Gráficos por ordenador

Joseba Makazaga Ander Murua

Contenido

COMPUTER GRAPHICS

Parte de la informática que se encarga de las aplicaciones que utilizan gráficos, imágenes o dibujos.

1. Aplicaciones que manipulan gráficos ó imágenes.

   utilizan información gráfica

2. Aplicaciones que para mostrar resultados utilizan gráficos.

   Utilización gráfica de la información

Un poco de historia

- Nacimiento lento por carestía de Hard.
- La aparición de pantallas gráficas le dió un gran empuje.
- Su abaratamiento hizo posible su extensa difusión.
- Los estandares han fracasado.
  1. GKS, CORE
  2. PHIGS, PHIGS+
  3. Open GL?
- Hay muchos procesos de bajo nivel por Hard.

Necesidades

- Formatos de imágenes:
  1. Video: 720 x 512 x 24
  2. Se tiene 1024 x 1024 x 96 en cualquier máquina domestica.
- En las animaciones, por lo menos, 20 imágenes por segundo.

HARDWARE

Requisitos:

- Mucha memoria.
- Gran capacidad de cálculo.
- Velocidad de visualización.

SOFTWARE

- Fracaso de los estandares.
- Se ha trabajado a bajo nivel. Xorg y tarjetas gráficas
- Librerias:
  1. Funciones de trazado ó dibujo.
  2. Funciones de entrada-salida.
  3. Funciones que informen sobre el estado.
  4. Independiente del Hard.
  5. Manejo de distintos sistemas de referencia.

Características

- Hay que trabajar con modelos (exactos y/o aproximaciones, posibilidad de crear y manipularlos...)
- Tratamiento. Transformar los modelos.
- Procesos de visualización.
  1. Dibujo de líneas y curvas, relleno de poligonos.
  2. Delimitación de la ventana de proyección.
  3. Cálculo de proyecciones.
  4. Eliminar partes ocultas.
  5. Iluminar.
- Interacción con el ususario.
  1. Facil de usarlo y veloz.
  2. Con posibilidad de corrección.
  3. Inteligente?

LIBRERIAS

- Mayor nivel de abstracción.
- Programas fuente portables.
- Convienen estandares.
- Si hubiera que crearlas, parametrizadas!:
  1. Dibujo y relleno.
  2. Entrada-salida.
  3. Informacion del estado.

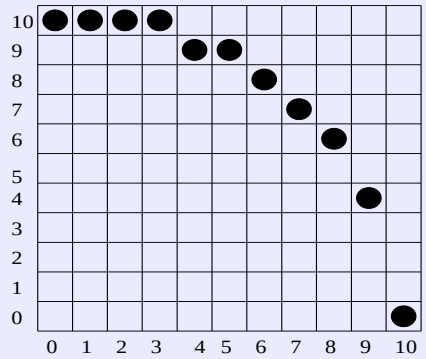
OBTENCION DE LA IMAGEN

1. Representación y manipulación de la escena.
2. Definir la cámara y representarlo todo en su sistema de referencia.
3. Delimitación y obtención de la perspectiva.
4. Eliminación de caras traseras u ocultas.
5. Definición de fuentes de luz y calculo de su repercusión en la escena.
6. Iluminación y sobreado de las superficies.
7. Dibujo.

Contenido

Discretización

- CRT ó pantalla. Formado por pixels.
  1. on-off.
  2. Niveles de gris.
  3. Coloreados.
- Es una matriz, problemas:
  1. Pérdida de información.
  2. Pueden aparacer huecos ó falta de continuidad.
  3. Aliasing. Nuestro cerebro busca símiles (alias).
  4. Gran cantidad de cálculo. Circuitos integrados.



x	0	1	2	3	4	5
y	10	9.9	9.7	9.5	9.1	8.6
biribilduz	10	10	10	10	9	9

6	7	8	9	10
8.0	7.1	6.0	4.3	0
8	7	6	4	0

Líneas y curvas

Algoritmos incrementales para líneas.

- Ecuación explícita.
- Pendiente entre 0 y 1.
- Basandose en el pixel anterior: multiplicación, suma y redondeo.

$$x_i = x_{i-1} + 1$$

$$y(x_i) = mx_i + k$$

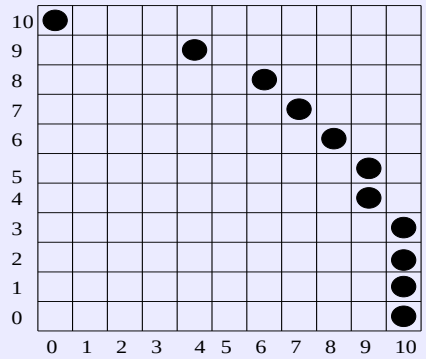
- Uso muy frecuente y debe ser muy eficiente.

$$y(x_{i+1}) = mx_{i+1} + k =$$

$$m(x_i + 1) + k = mx_i + k + m =$$

$$y(x_i) + m$$

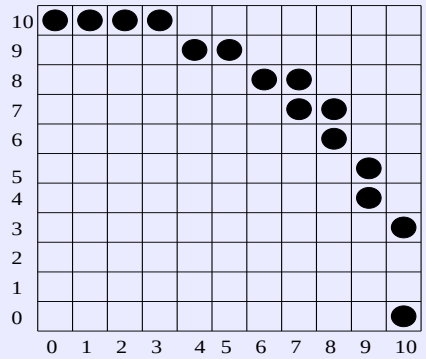
- Pendiente menor que 1!



y	0	1	2	3	4	5
x	10	9.9	9.7	9.5	9.1	8.6
biribilduz	10	10	10	10	9	9

6	7	8	9	10
8.0	7.1	6.0	4.3	0
8	7	6	4	0

No vale para circunferencias, ni duplicando el numero de pixels.



x	0.5	1.5	2.5	3.5	4.5
y	9.9	9.8	9.6	9.3	8.9
redondeo	10	10	10	9	9

5.5	6.5	7.5	8.5	9.5
8.3	7.6	6.6	5.2	4.3
8	8	7	5	4

Algoritmos DDA

- Basándonos en la ecuación diferencial de primer orden.
- Solución mediante métodos numéricos.
- Forma: $y^{'} = F(x, y)$ siendo $y(x_0) = y_0$ conocido.
- Como ejemplo Euler:

$$y^{'} = \frac{\Delta y_i}{\Delta x_i}$$

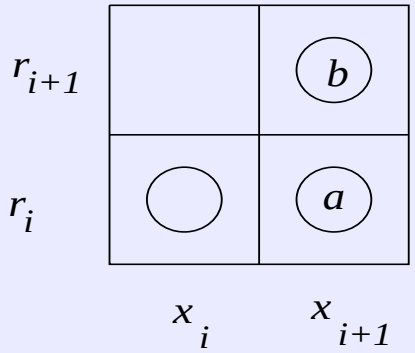
- Por tanto:

$$y_{i+1} = y_i + F(x_i, y_i)\Delta x_i$$

$$x_{i+1} = x_i + \delta x$$

Brasenham ó algoritmo del valor medio

Si la pendiente de la recta esta entre 0 y 1 el siguiente pixel a dibujar se elije entre dos:



Caso a, hay que dibujar el pixel de la derecha:

$$r_{i+1} = r_i \Longleftrightarrow \text{int}(y_{i+1} + 0{,}5) = r_i$$

$$r_i \leq y_{i+1} + 0{,}5 < r_i + 1$$

Pero $r_i = y_i - e_i$ eta $y_{i+1} = y_i + m$

$$y_i - e_i \leq y_i + m + 0{,}5 < y_i - e_i + 1$$

$$-e_i \leq m + 0{,}5 < -e_i + 1$$

$$-0{,}5 \leq m + e_i < 0{,}5$$

$$m + e_i < 0{,}5$$

Por tanto:

$$e_{i+1} = y_{i+1} - r_{i+1} = y_i + m - r_i =$$

$$= e_i + m$$

Caso b, pixel de encima:

$$r_{i+1} = r_i + 1 \Longleftrightarrow \text{int}(y_{i+1} + 0{,}5) = r_i + 1$$

$$r_i + 1 \leq y_{i+1} + 0{,}5 < r_i + 2$$

Pero $r_i = y_i - e_i$ eta $y_{i+1} = y_i + m$

$$y_i - e_i + 1 \leq y_i + m + 0{,}5 < y_i - e_i + 2$$

$$0{,}5 \leq m + e_i < 1{,}5$$

$$0{,}5 \leq m + e_i$$

Por tanto:

$$e_{i+1} = y_{i+1} - r_{i+1} = y_i + m - r_i - 1$$

$$= e_i + m - 1$$

Brasenham. Coma flotante

dibuja-recta(int x_0,int y_0,int x_1,int y_1)
{ int x_i,r_i; double e_i,m;
$m = (y_1 - y_0) \ / \ (x_1 - x_0)$;
$e_i = 0$; $r_i = y_0$;
dibuja(x_0, r_i;)
for($x_i = x_0 + 1$;$x_i < x_1$;x_i++)
{ $e_i += m$;
if ($e_i \geq 0{,}5$)
{ r_i++; e_i--; }
dibuja(x_i, r_i;)
}
}

Brasenham. Enteros

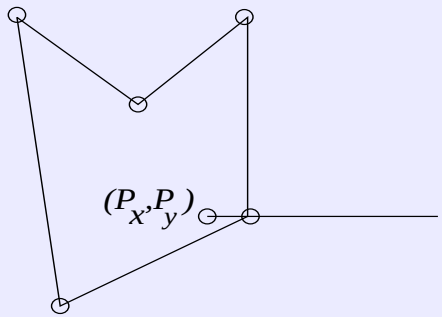
dibuja-recta(int x_0,int y_0,int x_1,int y_1)
{ int x_i,r_i,Δx,Δy,E_i;
$\Delta x = (x_1 - x_0)$; $\Delta y = (y_1 - y_0)$;
$E_i = -\Delta x$; $r_i = y_0$;
dibuja(x_0, r_i;)
for($x_i = x_0 + 1$;$x_i < x_1$;x_i++)
{ $E_i += 2\Delta y$;
if ($E_i \geq 0$);
{ r_i++; $E_i -= 2\Delta x$; }
dibuja(x_i, r_i;) }
}

Relleno de polígonos

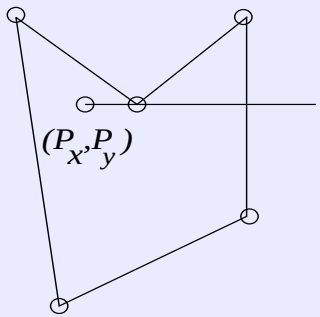
- Dibujar los lados es facil.
- Rellenar el interior no tan facil.
- Punto interno? Teorema de Jordan.

  Si la recta que va del punto al infinito corta un numero impar de veces es interno.

- En la práctica hay problemas:

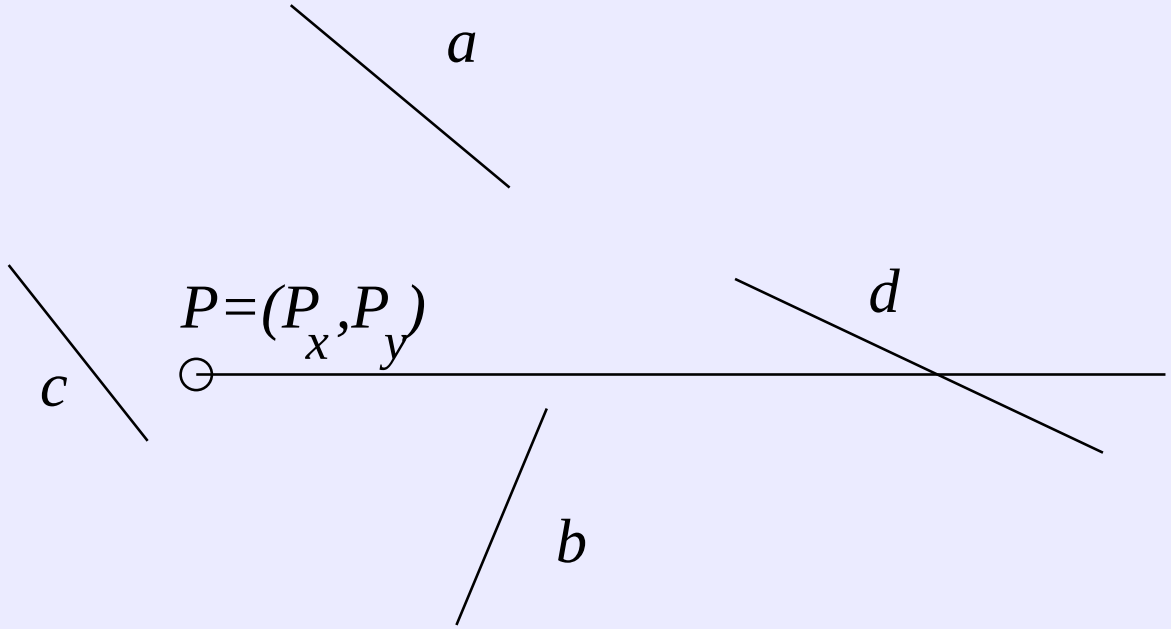


a kasua



b kasua

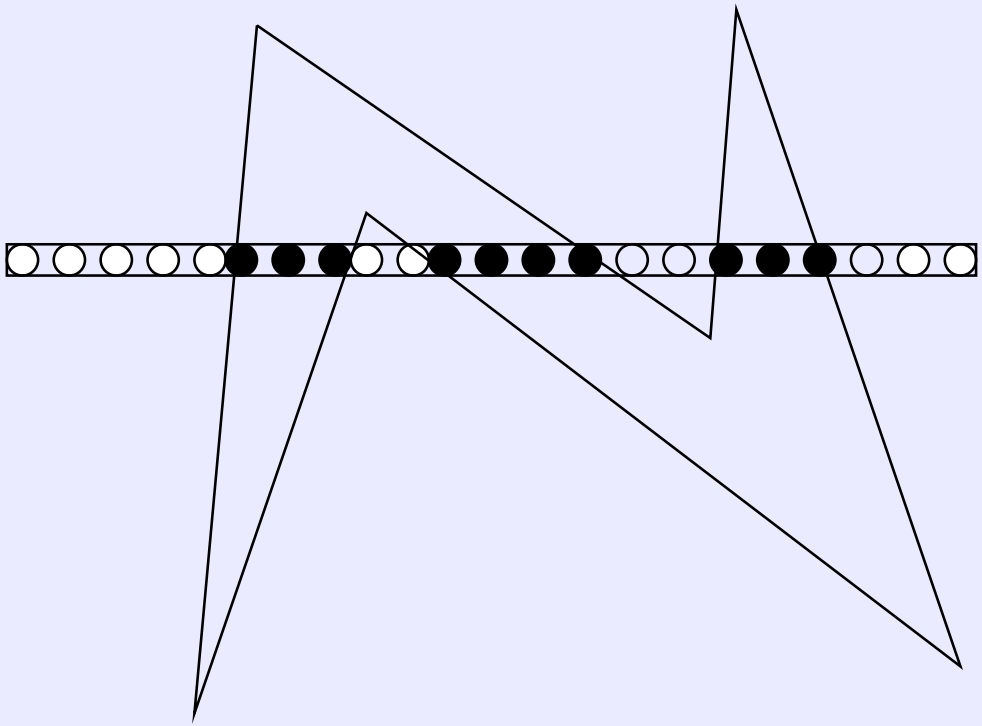
- La mayoría de los casos:



- Solución: En caso de vértice superior contarlo.
- Línea a línea.

