayor que 1 el objeto aumenta en esa dimensión.
Si un factor de escala es negativo el objeto se invierte en esa dimensión.
Si los factores de escala son diferentes se cambian las proporciones.
El escalado se realiza respecto a un punto fijo.
Se realiza respecto a un punto de referencia.
Equivalente a: trasladar el objeto (-x,-y) -> escalado -> deshacer traslación.

3. Rotación

Giro de un punto respecto de otro.
Coordenadas cartesianas de un punto son (x,y)
Sus correspondientes coordenadas polares son:

$x = r * \cos(ß)$

$y = r * \sin(ß)$

Si aplicamos giro de α grados respecto a un punto:

$x' = r * \cos(ß + \alpha)$

$y' = r * \sin(ß + \alpha)$

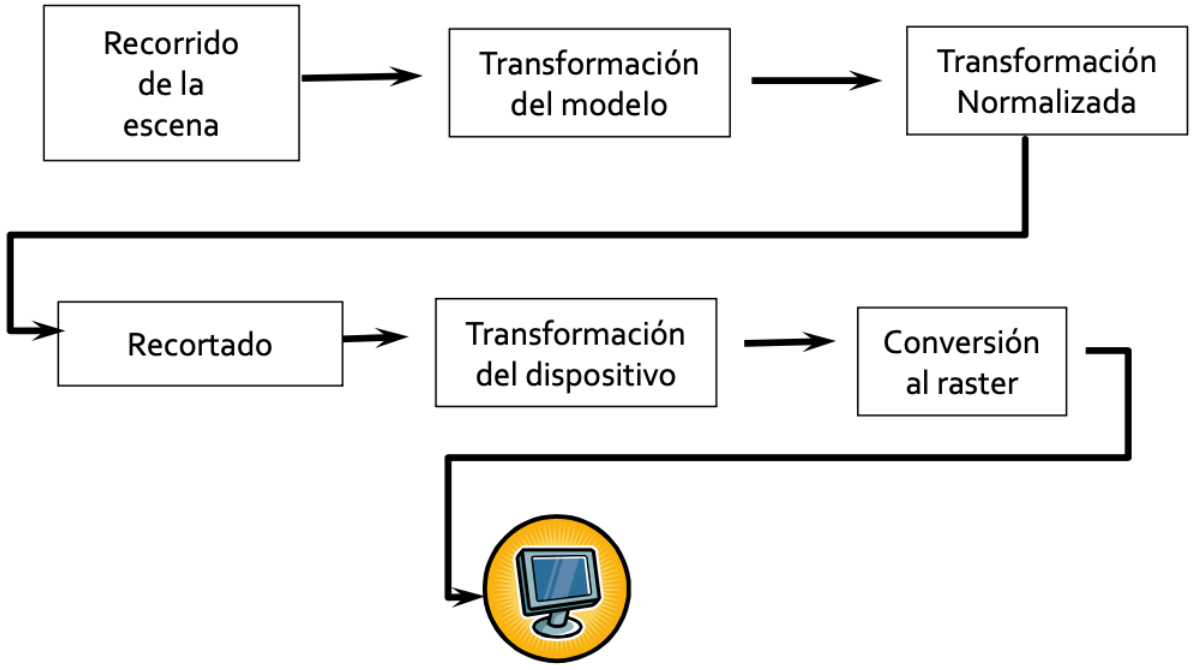
Se realizan desde un punto llamado centro de rotaciones (CR).
Equivalente a: trasladar objeto (-CR) -> rotación -> deshacer traslación.

4. Orden de las transformaciones

Conmutativas:
- traslación-traslación
- escalado-escalado
- rotación-rotación
- escalado proporcional-rotación

No conmutativas:
- traslación-escalda
- traslación-rotación
- escalado no proporcional-rotación

3. Visualización 2D



Sistema de coordenadas del mundo real (***CMR***):
- Descripción del sistema físico real.
- Las unidades de medida dependen de la representación.
- Se deben transformar en medidas del dispositivo (*mapping*).
- La transformación (*mapping*) se compone de un escalado y de un desplazamiento.

Sistema de coordenadas del dispositivo (***CD***) :

Depende del dispositivo:

Tamaño de la ventana, en píxels si es una pantalla.

Situación del origen de coordenadas.

Sentido de avance de cada coordenada.

Sistema de coordenadas del dispositivo normalizado (***CDN***):

Para realizar el *mapping* de forma normalizada se utiliza un dispositivo ficticio de anchura 1 y altura 1

Problemas de una transformación directa ***WCS*** (*World Coordinate System*)-> ***DCS*** (*Device Coordinate System*):

No se conoce el dispositivo

El *mapping* se tiene que reescribir si cambia el dispositivo

Díficil definir el marco en ***DCS***

Pasar de un sistema de coordenadas a otra de la forma más eficiente y a ser posible de forma eficiente y normalizada:

Definir zona mundo real: ventana (window)

Definir zona dispositivo: marco (viewport) : tamaño marco va de un píxel a toda la pantalla; lo que queda fuera de la ventana queda fuera del marco.

Transformación de coordenadas de la aplicación en coordenadas del dispositivo físico (***CD***):

De ***CMR*** a ***CDN***: transformación normalizada.

De ***CDN*** a ***CD***: transformación del dispositivo.

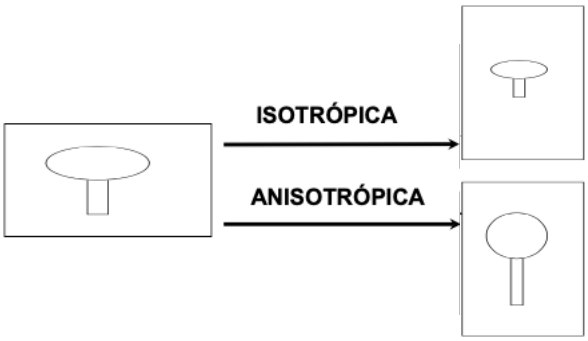
Pasos para transformar de ***SCM*** *(Sistema de Coordenadas del Mundo)* a ***SCD*** *(Sistema de Coordendas del Dispositivo)*:

1. Definir el formato final deseado (A4, 1024x768, etc) -> calcular el área correspondiente en el dispositivo normalizado
2. Definir la ventana en *SCM* -> qué parte de la escena se dibujará. En coordenadas del mundo
3. Definir el marco en ***SCDN*** *(Sistema de Coordendas Del Dispositivo Normalizado)* -> la posición deseada de la figura en el formato destino
4. Calcular la transformación *SCM -> SCDN*
5. Definir el marco en *SCD* -> permite múltiples páginas por hoja, escalar la imagen, etc.
6. Calcular la transformación *SCDN → SCD*

Transformación ventana-marco:

Se traslada la esquina inferior izquierda de la ventana al origen.
Se aplican los factores de escala para que marco y ventana tengan el mismo tamaño.
Se traslada el origen a la esquina inferior izquierda del marco.

La transformación de *CDN* a *CD* sería similar a la anterior .
En general las ventanas y los marcos serán rectangulares.
Tipos de transformaciones entre sistemas:



Isotrópica -> conservando proporciones, 1 factor de escala. Definición del marco deberá ser proporcional al de la ventana.

Anisotrópica -> 2 factores de escala.

Después de cambiar de *CM* a *CDN*, parte de la escena puede permanecer fuera del marco: utilizamos el recortado para quitar esa parte de la escena (no es visible).

El DCS depende del tamaño del dispositivo y es un sistema de coordenadas discreto.
En el WCS las unidades de medida dependen de la representación.
El WCS es independiente del dispositivo.

