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.

- Formatos de imágenes:

    1. Video: 720 x 512 x 24

    2. Se pueden conseguir 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. X11R5

- 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.

CARACTERISTICAS

- Hay que trabajar con modelos.
  - Exactos.
  - Aproximaciones.
  - Obtenidos mediante dispositivos.
  - Creados. Posibilidad para crearlos.
- Tratamiento. Transformar los modelos.
- Procesos de visualización.
  - Dibujo de líneas y curvas.
  - Delimitación de la ventana de proyección.

  - Cálculo de proyecciones.
  - Relleno de superficies.
  - Eliminación de partes ocultas.
  - Animación.

- Interacción con el ususario.
  - Facil de usarlo.
  - Veloz.
  - Con posibilidad de corrección..
  - 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.

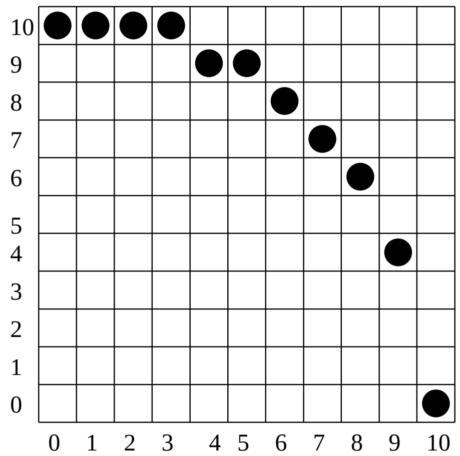
6. Iluminación y sobreado de las superficies.

7. Dibujo.

DISCRETIZACION

- 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.



x	0	1	2	3	4	5	6	7	8	9	10
y	10	9.9	9.7	9.5	9.1	8.6	8.0	7.1	6.0	4.3	0
redondeo	10	10	10	10	9	9	8	7	6	4	0

3. Aliasing. Nuestro cerebro busca símiles (alias).

4. Gran cantidad de cálculo. Circuitos integrados.

LINEAS Y CURVAS

Algoritmos incrementales.

- Para lineas.

- Ecuación explícita.

- Pendiente entre 0 y 1.

- Basandose en el pixel anterior.

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

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

- Multiplicación, suma y redondeo.

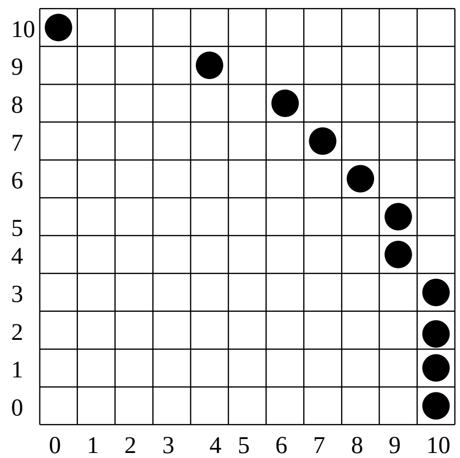
- Uso muy frecuente y debe ser muy eficiente.

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

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

$$y(x_i) + m$$

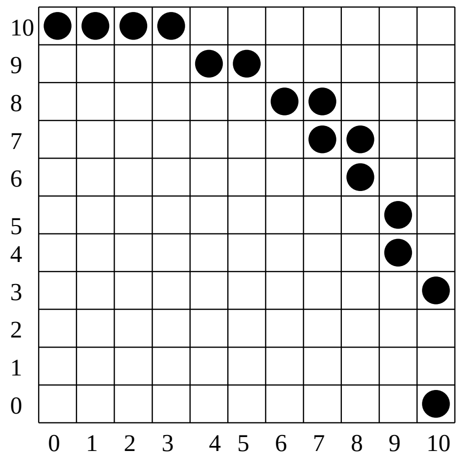
- Pendiente menor que 1!



y x	0 10	1 9.9	2 9.7	3 9.5	4 9.1	5 8.6
redondeo	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.



Hay que completar los datos con los anteriores.

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

- Basndonos en la ecuación diferencial de 1. orden.

- Solución mediante métodos numéricos.

- Forma: $y^{'} = F(x, y)$ siendo $y(x_0) = y_0$ conocido.

- Como ejemplo Euler:

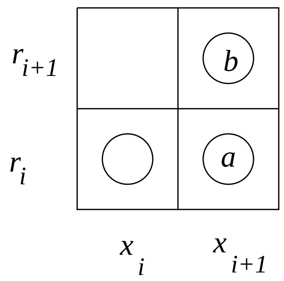
$$y^{'} = \frac{\Delta y_i}{\Delta x_i}$$

- 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.



- Caso a:

$$r_{i+1} = r_i \Longleftrightarrow \text{int}(y_{i+1} + 0.5) = r_i$$

$$r_i \leq y_{i+1} + 0.5 < r_i + 1$$

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:

$$r_{i+1} = r_i + 1 \Longleftrightarrow \mathrm{int}(y_{i+1} + 0.5) = r_i + 1$$

$$r_i + 1 \leq y_{i+1} + 0.5 < r_i + 2$$

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 ($E_i \geq 0$);

$\{$ $r_i ++$; $E_i -= \Delta x$; $\}$

dibuja(x_i, r_i;) $\}$

$\}$

RELLENO DE POLIGONOS.

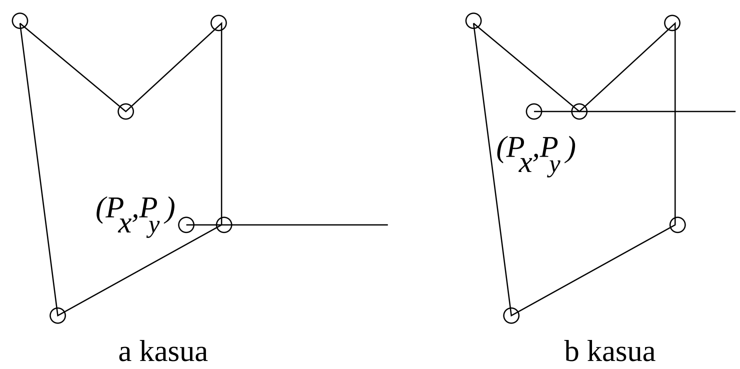
- Dibujar los lados es facil.

- Rellenar el interior no tan facil.

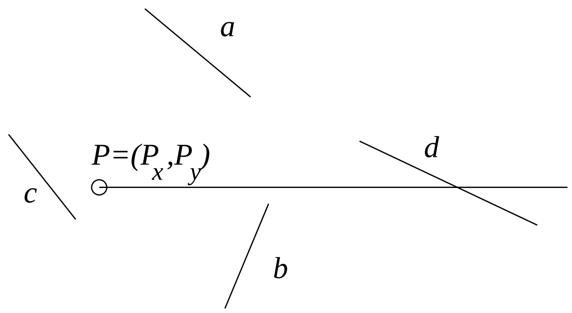
- Punto interno? Teorema de Jordan.

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

- En la práctica hay problemas:



- La mayoría de los casos:



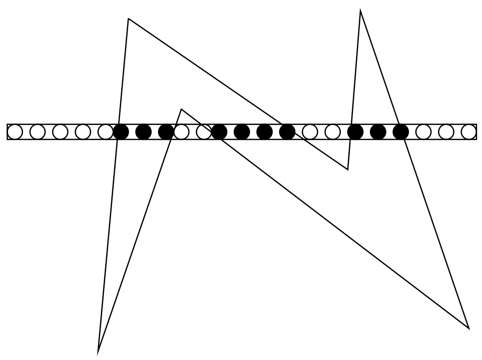
- Solución: En caso de vértice superior contarlo.

- 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:

  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 (*$numerador >= \Delta y$)

    {/* puede pasar $\Delta x > \Delta y$ */

    *x ++;

    *$numerador - = \Delta y$;

    }

}

Formas de evitar 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.

- 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.

- A mas subpixels mas cálculos.

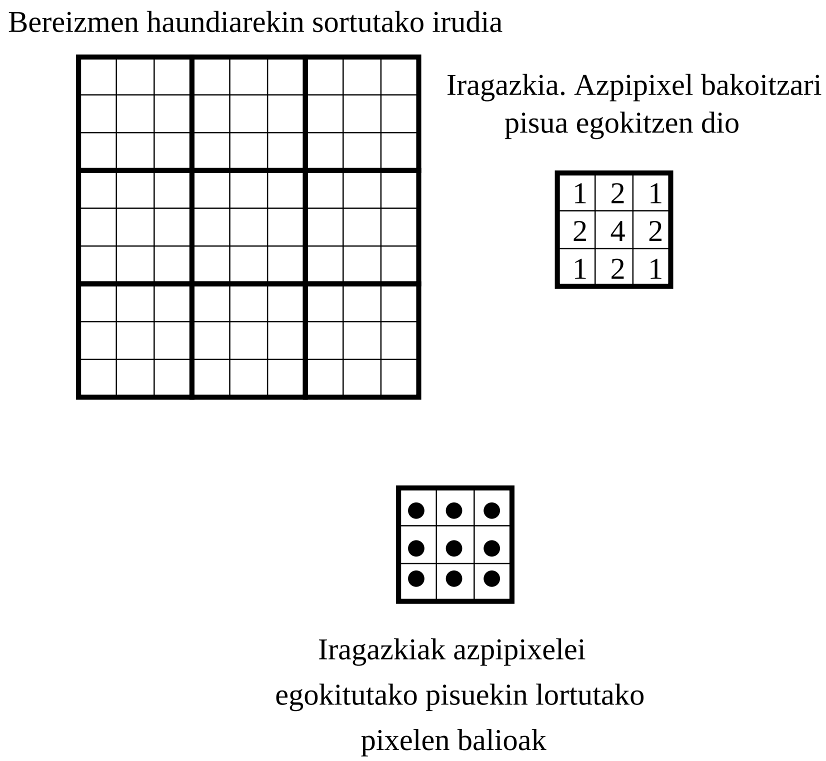
- El incremento de calculo es del orden de n^2.

- 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.

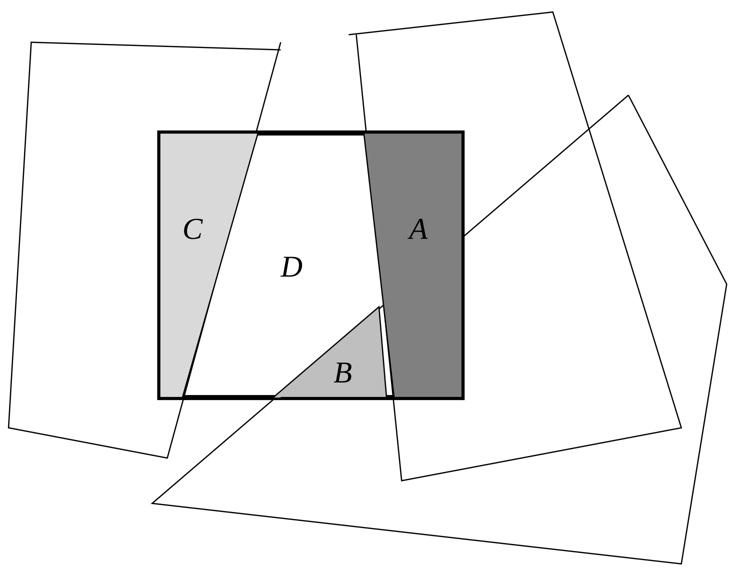

Bereizmen haundiarekin sortutako irudia
Iragazkia. Azpipixel bakoitzari pisua egokitzen dio
1 2 1
2 4 2
1 2 1
Iragazkiak azpipixelei egokitutako pisuekin lortutako pixelen balioak

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.



$$I = AI_A + BI_B + CI_C + DI_D$$

- 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.

- Se adecua bien a los metodos de 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.

Transformaciones geométricas

- Objetos representados por caras (vértices y aristas).

- Transformaciones a estudiar:

  1. Traslación.
  2. Escalado.
  3. Rotación.
  4. Reflexión.

- 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:

$$v^{`} = v + \begin{pmatrix} m \\ n \end{pmatrix}$$

- 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:

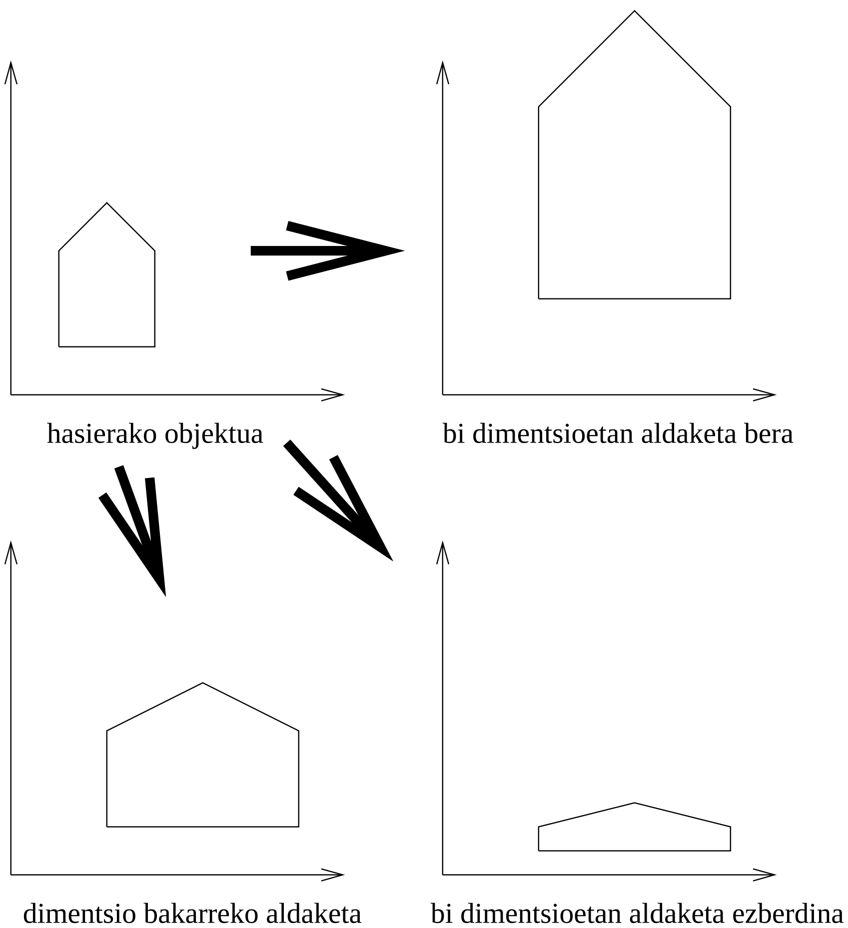
$$\begin{pmatrix} p & 0 \\ 0 & q \end{pmatrix}$$

- Deben cumplir la condición $p, q > 0$.

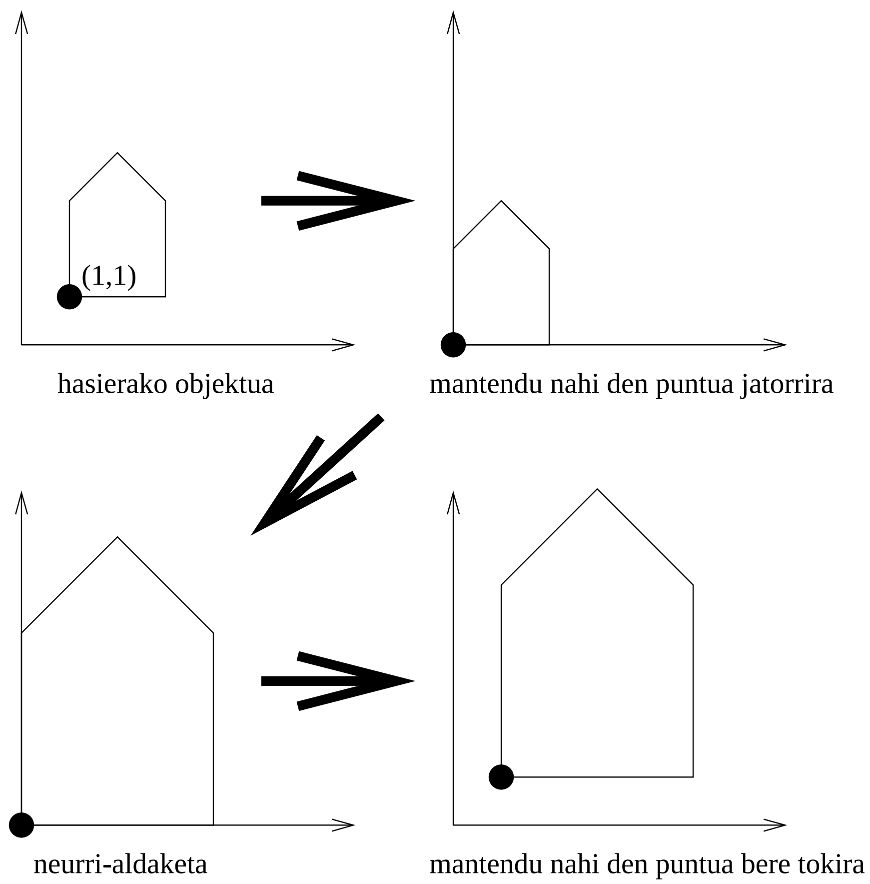
- Si $p, q > 1$ estaríamos incrementando tamaño.

- Si $p, q < 1$ decrementamos tamaño.

- El escalado puede provocar cambio de posición del objeto.