Algoritmo línea a línea.

- Tratar las líneas entre Y_{max} e Y_{min}.
- Coherencia de la línea. Secuencias.



- Guardar los cortes y utilizarlos de dos en dos.

Algoritmo línea a línea. II

- Algoritmo:
  1. Obtener los cortes de la línea con el poligono.
  2. Ordenarlos por x.
  3. Tratarlos de par en par.
- Reutilizar informacioón al cambiar de línea?

Spanning Scan Line

```
void nuevo_corte( int *x,int *numerador,int Δx int Δy)
{
*numerador += Δx;
while (*numerador >= Δy)
        {/* puede pasar Δx > Δy */
        *x ++;
        *numerador− = Δy;
        }
}
```

Aliasing

- En base a la discretización de un objeto ver otro distinto.
- Razon principal: representación mediante muestreo.
- Asignamos a todo el pixel el valor que le corresponde al centro.
- En las animaciones pueden aparecer pixels destelleantes.
- Se da mucho en los siguientes casos:
  1. Cambio de intensidad de luz.
  2. Cambio de color.
  3. Parpadeos que pueden ser producidos por objetos pequeños en las animaciones.

Formas de evitar aliasing

- Métodos principales de atialiasing:
  1. sobre muestreo ó supersampling.
  2. Realizar una aproximación de filtros anti-aliasing 2D y eliminar altas frecuencias.
  3. Muestreo estocastico.
- Como el origen es el muestreo conviene que sea lo mas continuo posible. La tecnología es un límite.

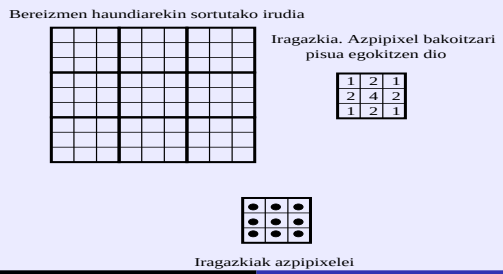
Supersampling

Se realiza en tres pasos:

1. Realizar un muestreo superior a la resolución del dispositivo. Por cada pixel se calculan *nxn*.
2. Aplicar un filtro al muestreo.
3. Obtenemos la imagen a la resolución del dipsositivo.

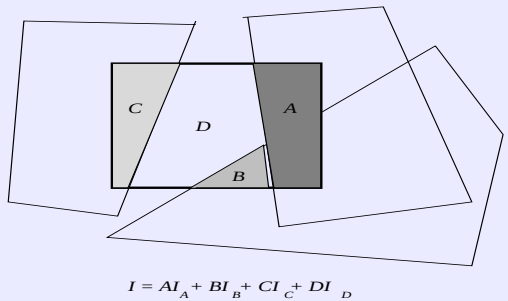
aspectos del supersampling

- A mas subpixels mas cálculos.
- El incremento de calculo es del orden de n^2.
- En la adecuación a pixels reales se puede utilizar un filtro con pesos.
- Asignando pesos mayores a los subpixels centrales se obtienen mejores resultados.
- Para una imagen de 512 x 512 y utilizando pixels de 5 x 5 subpixels habrá que realizar 512 x 512 x 25 multiplicaciones y sumas.
- Puede requerir mucha memoria.



Muestreo de superficies.

- Se basa en la geometríinterna del pixel.
- Tiene en cuenta las partes del pixel que ocupa cada objeto.
- Calculando lo que ocupa cada objeto asigna la intensidad del pixel.



- Cálculos:
  1. Calcular lo que ocupa cada objeto en la superficie correspondiente al pixel.
  2. Determinar la visibilidad de esas partes.
  3. Calcular la intensidad en función de los que son visibles.

Muestreo estocástico.

- Los fotoreceptores del ojo no estan uniformemente distribuidos.
- Cambiamos aleatoriamente los puntos de muestreo.
- pasos:
  1. Obtener muestreo de la imagen asignando a cada punto un cambio aleatorio.
  2. Aplicar un filtro a la muestra del paso anterior a fin de obtener la intensidad de los pixels.
- El cambio introduce ruido en la imagen, pero a su vez disminuye el efecto aliasing.
- Adecuado para trazado de rayos ó Ray-tracing.
- Para Z-buffer y Scanline presenta mas dificultades pero han surgido algoritmos basados en microsuperficies que realizan muestreo estocastico.

Contenido

Transformaciones geomiétricas

- Objetos representados por caras (vértices y aristas).
- Transformaciones a estudiar:
  1. Traslación.
  2. Escalado.
  3. Rotación.
  4. Reflexión.

Utilización de matrices

- Representación matricial:

$$x' = ax_1 + by_1$$

$$y' = cx_1 + dy_1$$

- Linealidad:

$$A(v_1 + v_2) = Av_1 + Av_2$$

$$A(rv) = r(Av)$$

- Consecuencia, las rectas (poligonos) se transforman en rectas (poligonos).
- El origen no se transforma.

Traslación

- A cada punto del objeto hay que sumarle el vector que corresponde a la traslación.
- Si el movimiento fuera $\begin{pmatrix} m \\ n \end{pmatrix}$, la transformación sería:

$$p' = p + \begin{pmatrix} m \\ n \end{pmatrix}$$

- Transforma el origen, lo que imposibilita su representación matricial.
- Mantiene la distancia entre puntos. No deforma los objetos.

Escalado

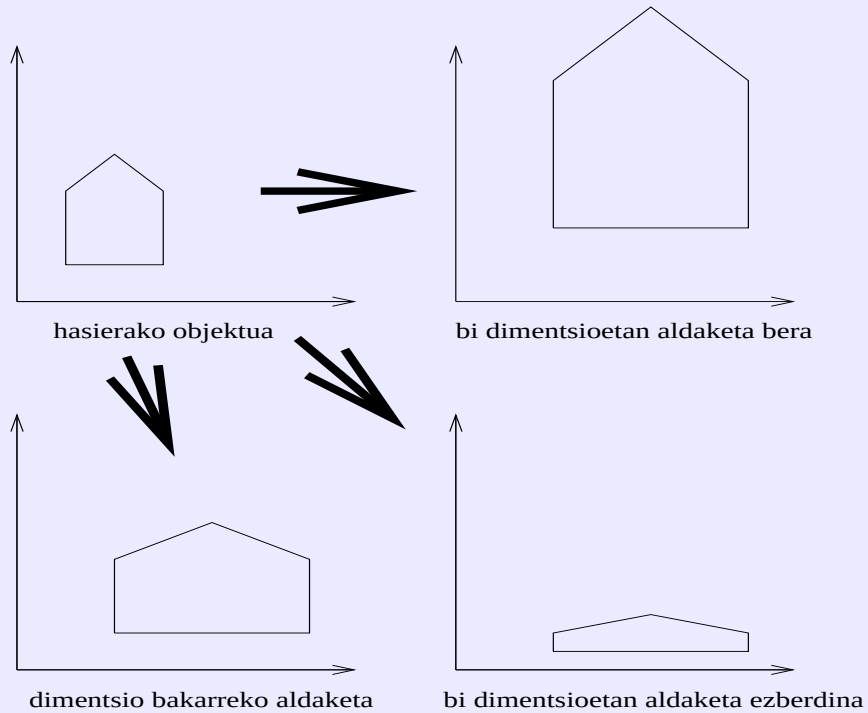
- Para hacer que el objeto sea p veces mayor basta con multiplicar por p las coordenadas. $x' = px$ y $y' = py$
- Admite representación matricial:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} p & 0 \\ 0 & p \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

- Si quisieramos perder la proporcionalidad en las dos dimensiones, habría que utilizar la siguiente matriz:
$\begin{pmatrix} p & 0 \\ 0 & q \end{pmatrix}$
- Deben cumplir la condición $p, q > 0$.
- Si $p, q > 1$ estaríamos incrementando tamaño.
- Si $p, q < 1$ decrementamos tamaño.

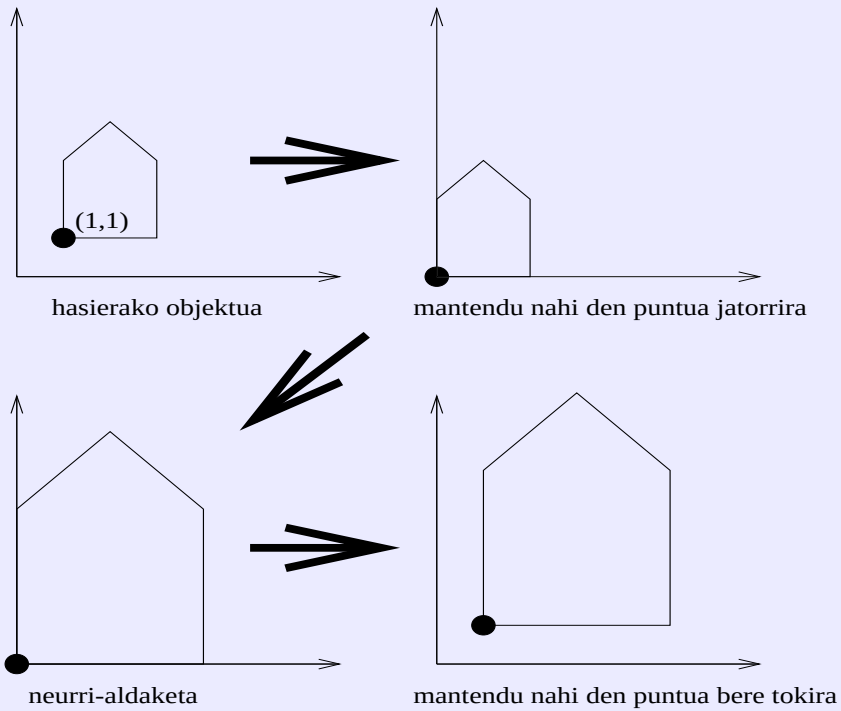
Traslación debida al escalado I

El escalado puede provocar cambio de posición del objeto.



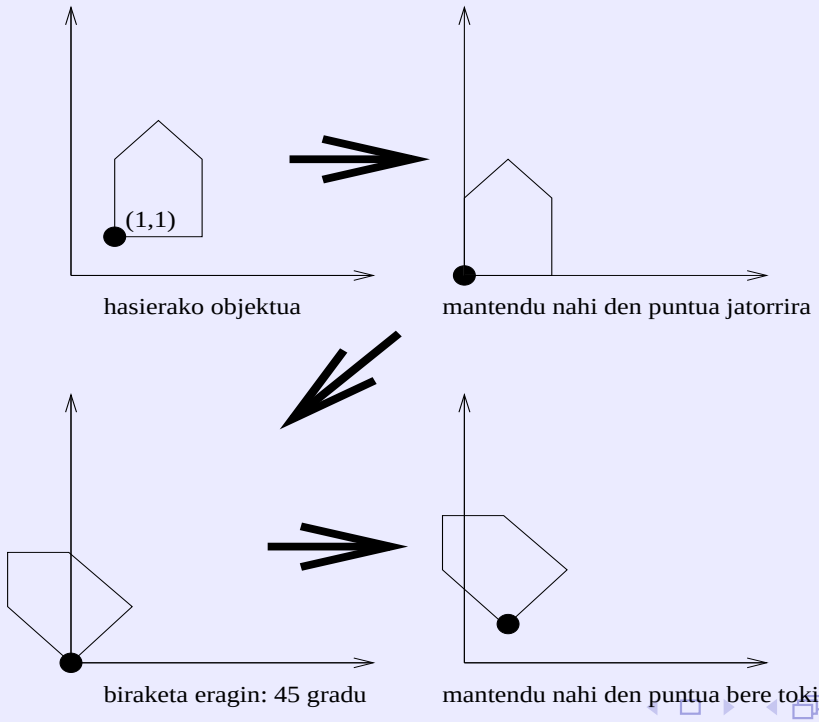
Escalado sin que se mueva el objeto

Para mantener un punto en su posición hay que concatenar tres transformaciones:



Rotación

- Alrededor de un punto: El origen.
- Si es alrededor de otro punto, habrá que realizar un cambio de origen (traslación), rotar y deshacer el cambio de origen (traslación).