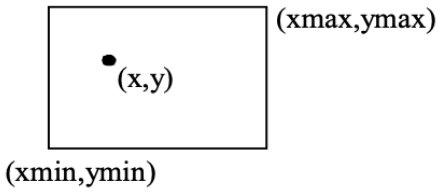
4. Recortado

Calcula las partes de la escena que son visibles a través de la ventana del mundo real.
Se puede hacer de dos formas:
Analíticamente: contra la ventana del mundo real.
Durante la conversión al raster.
Generalmente las ventanas de recortado son rectangulares y conn los lados paralelos a los ejes del dipositivos.

1. Recortado de puntos

Es el método más simple.
Un punto es visible si cumple las siguientes condiciones:

$$Xmin <= X <= Xmax \quad y \quad Ymin <= Y <= Ymax$$



2. Recortado de líneas (contra ventanas rectangulares)

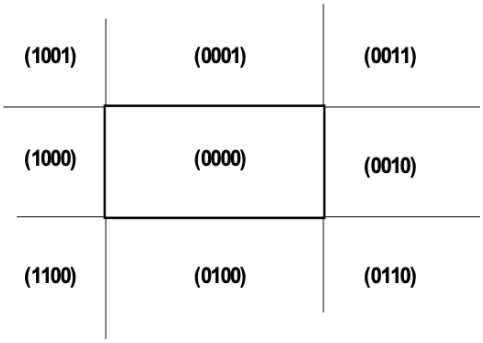
Para recortar una línea solo se consideran sus dos extremos.
3 situaciones posibles:

- Test trivial de aceptación: si los dos extremos de la línea están dentro de la ventana, la línea también lo está.
- Si sólo un extremo está dentro de la ventana, la línea intersecta el rectángulo: calcular la intersección
- Ningún extremo está dentro de la ventana: es necesario hacer más cálculos

Algoritmo de recortado de líneas de Cohen-Sutherland:

Realiza tests triviales iniciales para evitar calcular intersecciones innecesarias.
El espacio 2D se divide en nueve zonas, cada una con un código de cuatro bits.
Cada extremo de una línea se clasifica en una de las nueve zonas.

Bit Nº	Comentario	Condición
0	Extremo por encima de la ventana	Y > Ymax
1	Extremo a la derecha de la ventana	X > Xmax
2	Extremo por debajo de la ventana	Y < Ymin
3	Extremo a la izquierda de la ventana	X < Xmin



Se asigna un código de 4 bits a los extremos de la recta (P0,P1):

Test trivial de aceptación: CODIGO(P0) OR CODIGO(P1) = 0000

Test trivial de rechazo: CODIGO(P0) AND CODIGO(P1) <> 0000 (ambos extremos están en el mismo semiplano)

Ventajas de los tests triviales:

Fáciles de implementar en hardware.

Se pueden utilizar con cualquier algoritmo de recortado.

Los bits a 1 indican las fronteras de la ventana que pueden intersectar con la línea (caso de fallar el rechazo trivial).

Se pueden aceptar o rechazar rectas sin calcular las intersecciones.

Si la línea **no es aceptada ni rechazada** (trivialmente):

Elegir un extremo fuera de la ventana.

Calcular la intersección I de la línea con el límite de la primera región a la que pertenece el extremo elegido (el primer bit a 1 del código del punto).

Se elimina el tramo que va desde el extremo elegido a la intersección.

Si el otro tramo es aceptado trivialmente, el algoritmo termina.

Si no es aceptado, se calcula la siguiente intersección.

En el peor caso: se calculan las cuatro intersecciones.

Ventaja: como máximo se calculan las intersecciones que se calculan siempre en el de fuerza bruta.
Como máximo se calcularán tantas intersecciones como aristas tenga la ventana (peor caso).

3. Recortado de polígonos

Dificultad -> muchos casos posibles
Cada arista del polígono se ha de comprobar contra cada arista de la ventana del recortado.
No es correcto aplicar algoritmos de recortado de aristas.

Algoritmo de Sutherland-Hodgman

Volumen de recorte necesariamente convexo.

Algoritmo re-entrante para cada arista del área de recorte:
- Se realiza el recortado contra cada una de las aristas de la ventana de recorte.
- Se recorre cada arista del polígono a recortar.
- Se genera una nueva lista de vértices en cada iteración.
- Cuando el algoritmo se ha aplicado a todas las aristas de la ventana el resultado es el polígono recortado.

Extensión inmediata a 3D.
Resultado correcto: es necesatio generar más de una lista de polígonos como resultado -> algoritmo de Weiler-Atherton -> crea polígonos para cada fragmento visible.

Cuando los polígonos no son convexos puede ocurrir que el resultado del recorte no sea un único polígono.

El término "raster" se refiere a una representación de imágenes o gráficos mediante el uso de una cuadrícula de píxeles. El raster, también conocido como mapa de bits, organiza la información visual en una cuadrícula bidimensional, donde cada elemento de la cuadrícula es un píxel y tiene su propio color o intensidad.

TEMA 5: TRANSFORMACIONES Y VISUALIZACIÓN 3D

1. Transformaciones 3D

Para manipular objetos en el espacio 3D - traslación, giro y escalado- y para ayudar a visualizar y examinar objetos.
Los sistemas de coordenadas pueden ser:
- Dextrógiros: para las transformaciones
- Levógiro: para la proyección

Se utilizan las **coordenadas homogéneas** -> punto 3D (x,y,z) -> (x,y,z,w) y vector columna.
Matrices de transformación 4x4 -> premultiplicarán a los puntos

Para realizar una rotación de α grados respecto a un eje cualquiera:
- *Traslación* para que el eje pase por el origen.
- *Rotar* eje para que coincida con alguno de los ejes de coordenadas.
- *Rotación* de α grados alrededor del eje anterior.
- Aplicar *rotaciones inversas* para que el eje vuelva a orientación original.

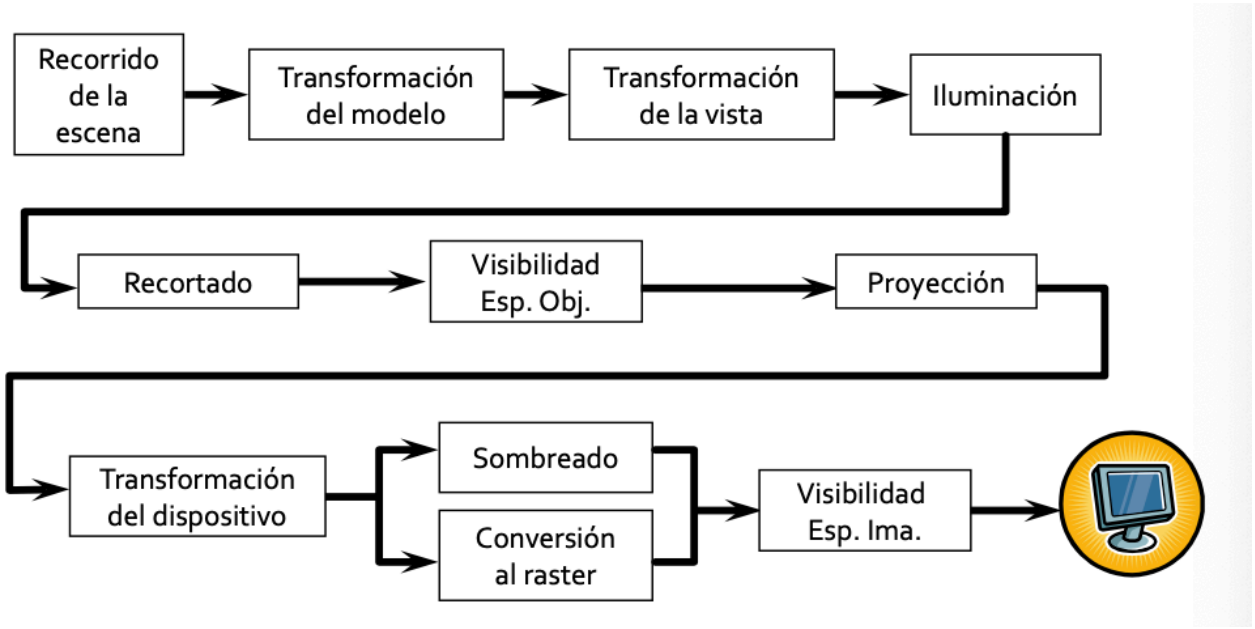
Aplicar *traslación inversa* para que vuelva a posición original.