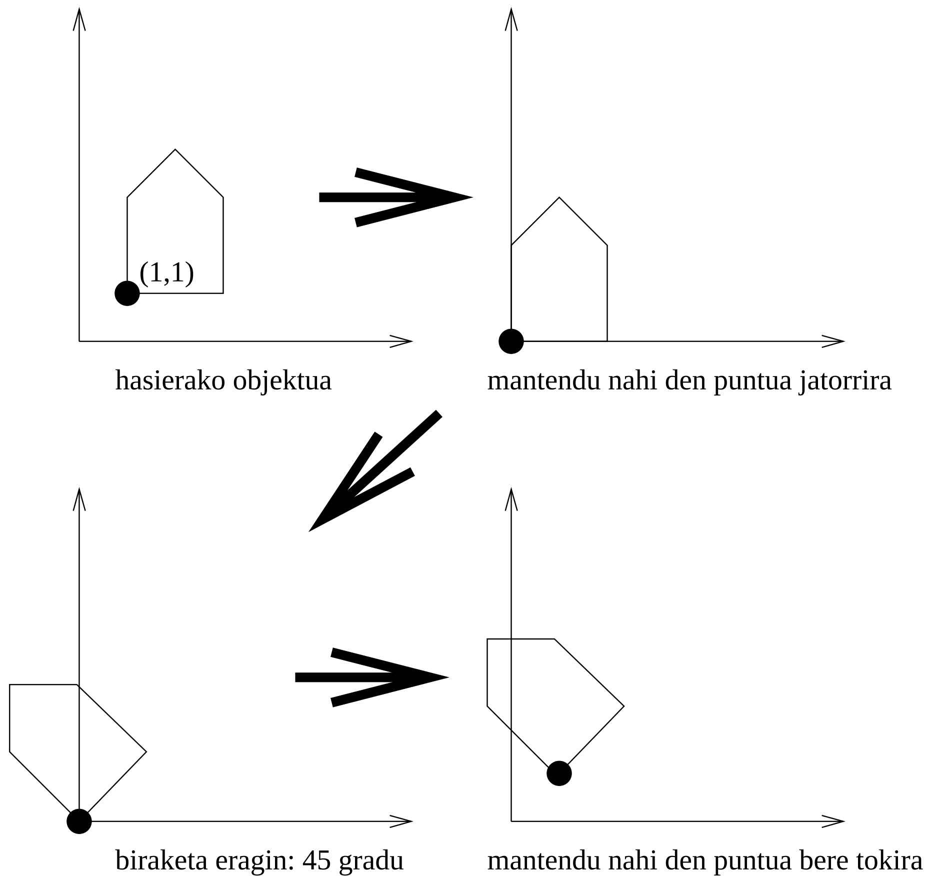
- Para mantener un punto en su posición hay que concatenar tres transformaciones:



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).



- 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:

$$x^{'} = L\,(\cos\alpha\cos\beta - \sin\alpha\sin\beta)\,^{*}$$

$$y^{'} = L(\cos\alpha\sin\beta + \sin\alpha\cos\beta)^{\dagger}$$

$^{*}\cos(\alpha+\beta) = \cos\alpha\cos\beta - \sin\alpha\sin\beta$
$^{\dagger}\sin(\alpha+\beta) = \sin\alpha\cos\beta + \cos\alpha\sin\beta$

- Visto de otra forma:

$$x^{`} = x\cos\beta - y\sin\beta$$

$$y^{`} = x\sin\beta + y\cos\beta$$

- Su representación matricial:

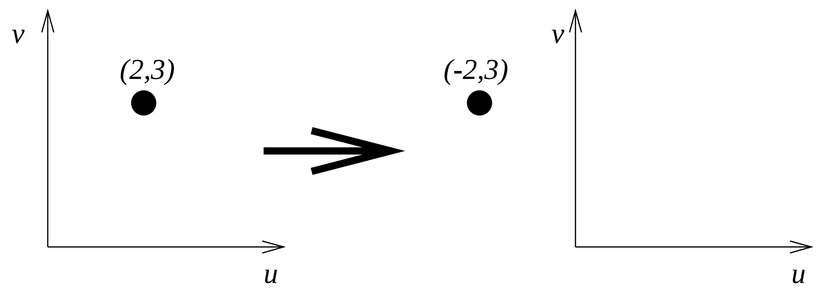
$$\begin{pmatrix} x^{`} \\ y^{`} \end{pmatrix} = \begin{pmatrix} \cos\beta & -\sin\beta \\ \sin\beta & \cos\beta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

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

Reflexión.

- Hay que obtener la figura simetrica respecto a una recta.

- Supondremos que pasa por el origen.



- 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 obtendría 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 $\begin{pmatrix} \cos\alpha \\ \sin\alpha \end{pmatrix}$.

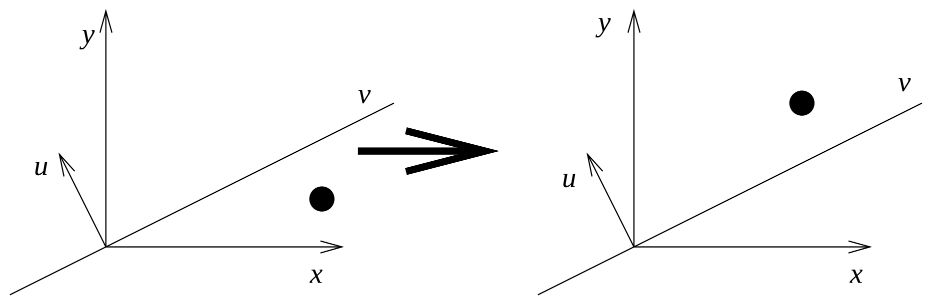
- Necesitamos el vector u que es perpendicular al eje: $\begin{pmatrix} -\sin\alpha \\ \cos\alpha \end{pmatrix}$.

- 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: $\begin{pmatrix} x^{'} \\ y^{'} \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} + \begin{pmatrix} m \\ n \end{pmatrix}$.

- 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.

- 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:

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

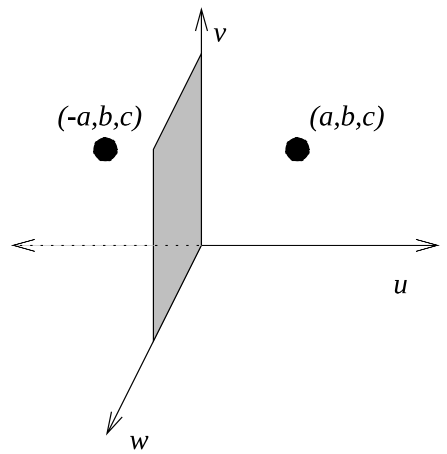
Mantiene el origen, pero el resto de puntos los transforma. Al escalarlos puede parecer que el objeto se mueve, si se quiere mantener algun punto habrá que concatenar distintas transformaciones.

- Reflexión: Matrices reflectoras de Housenholder, pero respecto a un plano:

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

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

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



v
(-a,b,c)
(a,b,c)
u
w

Rotación. Eje de rotación. Puede ser del sistema de referencia o cualquiera.

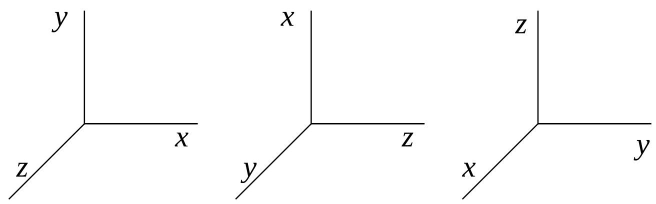
respecto al eje w:

$$u^{'} = u \cos\alpha - v \sin\alpha$$

$$v^{'} = u \sin\alpha + v \cos\alpha$$

$$w^{'} = w$$

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



rotación respecto a z

$$x^{'} = x\cos\alpha - y\sin\alpha$$

$$y^{'} = x\sin\alpha + y\cos\alpha$$

$$z^{'} = z$$

Su representación matricial será:

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

rotación respecto a x

$$y^{'} = y\cos\alpha - z\sin\alpha$$

$$z^{'} = y\sin\alpha + z\cos\alpha$$

$$x^{'} = x$$

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

rotación respecto a y

$$z^{`} = z\cos\alpha - x\sin\alpha$$

$$x^{`} = z\sin\alpha + x\cos\alpha$$

$$y^{`} = y$$

visto de otra forma:

$$x^{`} = x\cos\alpha + z\sin\alpha$$

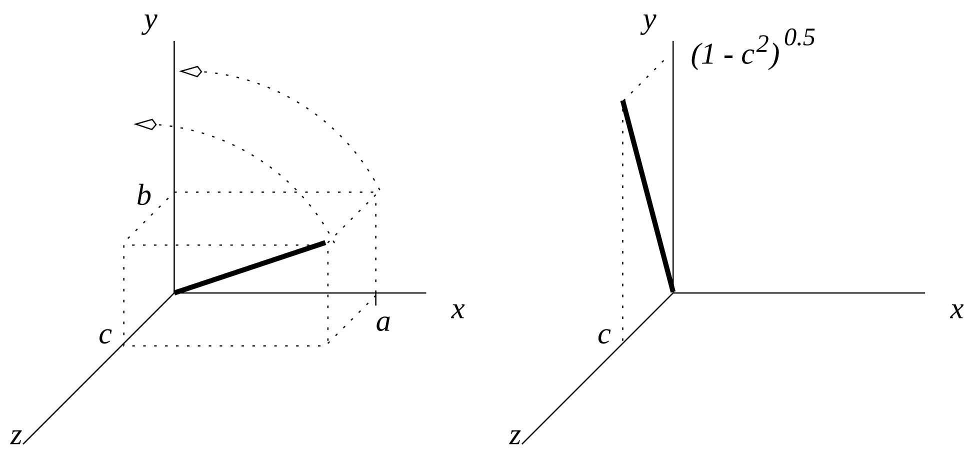
$$y^{`} = y$$

$$z^{`} = -x\sin\alpha + z\cos\alpha$$

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

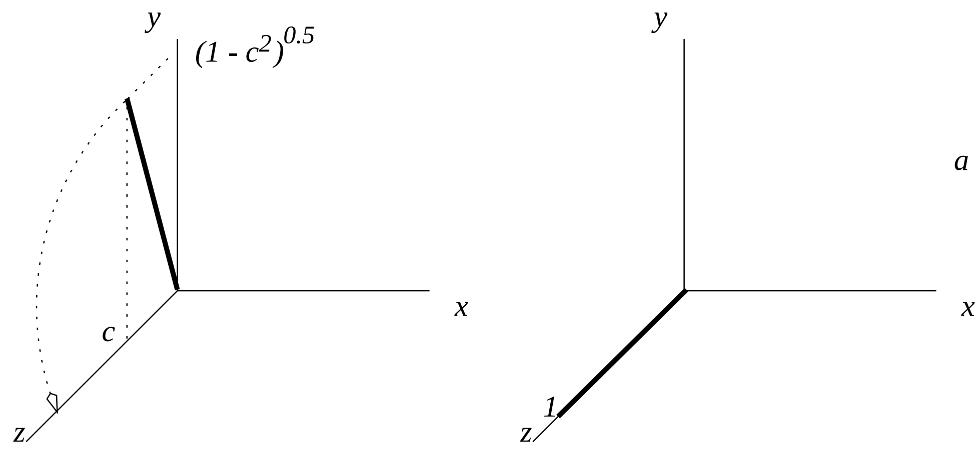
Rotación respecto a cualquier eje. Podemos basarnos en las rotaciones vistas, hay que conseguir que el eje de rotación sea uno de los tres del sistema de referencia.

1. Primera rotación:



   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.

3. Tercera rotación: realizar la rotación respecto a z (eje de rotación).

4. Deshacer las dos primeras rotaciones para dejar el sistema como al principio pero con la rotación respecto al eje realizada.

   - deshacer rotación respecto a x.

   - 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 realize todo el trabajo.

REPRESENTACION

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.**

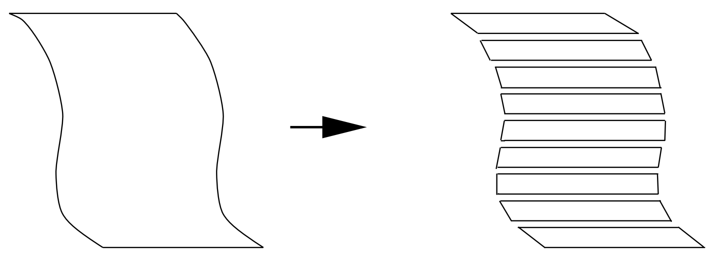
5. Tambien se pueden utilizar funciones matemáticas. Por ejemplo:

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

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.



- 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.

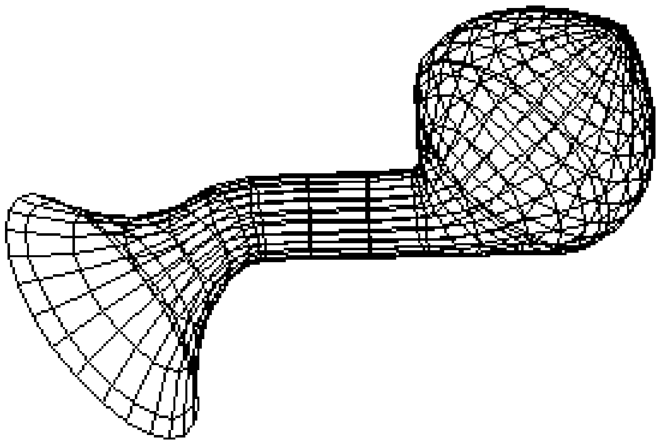
Creación del objeto

- 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.

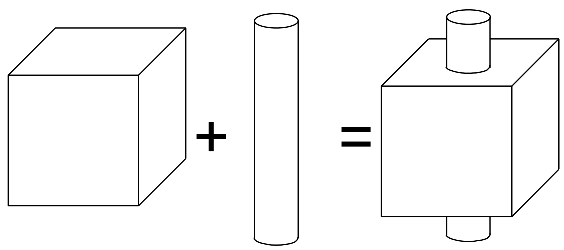
Se usan, sobre todo, dos técnicas:

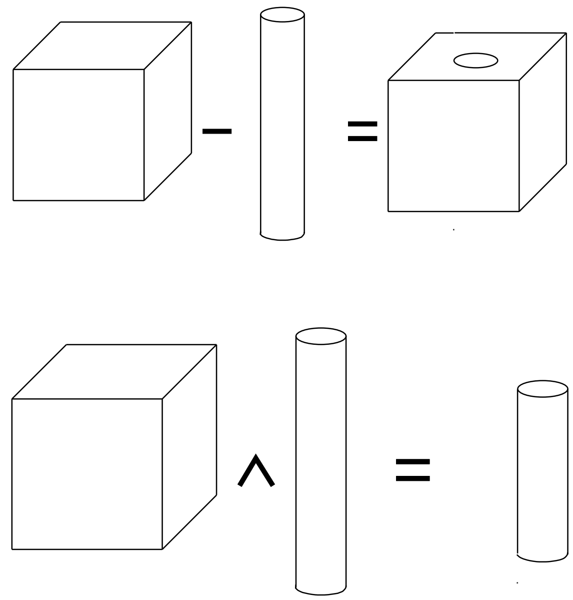
- **Ajuste de superficies:** Es un método de interpolación. El resultado será una aproximación del objeto, pero cumple condiciones de continuidad.

- **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.

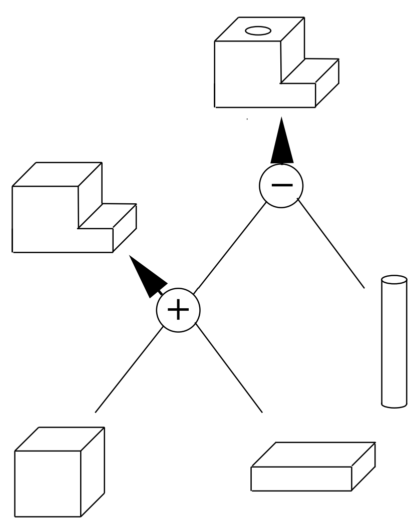
CSG

- 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.





- Como 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.

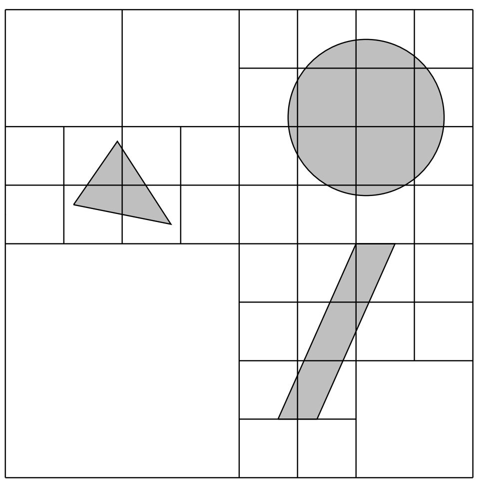
Octrees

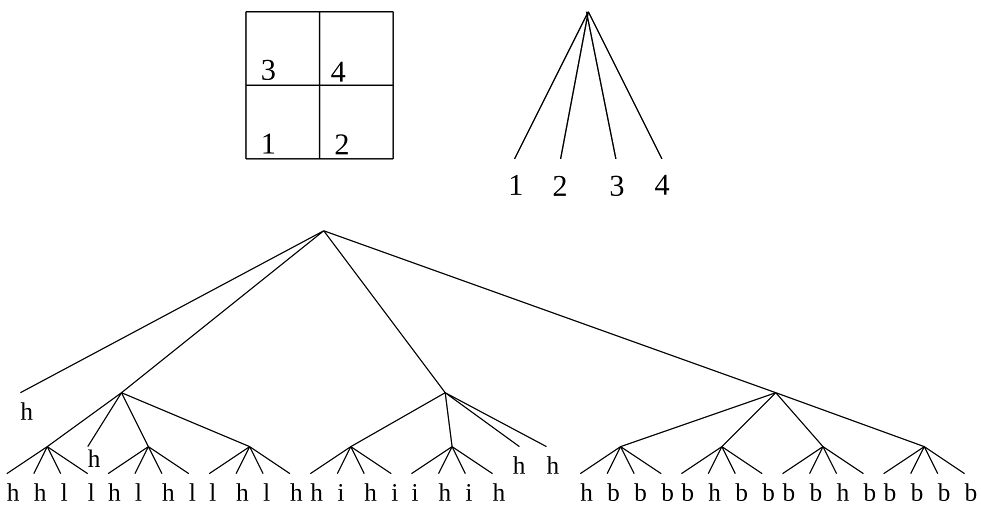
- Se trata de una estructura de árbol.