Rotación: coordenadas polares

- Se ve mejor con coordenadas polares: $P = \begin{pmatrix} L \\ \alpha \end{pmatrix}$
- Para volver a las coordenadas normales:

$$P = \begin{pmatrix} L\cos\alpha \\ L\sin\alpha \end{pmatrix}$$

- El punto tras rotar β grados:

$$\begin{pmatrix} L\cos(\alpha+\beta) \\ L\sin(\alpha+\beta) \end{pmatrix}$$

- Con lo que el punto queda como:

$$x' = L(\cos\alpha\cos\beta - \sin\alpha\sin\beta)^1$$

$$y' = L(\cos\alpha\sin\beta + \sin\alpha\cos\beta)^2$$

Rotación: representación matricial

- Visto de otra forma:

$$x' = x\cos\beta - y\sin\beta$$

$$y' = x\sin\beta + y\cos\beta$$

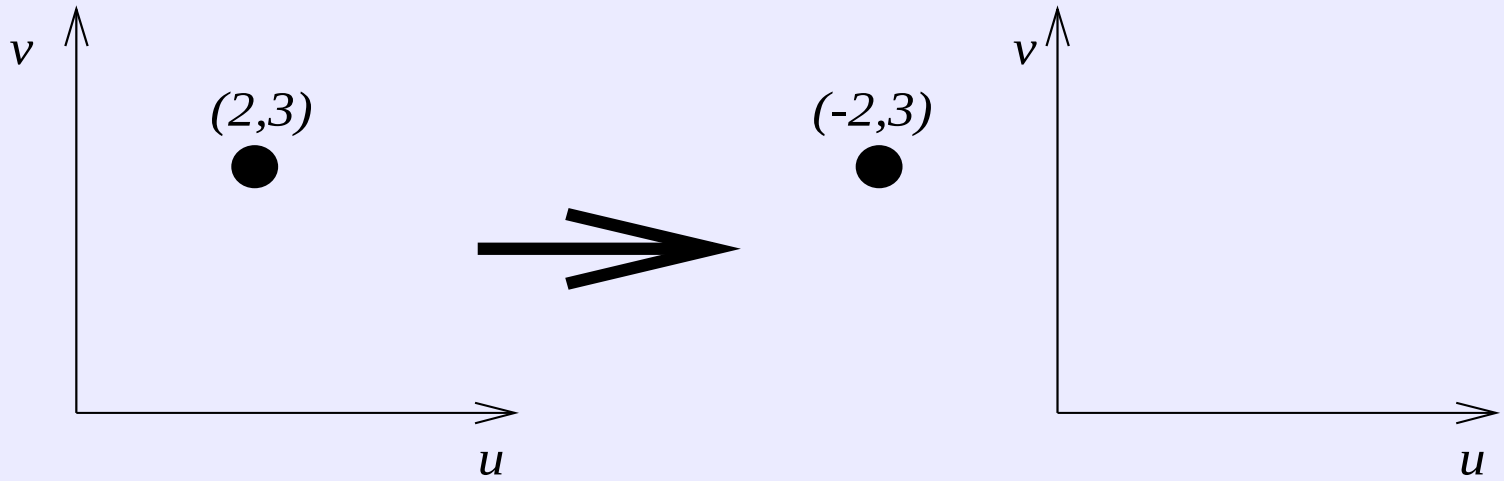
- Su representación matricial:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos\beta & -\sin\beta \\ \sin\beta & \cos\beta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

- Se trata de una matriz ortonormal, y una propiedad es $A^{-1} = A^t$. Además mantiene las distancias entre puntos.

Reflexión.

- Hay que obtener la figura simetrica respecto a una recta.
- Supondremos que pasa por el origen.



Reflexión: representación matricial

- Podemos usar las matrices reflectantes de Householder:

$$A = I - 2uu^i$$

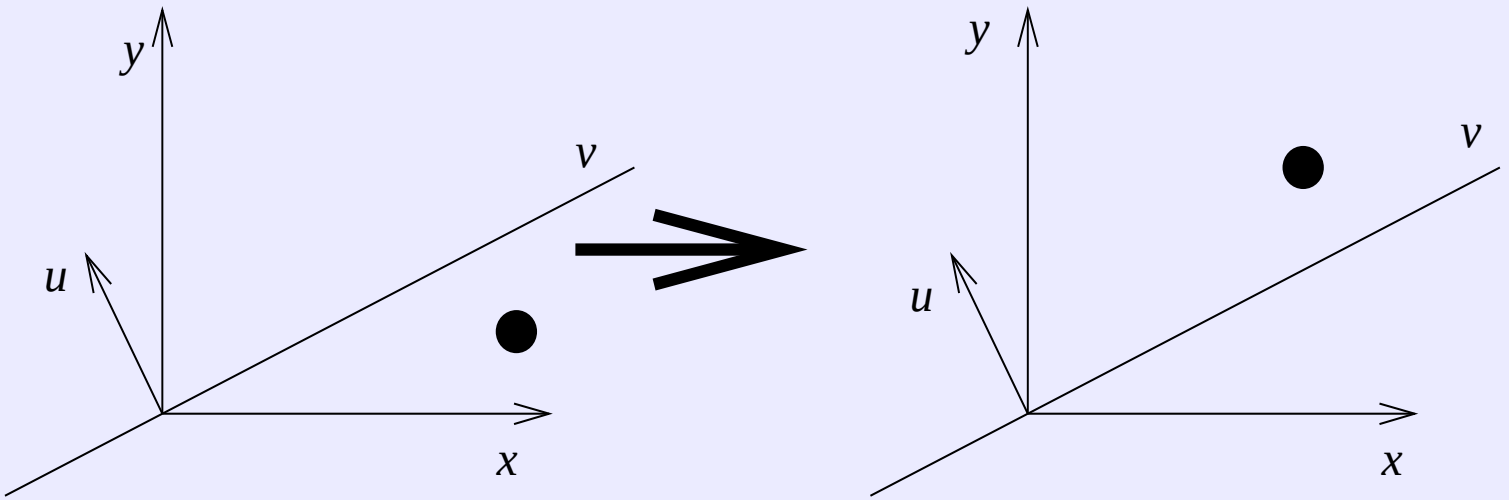
- Tienen dos propiedades:
  1. $Au = -u$
  2. Si el vector v es perpendicular respecto a u: $Av = v$
- Si u y v fueran vectores de la base de nuestro sistema de referencia la matriz A obtendrá el reflejo respecto el vector v.

- Reflexión respecto a cualquier eje: suponiendo que el eje tiene una inclinacion de α grados respecto a la horizontal, sería $\begin{pmatrix} \cos\alpha \\ \sin\alpha \end{pmatrix}$.
- Necesitamos el vector u que es perpendicular al eje: $\begin{pmatrix} -\sin\alpha \\ \cos\alpha \end{pmatrix}$.
- La matriz reflectora será:

$$A = I - 2uu^i = \begin{pmatrix} 1 - 2\sin^2\alpha & 2\sin\alpha\cos\alpha \\ 2\sin\alpha\cos\alpha & 1 - 2\cos^2\alpha \end{pmatrix}$$

Sustituyendo $\sin 2\alpha$ y $\cos 2\alpha$:

$$A = \begin{pmatrix} \cos 2\alpha & \sin 2\alpha \\ \sin 2\alpha & -\cos 2\alpha \end{pmatrix}$$


y
v
u
x
y
v
u
x

concatenación de transformaciones

- Una secuencia de transformaciones provoca una cierta transformación a un punto (el orden de la secuencia tiene su importancia), y esa transformación puede representarse de una forma muy simple: $\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} m \\ n \end{pmatrix}$.
- Con todo ello podemos asegurar que una recta se transforma en otra recta. Además, rectas que se cruzan seguiran cruzandose tras la transformación, y las paralelas seguiran siendo paralelas.

Coordenadas homogeneas.

- En el Espacio Afin no queda claro la diferencia entre punto y vector.
- En coordenadas homogeneas el punto se represnta como $\begin{pmatrix} hx \\ hy \\ h \end{pmatrix}$ ($h = 1$ en la mayorí de los casos). El vector será $\begin{pmatrix} x \\ y \\ 0 \end{pmatrix}$.
- Hace posible la unificación de representación de todas las transformaciones vistas: la traslación se puede realizar multiplicando una matriz.

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & m \\ 0 & 1 & n \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Transormaciones 3D

- Generalización de 2D.
- Afinidad: el plano se transforma en plano.
- Uso de coordenadas homogéneas.

Traslación

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & m \\ 0 & 1 & 0 & n \\ 0 & 0 & 1 & o \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

Escalado

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} p & 0 & 0 & 0 \\ 0 & q & 0 & 0 \\ 0 & 0 & r & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

Mantiene el origen, pero el resto de puntos los transforma. Al escalarlos puede parecer que el objeto se mueve, si se quiere mantener algun punto habrá que concatenar distintas transformaciones.

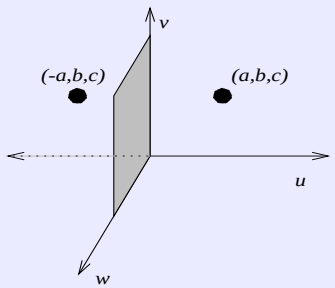
Reflexión

Matrices reflectoras de Housenholder, pero respecto a un plano:

$$A = I - 2uu^i$$

el vector u perpendicular al plano. Si $u = \begin{pmatrix} u_x \\ u_y \\ u_z \end{pmatrix}$

$$A = \begin{pmatrix} 1-2u_x^2 & -2u_xu_y & -2u_xu_z \\ -2u_xu_y & 1-2u_y^2 & -2u_yu_z \\ -2u_xu_z & -2u_yu_z & 1-2u_z^2 \end{pmatrix}$$



Rotación. I

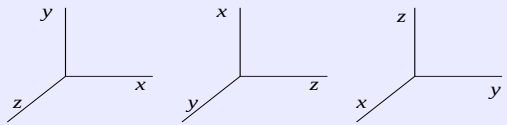
- Eje de rotación. Puede ser del sistema de referencia o cualquiera.
  respecto al eje w:

$$u' = u\cos\alpha - v\sin\alpha$$

$$v' = u\sin\alpha + v\cos\alpha$$

$$w' = w$$

  w puede ser cualquiera de los tres de nuestro sistema de referencia.



Rotación. II

- rotación respecto a z

$$x' = x\cos\alpha - y\sin\alpha$$
$$y' = x\sin\alpha + y\cos\alpha$$
$$z' = z$$

Su representación matricial será:

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} \quad (1)$$

Rotación. III

- rotación respecto a x

$$y' = y\cos\alpha - z\sin\alpha$$

$$z' = y\sin\alpha + z\cos\alpha$$

$$x' = x$$

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} \quad (2)$$

Rotación. IV

- rotación respecto a y

$$z' = z\cos\alpha - x\sin\alpha$$

$$x' = z\sin\alpha + x\cos\alpha$$

$$y' = y$$

visto de otra forma:

$$x' = x\cos\alpha + z\sin\alpha$$

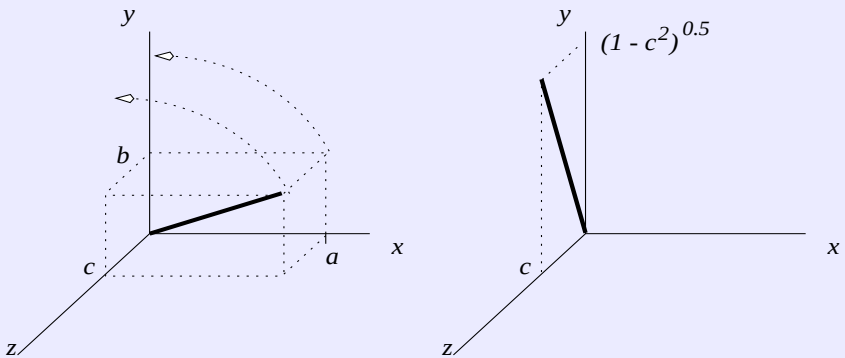
$$y' = y$$

$$z' = -x\sin\alpha + z\cos\alpha$$

Rotación. V

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} \cos\alpha & 0 & \sin\alpha \\ 0 & 1 & 0 \\ -\sin\alpha & 0 & \cos\alpha \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

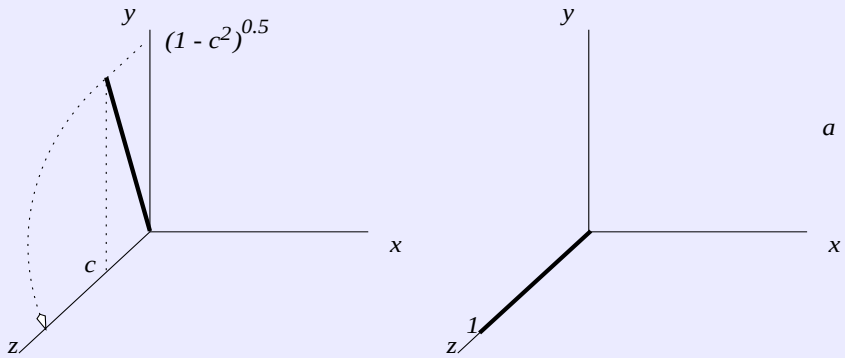
- Rotación respecto a cualquier eje. Podemos basarnos en las rotaciones vistas, hay que conseguir que el eje de rotación sea uno de los tres del sistema de referencia.
  1. Primera rotación:



Rotación. VI

Asi el eje se situa en el plano YZ (rotación respecto a z).