La composición de transformaciones nos permite manipular objetos en el espacio 3D.
La composición de transformaciones sirve para realizar cambios de sistemas de coordenadas.
La composición de transformaciones nos permite realizar giros mediante coordenadas polares.

2. El proceso de visualización 3D

El **proceso de síntesis** de una imagen (proceso de visualización) es el conjunto de operaciones (en 3D y en 2D) sobre un modelo informático de datos que resultan en una representación gráfica del mismo en un dispositivo físico de representación.

Pipeline 3D



Escena -> se transforma a una posición estándar -> cámara al origen, vector vista sobre eje Z.
Fuentes de luz definadas por el color e intensidad de la luz que emiten y su posición.
En la **etapa de iluminación** se callcula el color que debe tener cada vértice teniendo en cuenta todos los parámetros anteriores.
En la **etapa de recortado** se define el volumen de la vista y se recortan los objetos que quedan en el exterior de dicho volumen.

En la **etapa de visibilidad** se eliminan los objetos de la escena que no se pueden ver (tapados por partes del propio objeto).
La **etapa de proyección** convierte un espacio 3D en uno 2D.
La **etapa de transformación** del dispositivo consiste en adaptar el sistema de coordenadas de la vista al sistema de coordendas del dispositivo.
La **conversión al raster** consiste en convertir un conjunto de primitivas matemáticas 2D en píxeles.
El **sombreado** consiste en decidir el color de cada píxel de un polígono en función del color de sus vértices.
La **visibilidad de los píxeles** es seleccionar para todos los píxeles de la imagen, el color del objeto más cercano.

Una **proyección** es una conversión de 3D a 2D.

PP: Plano de proyección
CP: Centro de proyección
p: Punto en 3D
p': Proyección de p (intersección entre la visual y PP)

Hay dos tipos:

Perspectiva: definida por CP.
Paralela: definida por DP.

La proyección de un punto es la intersección de la visual que pasa por el punto y el observador.

3. Proyecciones paralelas

1. Multivista ortográfica: DP ⊥ PP

Planta: PP ⊥ eje Y
Alzado: PP ⊥ eje Z
Perfil: PP ⊥ eje X
Utilizada en diseños de ingeniería (máquinas) y en planos de arquitectura.
Ventajas: medidas precisas, todas las vistas tienen la misma escala.
Inconvenientes: no proporciona visión realista objetos 3D, suelen ser necesarias varias vistas para percibir el 3D.
Muestra 1 cara.

2. Proyecciones Axonométricas: DP ⊥ PP

Plano de proyecciones **no es perpendicula**r a ningún eje.
Tamaño líneas paralelas se reduce en la proyección en la misma medida.
No se puede tomar medidas pero tienes visión 3D.

Isométrica: los ángulos entre las proyecciones de los 3 ejes son iguales (120º), se aplica el mismo factor de escala (1) a lo largo de cada eje. Dirección de la proyección es [111].
Dimétrica: ángulos entre dos de los ejes son iguales. 2 factores de escala.
Trimétrica: ángulos entre 3 ejes diferentes. 3 factores de escala.
En función de ángulos utilizados variarán factores de escala.
Muestra caras adyacentes.

3. Proyección isométrica

Utilizada en ilustraciones de catálogos, registros de oficinas de patentes, diseño de muebles y algunos videojuegos.
Ventajas: no es necesario usar múltiples vistas, muestra la naturaleza tridimensional de los objetos, realizar medidas escalando en los ejes.
Inconvenientes: falta de disminución del tamaño en la proyección produce distorsiones, más útil para superficies planas que curvas.

4. Proyección oblicua:

Proyectores son oblicuos al pllano de proyecciones.
Plano de proyecciones es normal a uno de los ejes.
Ventajas: pueden representar de forma exacta una cara del objeto (tomar medidas exactas), mejor para formas elípticas que axonométricas, comparación de tamaños más sencilla que con perspectiva, representa apariencia tridimensional.
Inconvenientes: objetos aparecen distorsionados si no se elige bien el plano de proyecciones (círculos pueden parecer elipses), puede no parecer real.
Muestra caras adyacentes, 1 cara tiene medidas exactas, las otras presentan disminución del tamaño de la proyección de forma uniforme.
Tipos:
Caballera: ángulo entre plano de proyecciones y proyectores = 45º ; caras perpendiculares se proyectan a escala 1.
Gabinete: caras perpendiculares se proyectan a escala 1/2.

4. Proyección perspectiva

Centro proyección en un punto (x,y,z).
Proyección definida por CP y PP.
Ventajas: proporciona realismo visual y sensación tridimensional (efecto tamaño distancia).

Inconvenientes: no mantiene forma ni escala del objeto (excepto en planos paralelos al plano de proyección).
Diferencias con la paralela: líneas paralelas dejan de serlo al proyectar, tamaño objetos disminuye con la distancia, disminución tamaño no es uniforme.

En las proyecciones perspectivas los objetos que están más alejados del observador se representan de menor tamaño.

En la proyección paralela el tamaño de los objetos no varía con la distancia.

En las proyecciones paralelas ortográficas los planos de proyección son paralelos a alguno de los planos principallles del sistema de coordenadas.

En la proyección perspectiva simple, el sistema de coordenadas es levógiro, el centro de proyecciones está en el origen y el plano de proyecciones es perpendicular al eje Z a una distancia d.

En un proyección paralela, si se acercan los objetos al plano de proyecciones en la dirección de observación los objetos proyectados quedan igual.

5. Modelo de cámara

El modelo de cámara nos permite definir vistas generales utilizando determinados parámetros.

Parámetros:

> **Posición**: 3 grados de libertad -> x,y,z
> **Orientación**: dos vectores -> *LOOK* (indica hacia dónde mira la cámara) y *UP* (rotación alrededor del eje definido por *LOOK*) ; con la posición de la cámara y un punto de interés *LOOK-AT* también se puede obtener *LOOK*.

UP debe ser perpendicular a *LOOK* (en principio. Como es díficil los paquetes gráficos lo ajustan para que sea así -> *UP* puede ser cualquier vector no paralelo a *LOOK* -> generalmente [010]).

Volumen de la vista para proyección paralela ortográfica: define la parte de la escena visible, los objetos se recortan contra este volumen de la vista.
Volumen de la vista para proyección perspectiva: pirámide truncada.

Planos de recorte frontal y trasero:

El volumen entre seis planos de recortado define la porción de la escena que la cámara ve.
Las posiciones de los planos frontal y trasero vienen dadas por dos distancias a lo largo del vector *LOOK*.
Los objetos que quedan fuera del volumen no se dibujan.
Los objetos que intersectan con el volumen se recortan.