- 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.





h = hutsik

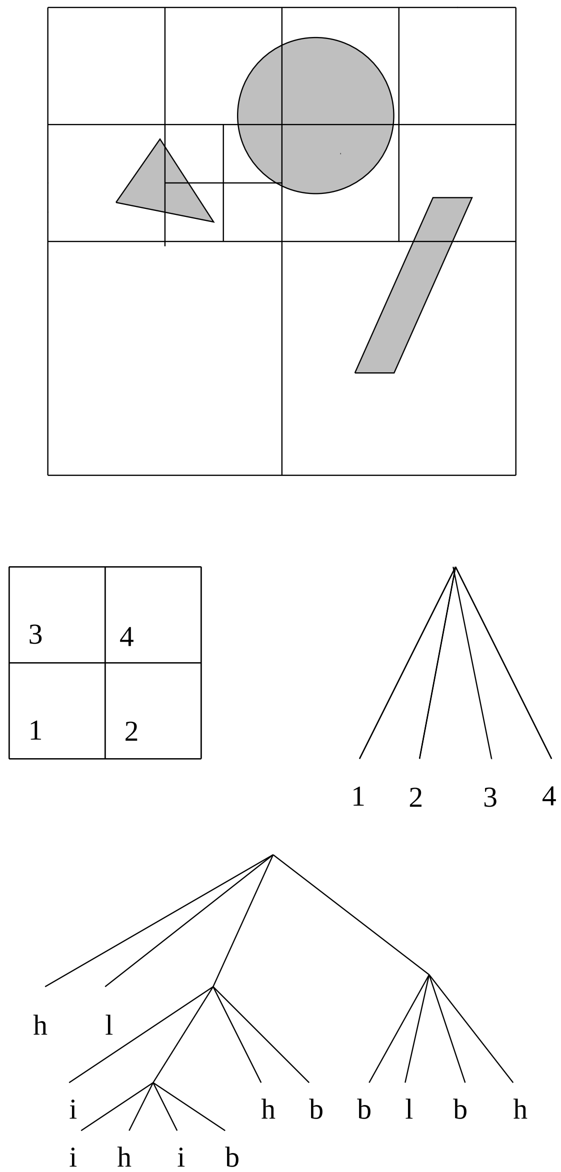
i = hirukia

b = borobila

l = laukizuzen okertua

- La subdivisión espacial se debe realizar en función de la resolución deseada.

- Las zonas sin objetos no hace falta dividirlas.



h = hutsik
i = hirukia
b = borobila
l = laukizuzen okertua

- Solo tiene en cuenta los limites del objeto.

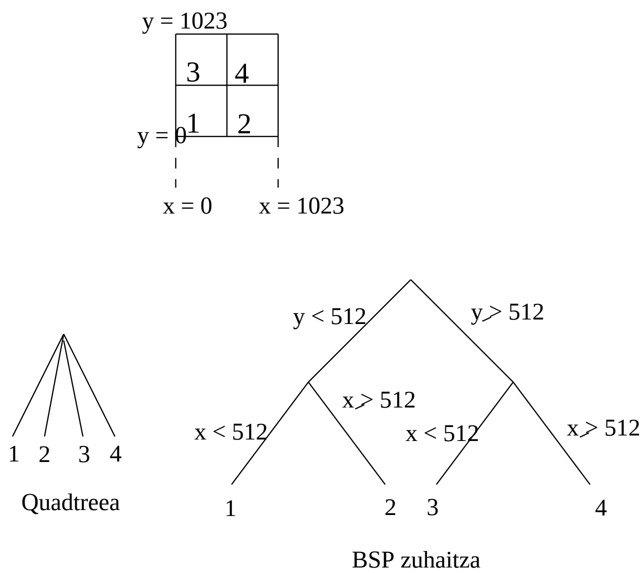
- En general, una hoja del árbol puede tener una lista de objetos.

Arboles BSP

- Se trata de una estructura de árbol.

- El espacio se divide siempre en dos.

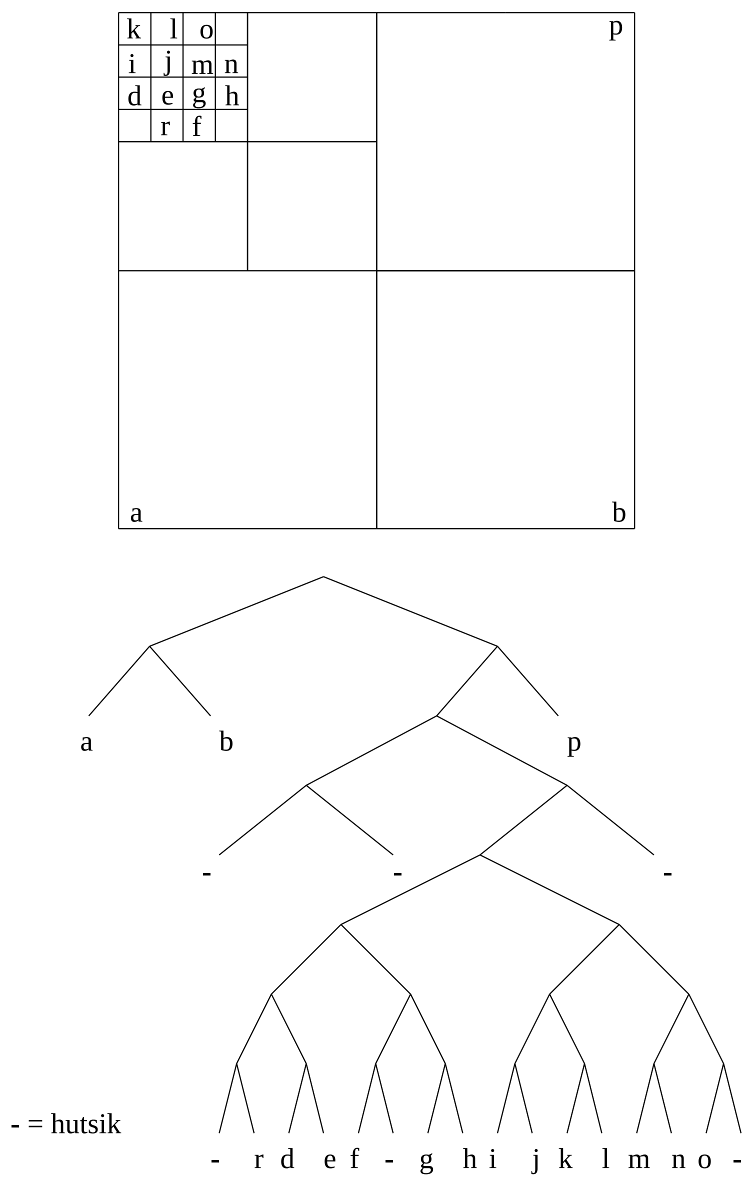
- Para la división se utilizan planos.

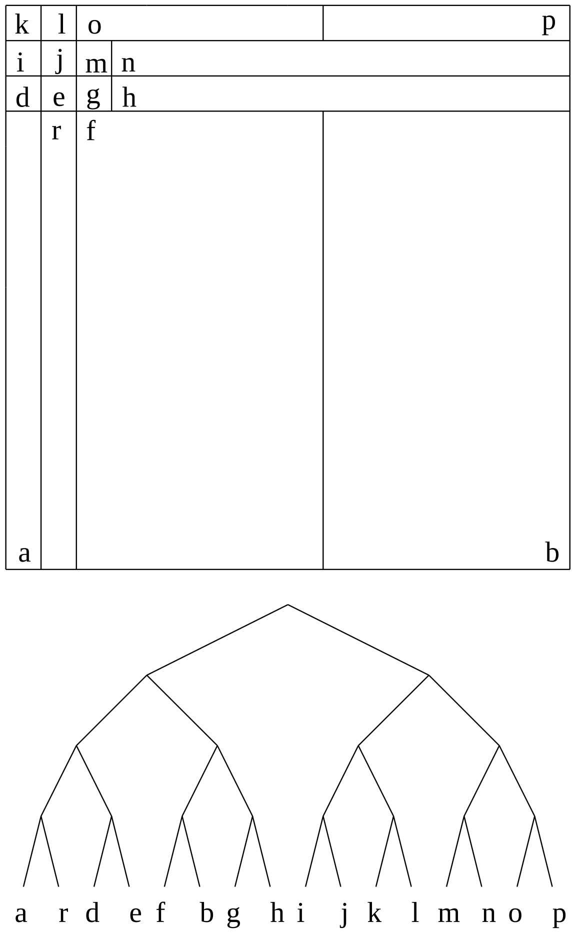


- Se adecua al ray tracing. El camino recorrido por un rayo se traduce a movimientos en la estructura del árbol, y en cada nodo habrá que testear si el rayo intersecta con los objetos que hay en esa zona.

subdivisión adecuada

- El plano para la división puede ser cualquier-a.



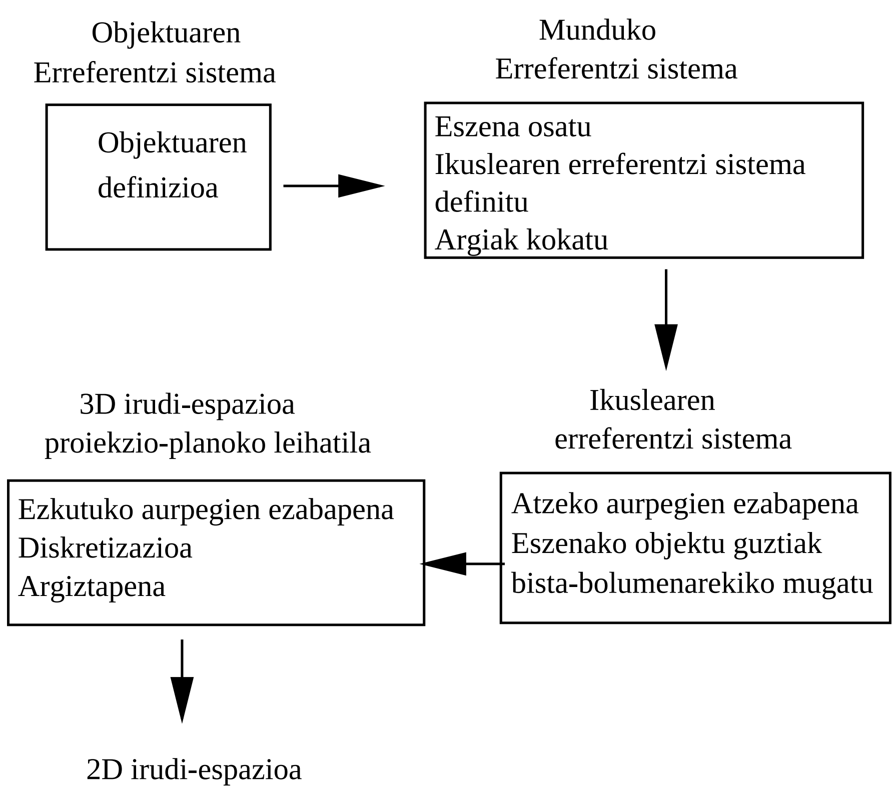


- Se obtienen álboles más equilibrados, de menor profundidad.

Obtención de la imagen mediante B-rep

- La entrada es una lista de polígonos, la salida un conjunto de pixels coloreados.

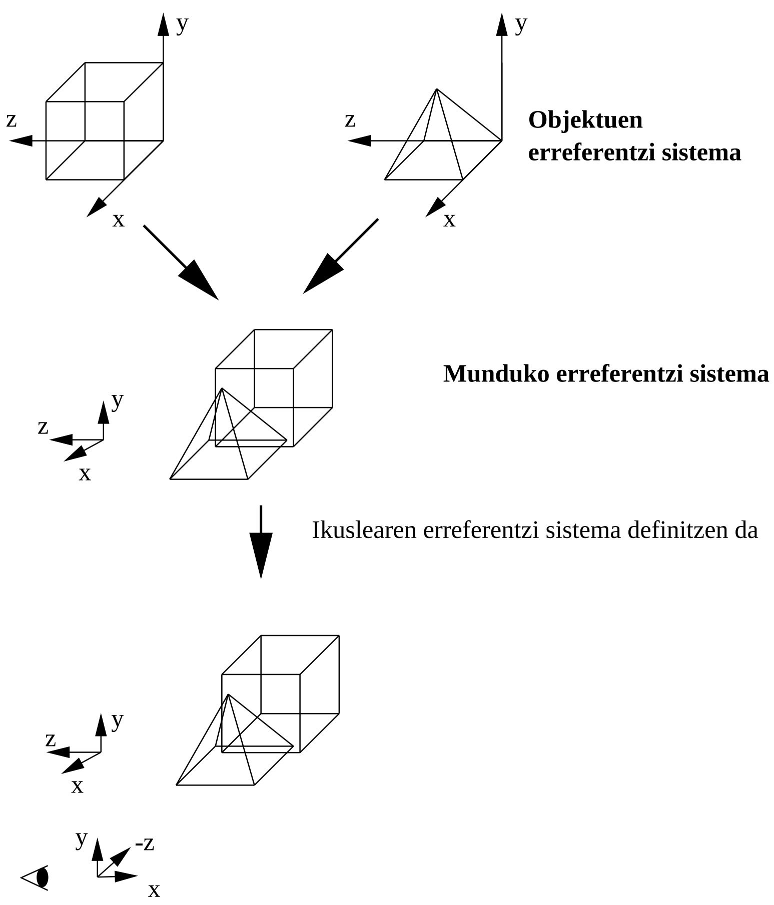
- Se pasan distintos sistemas de referencia.



- Los objetos tienen su Sist. de Ref.

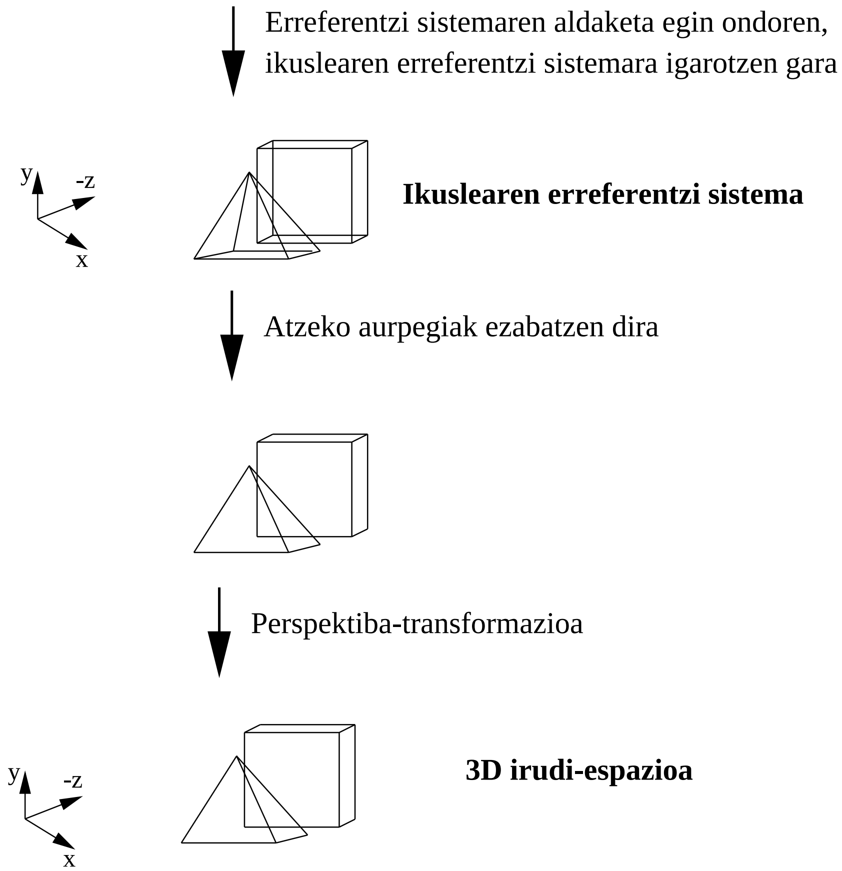
- Todos se introducen en un mundo con el sist. de Ref. de la escena.

- En ese sistema se especifican las luces, el observador . . . .



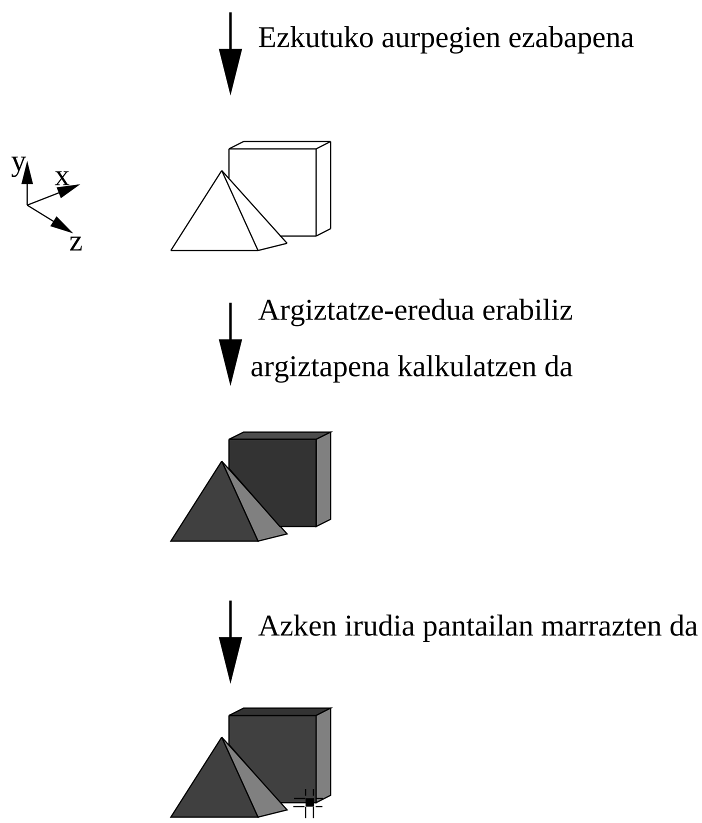
- Se establece el Sist. de Ref. del observador.