2. Segunda rotación: lo convertimos en eje de nuestro sistema de referencia.



Se trata de rotación respecto a x.

Ahora se puede realizar la rotación respecto a z (eje de rotación), y finalmente deshacer las dos primeras rotaciones para dejar el sistema como al principio pero con la rotación respecto al eje realizada.

Rotación. VII

1. deshacer rotación respecto a x.
2. deshacer rotación respecto a z.

En general, para rotar el punto P necesitamos 5 rotaciones:

$$P' = A^i B^i CBAP$$

Podemos multiplicar las 5 matrices para obtener una unica que realice todo el trabajo.

Contenido

Representación

Los objetos se pueden representar de muchas formas. Para elegir la representación que más nos convenga hay que mirar:

- Hardware, algoritmos, estructura de datos.
- El coste de la obtención de imagen.
- La resolución que requiera la imagen.
- La facilidad de edición del objeto.

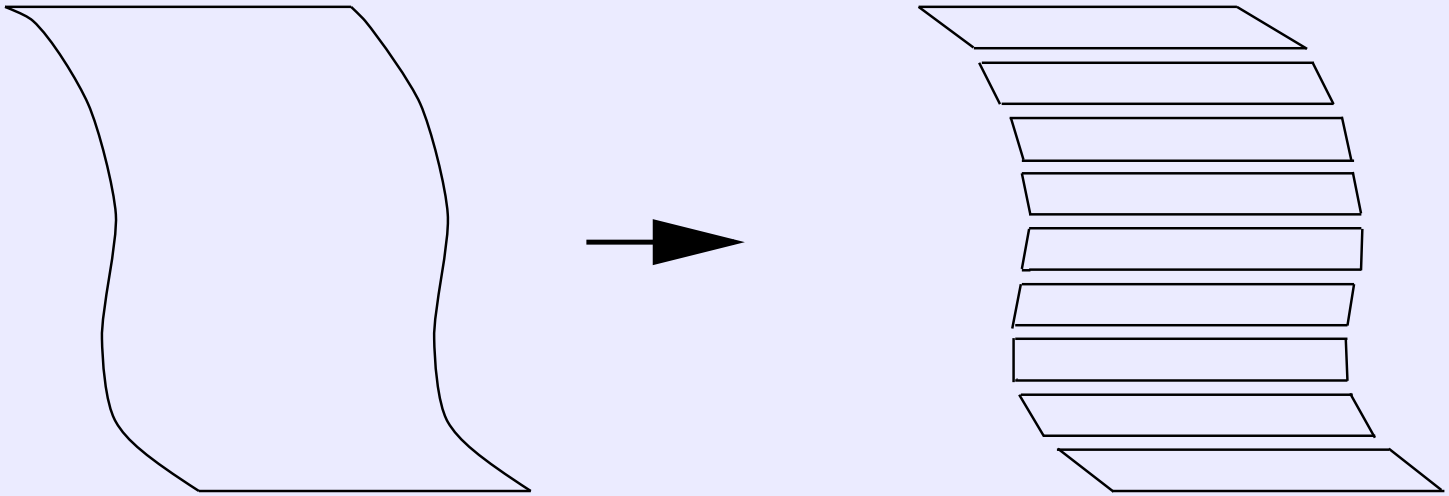
Tipos

1. **Representación poligonal.**
2. **Representación por parches bicúbicos paramétricos.**
3. **Representación por Geometría Sólida constructiva ó CSG.**
4. **División espacial.**
5. Tambien se pueden utilizar funciones matemáticas. Por ejemplo:

$$x^2 + y^2 + z^2 = r^2$$

Representación poligonal

- Red de poligonos. Descripción geométrica y topológica de los límites.
- Para representar objetos curvos se utilizan aproximaciones. Para no perder calidad de las formas habrá que utilizar muchos polígonos.

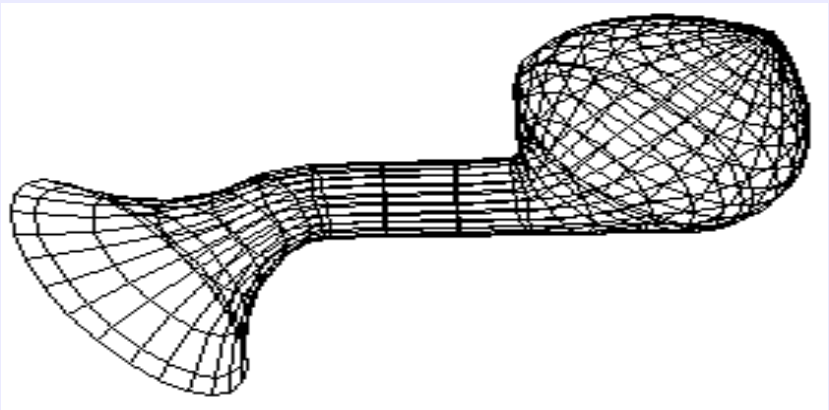


Características de la representación poligonal

- Muchos algoritmos de sombreado juegan a favor de esta representación. Para dar sensación de volumen se utiliza la interpolación.
- A menudo se utiliza esta representación para el rendering, ya que la generación de la imagen es muy simple y obtiene resultados aceptables.
- Se puede tratar cada polígono independientemente. Adecuado para Z-buffer.
- La representación básica se limita a poligonos formados por vértices. Además podemos tener información adicional.
- Se pueden organizar los polígonos para mejorar el tratamiento.
- En el caso de imágene alámbricas podemos basarnos en las aristas.

Creación del objeto

- A mano o bien a base de digitalizadoras. Los vértices pertenecen al objeto pero los plígonos son aproximaciones a la superficie del objeto.
- Automatiamente: Scanner (laser).
- Matemáticamente: Trasladando una sección a lo largo de una curva. Pueden surgir problemas de resolución (toroide).
- Moviendo secciones y transformando la seccion a moverlo.



Redes de parches bicúbicos.

- Un parche define todos los puntos de una superficie curva.
- La función $Q(u, v)$ es polinómica y $0 \leq u, v \leq 1$
- Se suelen utilizar 16 puntos de control para la obtención de los coeficientes. Estos definen la forma del parche.
- Los parches vecinos, para mantener la uniformidad del objeto, requieren continuidad en los puntos de union.
- Necesitan menos memoria que la representación poligonal.
- Se pueden subdividir hasta lograr la resolución necesaria. De esta forma se realizan cálculos en funcion de la resolución en la imagen. Al contrario de los poligonos que para lograr una resolución alta necesitan una red muy precisa.

Desventajas de la representación

- Dificultad a la hora de crear la estructura de datos.
- Los 16 puntos de control han de cumplir ciertas condiciones para mantener la continuidad entre parches adyacentes.

Ventajas para CAD

- Posibilidad de editar objetos ya existentes de forma intuitiva. Aunque requieren cierto conocimiento.
- Definen la forma exacta de los objetos, y permiten el cálculo de propiedades como masa, volumen ....
- Necesitan poca memoria.

Construcción de redes de parches.

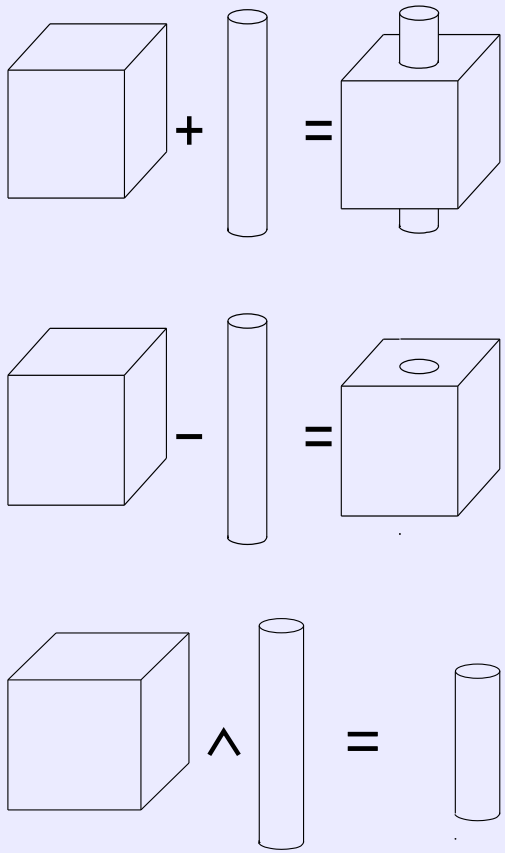
Se usan, sobre todo, dos técnicas:

- **Ajuste de superficies:** Es un método de interpolación. El resultado será una aproximación del objeto, pero cumple condiciones de continuidad.
- **Movimiento de secciones:** Se trata de una superficie obtenida mediante el movimiento de una curva cubica. El propio movimiento puede definirse mediante otra curva cúbica. En cada paso del movimiento habrá que calcular los parches de la superficie.

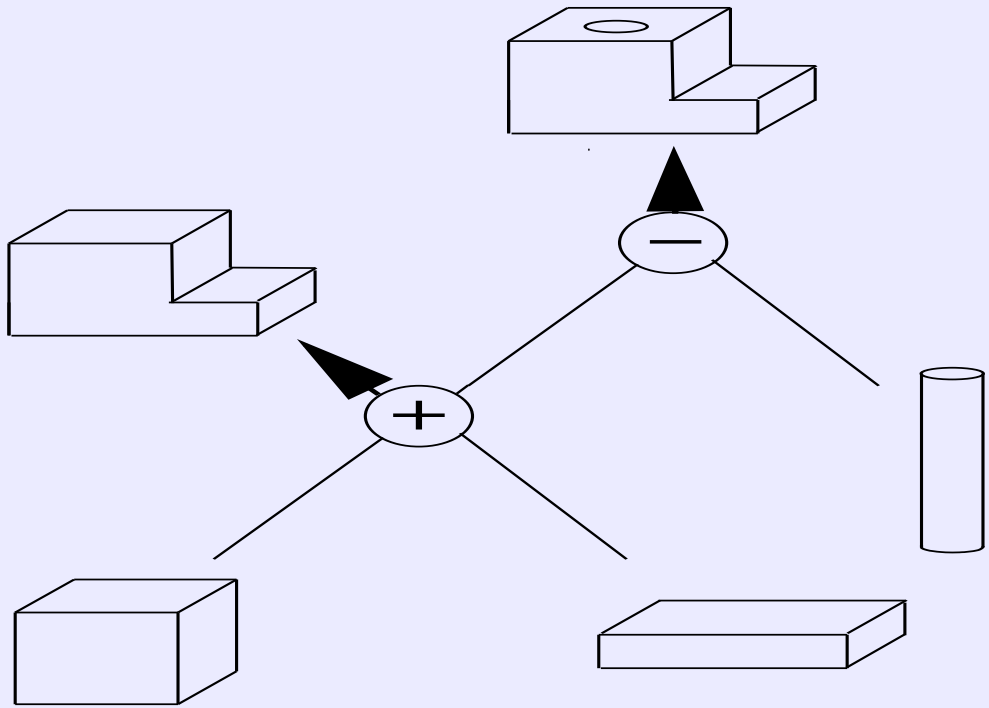
CSG

- El objeto se representa mediante un arbol.
- Las hojas representan objetos básicos.
- Los nodos internos representan operaciones entro los subobjetos.
- Como objetos básicos podemos usar esferas, cubos, conos . . . .
- Las operaciones pueden ser transformaciones u operaciones booleanas.

CSG: operaciones



CSG: ejemplo



Desventajas del CSG

- La obtención de la imagen requiere mucho tiempo.
- Las operaciones son globales al objeto, no pudiendo realizar operaciones a partes del objeto.

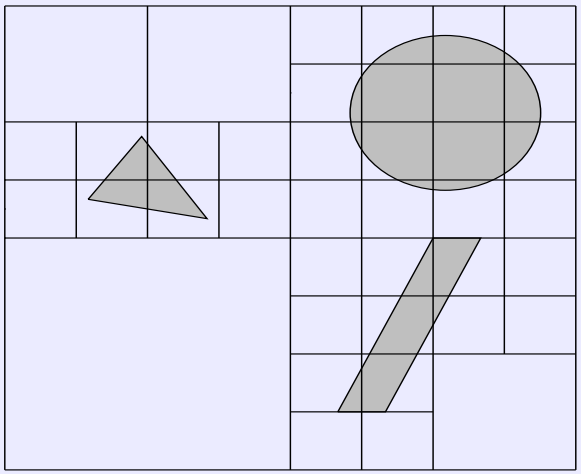
De todas formas, la creación de objetos es muy facil con esta representación.