Razón de aspecto: análogo al tamaño de lals fotografías, indica la proporción entre anchura y altura.
Campo de visión: anchura de campo: análogo a escoger una lente para una cámara fotográfica -> ajusta zoomm y cantidad de distorsión perspectiva.
Efecto gran angular.
Profundidad de campo: algunos modelos de cámara tienen profundidad de campo para fijar el rango de enfoque ideal y así aproximar el comportamiento de una cámara real ; objetos situados a la distancia focal se visualizarán nítidos (enfocados), los que estén más cercanos o más lejanos aparecerán borrosos (desenfocados).
Transformaciones de la vista:

vista general -> vista simple: cámara se mueve al origen, dirección proyección se lleva al eje -Z, dirección *UP* se lleva al eje Y.
CÁMARA PERSPECTIVA vista simple -> volumen canónico: ajusta tamaño volumen -> cambio a sistema levógiro.
CÁMARA PARALELA vista simple -> volumen canónico: trasladar volumen a z=0 -> ajusta tamaño volumen -> cambio a sistema levógiro.

TEMA 6: MODELADO GEOMÉTRICO

1. Introducción

Un sistema de modelado está formado por:

Estructura de datos: principal y auxiliares; exactas y aproximadas.
Operaciones: visualización, edición, conversión y determinación de propiedades.

Requisitos que debe cumplir para modelar sólidos:

No ambiguo.
Dominio de representación se debe ajustar al área de aplicación.
Válido: no se debe permitir la representación de objetos que no correspondan a un sólido.
Compacto: no redundante.
Eficiente en el manejo.

Creación de los objetos a modelar:

Aplicación de modelado.
A partir de objetos reales.
Matemáticamente.

2. Modelado plano de superficies

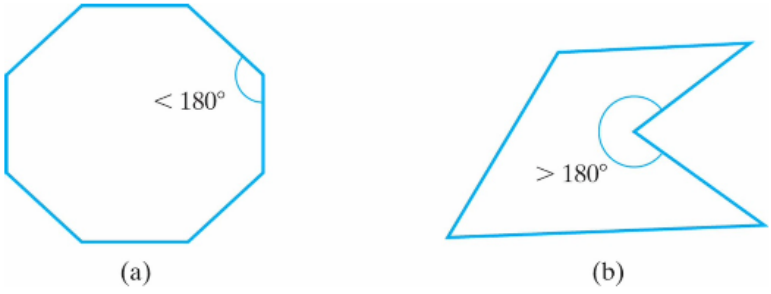
Modelo alámbrico

Se utilizaba cuando los monitores eran vectoriales.
Elementos: puntos, líneas, arcos, círculos, cónicas y curvas.
Ventajas: fácil de construir, pocaas necesidades de memoria y almacenamiento.
Desventajas: representación ambigua, falta de coherencia visual.
Actualidad: visualización auxiliar.

Polígono -> figura plana definida por un conjunto de 3 o más coordenadas (**vértices**) -> conectados en secuencia por **aristas**. Todos vértices deben estar en el mismo plano y las aristas no pueden cruzarse.

Un **ángulo interior** de un polígono es el ángulo que forman dos aristas adyacentes en el interior del polígono.

Si todos los ángulos interiores de un polígono son menores o iguales 180º, el polígono es **convexo**.
También, si unimos cualesquiera dos puntos del interior de un polígono convexo, el segmento de línea que los une, también está en el interior.
Si un polígono no es convexo, entonces es cóncavo.



Se dice que un polígono es **degenerado** si contiene vértices colineales o cuyas coordenadas sean coincidentes.
Los vértices **colineales** generan un segmento de línea recto.
Los vértices **coincidentes** generan, solapamiento de aristas o aristas de longitud 0.
También se utiliza este término para polígonos con menos de tres vértices.
Es importante en una aplicación gráfica, tener mecanismos para evitar los polígonos degenerados.
Los polígonos cóncavos también generan problemas, por lo que es común dividirlos en un conjunto de polígonos convexos.

Determinar si un polígono es **cóncavo:**

Calcular el vector en la dirección de cada arista.
Calcular el producto vectorial de cada par de aristas adyacentes.
Si la coordenada zeta de todos ellos es positiva o negativa, el polígono es convexo, en otro caso es cóncavo.

Triangularizar un polígono **convexo:**

Se seleccionan tres vértices consecutivos y se crea un triángulo.
Se elimina de la lista de vértices el del medio.
Se vuelve a hacer lo mismo con la lista modificada hasta que sólo queden 3 vértices.

Malla de polígonos: colección de vértices, aristas y polígonos conectados de forma que cada arista es compartida como máximo por dos polígonos .

vértice: punto de coordenadas (x,y,z).
arista: segmento de línea que une dos vértices.
polígono: secuencia cerrada de aristas.

Existen diferentes **tipos de representaciones** que pueden usarse a la vez en una misma aplicación:

Representación explícita

Cada polígono se representa por una **lista de coordenadas de vértices.**
Los vértices se almacenan en **orden** (horario o antihorario).
Los vértices compartidos están **duplicados**.
No existe representación explícita de los vértices y aristas compartidas.
Utilizado para almacenar polígonos complejos que no forman partes de mallas.
Ventajas: representación eficiente para polígonos individuales
Problemas: alto coste de almacenamiento, para mover un vértice es necesario recorrer todos los polígonos, si se dibujan las aristas las compartidas se dibujan dos veces.

Representación con punteros a lista de vértices

Cada vértice se almacena una sola vez en una **lista de vértices**.
Un **polígono** se define como una lista de índices (o punteros) a sus vértices en la lista de vértices.
Ventajas: cada vértice se almacena una sola vez, las coordenadas de los vértices pueden cambiarse fácilmente.
Problemas: difícil encontrar polígonos que compartan una arista, las aristas compartidas se siguen dibujando dos veces.

Representación con punteros a lista de aristas

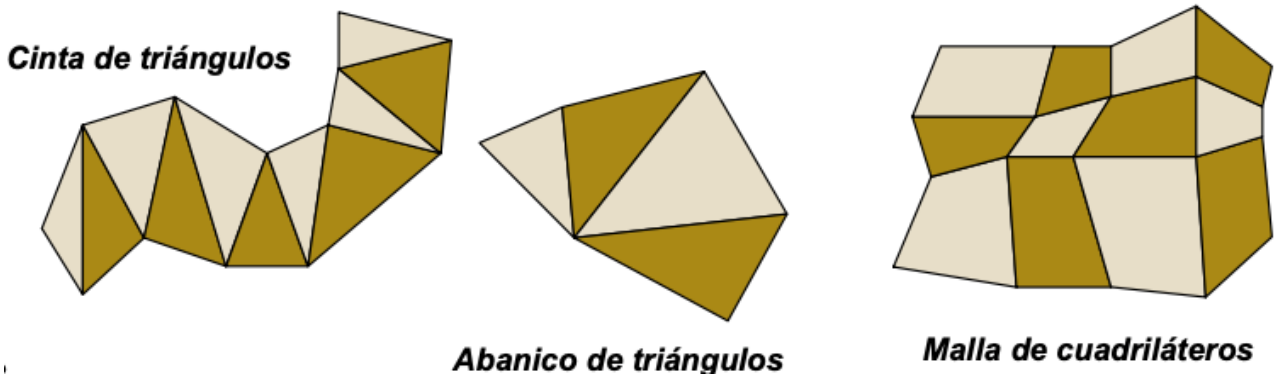
Se mantiene la **lista de vértices.**
Un **polígono** se representa como una lista de índices a una lista de aristas.
Cada arista apunta a dos vértices y a los polígonos a los que pertenece.
Ventajas: cada vértice se almacena una sola vez, las aristas compartidas se dibujan una sola vez.
Problemas: difícil determinar qué aristas comparten un vértice (en todas las representaciones).