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



- 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

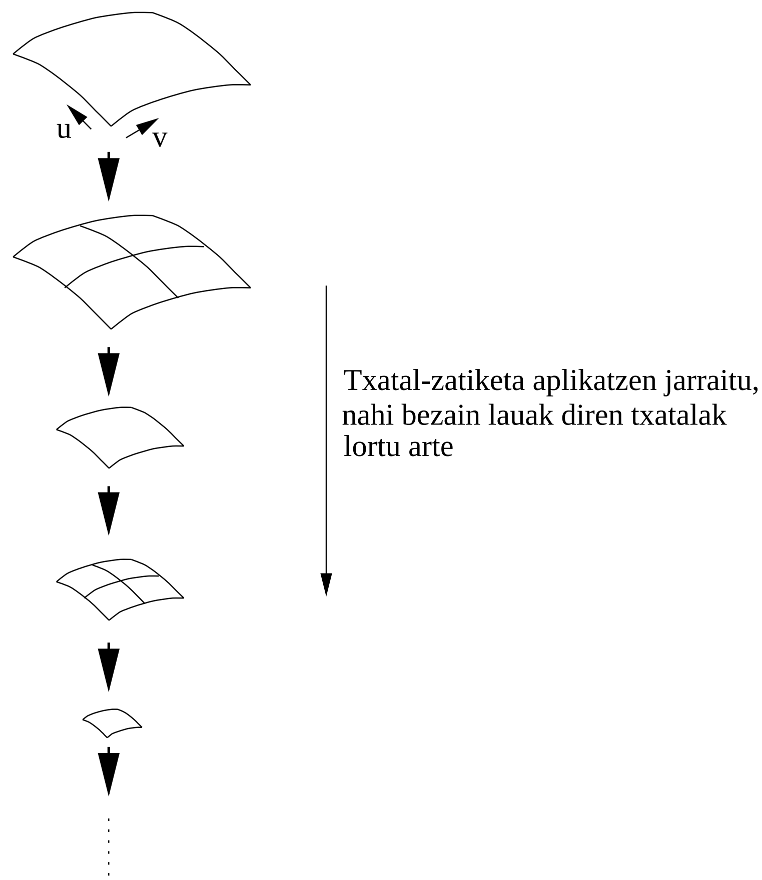
- Lo más facil es pasar los parches a polígonos.

- Los parches pueden tener cualquier resolución, pero los podemos

- Estos objetos los crea el usuario de forma interactiva, pero la traducción nos permite obtener la imagen mediante B-Rep.

- El número de poligonos supera al de parches.

- La traducción se realiza en dos fases: Traduccion a Bezier y subdivision de Bezier y posterior traducciíon de Bezier a polígonos.



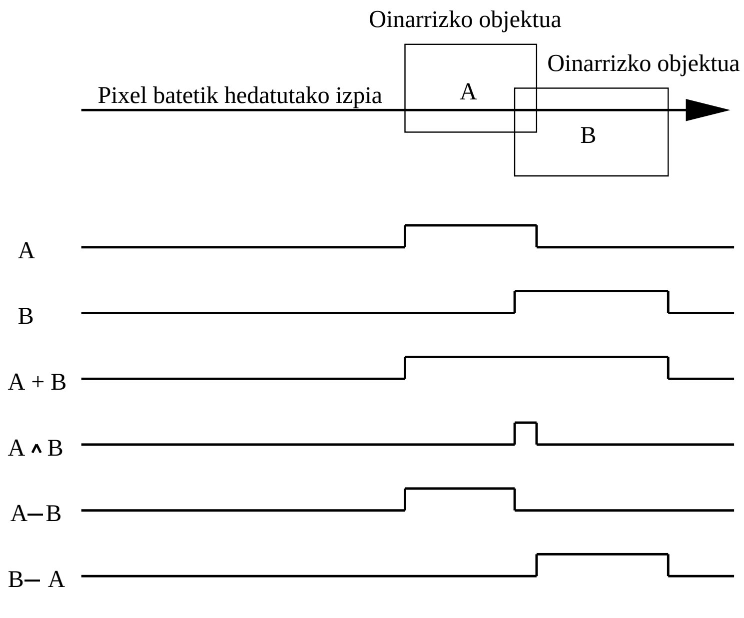
- Para terminar la subdivisión hay que establecer condiciones de planaridad.

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.

Sistemas de Visualización

- Hay que pasar de 3D a 2D.

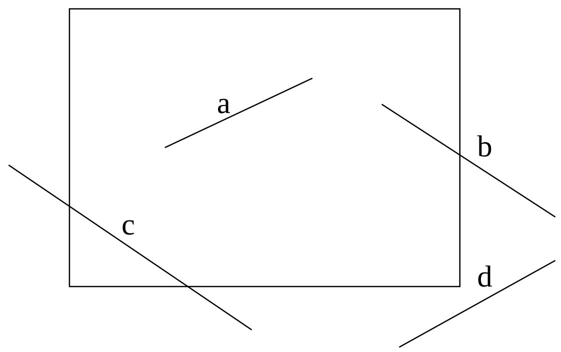
- De 2D hay que llevarlo a la ventana de la pantalla. Habrá que especificar el rectángulo que queremos que aparezca en la pantalla.

  1. Clipping ó recorte: hay que recortar lo que queremos que aparezca.

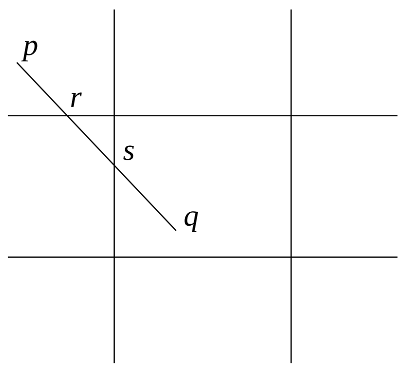
  2. Lo recortado hay que pasarlo al sistema de referencia de la pantalla.

Recorte

- Recorte de lineas: algoritmo Cohen-Sutherland.

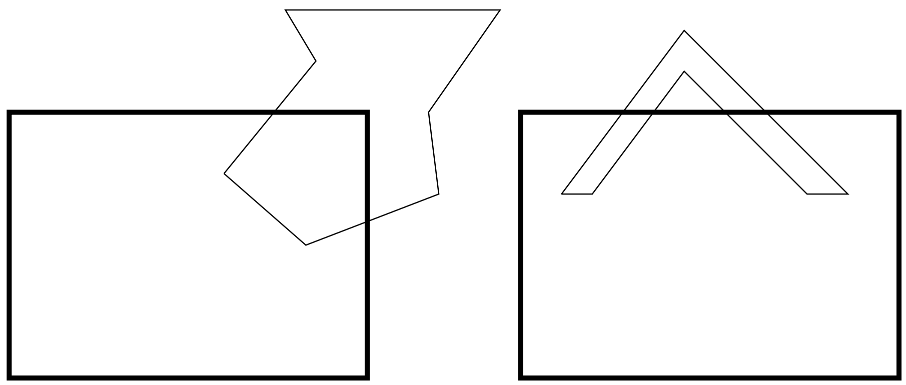


Divide el espacio en 9 zonas y facilita el trabajo utilizando ciertos codigos.



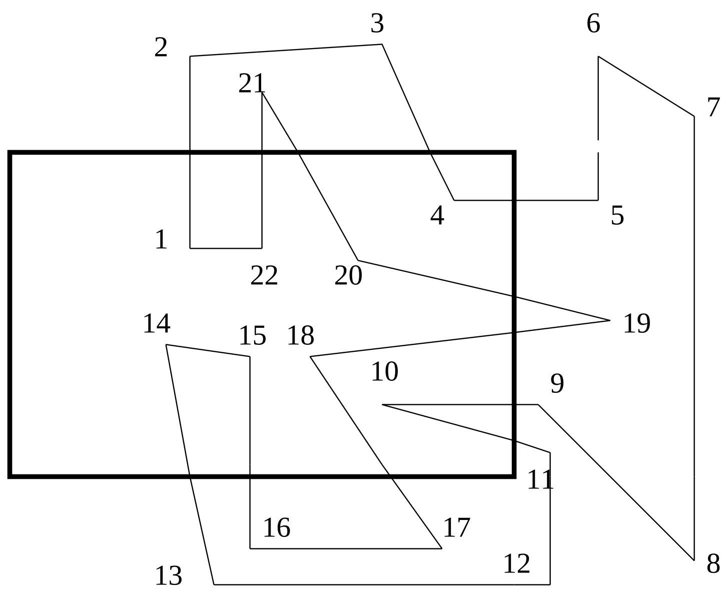
- Recorte de poligonos: algoritmo Sutherland-Hodgman.

Si hay que rellenar el poligono, el algoritmo anterior puede dar como resultado poligonos no cerrados y habrá que evitarlo.

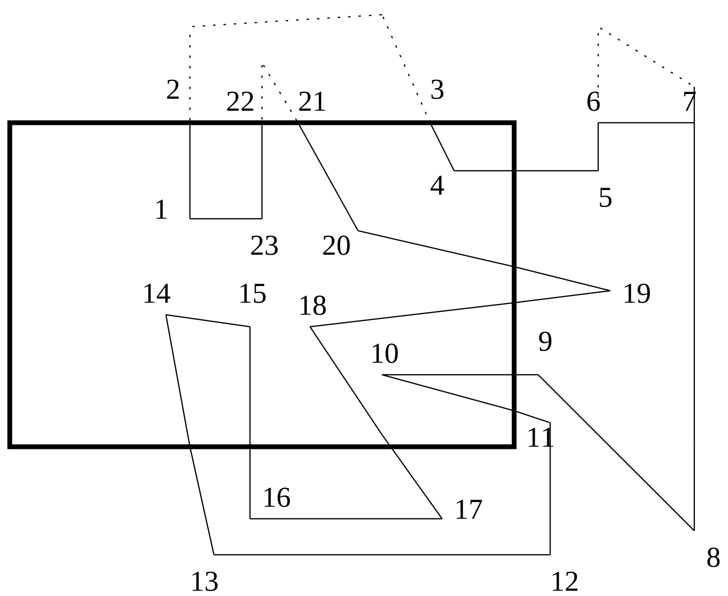


(a) (b)

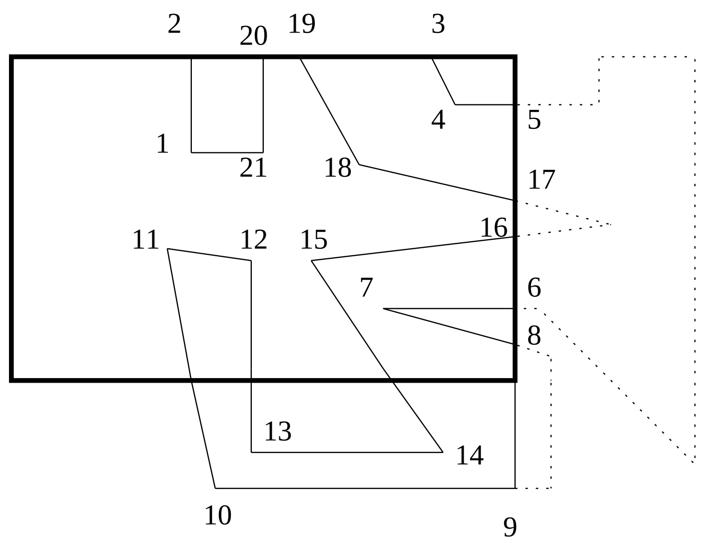
El algoritmo parte de un poligono y tras recortarlo devuelve otro poligono.

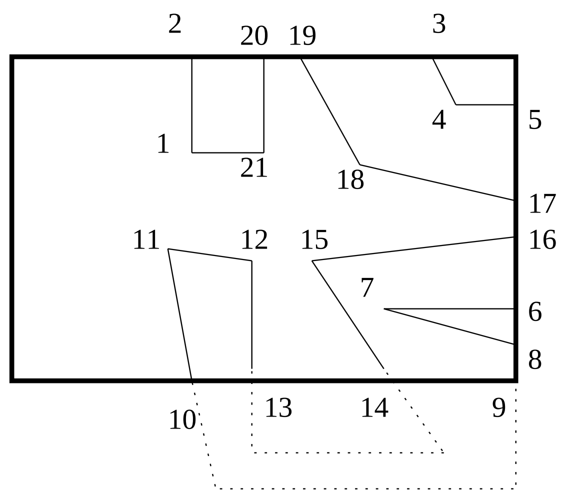


2
3
6
21
7
4
5
1
22
20
14
15
18
19
10
9
11
16
17
12
13
8



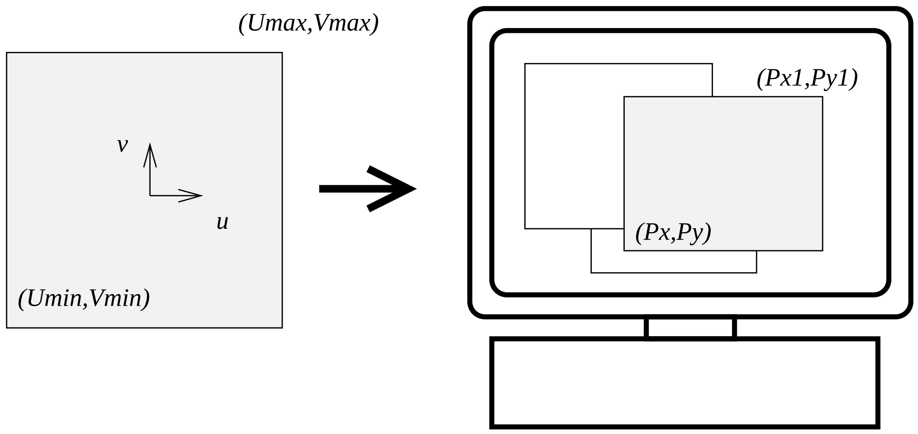
2
22
21
3
6
7
4
5
1
23
20
14
15
18
19
10
9
11
16
17
13
12
8


2
20
19
3
4
5
1
21
18
17
16
11
12
15
7
6
8
13
14
10
9


2
20
19
3
4
5
1
21
18
17
16
11
12
15
7
6
8
13
14
9
10

Encuadre

- Se trata de un cambio de sistema de referencia.



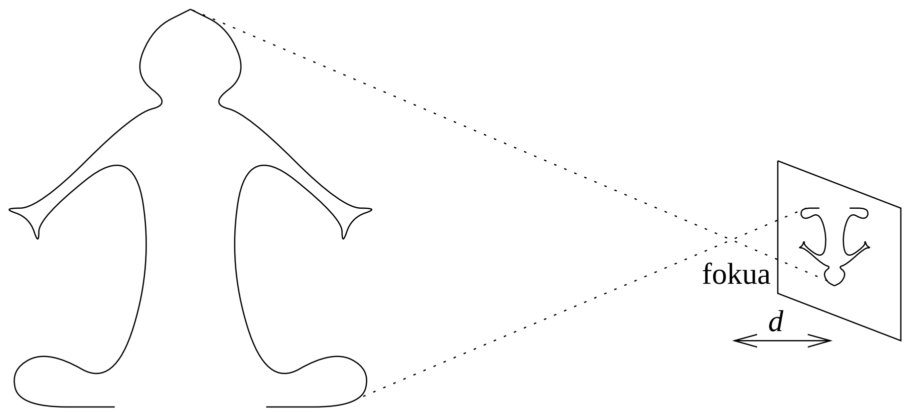
- No es más que un cambio del tipo $\overline{u} = Au + T$. (cambio de posición y de escala).

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

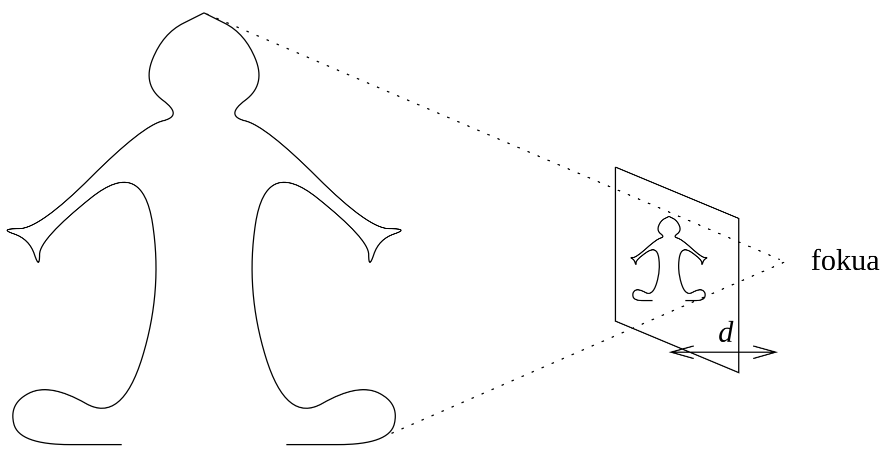
$$T = \begin{pmatrix} \frac{P_x.U_{max}-P_{x1}.U_{min}}{U_{max}-U_{min}} \\ \frac{P_y.V_{max}-P_{y1}.V_{min}}{V_{max}-V_{min}} \end{pmatrix}$$

De 3D a 2D

- Nuestros ojos ó una cámara fotográfica:

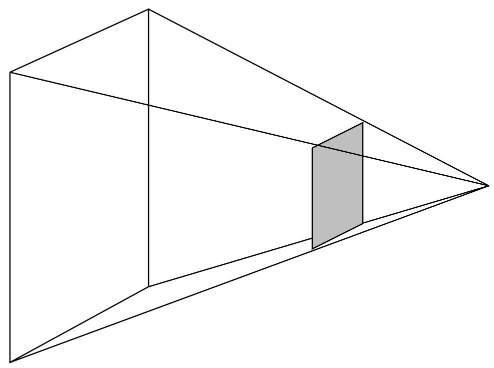


- Para evitar que este al reves podemos poner el plano de proyección delante del foco.



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

- 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**.

- 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

- 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.