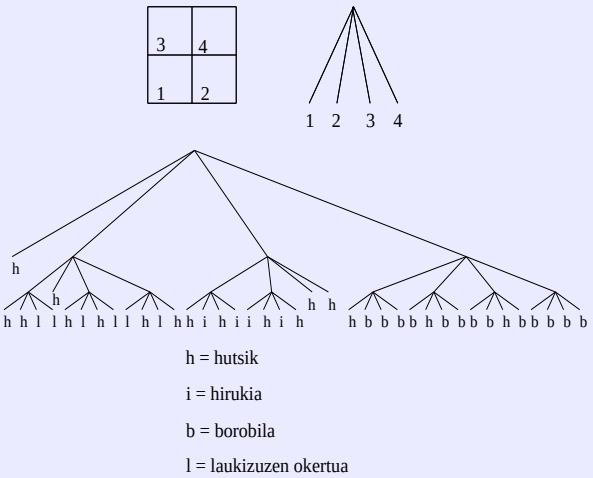
Partición espacial

- El espacio se divide en voxels.
- En cada voxel se guarda el objeto que lo ocupa.
- Cada objeto ocupará varios voxels.
- Requiere mucha memoria.
- Existen varias estructuras, y se utilizan como estructuras auxiliares.
- Se adecuan mucho al ray tracing.

Octrees

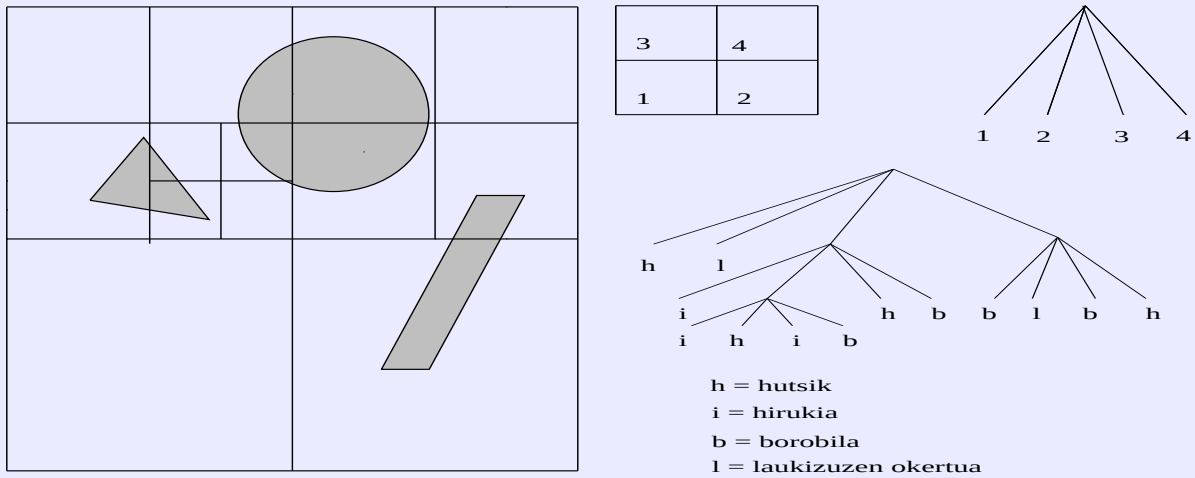
- Se trata de una estructura de árbol.
- Cada unidad espacial se divide en 8, por tanto cada nodo tiene 8 subnodos.,
- Cada división corresponde a un octante.
- En dos dimensiones sería quadtree.





Octrees: la subdivisión espacial

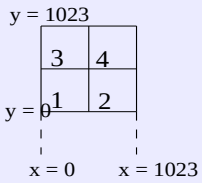
- La subdivisión espacial se debe realizar en función de la resolución deseada.
- Las zonas sin objetos no hace falta dividirlas.

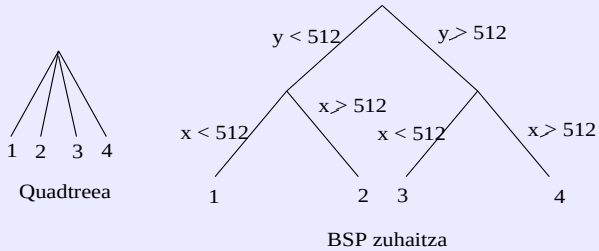


- Solo tiene en cuenta los limites del objeto.
- En general, una hoja del árbol puede tener una lista de objetos.

Arboles BSP

- Se trata de una estructura de árbol.
- El espacio se divide siempre en dos.
- Para la división se utilizan planos.

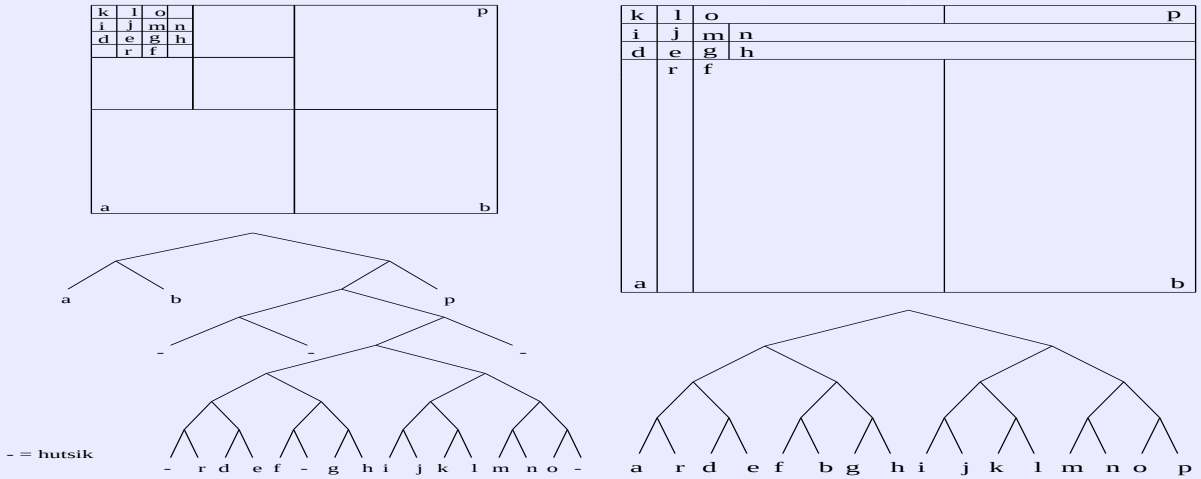




- Se adecua al ray tracing. El camino recorrido por un rayo se traduce a movimientos en la estructura del árbol, y en cada nodo habrá que testear si el rayo intersecta con los objetos que hay en esa zona.

subdivisión adecuada

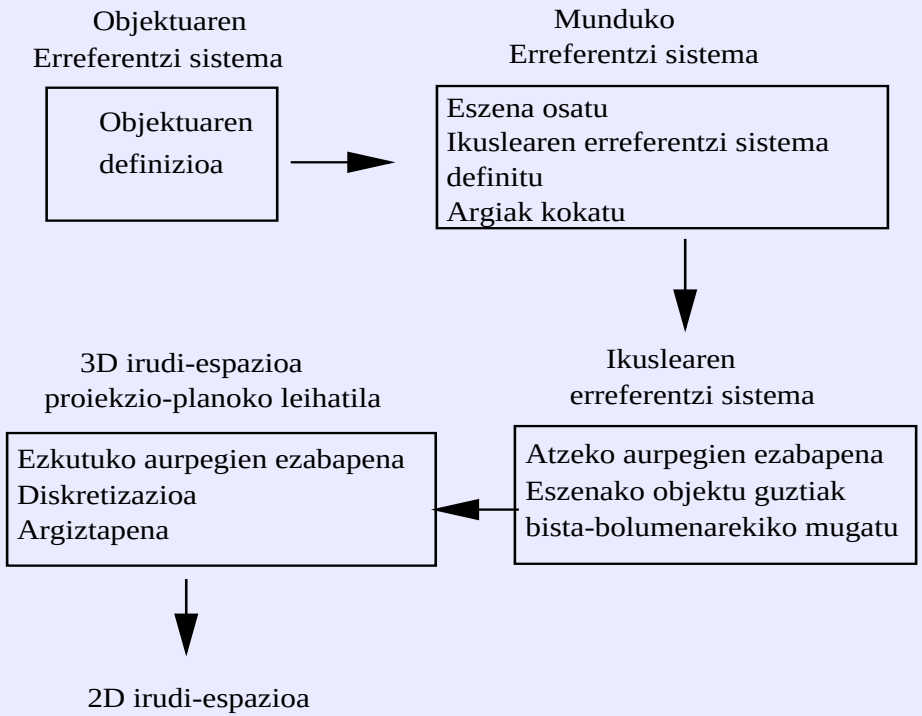
- El plano para la división puede ser cualquiera.



- Se obtienen álboles más equilibrados, de menor profundidad.

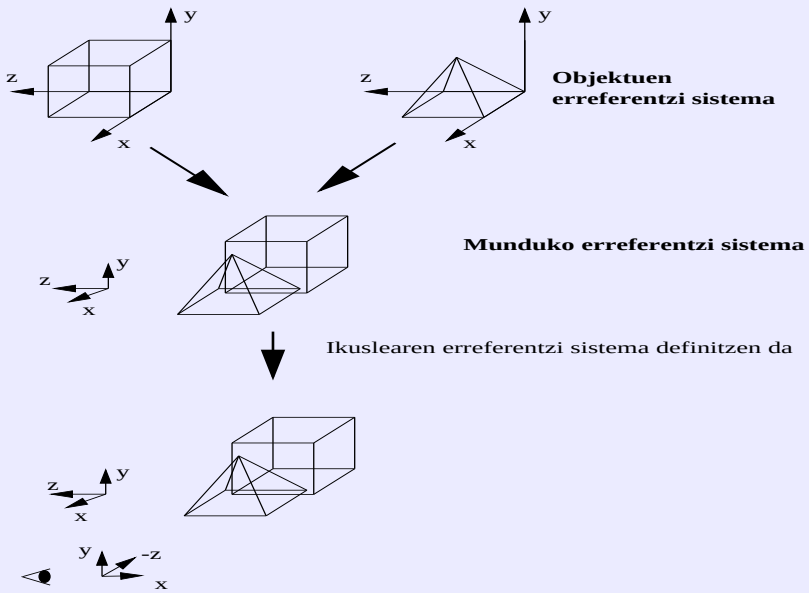
Obtención de la imagen mediante B-rep

- La entrada es una lista de polígonos, la salida un conjunto de pixels coloreados.
- Se pasan distintos sistemas de referencia.



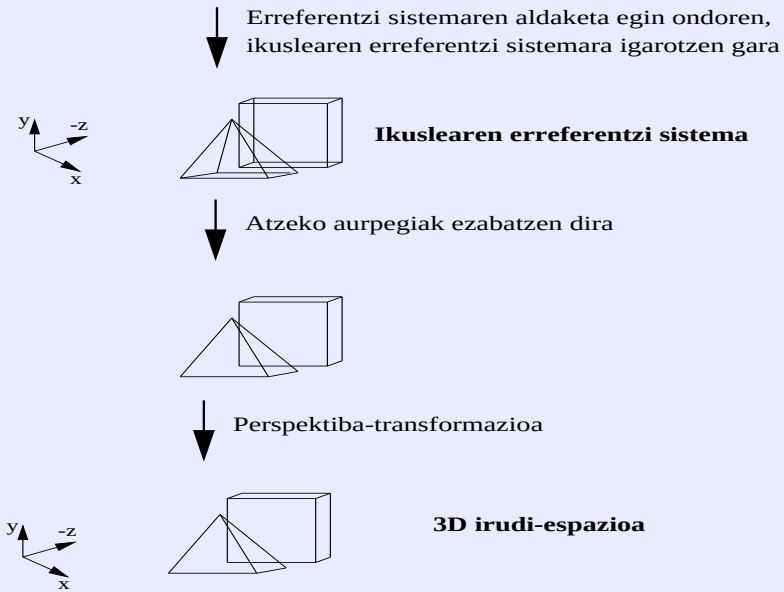
Obtención de la imagen mediante B-rep I

- Los objetos tienen su Sist. de Ref.
- Todos se introducen en un mundo con el sist. de Ref. de la escena.
- En ese sistema se especifican las luces, el observador ....



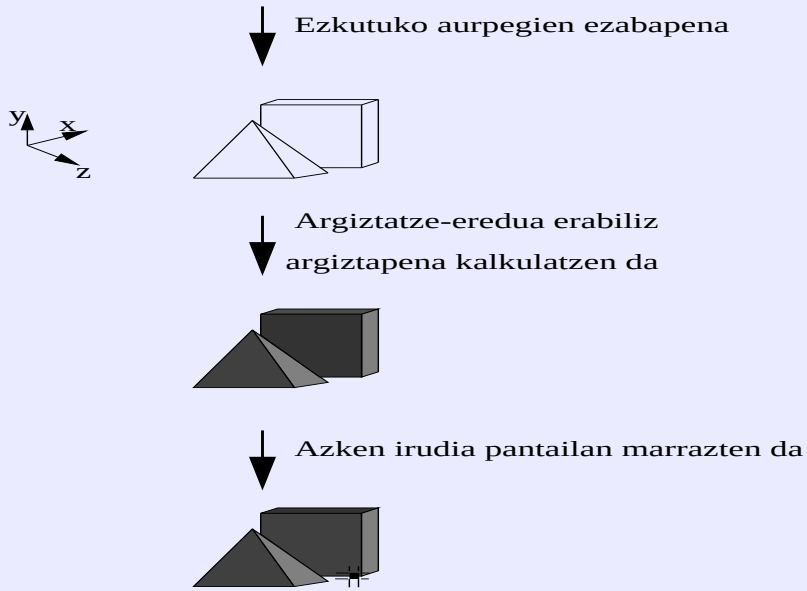
Obtención de la imagen mediante B-rep II

- Se establece el Sist. de Ref. del observador.
- Se pasa la escena al Sist. de Ref. de la cámara, se eliminan caras traseras y se realiza el recorte.