Criterios de evaluación de las representaciones:

tiempo
espacio
información topológica

Mallas poligonales:

Cinta de triángulos -> para n vértices, produce (n-2) triángulos conexos.
Abanico de triángulos -> para n vértices, produce (n-2) triángulos conexos.
Malla de cuadriláteros -> genera una malla de (n-1) por (m-1) cuadriláteros para n por m vértices.



3. Modelos de sombreado

Aproximación de superficies curvas con mallas de polígonos

Cálculo aproximado:
Sumar las normales de las caras adyacentes al vértice
Normalizaar el vector resultante -> ecuaciones de sombreado usan vectores unitarios.

Problema -> cilindro tiene vértice común a la cara lateral y a la 'tapa' -> solución "*crease angle*"

4. Modelado curvo de superficies

No paramétricas

Explícitas: incapaces de representar curvas cerradas (círculos) o de valores múltiples (parábolas), dificultad en la aplicación de rotaciones y de tangentes verticales.

Implícitas: la ecuación puede tener más de una solución, problemas con la especificación de continuidad entre curvas, cuádricas y superficies equipotenciales.

Paramétricas: fáciles de usar para edición y manipulación (*CAD/CAM*), posibilidad de describir objetos mediante trozos adyacentes, sustituyen las pendientes (que pueden ser infinitas) por vectores tangentes.

Cuádricas: descritas mediante **ecuaciones de segundo grado.**

Supercuádricas: generalización de las superficies cuádricas que incorporan **parámetros adicionales.**

Splines:

Curvas -> definidas mediante funciones polinómicas cúbicas, cuyas derivadas de primer y segundo orden son continuas.
Superficies -> definidas mediante dos conjuntos ortogonales de curvas.
Puntos de control -> conjunto de puntos que indican la forma general de la curva .
Convex hull -> frontera del polígono convexo que encierra el conjunto de puntos de control.
Polígono característico -> conjunto de segmentos que conectan los puntos de control en orden.
Tipos de Splines:

- Interpolación -> la curva pasa por los puntos de control.
- Aproximación -> la curva se ajusta a los puntos de control.

Condiciones de continuidad: permiten especificar la forma en que se unen dos segmentos curvos. Si cada segmento se describe como un conjunto de funciones paramétricas de la forma

x = x(u), y = y(u), z = z(u), u1 <= u <= u2

Continuidad paramétrica:

- C^0: curvas unidas, valores x,y,z evaluados en u2 para el primer segmento son iguales a los valores x,y,z evaluados en u1 para el segundo segmento.
- C^1: primeras derivadas (tangentes) de las curvas son iguales en los puntos de unión.
- C^2: primeras y segundas derivadas de los dos segmentos de las curvas son iguales en la intersecciónn -> variación de las tangentes entre los dos segmentos de curvas es suave.

Continuidad geométrica:

- G^0: similar a C^0.
- G^1: tangentes proporcionales.
- G^2: primera y segunda derivada son proporcionales.

Curvas y superficies de Hermite:

Representación por **interpolación**.
Las curvas suelen utilizarse para la especificación de trayectorias, se definen dando los puntos inicial y final y sus respectivas tangentes, son curvas cúbicas, si se quiere interpolar un conjunto n de puntos con sus dos vectores tangentes finales P0' y P'n-1 se imponen unas condiciones de continuidad y se plantea un sistema de ecuaciones para obtener las tangentes intermedias.
Las superficies se definen con los puntos y sus tangentes en cada una de las direcciones paramétricas.

Curvas y superficies de Bezier:

Aproximan cualquier número de puntos de control.
Una curva con n+1 puntos tiene grado n.

Con 4 puntos tenemos **curvas cúbicas.**
Continuidad de tramos cúbicos: C0, C1 y C2.
Manipulación de la curva: movimiento de puntos, acumulación de puntos, control global.

Curvas y superficies B-Splines:

Spline: banda flexible que produce una curva suave a través de un conjunto de puntos.
Curva con secciones polinómicas.
Diseñar curvas y superficies.
El grado es **independiente** del número de puntos de control.
Permiten **control local.**

Beta-Splines:

Generalización de las B-*Splines* en las que se imponen condiciones de continuidad sobre la primera y la segunda derivada paramétrica.
Parámetros de continuidad se llaman **ß parámetros.**

Splines racionales (NURBS):

Similares a las B-Splines pero con **pesos asociados** a cada uno de los puntos de control -> permiten ajustar más o menos la curva al punto de control -> cuando todos son iguales a 1 tenemos una B-*Spline*. **Representación exacta para las cuádricas (cónicas).**

5. Modelado de sólidos

Representación útil para la construcción de objetos que tienen **simetrías rotacionales, traslacionales o de otros tipos.**
Los objetos se definen mediante una primitiva en 2D y una trayectoria en el espacio 3D.

Geometría sólido constructiva (CSG):

Un nuevo sólido se obtiene aplicando operaciones de unión, intersección y diferencia sobre dos sólidos iniciales.
Existe un conjunto inicial de primitivas (bloques, conos, cilindros, esferas, superficies de revolución, …).

Los objetos diseñados con este método se representan con un **árbol binario.**

Enumeración espacial (Octrees):

Estructura de **árbol jerárquico**, en la que cada nodo se corresponde con una región del espacio 3D.
Las regiones del espacio 3D se dividen en **octantes** (cubos), se almacenan 8 elementos en cada nodo del árbol.
Los nodos pueden ser de 3 tipos: **blancos** (fuera del objeto), **verdes** (dentro del objeto) o **grises** (ni dentro ni fuera).
Los elementos individuales del espacio 3D se denominan ***voxels***.

6. Otros modelos

Fractales

Permiten representar objetos que por su **complejidad** no se ajustan a la geometría euclídea (montañas, nubes, plantas, etc).
Se utilizan **procedimientos** en lugar de ecuaciones para modelar los objetos.

Sistemas de partículas

Técnica de modelado ideal para **objetos o fenómenos dinámicos** (nubes, humo, fuego, explosiones, …).
Las **partículas** son los elementos básicos del sistema pueden tener distintas formas (esferas, elipsoides, blobs, ...) y un comportamiento que determina la evolución del sistema (tiempo de vida, aspecto de las partículas, trayectoria, ...) y que puede estar influido por **fuerzas específicas** (gravedad, campos magnéticos, …).

Blobs

Esferas flexibles entre las que existe un campo de fuerza que las repele o atrae, de forma que cuando entran en contacto se funden en un único cuerpo con conexiones suaves.
Otras denominaciones: *Metaballs* , isosuperficies, objetos blandos.
Aplicaciones: modelado de moléculas, representación de fluidos.

TEMA 7: VISIBILIDAD

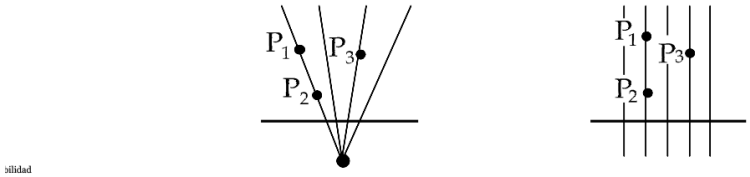
Dado un conjunto de objetos 3D y un modelo de cámara sintética, determinar qué líneas y superficies son visibles desde el punto de vista (para una cámara perspectiva) y a lo largo de una dirección de proyección (para una cámara ortográfica).