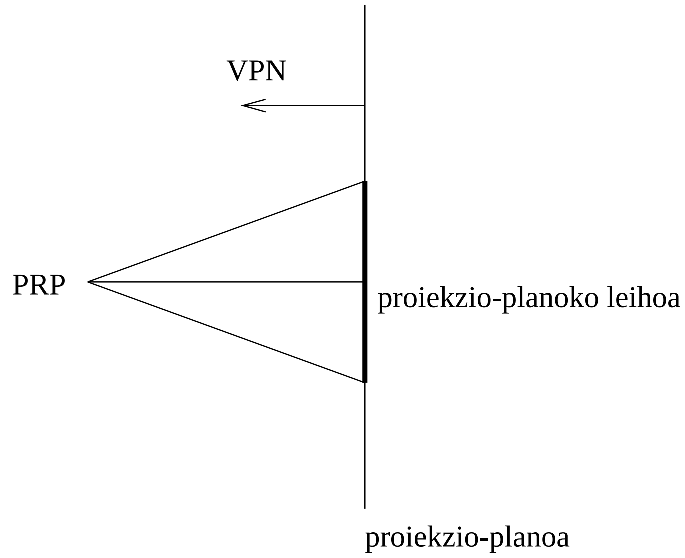
6. Distancia al plano lejano, b. Los objetos que estan más alla de este plano se consideraran invisibles.

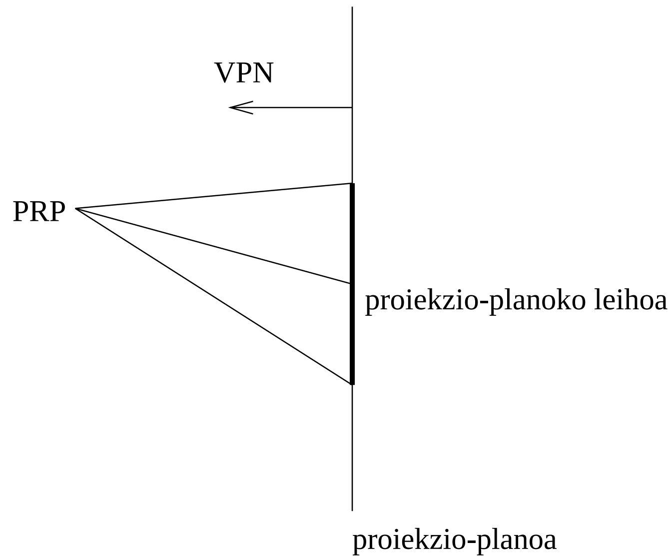
7. Distancia del plano cercano, f. Solo se verán los objetos comprendidos entre estos dos planos.

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:



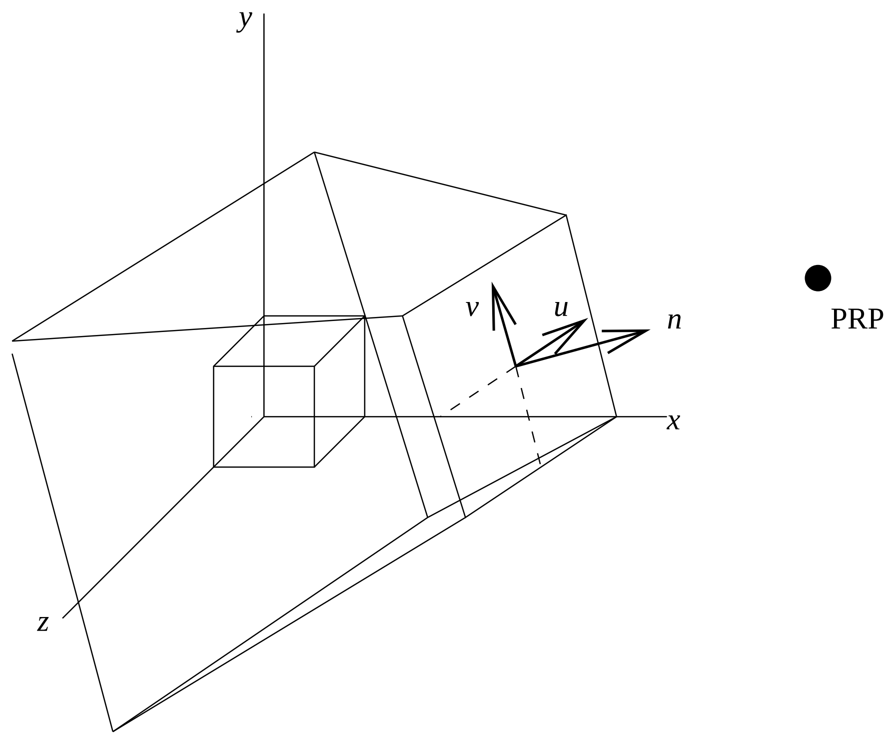


- La dirección de mira de la cámara puede ser cualquiera.

- La dirección vertical puede ser cualquiera.

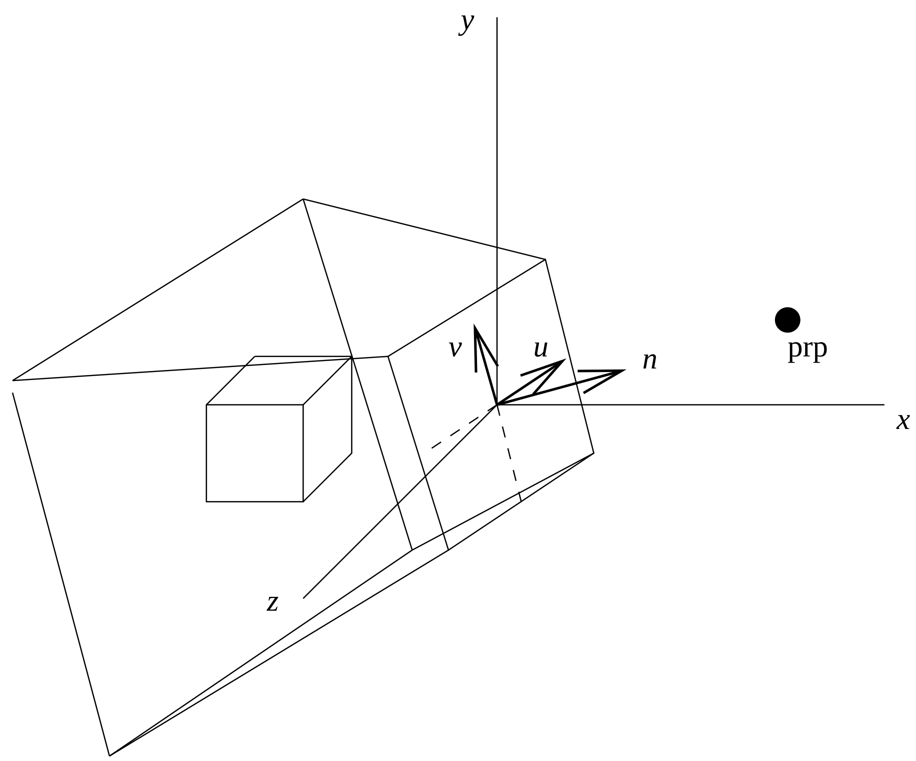
- Se da el volumen de visión. Para ello se ha de definir la ventana del plano de proyección, el plano cercano y el lejano.

Etapas en la generación de la imagen.



- Cambio del sistema de referencia del mundo al de la cámara: Se utilizan VRP, VPN y VUP. Han de establecerse el origen y la base del nuevo sistema de referencia:

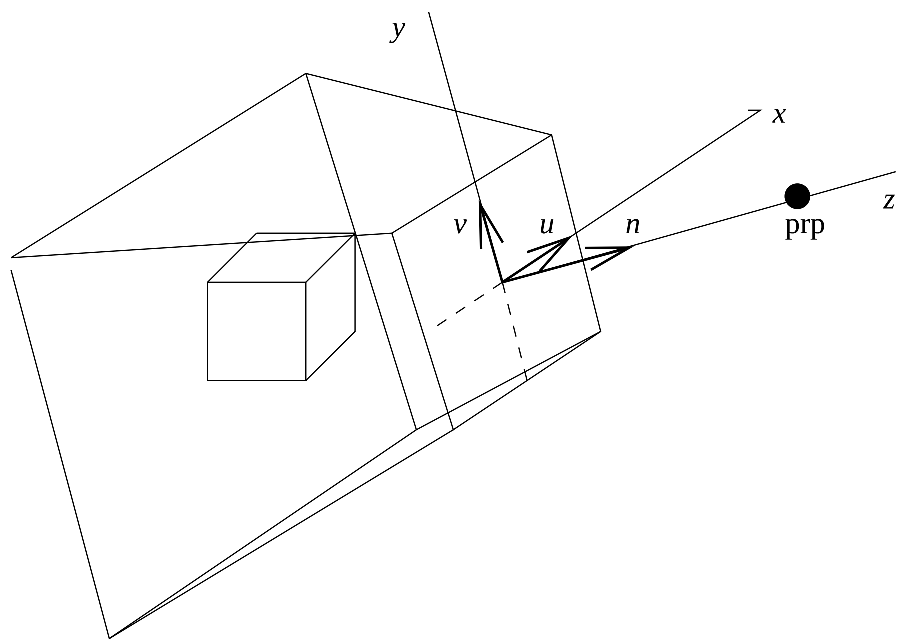
  1. El origen es VRP.



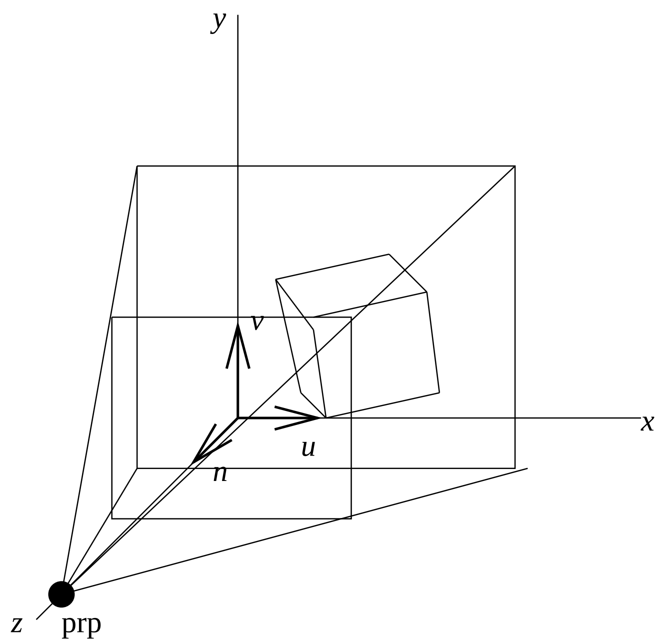
2. z_k: La cercanía ó lejanía depende de VPN.

3. x_k: Es perpendicular al plano definido por el vector VUP y z_k, por tanto, su producto vectorial nos dara su dirección.

4. y_k: Producto vectorial entre los dos anteriores.



- Hay que pasar la escena al nuevo sistema de referencia:



- Obtener la proyección, pero si se trata de una proyección oblicua hay que hacer coincidir la dirección z_c con el vector que va del centro de la ventana a PRP.

- Paso de ventana del plano de proyección a ventana de 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.

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

Superficies y curvas

Los objetos se pueden representar mediante ecuaciones y una ecuación la podemos representar de varias formas:

- Ecuación explícita: y=f(x) desventajas:
  - No admite curvas cerradas o curvas que para un valor de x permitan mas de una y.
  - Tras la aplicación de transformaciones afines la curva puede que no mantenga el subespacio de la ecuación original: La curva y=x^2 Tras rotar 90 grados se convierte en y=$\pm\sqrt{x}$
- Ecuación implicita: f(x,y)=0 Desventajas:

- Obtener los puntos de la curva puede ser complicado.

- Las transformaciones afines pueden hacer que la curva salga del espacio al que pertenece.

- Ecuaciones paramétricas: x=f(t), y=g(t) Los puntos de la curva se pueden obtener de una forma muy sencilla y si f y g son polinomios las transformaciones afines no alteran el subespacio. Desventajas:

  - Dado un valor x la obtención del punto (x, y) de la curva puede ser dificil.

Distintos usos de las curvas

- En general, la curva se desconoce.

- Se calcula en función de unos puntos de control.

- Dependiendo de las necesidades se utilizan distintos metodos:

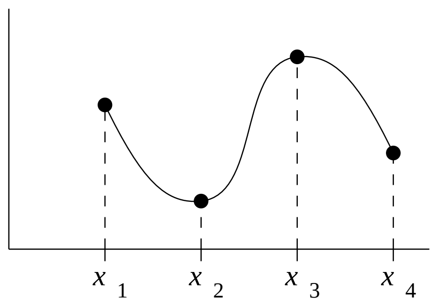
  1. Interpolación.

  2. Ajuste.

  3. Control de forma.

Interpolación

- Curva que pasa por los puntos de control.



- Sean en el eje OX $n+1$ puntos:

  $$x_0, \ldots, x_n \ / \ x_i < x_{i+1}$$

- La distancia entre x_i y x_{i+1} puede ser cualquiera pero normalmente será uniforme.

- A cada x_i le corresponde una y_i y buscamos la función que devuelva ese valor.

- Para representar esa función utilizaremos una base del espacio vectorial de polinomios de dimensión $n+1$.

- Representaremos nuestra función como combinación lineal de las funciones de la base.

  $$B = \{g_0(x), g_1(x), \ldots, g_n(x)\}$$

  $$f(x) = a_0 g_0(x) + a_1 g_1(x) + \ldots + a_n g_n(x)$$

- Las únicas incognitas son a_0, . . . , a_n.

- Las condiciones a cumplir: $f(x_i) = y_i$

- Un sistema de $n+1$ incógnitas y $n+1$ ecuaciones.

  $$\begin{pmatrix} g_0(x_0) & g_1(x_0) & \ldots & g_n(x_0) \\ g_0(x_1) & g_1(x_1) & \ldots & g_n(x_1) \\ \vdots & \vdots & & \vdots \\ g_0(x_n) & g_1(x_n) & \ldots & g_n(x_n) \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{pmatrix}$$

- Para que tenga una única solución su determinante debe ser distinto de 0.

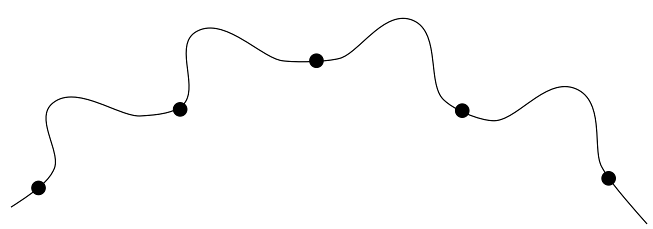
- El determinante depende del soporte y de la base.

- Se pueden utilizar bases que independientemente del soporte hagan que el determinante sea distinto de 0.

Problemas de la interpolación

- En función de la base del espacio vectorial pueden aparecer problemas; por ejemplo, la base $\{1, x, x^2, \ldots, x^n\}$ exige un alto costo computacional.

- Pueden aparecer oscilaciones, incluso aumentando el número de puntos de control.

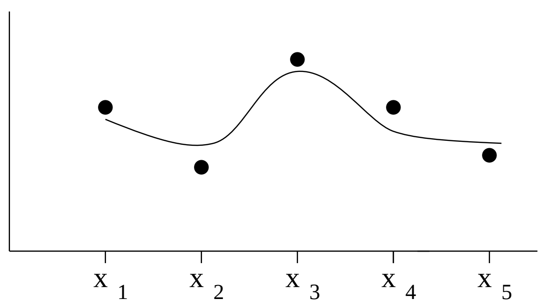


Ajuste

- La curva no tiene por qué pasar por los puntos de control.

- Se utiliza algun criterio de proximidad. Normalmente la norma euclídea o ajuste por mínimos cuadrados.

- Se utiliza un espacio de dimensión menor que $n-1$, en ese caso, la interpolación (en general) no tiene solución por lo que hay que buscar la curva que minimice el criterio de proximidad.



- Siendo el espacio de dimensión $m+1$ y la base $g_0(x)$, . . . , $g_m(x)$, podemos representar la función $g(x)$ como:

$$g(x) = a_0 g_0(x) + \ldots + a_m g_m(x)$$

- Y lo que hay que minimizar (utilizando la norma euclídea es:

$$F^2(x) = \sum_{i=0}^{n} \left(\; y_i - g(x_i) \; \right)^2 =$$

$$\sum_{i=0}^{n} \left(\; y_i - \textstyle\sum_{j=0}^{m} a_j g_j(x_i) \; \right)^2$$

- Lo consideramos como función de las variables a_0,. . . ,a_n.

- Para obtener el mínimo igualamos la derivadas parciales a cero:

$$\frac{\partial F(a_0, \ldots, a_n)}{\partial a_j} = 0$$

- Tenemos $m+1$ ecuaciones ($j = 0 \ldots m$):

$$2 \sum_{i=0}^{n} \left(\, y_i - \textstyle\sum_{k=0}^{m} a_k g_k(x_i) \, \right) (-g_j(x_i)) = 0$$

- Con unos pocos cambios:

$$\sum_{k=0}^{m} a_k \sum_{i=0}^{n} g_k(x_i) g_j(x_i) = \sum_{i=0}^{n} y_i g_j(x_i)$$

- Si decimos

$$g_{kj} = \sum_{i=0}^{n} g_k(x_i) g_j(x_i)$$

y

$$b_j = \sum_{i=0}^{n} y_i g_j(x_i)$$

Tenemos el siguiente sistema:

$$\begin{pmatrix} g_{00} & g_{01} & \cdots & g_{0m} \\ g_{10} & g_{11} & \cdots & g_{1m} \\ \vdots & \vdots & & \vdots \\ g_{m0} & g_{m1} & \cdots & g_{mm} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_m \end{pmatrix} = \begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_m \end{pmatrix}$$

Ya que g_{ij}=g_{ji} la matriz es simétrica y tiene por lo menos una solución.

Control de forma

- Los puntos de control no se usan para interpolarlos ni para que la curva se aproxime a ellos.

- Son para controlar la forma de la curva.

- Hay que escoger un Espacio vectorial concreto y una base concreta. La elección de la base tiene mucho que ver.

- Los parámetros de control se usan como coeficientes de las funciones de la base y en definitiva hay que evaluar las funciones de la base.

$$g(x) = a_0 g_0(x) + a_1 g_1(x) + \ldots + a_n g_n(x)$$

$a_0, a_1, \ldots, a_n$ Son los parametros de control.