Obtención de la imagen mediante B-rep III

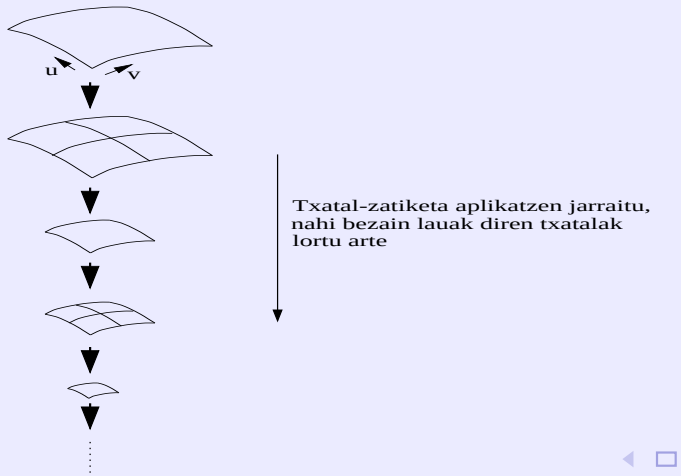
- Se eliminan las partes ocultas y se calcula la iluminación.
- Al final se pasa a pixels.



- Los últimos pasos se pueden dar de muy distintas formas, hay muchos algoritmos.

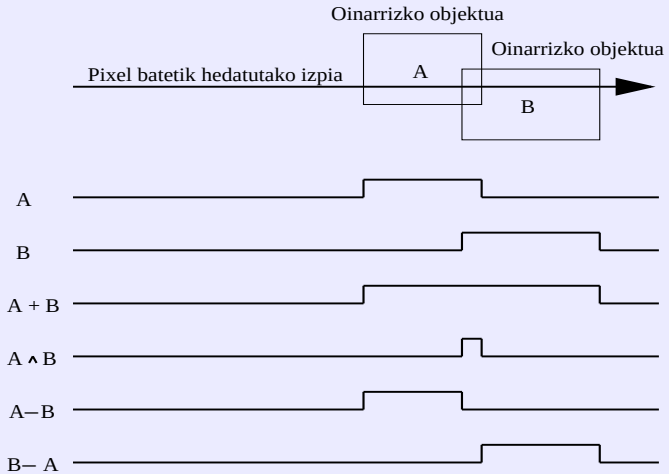
Obtención de la imagen de los parches

- Lo más facil es pasar los parches a polígonos.
- Los parches pueden tener cualquier resolución, pero los podemos subdividir hasta el tamaño deseado
- Los pequeños parches los tratamos como poligonos planos
- El número de poligonos supera al de parches.
- Para terminar la subdivisión hay que establecer condiciones de planaridad.



Obtención de la imagen de CSG

- Aunque la creación de objetos sea facil la obtención de la imagen es complicada.
- El problema radica en la obtención de los bordes del objeto. Tres técnicas:
  1. ray-tracing para CSG



  2. Trasladar de CSG a voxel y utilizar alguna técnica para volúmenes.
  3. Utilizar Z-buffer.

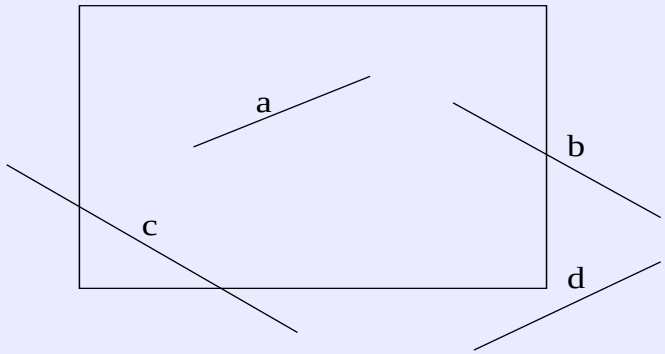
Contenido

Sistemas de Visualización

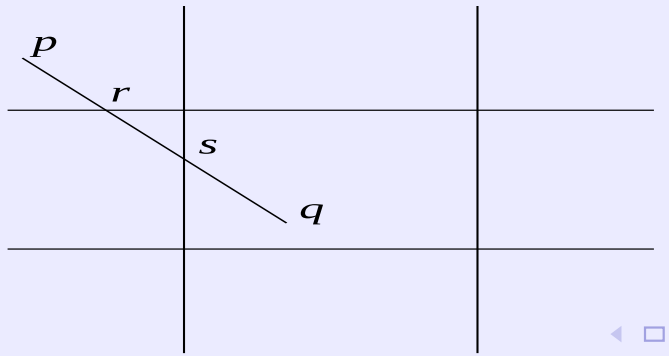
- Hay que pasar de 3D a 2D.
- De 2D hay que llevarlo a la ventana de la pantalla. Habrá que especificar el rectángulo que queremos que aparezca en la pantalla.
  1. Clipping ó recorte: hay que recortar lo que queremos que aparezca.
  2. Lo recortado hay que pasarlo al sistema de referencia de la pantalla.

Recorte de lineas

Recorte de lineas: algoritmo Cohen-Sutherland.

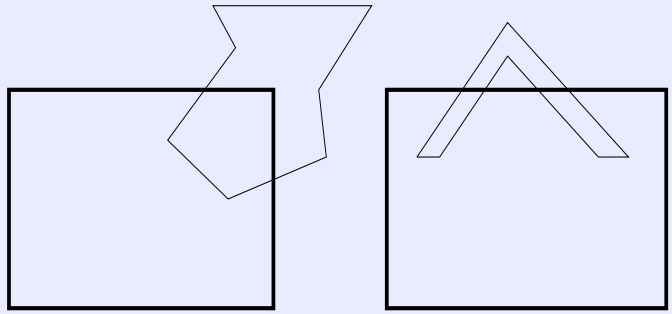


Divide el espacio en 9 zonas y facilita el trabajo utilizando ciertos codigos.



recorte de polígonos

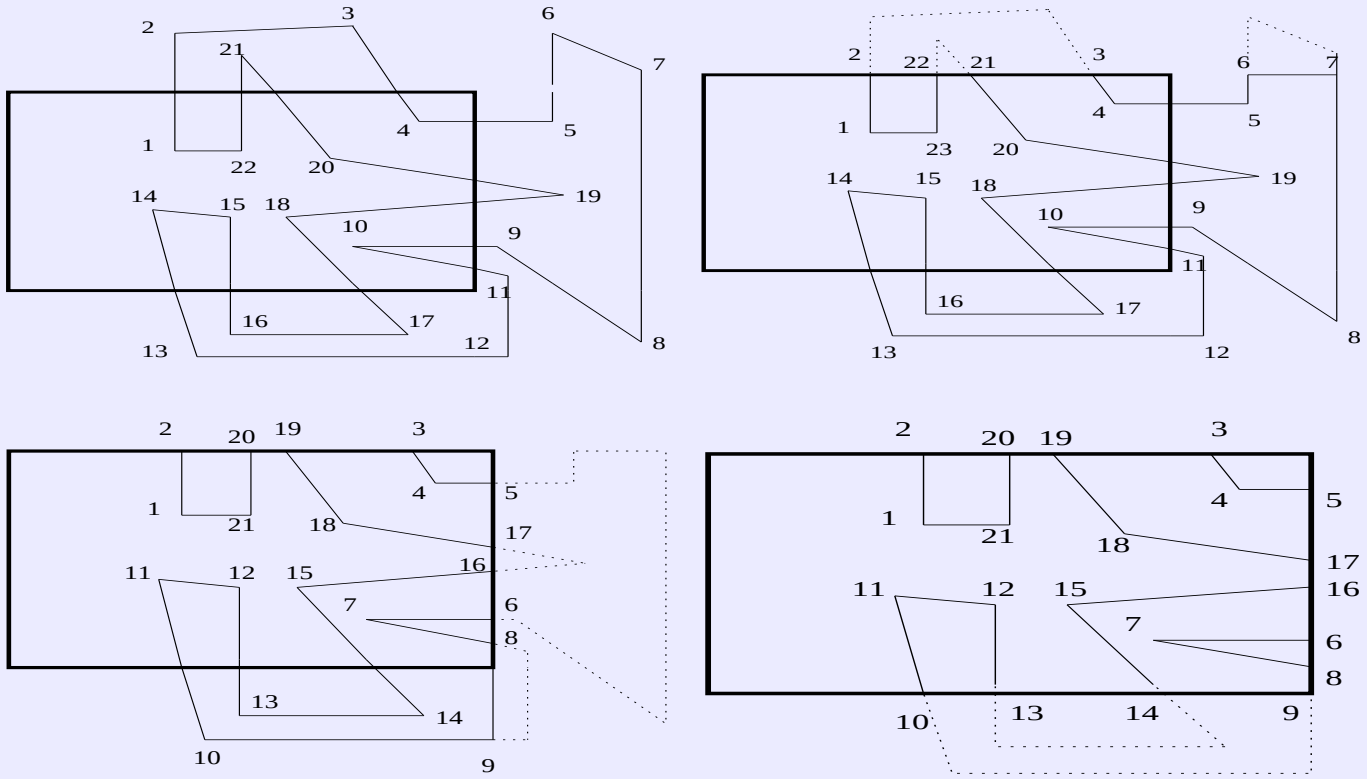
Recorte de poligonos: algoritmo Sutherland-Hodgman.
Si hay que rellenar el poligono, el algoritmo anterior puede dar como resultado poligonos no cerrados y habrá que evitarlo.



(a) (b)

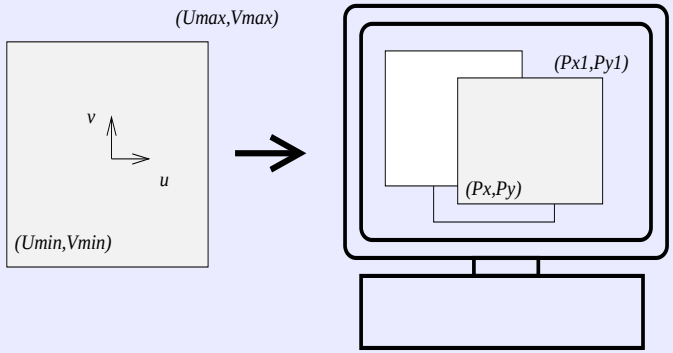
recorte de polígonos

El algoritmo parte de un poligono y tras recortarlo devuelve otro poligono.



Encuadre

- Se trata de un cambio de sistema de referencia.



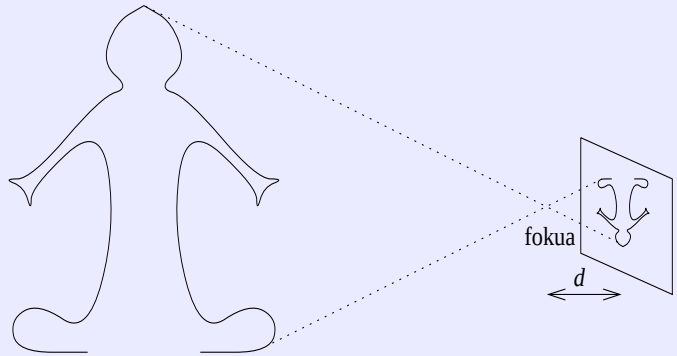
- No es más que un cambio del tipo $\overline{u} = Au + T$. (cambio de posición y de escala).

$$A = \begin{pmatrix} \frac{P_{x1}-P_x}{U_{max}-U_{min}} & 0 \\ 0 & \frac{P_{y1}-P_y}{V_{max}-V_{min}} \end{pmatrix}$$

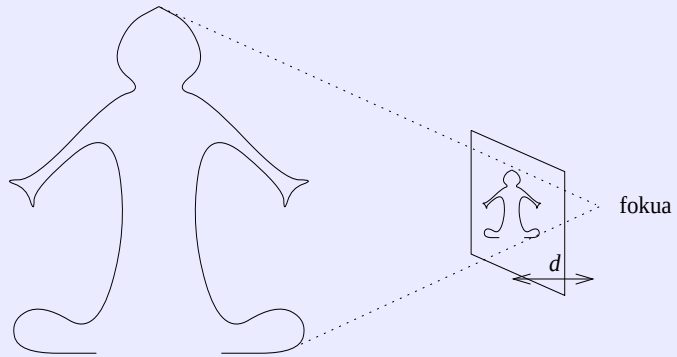
$$T = \begin{pmatrix} \frac{P_x.U_{max}-P_{x1}.U_{min}}{U_{max}-U_{min}} \\ \frac{P_y.V_{max}-P_{y1}.V_{min}}{V_{max}-V_{min}} \end{pmatrix}$$

De 3D a 2D

- Nuestros ojos ó una cámara fotográfica:



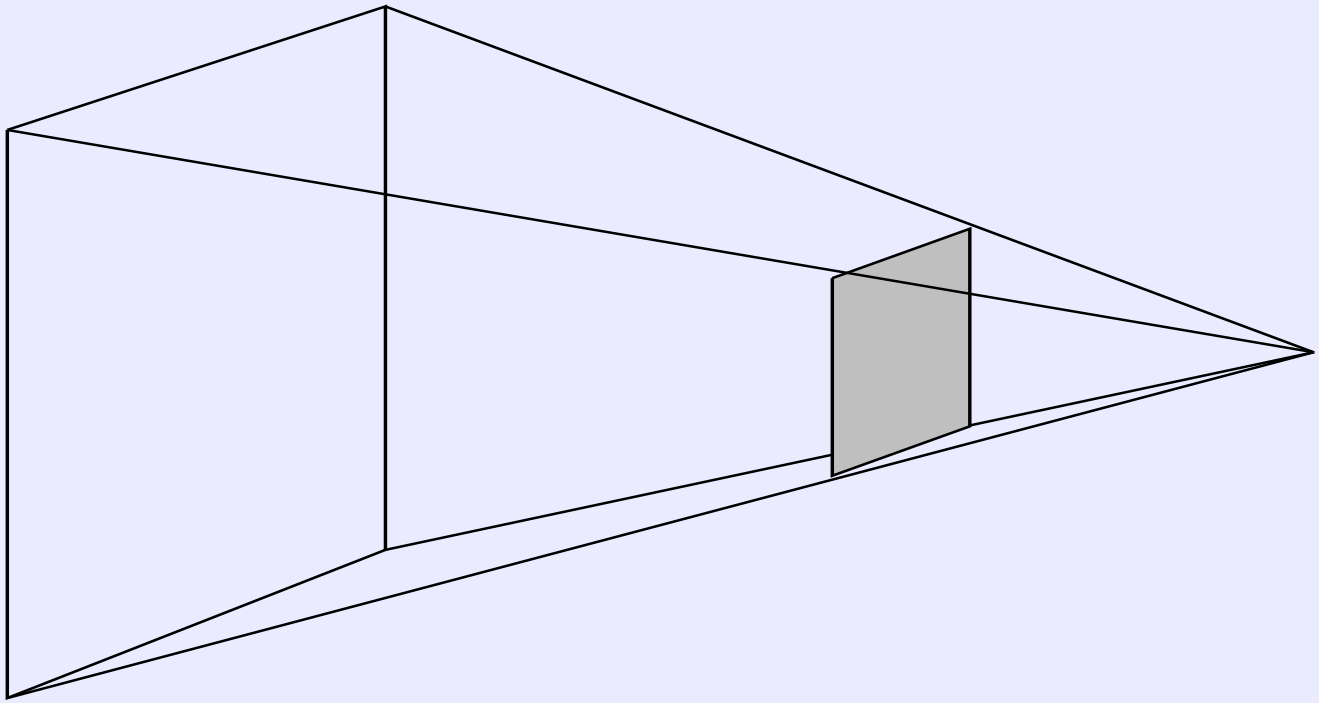
- Podemos poner el plano de proyección delante del foco.



- El punto bidimensional corresponde a la intersección del plano de proyección con el rayo que va del punto al foco

Proyección

- Teniendo en cuenta los límites del plano de proyección:



Sistema de referencia del observador

- Hay que especificar la pirámide.
- Para poder cambiarla debe darse de forma parametrizada.
- Conviene basarse en los estandares: **phigs** o **OpenGL** .
- Pasos que hay que dar:
  - Cambio de sistema de referencia: Hay que pasar la escena al sistema de referencia de la cámara.
  - Proyección.
- Información requerida:
  1. Habrá que especificar cómo ve la cámara, es decir, su sistema de referencia.
  2. De entre todo lo que puede ver hay que especificar lo que ve: las dimensiones del volumen de visión.
  3. Finalmente, hay que especificar la parte de la pantalla en la que se quiere dibujar lo que ve la cámara.

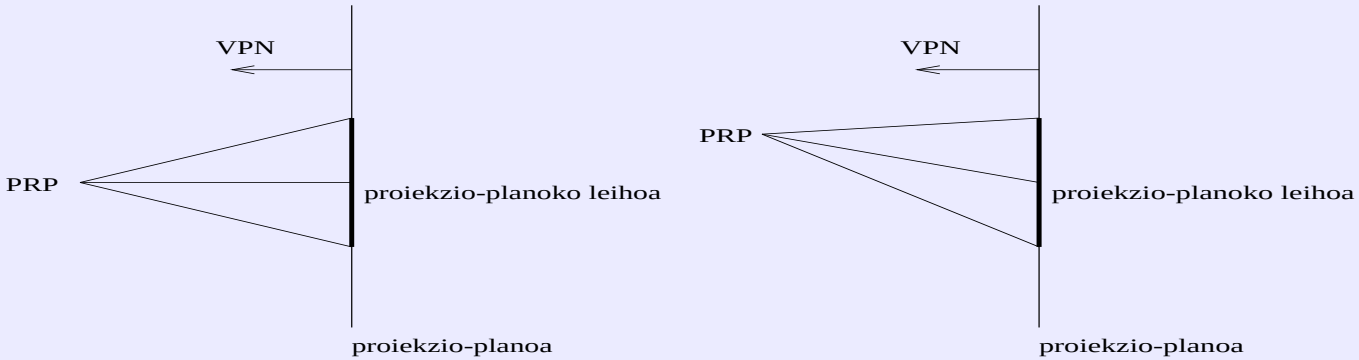
La cámara PHIGS I

- parametros:
  1. Un punto tridimensional, VRP; este punto será el origen del nuevo sistema de referencia.
  2. Vector que indique la dirección de mira, VPN.
  3. Vector que indique cual va a ser la dirección vertical, VUP.
  4. Tipo de proyección: perspectiva ó paralela.
  5. Foco de la proyección, PRP. Sus coordenadas se dan en el sistema de referencia de la cámara.
  6. Distancia al plano lejano, b. Los objetos que estan más alla de este plano se consideraran invisibles.
  7. Distancia del plano cercano, f. Solo se verán los objetos comprendidos entre estos dos planos.

La cámara PHIGS II

8. Distancia al plano de proyección, d. Estos tres planos son paralelos al plano XY del sistema de referencia de la cámara. Los valores b, f y d nos dan las distancias a las que estan en el eje z_c (Sistema de referencia de la cámara)
9. La ventana en el plano de proyección: x_{min}, y_{min}, x_{max} y y_{max}. Las rectas que van desde PRP hasta los cuatro vertices de esta ventana corresponden a las aristas del la pirámide que limita el volumen de visión. Hay que tener en cuenta que el vector que va del centro de la ventana al PRP no tiene por qué ser paralelo al eje z_c ó dirección de mira, por lo que la proyección puede ser oblicua:

La cámara PHIGS III



- La dirección de mira de la cámara puede ser cualquiera.
- La dirección vertical puede ser cualquiera.
- Se da el volumen de visión. Para ello se ha de definir la ventana del plano de proyección, el plano cercano y el lejano.

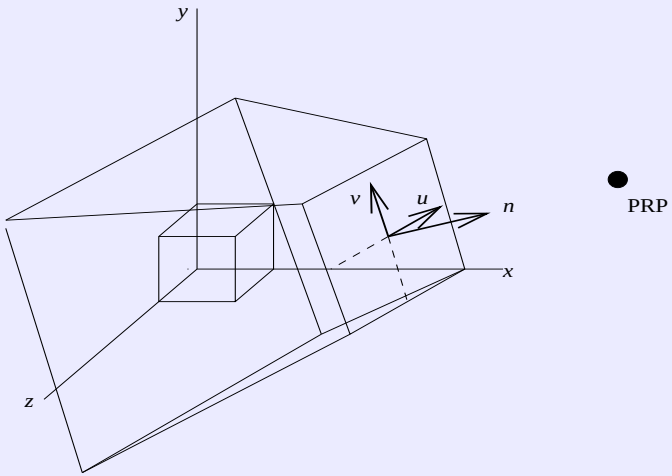
La cámara de OpenGL

- Posición de la cámara. Este punto será el nuevo origen.
- Punto al que mira. Con estos dos puntos obtenemos la dirección de mira.
- Vup. Dirección que indica la verticalidad.
- Distancia al plano de proyección y al plano lejano. En este caso el plano cercano coincide con el plano de proyección.
- Ventana en el plano de proyección: (xmin,ymin) (xmax,ymax).
- Modo de proyección: paralela / perspectiva.

El sistema de referencia de la cámara en el mundo

La cámara se define/posiciona en el mundo:

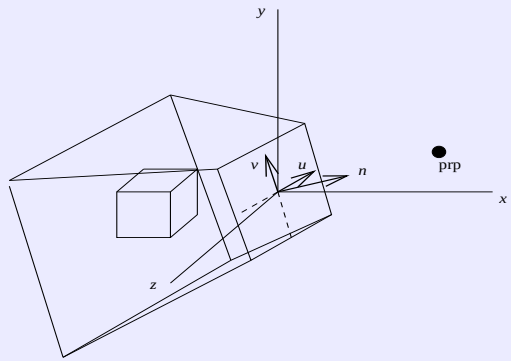
- origen (OpenGL: posición de la cámara. PHIGS: VRP)
- Base (u, v, w) o información necesaria para su obtención: OpenGL: posición, punto de mira y Vup. PHIGS: VPN y Vup
- Foco (Posición de la cámara / PRP)



Sistema de referencia de la cámara

Hay que pasar la información del sistema de referencia del mundo al de la cámara: los parámetros de la cámara establecen el origen y los vectores de la base del nuevo sistema de referencia:

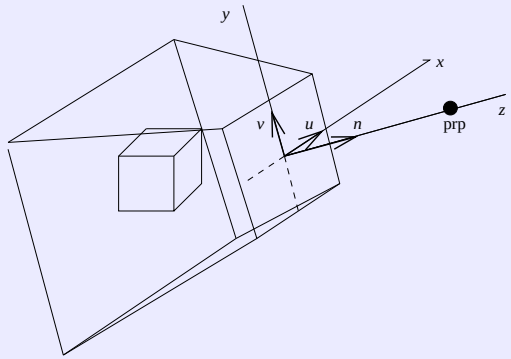
1. Origen: VRP (PHIGS) / posición de la cámara (OpenGL).



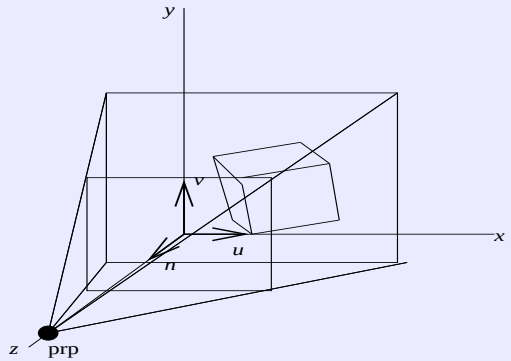
2. z_k: el vector VPN (PHIGS) o posición - punto de mira (OpenGL).
3. x_k: ya que es perpendicular al plano formado por Vup y z_k, se puede obtener mediante el producto vectorial de ambos.
4. y_k: producto vectorial de z_ky x_k (en ese orden).

Cambios de sistema de referencia

- Del sistema del mundo al de la cámara:



- Lo que ve la cámara:



- Obtener la proyección (teniendo en cuenta las oblicuas).
- Paso de la ventana de proyección a la pantalla: encuadre.

Eliminación de caras traseras.

- Cada poligono esta contenido en un plano:

$$Ax + By + Cz + D = 0$$

  Los valores A, B y C definen un vector perpendicular al plano.
- Si el ángulo entre este vector y el que va desde el poligono al observador esta entre los valores -90 y 90 grados, el observador podrá ver ese poligono, pero si el ángulo es superior ó inferior, se trata de una cara que esta de espaldas al observador.
- Podemos obtener la respuesta mediante el producto escalar:

$$VN = |V||N| \cos(\alpha) > 0$$

Contenido

Modelos de ilumininción

- Si buscamos realismo necesitamos tener en cuenta las luces.
- Hacen falta:
  1. Representación de los objetos que permita el calulo de la incidencia de la luz.
  2. Descripción aceptable de los efectos físicos de la luz.
- Efectos a tener en cuenta:
  1. Reflejos.
  2. Transparencias.
  3. Texturas.
  4. Sombras.

Modelo básico

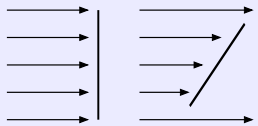
- Los cálculos se realizan en función de las propiedes de las superficies del objeto y de las fuentes de luz.
- Iluminación global: luz ambiental.
  I_a intensidad de luz que llega a todas partes por igual.
- Luces que tienen dirección:
  1. Solo dirección: sol
  2. La dirección depende de la posición: bombilla.
  3. Ademas de la posición de la luz los que alumbran en una cierta dirección: focos o linternas.