Especificación de la vista: punto de vista (PV), punto de interés (PI), plano de proyección (PP) y campo de visión.

Decidir si dos puntos P1 y P2 uno tapa al otro:

¿Están mismo proyector?

- **Proyección perspectiva:** $\frac{x_1}{z_1}=\frac{x_2}{z_2}$ & $\frac{y_1}{z_1}=\frac{y_2}{z_2}$
- **Proyección paralela:** $x_1 = x_2$ & $y_1 = y_2$



bilidad

Si lo están -> comparar z1 y z2 para decidir cuál está más próxima a la cámara.

Transformación perspectiva-paralela

Preserva profundidad relativa, líneas y planos.
En perspectiva -> disminución del tamaño en función de la profundidad.
Ventajas:

Recortado tras la transformación consiste en recortar contra los planos.
Profundidades se commparan después de la transformación.
Proyección de los puntos tras la transformación consiste en descartar componente z.

Clasificación de algoritmos

Algoritmos en el espacio de la imagen

La visibilidad se calcula en coordenadas del espacio de la imagen.
Se calcula cuál de los n objetos de la escena es visible para cada uno de los p píxeles de la imagen.

para cada píxel hacer
    trazar una visual que pase por PV y el píxel
    determinar el objeto más cercano que intersecciona con la visual
    dibujar el píxel del color del objeto
fin_para

Complejidad: O(np) -> para cada píxel tenemos que comprobar todos los polígonos.
La operación más compleja es el cálculo de la intersección recta-polígono.

Algoritmos en el espacio de objetos

La visibilidad se calcula en coordenadas del espacio de objetos
Cada objeto se compara con todos los demás
Los objetos (o partes de los objetos) que no son visibles, se eliminan de la escena

para cada objeto hacer
    determinar las partes del objeto que no están ocultas:
        por otras partes de sí mismo
        por otros objetos
    dibujar las partes no ocultas con el color apropiado
fin para

Complejidad: O(n2) -> cada objeto se tiene que comparar con todos los demás.

La operación más compleja es el cálculo de la intersección polígono-polígono.

Después de calcular qué objetos son visibles se tienen que convertir al raster para obtener la imagen final.

Algoritmo del pintor

Ordenación en Z (Newell, Newell y Sancha).
Espacio del objeto.
Estrategia: Ordenar los polígonos en profundidad, del más lejano al más cercano y después dibujarlos en ese orden.

Realizar una primera ordenación por la Z más alejada de cada polígono.
Seleccionar la superficie (S) al final de la lista.
Si no hay solape con otros polígonos de la lista,
Convertir S al raster.
En otro caso,
Realizar test posteriores para determinar si se necesita reordenar.

Si el polígono más alejado en la lista S solapa con otro u otros polígonos, se crea la lista de polígonos que solapan con S .

Realizar los siguientes tests entre S y cada polígono que se solape con él (S').
Si algún test es CIERTO, entonces S' se elimina de la lista porque se debería convertir al raster S antes de S'.

1. Los rectángulos de inclusión en las direcciones XY para las dos superficies no se solapan.
2. El polígono S está completamente detrás del polígono con el que se solapa (S') respecto a la posición de la vista.
3. El polígono con el que se solapa (S') está completamente delante de S respecto a la posición de la vista
4. Las proyecciones de las dos superficies sobre el plano de la vista no se solapan

Si cada polígono que se solapa con S pasa un test al menos (lista vacía) -> convertir al raster.
Si no -> intercambiar S y S' (polígono que falla los 4 tests) + repetir tests para cada polígono reordenado.
Puede entrar en bucle infinito cuando dos o más alternativamente se tapan unos a otros.
Una vez un polígono se ha reordenado debería marcarse. Si se va a reordenar de nuevo -> se divide el polígono y se continúa.

Ventajas: rápido para escenas simples, independiente de la resolución, ordenación en profundidad es útil para seleccionar objetos.
Desventajas: lento para escenas de complejidad media, díficil de implementar y depurar, muchos casos especiales.

Z-Buffer

Características destacables

Precisión de imagen.
Gran simplicidad.
Alto consumo de memoria.
Almacenamiento de la profundidad máxima (Zmax) a lo largo de X, Y.
Trabaja con escenas de cualquier complejidad.
No necesita ningún tipo de ordenación.
Facilidad de implementación hardware.

Estructura de datos

Además del mapa de píxeles se utiliza otra matriz (Z-Buffer) para almacenar la profundidad (valor de Z) para cada píxel.

Algoritmo

Preproceso

El raster se inicializa al color del fondo.
El Z-Buffer se inicializa a Zmax (1 para el volumen canónico).

Proceso

Los polígonos se convierten al raster (cálculo de los píxeles que ocupa el polígono) en orden arbitrario.
Comprobación de cercanía y actualización.

```
Inicializar imagen a color del fondo
Inicializar zbuffer a la z de máximo alejamiento
para cada polígono
    para cada píxel en la proyección del polígono
        z:=z(x,y)
        si z más cercana que zbuffer(x,y)
            zbuffer(x,y):= z
            escribir píxel (x,y) al color conveniente
fin Z-Buffer
```

Eliminación de caras traseras (Backface culling)

Invarianza de signo del producto escalar entre la normal al plano y cualquier visual que lo atraviese.

No son visibles los polígonos cuya normal forma un ángulo entre -90 y +90 con la dirección de observación.

Next = Normal externa

DP = Dirección de Proyección

Aprovechamiento de la transformación perspectiva-paralela:

DP = (0,0,1)

cara trasera = componente z de Next positiva (levógiro)

Reducción media del 50% de polígonos

TEMA 8: ILUMINACIÓN Y SOMBREADO

Fuentes de luz:

Posición: localizada o infinitamente alejada.

Intensidad: color de la fuente.

Cantidad: integración de efectos.

Distribución lumínica: uniforme, focalizada, direccional, etc.

Geometría: puntual, esférica, lineal, etc.

Objetos:

Distancias: al observador y a la fuente de luz.

Material: pulido, metálico, rugoso, etc.

Propiedades ópticas: transparencia, refracción, etc.

Cromaticidad: Color superficial propio.

Interacción lumínica con otros objetos de la escena.

Observador:

Dirección de observación: cálculo de la intensidad

Modelo de iluminación simple

Un modelo de iluminación se define como: I = f (p, PV, {O}, {F})

p: punto de cálculo de la iluminación

PV: posición del punto de vista

{O}: modelo geométrico y material de los objetos

{F}: modelo geométrico e intensidad de las fuentes de luz

I: intensidad luminosa observada en p

Los modelos que se mostrarán son empíricos -> no siguen leyes físicas -> simulación visualmente aceptable del fenómeno de la iluminación.
Simplificaciones aplicadas: fuentes puntuales, objetos opacos, modelo de iluminación local.