- Al cambiar el parámetro de control a_i cambiamos la función $g_i(x)$.

$$g(x) = a_0 g_0(x) + \ldots + a_n g_n(x)$$

$$\overline{g}(x) = \overline{a_0} g_0(x) + \ldots + a_n g_n(x)$$

Y la diferencia entre las dos funciones es:

$$g(x) - \overline{g}(x) = (a_0 - \overline{a_0}) g_0(x)$$

- El cambio depende del comportamiento de g_0.

- Lo mejor es elegir funciones que devuelvan valores pequeños y que esten definidos en dominios limitados. Splines.

Bases de polinomios

- **Base de potencias:** $1, (x-a), (x-a)^2, \ldots, (x-a)^n$

  No sirve para control de forma, para interpolación exige $\frac{2n^3}{3}$ operaciones; y solo se puede utilizar para ajuste de minimos cuadrados.

- **Base de Lagrange:** Se trata de una base que se puede representar como función de los puntos del eje OX:

$$L_i(x) = \frac{(x-x_0)\ldots(x-x_{i-1})(x-x_{i+1})\ldots(x-x_n)}{(x_i-x_0)\ldots(x_i-x_{i-1})(x_i-x_{i+1})\ldots(x_i-x_n)}$$

$$P(x) = \sum_{i=0}^{n} a_i L_i(x)$$

No sirve para ajuste ni control de forma. para la interpolación, como $L_i(x_k) = 0$,cuando $x = x_k$:

$$P(x_k) = a_k$$

La solución a la interpolación es inmediata, lo dificil es el cálculo de la base, pero cuando el soporte es regular se sinplifica mucho.

- **Base de Newton:** Para obtener los coeficientes a_i que nos den una solución a la interpolación podemos utilizar un metodo repetitivo:

  $$N_i = (x - x_0)(x - x_1)\ldots(x - x_i)$$

  $$N_0 = 1$$

  y siendo

$y(x_i) = y_i$

$y(x_i, x_j) = \frac{y(x_j)-y(x_i)}{x_j-x_i}$

$y(x_i, x_j, x_k) = \frac{y(x_j,x_k)-y(x_i,x_j)}{x_k-x_i}$

$a_i = y(x_0, x_1, \ldots, x_i)$

interpola los puntos

$$P(x) = \sum_{i=0}^{n} a_i N_i(x)$$

- **Base de Bernstein:**

$$B_k(x) = \binom{n}{k} x^k (1-x)^{n-k}$$

definido en el intervalo $[0,1]$, debido a $(1-x)$. Eta

$$P(x) = \sum_{i=0}^{n} a_i B_i(x)$$

Los puntos (x_i, a_i) forman un polígono de control que cumple:

1. $P(0) = a_0$ eta $P(1) = a_n$

2. La tangente de $P(x)$ en los puntos 0 y 1 se pueden obtener mediante el primer y último lado del polígono.

3. Los puntos del plinomio $P(x)$ estan dentro de la envolvente convexa de los puntos de control.

4. No existe recta que corte más veces a la curva que al poligono formado por los puntos de control.

Es adecuado para control de forma.

- Resumiendo:

  1. Los polinomios no son adecuados para interpolación; tienen problemas de oscilación y exigen alto coste de computación ya que el número de puntos de control es el que establece la dimensión.

  2. En control de forma, y aunque pueda dar problemas de dimensión, se puede utilizar la base de Bernstein.

Curvas de Bezier

- Una curva de Bezier de grado 3 es el siguiente polinomio:

$$P(x) =$$

$$a_0(1-x)^3+a_1 3x(1-x)^2+a_2 3x^2(1-x)+a_3 x^3$$

- a_0, a_1, a_2 eta a_3 son puntos de control y definen la forma de la curva.

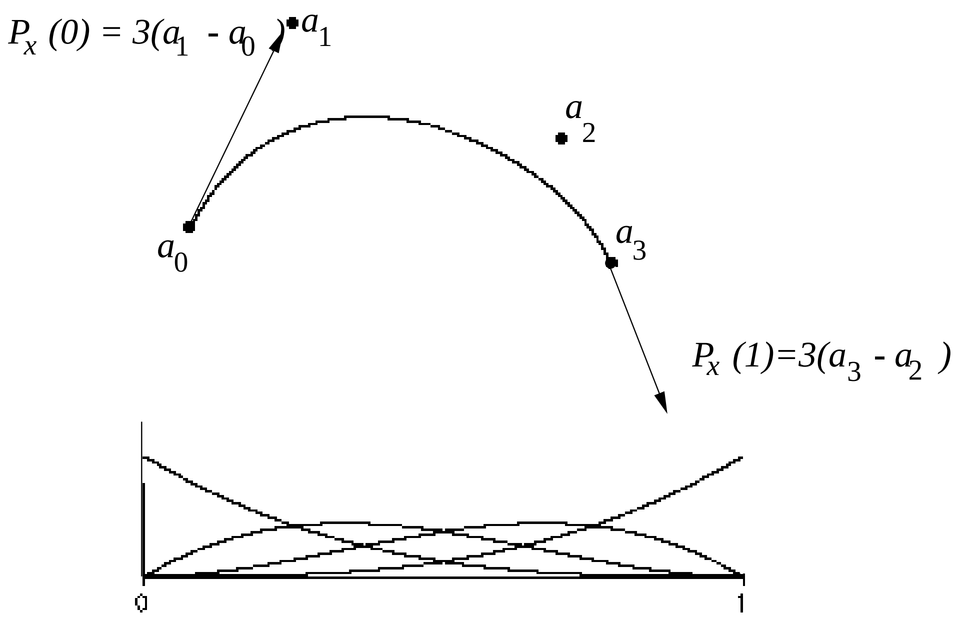
- $(1-x)^3, 3x(1-x)^2, 3x^2(1-x), x^3$ Son las funciones de la base de Bernstein.

- en el punto $x = 0$ toma el valor a_0 y en $x = 1$ su valor es a_3. Interpola los dos puntos.

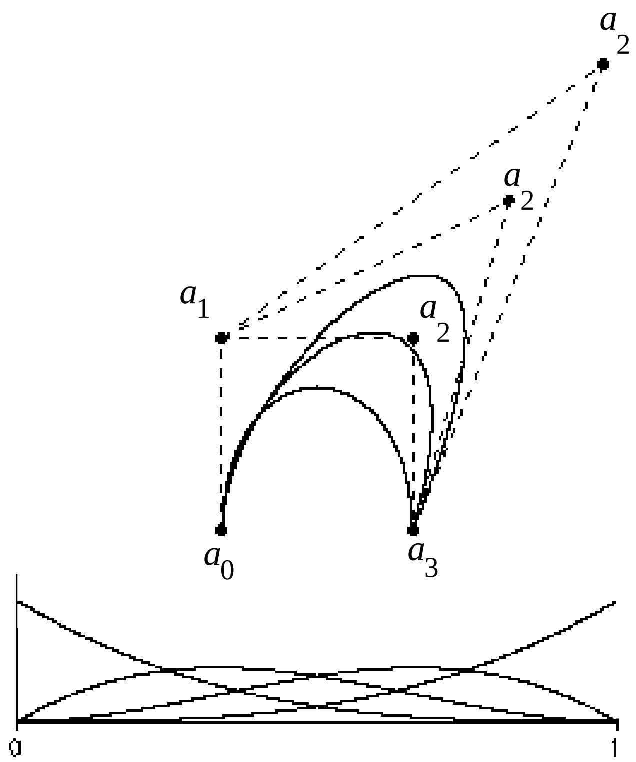
- La derivada en esos dos puntos:

$$P_x(0) = 3(a_1 - a_0)$$

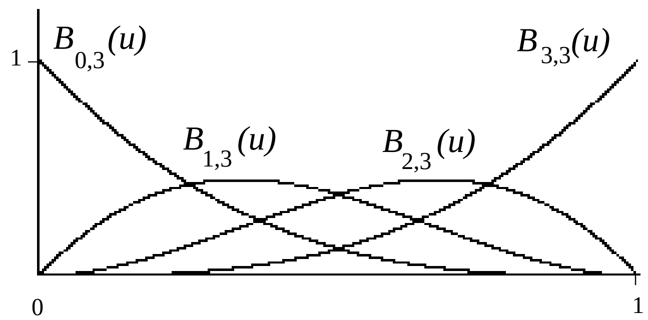
$$P_x(1) = 3(a_2 - a_3)$$



- Por tanto,cambiando los cuatro puntos de control cambiaremos la forma de la curva.



- El gráfico de las funciones de la base:



- Conviene fijarse en la zona de influencia de cada una.

- El cambio de cualquier punto de control tiene efecto en toda la curva.

- En general, con $n+1$ puntos de control tendremos una curva de Bezier de grado n pero las funciones de la base tambien han de ser de grado n.

  $$B(x) = \sum_{i=0}^{n} a_i B_{i,n}(x),$$

  donde $B_{i,n}(x) = \left(\begin{array}{c} n \\ i \end{array} \right) x^i (1-x)^{n-i}$ y donde $\left(\begin{array}{c} n \\ i \end{array} \right) = \frac{n!}{(i!(n-i)!)}$ es el coeficiente binómico.

- Para aplicar a la curva una transformación afin basta con aplicar la transformación a los puntos de control.

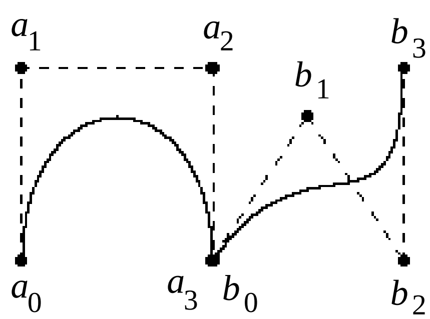
- El cálculo de la perspectiva no es afin.

Concatenación de curvas de Bezier

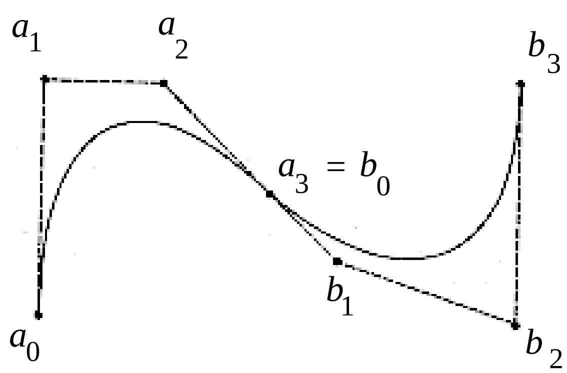
- Se pueden concatenar curvas.

- Para mantener la continuidad hay que cumplir ciertas condiciones.

- Si el ultimo punto de control de una curva coincide con el primero de la siguiente curva, las dos curvas se tocan.



- Para conseguir continuidad de derivada hay que cumplir más condiciones:


a_1
a_2
b_3
$a_3 = b_0$
b_1
a_0
b_2

Propiedades de Bezier

- Con n puntos de control, grado $n-1$.
- La corva toma la forma del poligono formado por los puntos de control y no sale de la envolvente de los mismos.
- El control de forma es global y no local. Un cambio de un punto de control afecta a toda la curva.
- Se interpolan el primer y último punto.
- La tangente en los extremos de la curva tiene la misma dirección que las aristas que forman los dos primeros y los dos últimos puntos de control.

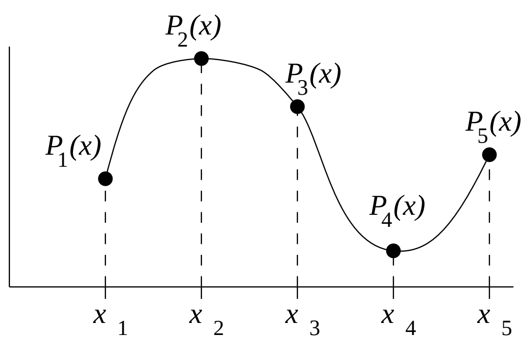
- Para aplicar cualquier transformación afin (cualquier combinación de transfromaciones lineales) basta con aplicarsela a los puntos de control.

Splines

- Se trata de una unión de distintos polinomios.

- Cada polinomio define la curva entre dos puntos de control.

- Es deseable obtener continuidad de máximo orden posible en los puntos de unión (mínimo orden 2).



- Si el grado de los polinomios es k (spline de orden k)la continuidad será como mucho $k-1$.

- Se utilizan splines cúbicos.

- El spline cúbico es la curva que traza un alambre flexible al hacerle pasar por ciertos puntos de control.

- Minimiza la curvatura media y la oscilación.

Obtencion de los Splines

- Sea $x_1, x_2, \ldots, x_n$ un soporte regular y sean $y_1, y_2, \ldots, y_n$ las alturas de los puntos.

- El polinomio P_i de grado 3 que interpola los puntos (x_i, y_i) y (x_{i+1}, y_{i+1}) es:

$$P_i(x) = c_{0i} + c_{1i}\left(\frac{x-x_i}{h}\right) + c_{2i}\left(\frac{x-x_i}{h}\right)^2 + c_{3i}\left(\frac{x-x_i}{h}\right)^3$$

- Ha de interpolar los dos puntos:

$$P_i(x_i) = y_i = c_{0i}$$

$$P_i(x_{i+1}) = y_{i+1} = c_{0i} + c_{1i} + c_{2i} + c_{3i}$$

- Para que los dos polinomios adyacentes tengan una unión suave:

$$P_i^{'}(x_i) = P_{i-1}^{'}(x_i)$$

$$P_i^{'}(x_{i+1}) = P_{i+1}^{'}(x_{i+1})$$

$$P_i'(x) = \frac{c_{1i}}{h} + 2\frac{c_{2i}}{h}\left(\frac{x-x_i}{h} \right) + 3\frac{c_{3i}}{h}\left(\frac{x-x_i}{h} \right)^2$$

$$P_i'(x_i) = \frac{c_{1i}}{h} = \sigma_i$$

$$P_i'(x_{i+1}) = \frac{c_{1i}}{h} + 2\frac{c_{2i}}{h} + 3\frac{c_{3i}}{h} = \sigma_{i+1}$$

- Con esas condiciones los coeficientes son:

$$c_{0i} = y_i$$

$$c_{1i} = h\sigma_i$$

$$c_{2i} = 3(y_{i+1} - y_i) - h(2\sigma_i + \sigma_{i+1})$$

$$c_{3i} = h(\sigma_i + \sigma_{i+1}) - 2(y_{i+1} - y_i)$$

- La condición de continuidad de la segunda derivada:

$$P_{i-1}''(x_i) = P_i''(x_i)$$

$$P_i''(x) = \frac{2c_{2i}}{h^2} + \frac{6c_{3i}}{h^2}\left(\frac{x-x_i}{h} \right)$$

- Por tanto:
$$\frac{2c_{2i-1}}{h^2}+\frac{6c_{3i-1}}{h^2}\left(\frac{x_i-x_{i-1}}{h}\right)=\frac{2c_{2i}}{h^2}+\frac{6c_{3i}}{h^2}\left(\frac{x_i-x_i}{h}\right)$$

- Como $x_i - x_{i-1} = h$:
$$\frac{2c_{2i-1}}{h^2}+\frac{6c_{3i-1}}{h^2}=\frac{2c_{2i}}{h^2}$$

- Sustituyendo c_{2i-1}, c_{3i-1} y c_{2i}:
$$\sigma_{i-1}+4\sigma_i+\sigma_{i+1}=\frac{3(y_{i+1}-y_{i-1})}{h}=b_i$$

- n ecuaciones y $n-2$ incógnitas.

- Suponiendo conocidas σ_1 y σ_n tenemos $n-$

2 ecuaciones y $n-2$ incógnitas:

$$\begin{pmatrix} 4 & 1 & 0 & \dots & & & 0 \\ 1 & 4 & 1 & \dots & & & 0 \\ 0 & 1 & 4 & \dots & & & 0 \\ \vdots & \vdots & \vdots & & & & \vdots \\ 0 & 0 & 0 & \dots & 1 & 4 & 1 \\ 0 & 0 & 0 & \dots & & 1 & 4 \end{pmatrix} \begin{pmatrix} \sigma_2 \\ \sigma_3 \\ \sigma_4 \\ \vdots \\ \sigma_{n-2} \\ \sigma_{n-1} \end{pmatrix} = \begin{pmatrix} b_2 - \sigma_1 \\ b_3 \\ b_4 \\ \vdots \\ b_{n-2} \\ b_{n-1} - \sigma_n \end{pmatrix}$$

- Matriz de Hadaward. Tiene una única solución:

- Sustituyendo σ_i obtenemos los coeficientes c_{ki} del polinomio interpolador.

- depende de las coordenadas Y y σ_1 y σ_n.

- El conjunto de splines cúbicos forma un espacio de dimensión $n+2$.

B-splines: Splines Básicos

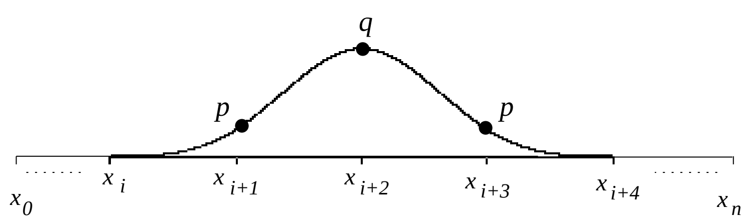
- En el control de forma la base tiene gran importancia.

- Ante una variación del coeficiente a_k el cambio sufrido por la curva:

$$g(x) - \overline{g}(x) = (a_0 - \overline{a_0})b_k(x)$$

- Conviene que el cambio sea local, para que varie unicamente los alrededores de a_0.