Tipos de reflejos de luz

- Reflexión difusa.
  Depende de la cantidad de luz que reciba el bojeto:
  1. la luz ambiental siempre es la misma:

$$I_{amb-dif} = K_d I_a$$

  2. La cantidad de luz que recibe del resto de fuentes es variable:



  Depende del ángulo de incidencia. Para cada fuente de luz:

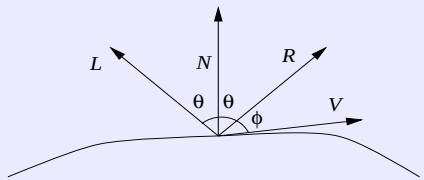
$$I_{l,dif} = K_d I_l \cos(\alpha) = K_d I_l \, (\overline{N} \, \overline{L})$$

En total:

$$I_{dif} = K_a I_a + K_d I_l \, (\overline{N} \, \overline{L})$$

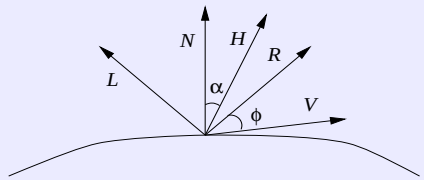
Reflexión especular. Modelo de Phong

En ciertos puntos del objeto, en funcion de la posición del obsevador, se refleja la luz de forma parecida a lo que sucede en un espejo, creando zonas brillantes.



$$I_{esp} = W(\theta) I_l \cos^{ns}(\phi) \text{ edota } I_{esp} = K_s I_l \, (\overline{V} \, \overline{R})^{ns}$$

Podemos usar el vector intermedio:



$$I_{esp} = K_s I_l \, (\overline{N} \, \overline{H})^{ns}$$

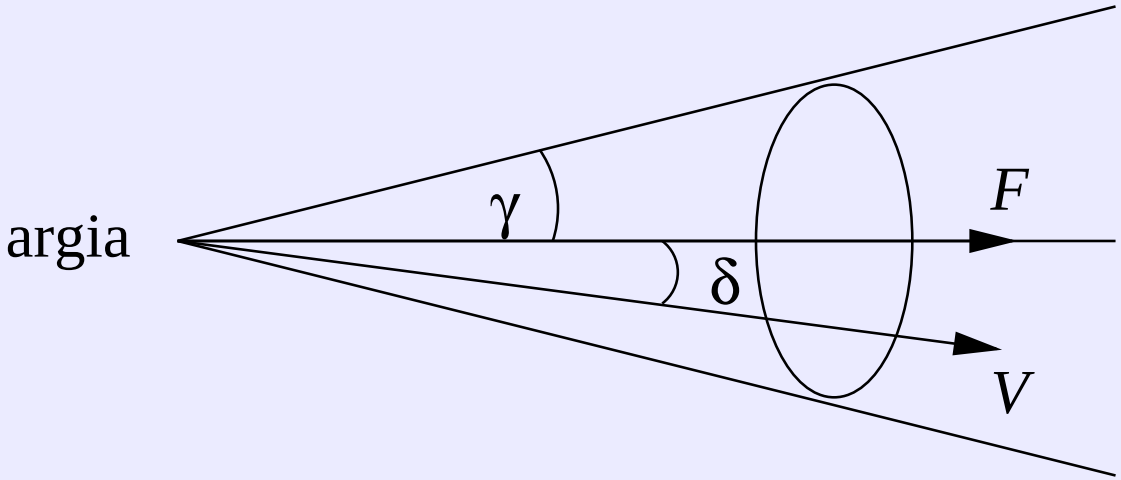
Calculo general de los efectos luminosos

En general, con n fuentes de luz:

$$I = K_a I_a + \sum_{i=1}^{n} I_{li}[K_d(\overline{N}\ \overline{L_i}) + K_s(\overline{N}\ \overline{H_i})^{ns}]$$

Modelo de Warn: Focos

- Para fuentes que iluminan solo en una cierta dirección.



$$I_{konoa} = I_l(-\overline{V}\ \overline{F})^p$$

- Se utiliza para alumbrar zonas limitadas.

Atenuación de la Intensidad

- Si la fuente es lejana recibimos menor intensidad.

$$f(d) = \frac{1}{(a_0 + a_1 d + a_2 d^2)}$$

El denominador debe ser mayor que 1:

$$f(d) = \text{mín}(1, \frac{1}{(a_0 + a_1 d + a_2 d^2)})$$

- Si lo incluimos en el modelo de iluminación:

$$I = K_a I_a +$$

$$+ \sum_{i=1}^{n} f(d_i) I_{li} [K_d (NL_i) + K_s (NH_i)^{ns}]$$

Correcciones para el color

- Lo normal es que utilicemos 3 componentes para cada luz.
- Para el componente azul sería:

$$I_B = K_{aB} I_{aB} +$$

$$\sum_{i=1}^{n} f(d_i) I_{lBi} [K_{dB}(NL_i) + K_{sB}(NH_i)^{ns}]$$

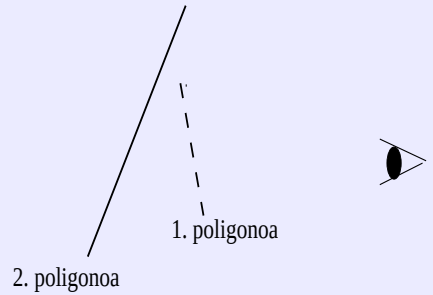
- podríamos usar vectores para las constantes:

$$I_B = K_a S_{dB} I_{aB} +$$

$$\sum_{i=1}^{n} f(d_i) I_{lBi} [K_d S_{dB}(NL_i) + K_s S_{sB}(NH_i)^{ns}]$$

Transparencia

- Transparencia sin refracción.
  1. Transparencia interpolada.



$$I = (1 - K_t)I_1 + K_t I_2$$

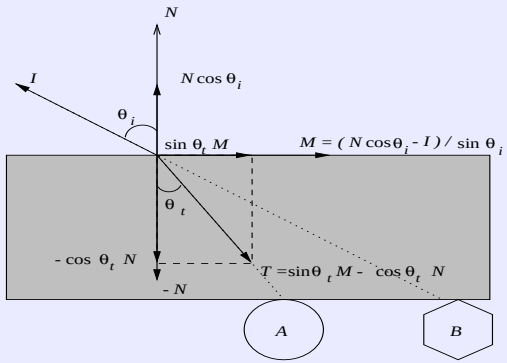
  2. Transparencia filtrada.

$$I = I_1 + K_t O_{t\lambda} I_2$$

Transparencia refractada

- Calculo de la refracción: Relación entre el ángulo del rayo incidente y el del refractado:

$$\frac{\sin(\theta_i)}{\sin(\theta_t)} = \frac{\eta_t}{\eta_i}$$

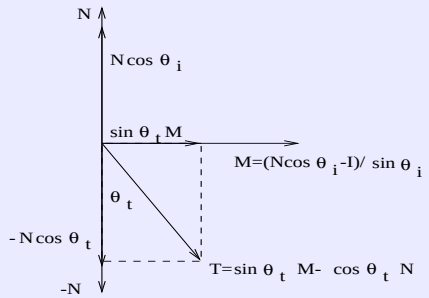


Cálculo del vector del rayo refractado

Vector de refracción:

$$\overline{T} = \left(\sin(\theta_t)\overline{M}, -\cos(\theta_t)\overline{N} \right)$$

$$\overline{T} = \left(\eta_r(\overline{N}\,\overline{I}) - \sqrt{1 - \eta_t^2(1 - (\overline{N}\,\overline{I})^2)} \right) \overline{N} - \eta_t \overline{I}$$

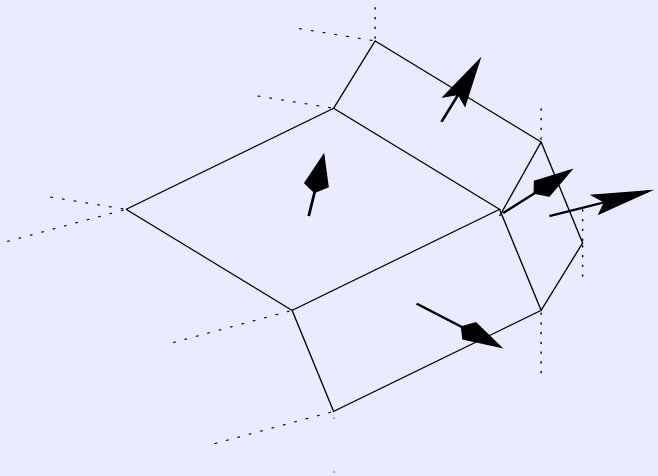


Sombras

- Podemos usar cualquier algoritmo de eliminación de partes ocultas. El algoritmo nos indicará lo que ve la luz.
- Mientras la luz y los objetos no cambien de posición el cálculo no se tiene que volver a realizar.
- En los puntos visibles para la fuente de luz hay que tener en cuenta la intensidad de la fuente, en otro caso no.

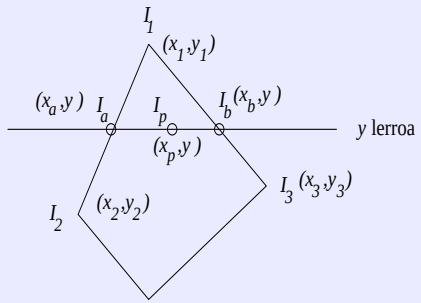
Técnicas de iluminación de polígonos

- Un buen modelo de iluminación debería calcular la intensidad para todos los puntos del polígono.
- Lo normal es que un polígono sea la continuación del polígono adyacente. Habría que suavizar el cambio.
- Se necesita el vector normal en cada punto y vértice.



Técnicas de iluminación de polígonos: Gouraud y phong I

- Se usan dos técnicas para los puntos internos:
  1. Gouraud.
     Se calcula la intensidad en los vértices y se interpola para los puntos internos.



Técnicas de iluminación de polígonos: Gouraud y phong II

$$I_a = \frac{1}{y_1 - y_2}(I_1(y - y_2) + I_2(y_1 - y))$$

$$I_b = \frac{1}{y_1 - y_3}(I_1(y - y_3) + I_3(y_1 - y))$$

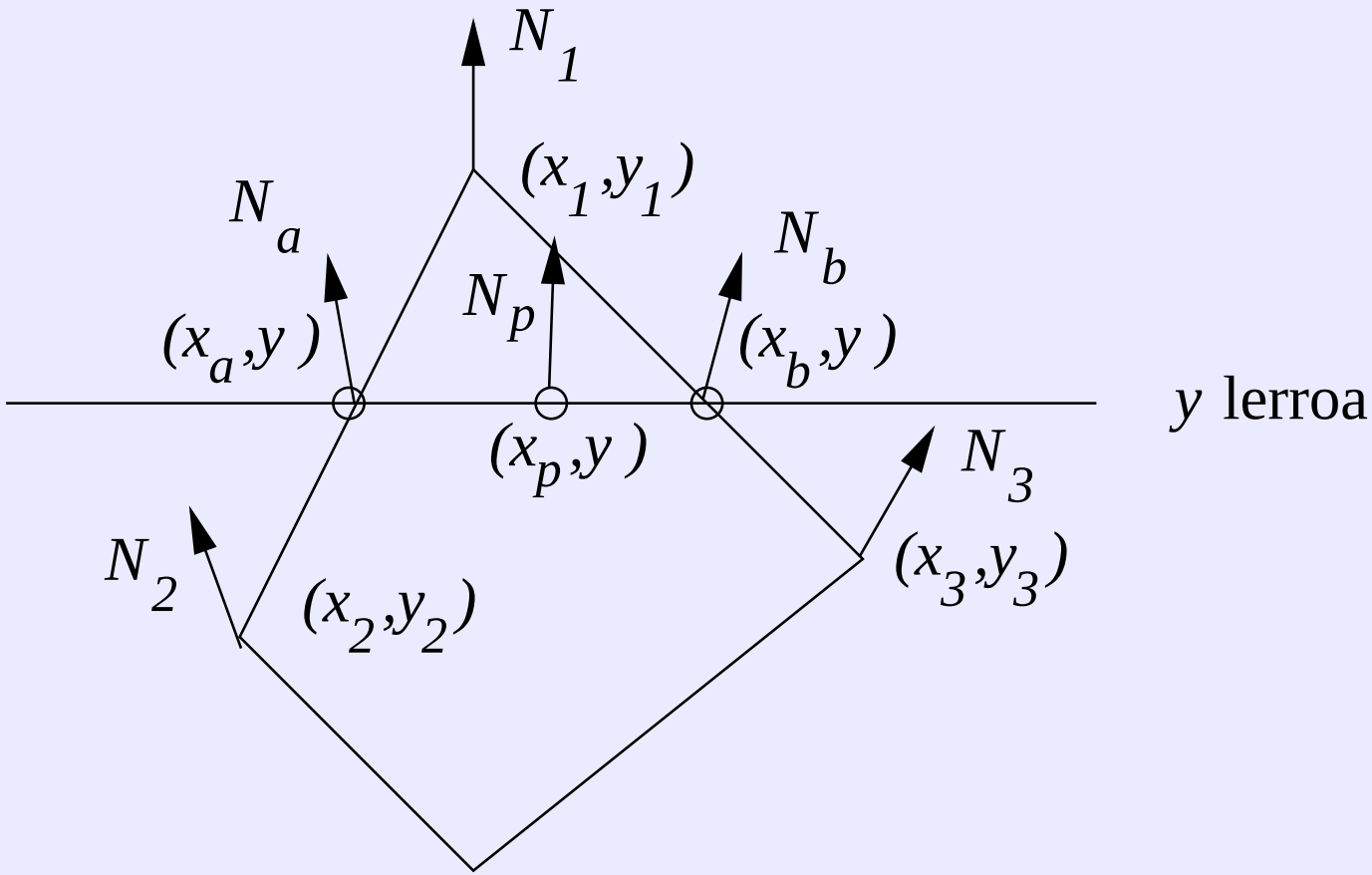
$$I_p = \frac{1}{x_b - x_a}(I_a(x_b - x) + I_b(x - x_a))$$

Mayor eficacia:

$$I_{x+1} = I_x + \Delta I_u$$

2. Phong.
   Se interpola el vector normal con el que se cálcula la intensidad en cada punto del polígono.

Técnicas de iluminación de polígonos: Gouraud y phong III



Técnicas de iluminación de polígonos: Gouraud y phong IV

$$N_a = \frac{1}{y_1 - y_2}(N_1(y - y_2) + N_2(y_1 - y))$$

$$N_b = \frac{1}{y_1 - y_3}(N_1(y - y_3) + N_3(y_1 - y))$$

$$N_p = \frac{1}{x_b - x_a}(N_a(x_b - x) + N_b(x - x_a))$$

edo:

$$N_{px,x+1} = N_{px,x} + \Delta N_{ux}$$

$$N_{py,x+1} = N_{py,x} + \Delta N_{uy}$$

$$N_{pz,x+1} = N_{pz,x} + \Delta N_{uz}$$

Diferencias entre Phong y Gouraud

- Gouraud es más rápido, tiene menos c'alculos.
- Phong puede obtener reflejo especular en el interior de un polígono. Gouraud no.
- Existe la posibilidad de combinar los dos métodos.