Modelo más simple -> cada objeto es mostrado usando intensidad propia -> $I = k_i$ -> k_i es la intensidad intrínseca [0..1]

Iluminación ambiente

Luz ambiente -> se considera que hay una fuente de luz no direccional, producto de múltiples reflexiones de luz desde muchas fuentes presentes en el entorno -> incide igualmente en todas las superficies en todas las direcciones -> $I = I_a * k_a$ -> I_a es la intensidad de la luz ambiente y k_a el coeficiente de reflexión ambiente (depende de las propiedades del material del objeto)

Iluminación difusa

Reflexión difusa ->parte de la luz reflejada por la superficie de un objeto de manera adireccional (igual intensidad en cualquier dirección) -> la intensidad de luz reflejada en un punto es independiente de la posición del observador

Ley de Lambert: "la componente difusa de la luz reflejada por una superficie es proporcional al coseno del ángulo de incidencia"

Es característico de las superficies mate: (tiza, paredes, telas,...)
La intensidad resultante depende del ángulo θ entre la dirección de L y de N (ángulo de incidencia) -> $I = I_L * k_d * cos(\theta)$ -> ecuación de reflexión difusa ->

$$I = I_a * k_a + I_L * k_d * cos(\theta) = I_a * k_a * I_L * k_d * (N * L)$$

I_L : Intensidad fuente
θ: Ángulo incidencia [0..90]
k_d : Coeficiente reflexión difusa [0..1]
N: Normal superficie (unitario)
L: Vector iluminación (unitario)

Reflexión especular

La reflexión especular es la componente de la luz reflejada sobre una superficie brillante o pulida en una dirección preferente formando un brillo.
La reflexión especular depende de la posición del observador, dada por el vector **V**.
En un espejo perfecto la dirección desde la que se observa el reflejo es la dirección de reflexión perfecta **R**.
En las superficies normales, el brillo decae cuando el observador se aleja de la dirección de reflexión perfecta. Se podría modelar este efecto teniendo en cuenta el ángulo entre **R** y **V**.
Dependiendo de la superficie (grado de pulido) el brillo está más o menos concentrado alrededor del punto donde R y V coinciden.
L, N y R son coplanares, V no.

Modelos de iluminación de Phong

Asume que la máxima reflectancia especular ocurre cuando α es 0 y decae rápidamente conforme α se incrementa.

Ecuación de la intensidad especular reflejada->

$$I_s = I_L \cdot k_s \cdot \cos^n(\alpha) = I_L \cdot k_s \cdot (\vec{R} \cdot \vec{V})^n$$

Ecuación de cálculo de intensidad en un punto ->

$$I = I_a \cdot k_a + I_L \cdot (k_d \cdot (\vec{N} \cdot \vec{L}) + k_s \cdot (\vec{R} \cdot \vec{V})^n)$$

Cálculo de la dirección de reflexión R ->

$$\vec{R} = \vec{N} \cdot \cos(\theta) + \vec{S}$$

$$\vec{S} = \vec{N} \cdot \cos(\theta) - \vec{L}$$

$$\vec{R} = 2 \cdot \vec{N} \cdot \cos(\theta) - \vec{L} = 2 \cdot \vec{N} \cdot (\vec{N} \cdot \vec{L}) - \vec{L}$$

Atenuación del foco

Problema: dos superficies a distinta distancia del foco son irradiadas con diferente intensidad -> esta diferenciación se introduce con un factor de atenuación ->

$$I = I_a \cdot k_a + f_{at} \cdot I_L \cdot \left(k_d \cdot (\vec{N} \cdot \vec{L}) + k_s \cdot (\vec{R} \cdot \vec{V})^n \right) \qquad f_{at} = min\left(\frac{1}{(c_1 + c_2 \cdot d_L + c_3 \cdot d_L^2)}, 1 \right)$$

Atenuación atomosférica

Tener en cuenta la distancia de los objetos al observador (proyección canónica = coordenada Z).
Efectos atmósfera -> interpolando entre una intensidad de fondo predifinida (Ib) y la intensidad calculada (I) utilizando 2 factores de escala Sa y Sb.

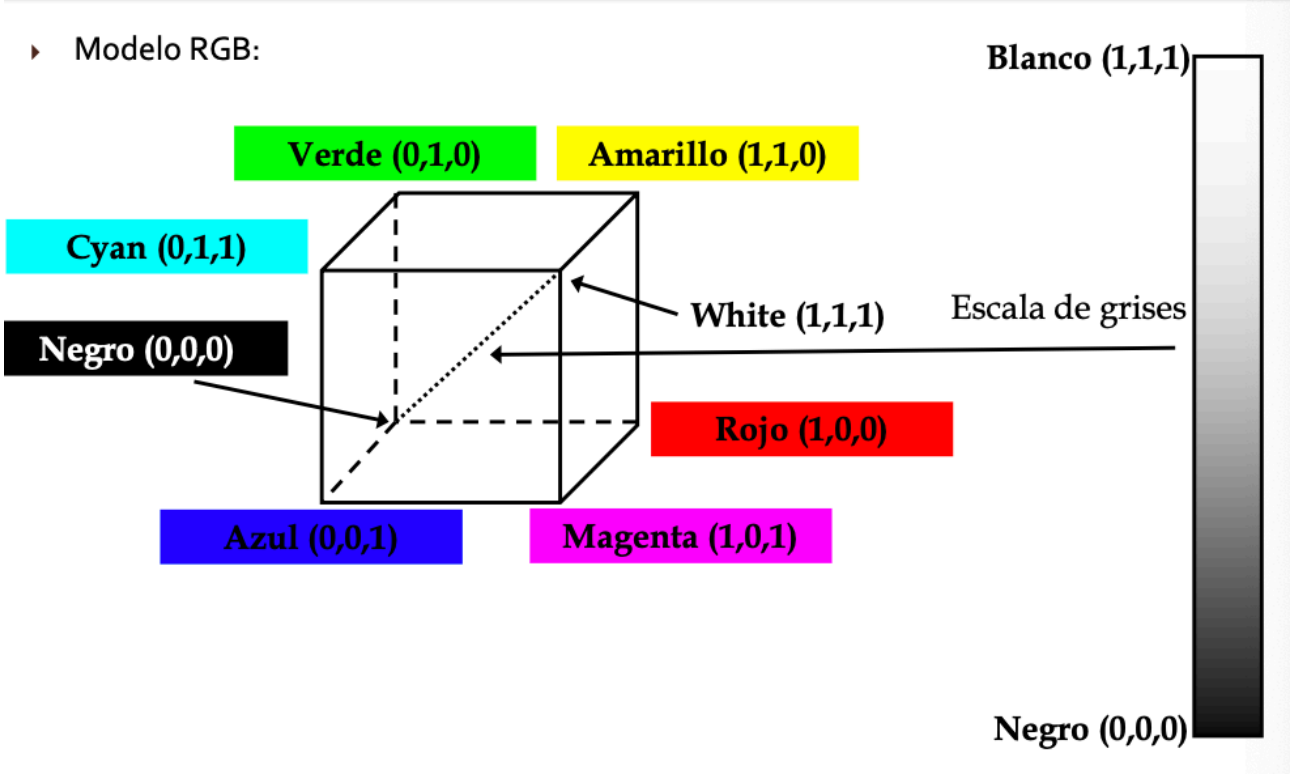
$$I' = (1-S) \cdot I + S \cdot I_B$$

Color en luces y superficies

Dependencia de los parámetros del modelo de iluminación con la longitud de onda de la luz (color) -> parámetros dependientes del color -> intensidades (color espectral de las fuentes) y coeficientes de reflexión (dependen del color de los objetos o de las preferencias de absorción de determinadas longitudes de onda por el material).

Modelo espectral -> $$I(\lambda) = I_{a\lambda} \cdot K_a \cdot O_{d\lambda} + f_{att} \cdot I_{L\lambda} \cdot \left[K_d \cdot O_{d\lambda} \cdot \left(\overline{N} \cdot \overline{L} \right) + K_s \cdot O_{s\lambda} \cdot \left(\overline{R} \cdot \overline{V} \right)^n \right]$$

- Modelo RGB:



Si hay m fuentes de luz, las aportaciones de cada fuente se suman.
Problema -> alguna componente R,G,B se sature por encima de 1 (valor máximo del dispositivo) -> soluciones: truncar vallor de I respecto al máximo / calcular imagen en su totalidad y normalizar las intensidades para que queden comprendidas en el intervalo [0..1].