- Las funciones de la bese deberían tener la siguiente forma:



- Facilmente obtenible mediante splines; las condiciones a cumplir:

$$
\begin{aligned}
&B(x_1) = B(x_2) = \ldots = B(x_{k-2}) = 0\\
&B(x_{k-1}) = p\\
&B(x_k) = q\\
&B(x_{k+1}) = p\\
&B(x_{k+2}) = B(x_{k+3}) = \ldots = B(x_n) = 0\\
&B^{'}(x_0) = 0 = \sigma_1\\
&B^{'}(x_n) = 0 = \sigma_n
\end{aligned}
$$

- Tomando $q = 4p$, los resultados del sistema que calcula $\sigma_i (i = 2 \ldots n-1)$:

$$
\begin{aligned}
&\sigma_2 = \sigma_3 = \ldots = \sigma_{k-2} = 0\\
&\sigma_{k-1} = \tfrac{3q}{4h}\\
&\sigma_k = 0\\
&\sigma_{k+1} = \tfrac{-3q}{4h}\\
&\sigma_{k+2} = \sigma_{k+3} = \ldots = \sigma_{n-1} = 0
\end{aligned}
$$

- Tomando $q = 2/3$ y sustituyendo los coeficientes de los polinomios:

$$
\begin{aligned}
c_{0i} &= y_i \\
c_{1i} &= h\sigma_i \\
c_{2i} &= 3(y_{i+1} - y_i) - h(2\sigma_i + \sigma_{i+1}) \\
c_{3i} &= h(\sigma_i + \sigma_{i+1}) - 2(y_{i+1} - y_i)
\end{aligned}
$$

- Obtenemos los polinomios que forman el spline básico B_i:

$$P_{i-2}(x) = \frac{1}{6}\left(\frac{x-x_{i-2}}{h} \right)^3$$

$$P_{i-1}(x) = \frac{1}{6} + \frac{1}{2}\left(\frac{x-x_{i-1}}{h} \right) + \frac{1}{2}\left(\frac{x-x_{i-1}}{h} \right)^2 -$$

$$-\frac{1}{2}\left(\frac{x-x_{i-1}}{h} \right)^3$$

$$P_i(x) = \frac{2}{3} - \left(\frac{x-x_i}{h} \right)^2 + \frac{1}{2}\left(\frac{x-x_i}{h} \right)^3$$

$$P_{i+1}(x) = \frac{1}{6} - \frac{1}{2}\left(\frac{x-x_{i+1}}{h} \right) + \frac{1}{2}\left(\frac{x-x_{i+1}}{h} \right)^2 -$$

$$-\frac{1}{6}\left(\frac{x-x_{i+1}}{h} \right)^3$$

- Conocemos los splines, pero solo tenemos $n-4$ splines: $B_3, B_4, \ldots, B_{n-2}$. Para obtener los 6 splines que nos faltan para completar la base (se trata de un espacio de dimensión $n+2$), introducimos 6 puntos, 3 a cada lado.

- Con esos B-splines podemos definir cualquier spline $S(x)$:

$$S(x) = \sum_{k=0}^{n+1} a_k B_k(x)$$

- Siendo $x \in [x_k, x_{k+1}]$, para calcular $S(x)$ basta con realizar 4 sumas, ya que solo 4 de los B-splines tienen valor distinto a 0 en ese tramo.

$$S(x) = \sum_{i=k-1}^{k+2} a_i B_i(x)$$

- Haciendo $t = \frac{x - x_k}{h}$:

$$S(t) = a_{k-1}(\tfrac{1}{6} - \tfrac{1}{2}t + \tfrac{1}{2}t^2 - \tfrac{1}{6}t^3) + a_k(\tfrac{2}{3} - t^2 + \tfrac{t^3}{2}) +$$

$$+ a_{k+1}(\tfrac{1}{6} + \tfrac{t}{2} + \tfrac{t^2}{2} - \tfrac{t^3}{2}) + a_{k+2}(\tfrac{t^3}{6})$$

  simplificando:

$$S(x) = \frac{a_{k-1} + 4a_k + a_{k+1}}{6} + \frac{a_{k-1} + a_{k+1}}{2}t +$$

$$\frac{a_{k-1} + 2a_k + 2a_{k+1}}{2}t^2 + \frac{a_{k-1} + 3a_k + 3a_{k+1} + a_{k+2}}{6}t^3 =$$

$$s_0 + s_1 t + s_2 t^2 + s_3 t^3$$

Propiedades de los B-splines

- La suma de los cuatro sumandos en un punto cualquiera es 1, por lo que:

$$S(x) \leq \max(a_{k-1}, a_k, a_{k+1}, a_{k+2})$$

- La curva esta dentro del poligono formado por los puntos de control.

- El cambio debido a la modificación de un punto de control:

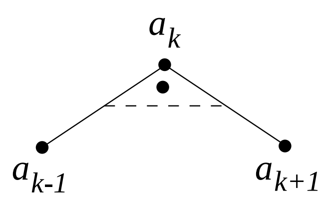
$$(a_i - \overline{a_i})B_i(x)$$

  Al tener valores distintos de 0 solo en 4 tramos la curva se ve afectada solo en esos 4 tramos. Control local.

- El valor que le corresponde a la curva en un punto del soporte, $x = x_i$ es

$$S(x_i) = (a_{k-1} + 4a_k + a_{k+1})/6$$

cercano a a_k.



Interpolacion mediante B-splines

- Ya que conocemos el valor de la curva en los puntos $x-i$, podemos plantear el problema a la inversa: Dados los puntos por los que pasa obtener los coeficientes a_i.

- n ekuaciones y $n+2$ incógnitas:

$$a_{i-1} + 4a_i + a_{i+1} = 6y_i \quad i = 1, 2, \ldots, n$$

$$\begin{pmatrix} 4 & 1 & 0 & \ldots & & & 0 \\ 1 & 4 & 1 & \ldots & & & 0 \\ 0 & 1 & 4 & \ldots & & & 0 \\ \vdots & \vdots & \vdots & & & & \vdots \\ 0 & 0 & 0 & \ldots & 1 & 4 & 1 \\ 0 & 0 & 0 & \ldots & & 1 & 4 \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_{n-1} \\ a_n \end{pmatrix} = \begin{pmatrix} 6y_1 - a_0 \\ 6y_2 \\ 6y_3 \\ \vdots \\ 6y_{n-1} \\ 6y_n - a_{n+1} \end{pmatrix}$$

B-splines uniformes

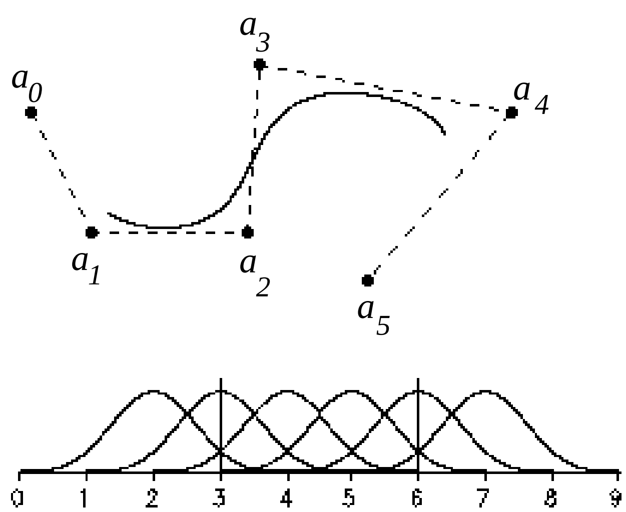
- Mediante B-splines para definir un tramo de curva necesitamos 4 puntos de control y 4 splines básicos.

- Cada spline básico tiene 4 polinomios, por lo que necesita 5 puntos (llamados tambien puntos de nudo).

- Cada B-spline utiliza 4 puntos de nudo del anterior B-spline mas otro nudo, por lo que en el vector de nudos hacen falta 4 nudos mas que B-splines.

- Para una curva que queremos que tenga m tramos necesitamos $m + 3$ B-splines (y puntos de control) y $m+7$ puntos de nudo.

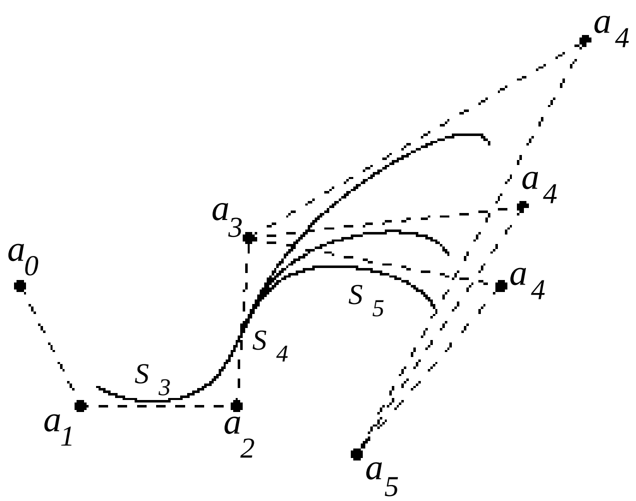
- por ejemplo, con 6 puntos de control podemos dibujar 3 tramos y necesitamos un vector de nudos de 10 elementos.



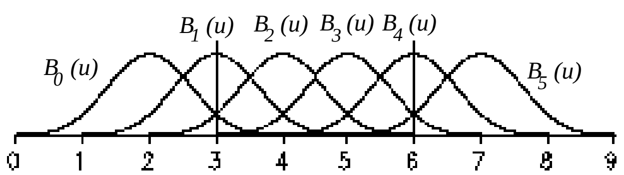
- Decir que un B-spline es uniforme equivale a decir que los puntos de nudo estan igualmente espaciados.

Control local

- Ejemplo del control local: aunque movamos a_4 el tramo S_3 no se modifica.



- Para obtener el valor de la curva en un punto dado hay que calcular el valor de 4 B-splines.



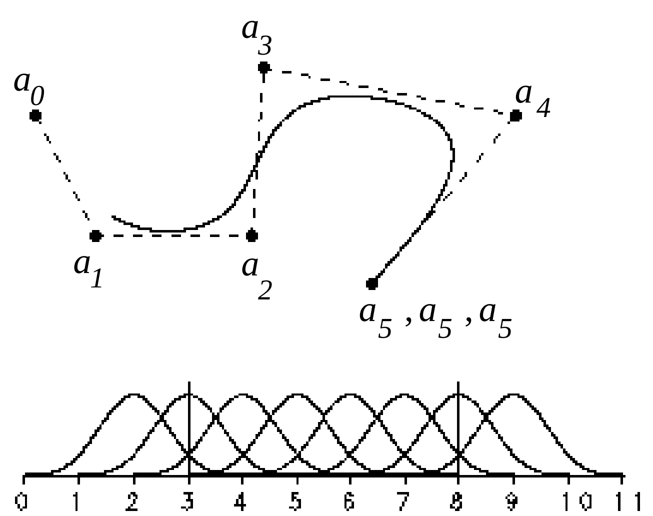
- El cuarto polígono del B-spline izquierdo, el tercer polígono del segundo B-spline, el segundo del siguiente B-spline y el primer polígono del B-spline derecho. Con $x_4 \leq x \leq x_5$

$$
\begin{aligned}
&B_1 = 1/6(1-x)^3 \\
&B_2 = 1/6(3x^3 - 6x^2 + 4) \\
&B_3 = 1/6(-3x^3 + 3x^2 + 3x + 1) \\
&B_4 = 1/6x^3
\end{aligned}
$$

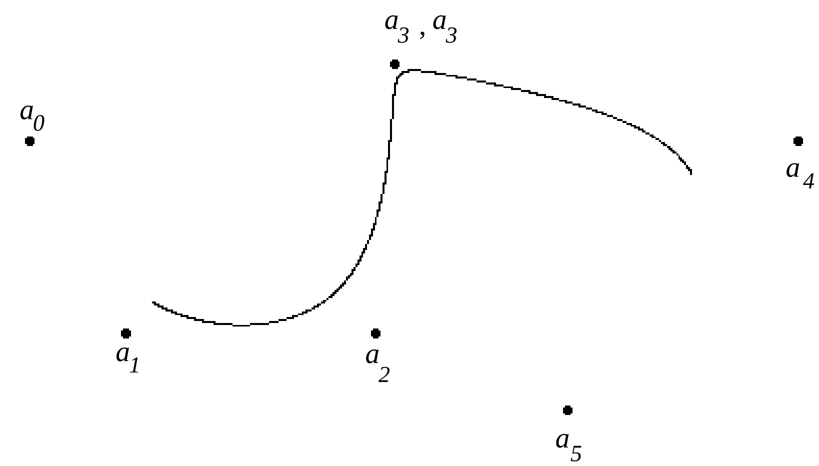
Puntos de control repetidos

- La B-splines no interpolan los puntos de control de los extremos.

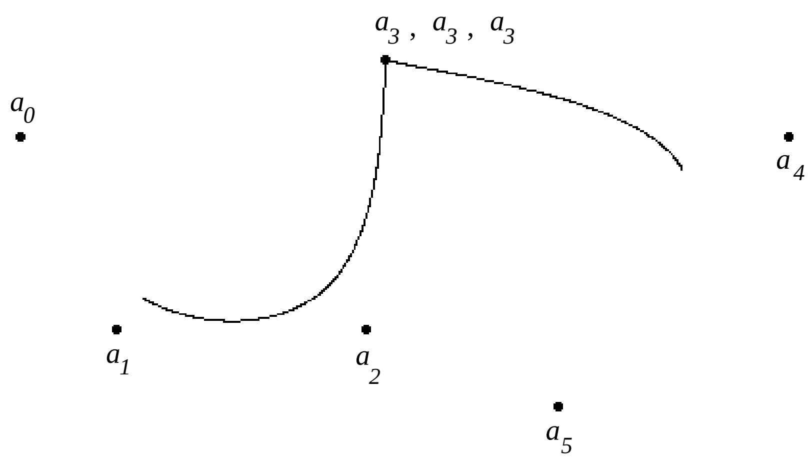
- Repitiendo los puntos de control podemos interpolarlos.



- introduciendo puntos de control repetidos disminuimos la continuidad.



a_3 , a_3
a_0
a_4
a_1
a_2
a_5



a_3 , a_3 , a_3
a_0
a_4
a_1
a_2
a_5

B-splines no uniformes

- Las distancias entre puntos de nudo puden tomar valor 0.

- Como consecuencia, los splines que forman la base tendran formas distintas.

- Cada spline básico necesita 5 puntos de nudo, al tener varios repetidos obligamos al spline a tener una longitud mas corta y su forma cambia.

- Existe un algoritmo recursivo para obtener los splines de la base segun el vector de nudos. Para calcular $B_i(x)$:

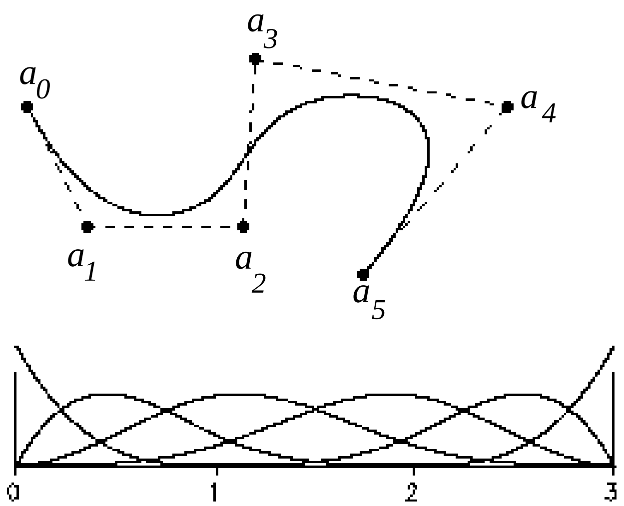
$$B_{i,1}(x) = \begin{cases} 1 & x_i \leq x \leq x_{i+1} \\ 0 & \text{bestela} \end{cases}$$

$$B_{i,2}(x) = \frac{x - x_i}{x_{i+1} - x_i} B_{i,1}(x) + \frac{x_{i+2} - x}{x_{i+2} - x_{i+1}} B_{i+1,1}(x)$$

$$B_{i,3}(x) = \frac{x - x_i}{x_{i+2} - x_i} B_{i,2}(x) + \frac{x_{i+3} - x}{x_{i+3} - x_{i+1}} B_{i+1,2}(x)$$

$$B_{i,4}(x) = \frac{x - x_i}{x_{i+3} - x_i} B_{i,3}(x) + \frac{x_{i+4} - x}{x_{i+4} - x_{i+1}} B_{i+1,3}(x)$$

- Para interpolar los puntos de control de los extremos se repiten los puntos de nudo iniciales y finales.



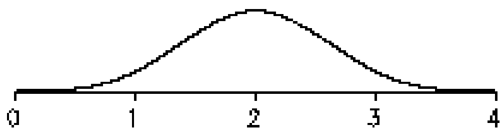
[0,0,0,0,1,2,3,3,3,3]

- El Knot-vector $[0, 0, 0, 0, 1, 1, 1, 1]$ da como resultado las curvas de Bezier.

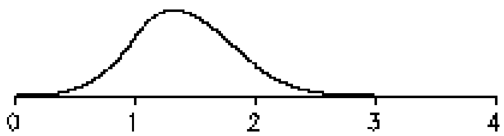
Puntos de nudo repetidos

- Al repetir los de los estremos logramos interpolar los puntos de control inicial y final.