Modelos de sombreado

Sombreado constante

Cada polígono se sombrea con un color: fuente de luz en el infinto y punto de vista en el infinito -> si alguna de las dos no se cumple -> calcular L y/o V para cada polígono.

Sombreado de Gouraud

Pasos

Calcular las normales para cada polígono.
Calcular las normales para cada vértice.
Interpolar la intensidad para cada vértice.

Sombrear el polígono (p. ej. Utilizando Scan-Line).

Sombreado de Phong

Pasos

Calcular las normales para cada polígono
Calcular las normales para cada vértice
Interpolar las normales para cada píxel
Sombrear cada píxel con las normales interpoladas

Características

Interpola normales
Recalcula I para cada píxel
Es apropiado para reflexión especular
Evita las bandas de Mach

Modelos avanzados: Sombras

Las sombras son necesarias para dar sensación de realismo.
Su cálculo aumenta el tiempo de render sensiblemente.
La sombra no depende de la posición del observador

Problema -> ¿es visible la luz desde el punto donde estoy calculando la iluminación? -> tener en cuenta tamaño forma y distancia de la fuente, umbría y penumbra, iluminación global.

Sombras en síntesis de imagen

Aplicaciones interactivas:

Iluminación local con aprovechamiento de la coherencia
Imágenes fotorrealistas: Iluminación global

Fuentes puntuales en escenas poligonales

Si A y B son dos polígonos, la sombra de A sobre B debida al foco F coincide con la proyección de A con centro de proyección F y plano de proyección B (problema de visibilidad)

Ninguna sombra es vista si el observador y el foco puntual coinciden

Si las fuentes son puntuales no hay penumbra

Efectos prácticos

- Las sombras son independientes del punto de vista
- Modificar el modelo de iluminación local con un factor de bloqueo

Sombras sobre planos

Producción sencilla de sombras sobre un plano

- Premisas
  - Modelo poligonal
  - Sombras sobre el suelo o paredes
  - Objetos aislados
  - Luz direccional
- Método
  - Calcular la proyección de cada polígono sobre el plano desde la fuente
  - Generar un polígono de sombra por cada proyección
  - Introducir los polígonos de sombra en el Z-Buffer
  - Procesar el resto de polígonos. A igual z, mantener el Z- buffer
- Método alternativo
  - Generar una textura para el suelo con las proyecciones (shadow map)

Z-Buffer de dos pasos

- Proceso
  - Calcular el Z-Buffer-Luz desde $\mathbf{L}$
  - Aplica Z-Buffer modificando:
    - Transformar P del sistema de la vista al sistema de la fuente
    - Comparar la z de P en el nuevo sistema con la almacenada en el Z- Buffer-Luz
  - Si la lzl es mayor que la almacenada, el punto está en sombra
- Inconvenientes -> memoria, cálculos innecesarios sobre píxeles repintados, Aliasing espacial, Aliasing numérico

Modelos Avanzados: Trazado de rayos

La luz se considera compuesta de partículas que viajan en línea recta cambiando de dirección en la superficie de los objetos.

Trazado:

Hacia adelante: Trazado desde las fuentes de luz.

Hacia atrás: Trazado desde el observador.

Combinación espacio objeto-imagen.

Discretización del plano de proyecciones (rayo-pixel)

Cálculo de intersecciones en espacio del objeto

Solución primaria: Cálculo de Visibilidad

Cálculo de intersección rayo-objeto + selección del más

Problemas y soluciones:

Cálculo de la intersección: Uso de superficies implícitas f(x,y,z)=0

Alto número de intersecciones: Uso de coherencia para aceleración

Clases de rayos

Primarios: PV-pixel P(t)=PV$\vec{+}$(pixel-PV)*t

Secundarios:

Rayos de sombra: dirección $\mathbf{L}$

Rayos reflexivos : dirección $\mathbf{R}$

Rayos refractivos: dirección $\mathbf{T}$

Algoritmo de Whitted

Árbol de rayos recursivos

Profundidad 1,sólo rayos primarios y sombra

Condiciones de parada:

rayos perdidos: color de entorno, fondo o ambiente

limitación constante: profundidad máxima o saturación de recursos

adaptativa: limitación de la valoración de la contribución de un rayo s obre el primario

Problemas

Refracción:rayos de sombra

Precisión numérica: problema de autosombra

Elevado número de intersecciones por pixel

Rayos Primario, $\mathbf{R}$ y $\mathbf{T}$ = 2^n-1 ; árbol binario de profundidad n

Rayos de sombra = m*2^n-1 ; m = nº luces

Imposibilidad de eliminación de caras traseras

Eficiencia

Cuello de botella del trazado de rayos

Cálculo en 3D de la intersección de un rayo (recta) con una primitiva geométrica
Primitivas: cualquiera que ofrezca un método de cálculo de la intersección con una recta.
Trazado de rayos exhaustivo: cálculo, uno a uno, de la intersección de cada rayo con cada uno de los objetos primitivos. Lineal con el número de objetos.

Problema del "aliasing": 1 rayo (1 muestra) por píxel -> solución: multimuestreo (más de un rayo por píxel).

TEMA 9: MODELADO JERÁRQUICO

Cada objeto se define en su propio **sistema de coordendas local.**
Posteriormente se le aplica la transformación del modelo.
Hasta ahora -> matriz separada para cada objeto -> normalmente los objetos están organizados o agrupados de alguna manera -> es más sencillo situar los objetos de forma relativa.

Inicialmente la **matriz de transformación es la identidad** -> al colocar cada objeto se actualiza -> todas las transformaciones son relativas a las que se han acumulado -> primero se dibujan los objetos contenedores y después los objetos que contienen.

Cuando se quiere dibujar toda la escena es necesario **recuperar estados anteriores** -> no es necesario empezar desde el principio para cada objeto -> en cada fase hay que recordar el estado para poder volver atrás -> *Push()* , *Pop()* -> *PushMatrix()* y *PopMatrix()*.

Escena -> representada mediante un **grafo jerárquico** -> conjunto de objetos organizados en grupos y relaciones mediante transformaciones jerárquicas.
Cada **nodo** -> asociado un sistema de coordenadas local, puede definir una forma que se dibuja en el s.c.local , puede tener hijos que heredan el s.c. y pueden definir objetos o transformaciones.

Nodos más comunes:

Transformación: incluye una transformación local que se acumula a la que hereda.
Instancia: instancia de una geometría -> almacena un puntero a un modelo -> no se sobrecarga el grafo y es posible cambiar la geometría de todas las instancias de forma sencilla. No pueden tener hijos -> siempre son nodos hoja.
Grupo: almacena un conjunto de nodos hijo, no realizan ningún cálculo, se utilizan cuando se necesita tener un padre común para varios nodos.

Modificar una escena consiste en:

Cambiar la estructura del grafo -> añadir, borrar, editar nodos.
Cambiar contenido grafo -> objetos, transformaciones.
Definir subclases para diferentes tipos de nodos -> color, luces, materiales, cámaras, comportamiento, animación, interactividad.

Qué se puede hacer con un grado de escena:

Esqueleto y deformaciones -> objetos articulados, nodos que cambian la geometría de los objetos.
Propiedades de un nodo que modifican valores de otros nodos.
Uso de **nombres** para los nodos.
Calcular transformacioens del modelo -> ***top-down, down-top.***
Cajas de inclusión: muy útiles para visibilidad.
Cálculos de **colisiones**.
Selección de objetos.