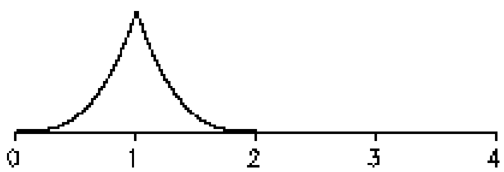
- Un ejemplo de funciones de la base:



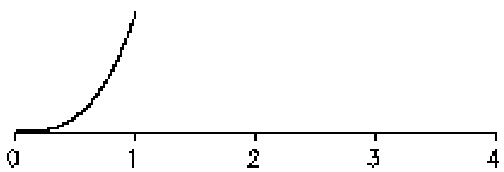
[0,1,2,3,4]



[0,1,1,2,3]

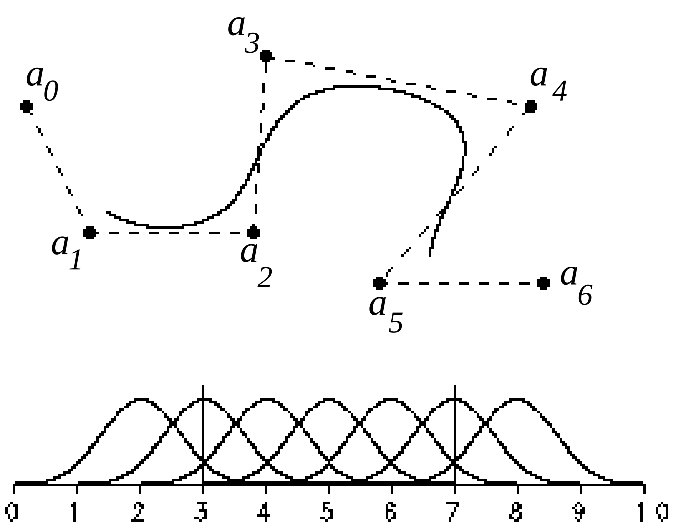


[0,1,1,1,2]

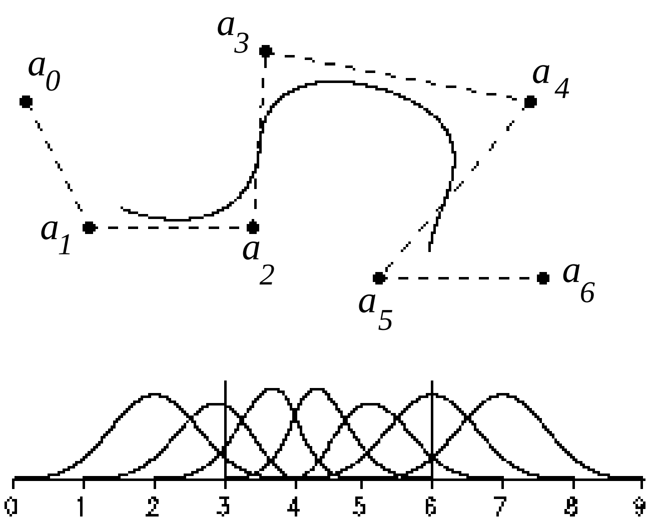


[0,1,1,1,1]

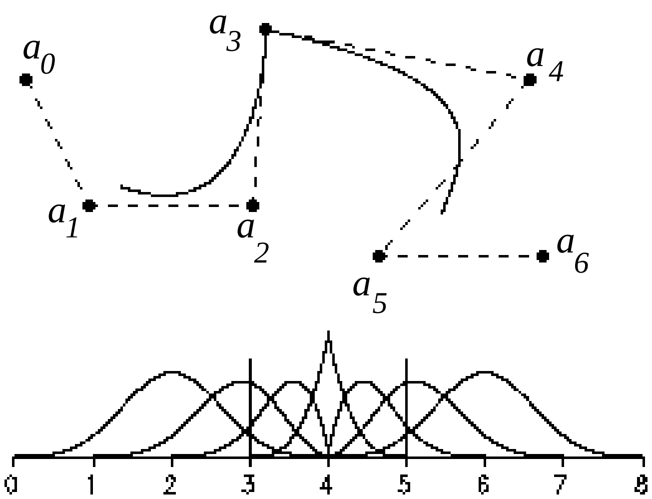
- Consecuencias que sufre la curva con los puntos de nodo repetidos:



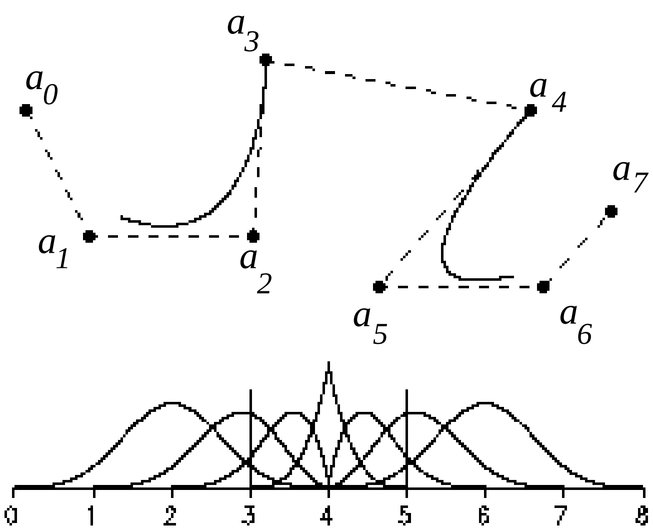
[0,1,2,3,4,5,6,7,8,9,10]



[0,1,2,3,4,4,5,6,7,8,9]



[0,1,2,3,4,4,4,5,6,7,8]



[0,1,2,3,4,4,4,4,5,6,7,8]

Superficies regulares

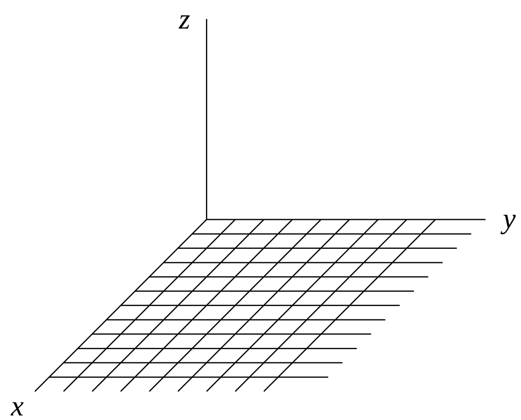
- 3D es más complejo.

- Para lograr realismo habrá que trabajar con esas superficies.

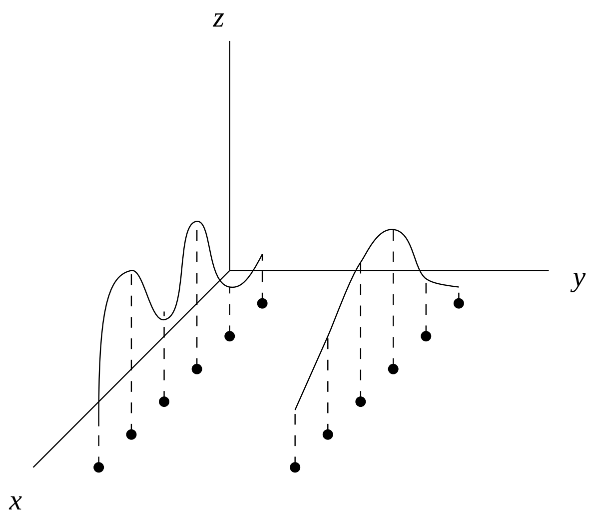
- Las ecuaciones más utilizadas son las paramétricas: $x = f(u, v), y = g(u, v)$ eta $z = h(u, v)$. El lapráctica son dificiles de manejar y se utilizan ecuaciones de la forma $x = f(u, v) = f_1(u)f_2(v)$ (producto tensorial).

Interpolación.

- El soporte lo forman $x_0, \ldots, x_n$ eta $y_0, \ldots, y_m$.



- Para interpolar la superficie $z_{ij} = f(x_i, y_j)$ podemos calcular $n+1$ funciones, para cada y_j una función.



- Utilizando la baseg_0,. . . ,g_n:

$$g(x,0) = \sum_{i=0}^{n} a_{i0} g_i(x) \quad \text{funtzioak} \quad z_{00} \ldots z_{n0} \quad \text{interpolatzen ditu}$$

$$g(x,1) = \sum_{i=0}^{n} a_{i1} g_i(x) \quad \text{funtzioak} \quad z_{01} \ldots z_{n1} \quad \text{interpolatzen ditu}$$

$$\vdots \qquad\qquad\qquad\qquad \vdots$$

$$g(x,m) = \sum_{i=0}^{n} a_{im} g_i(x) \quad \text{funtzioak} \quad z_{0m} \ldots z_{nm} \quad \text{interpolatzen ditu}$$

- Visto de otra forma:

$$g(x,0) = a_{00} g_0(x) + a_{10} g_1(x) + \ldots + a_{n0} g_n(x)$$

$$g(x,1) = a_{01} g_0(x) + a_{11} g_1(x) + \ldots + a_{n1} g_n(x)$$

$$\vdots$$

$$g(x,m) = a_{0m} g_0(x) + a_{1m} g_1(x) + \ldots + a_{nm} g_n(x)$$

- Por otro lado, mediante una base de $m+1$ funciones podemos interpolar $a_{00}, a_{01}, \ldots a_{0m}$:

$$\begin{array}{lcl} h(0,y) = \sum_{j=0}^{m} b_{0j} h_j(y) & \text{interpola} & a_{00} \ldots a_{0m} \\ \quad\vdots & & \quad\vdots \\ h(n,y) = \sum_{j=0}^{m} b_{nj} h_j(y) & \text{interpola} & a_{n0} \ldots a_{nm} \end{array}$$

- Con todo lo visto:

$$f(x,y) = \sum_{i=0}^{n} a_{ij} g_i(x) = \sum_{i=0}^{n} \sum_{j=0}^{m} b_{ij} h_j(y) g_i(x)$$

- Por ejemplo, en el caso (x_2, y_3):

$$\sum_{i=0}^{n} \left(\sum_{j=0}^{m} \left(b_{ij} h_j(y_3) \right) g_i(x_2) \right) = \sum_{i=0}^{n} a_{i3} g_i(x_2) = z_{23}$$

Control de forma.

- Al igual que en la interpolación, la superficie se representa como producto de dos familias de funciones:

$$f(x,y) = \sum_{i=0}^{n} \sum_{j=0}^{m} b_{ij} g_i(x) h_j(y)$$

- Las bases tienen gran importancia, sobre todo en el control local.

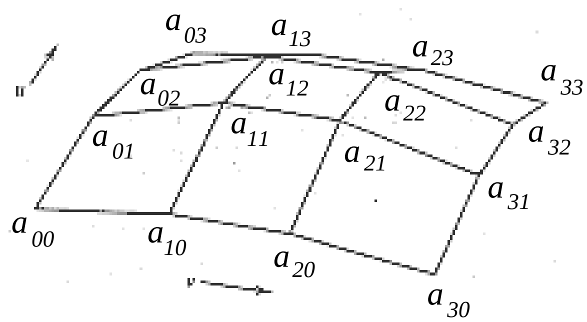
- Podemos usar la base de Bernstein o B-splines.

- Caso de Bezier:

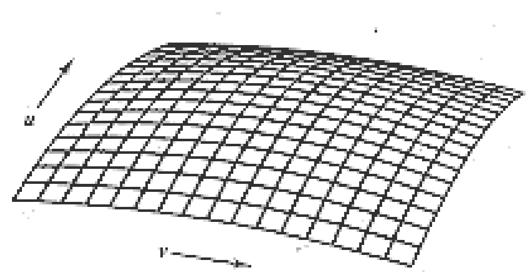
$$B_i(x) = \binom{n}{i} x^i (1-x)^{n-i}$$

$$P(x,y) = \sum_{i=0}^{n}\sum_{j=0}^{m}\binom{n}{i}\binom{m}{j} y^j(1-y)^{m-j}x^i(1-x)^{n-i}z_{ij}$$

- Los puntos de control y la superficie:

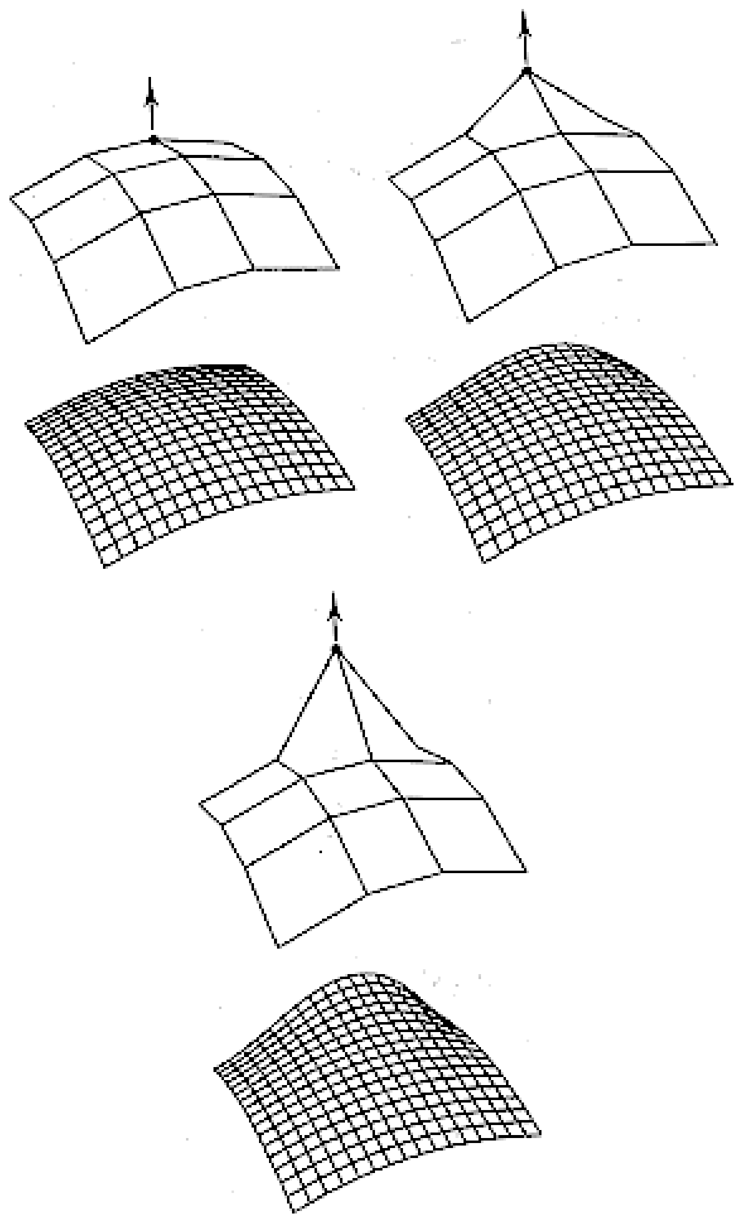


(a)

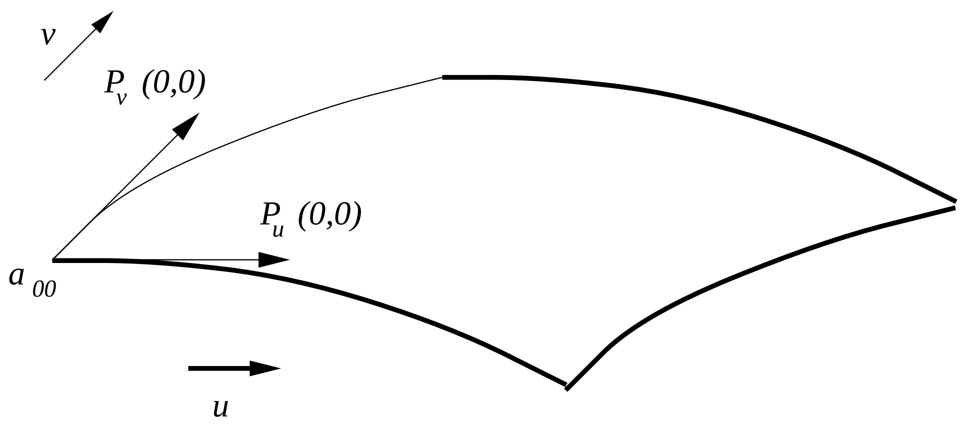


(b)

- Control-local:



- Tangentes:



- Para iluminación hace falta el vector normal, podemos utilizar las tangentes o la división de parches.

Unión de parches de Bezier

- Para que los parches P y R tengan una arista comun:

$$P(1, v) = R(0, v) \quad \text{non}0 \leq v \leq 1$$

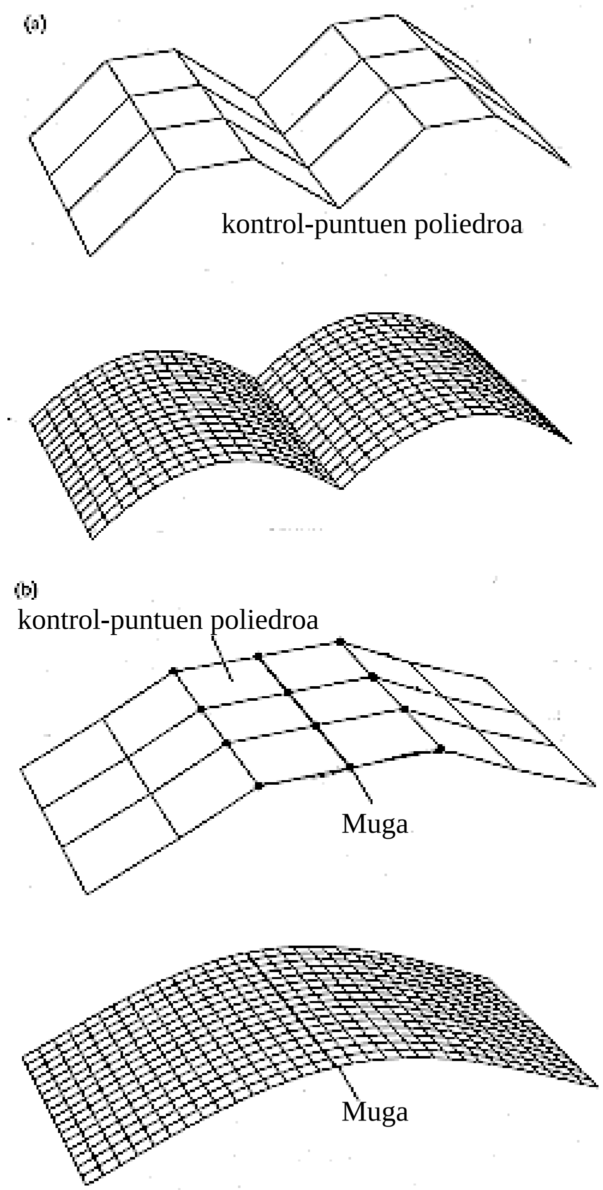
- En consecuencia:

$$\begin{aligned} P_{30} &= R_{00} \\ P_{31} &= R_{10} \\ P_{32} &= R_{20} \\ P_{33} &= R_{30} \end{aligned}$$

- Para la continuidad de primer orden:

$$(P_{3i} - P_{2i}) = k(R_{1i} - R_{0i}) \;\; i = 0, \ldots, 3$$

- Ejemplos:



(a)
kontrol-puntuen poliedroa
(b)
kontrol-puntuen poliedroa
Muga
Muga

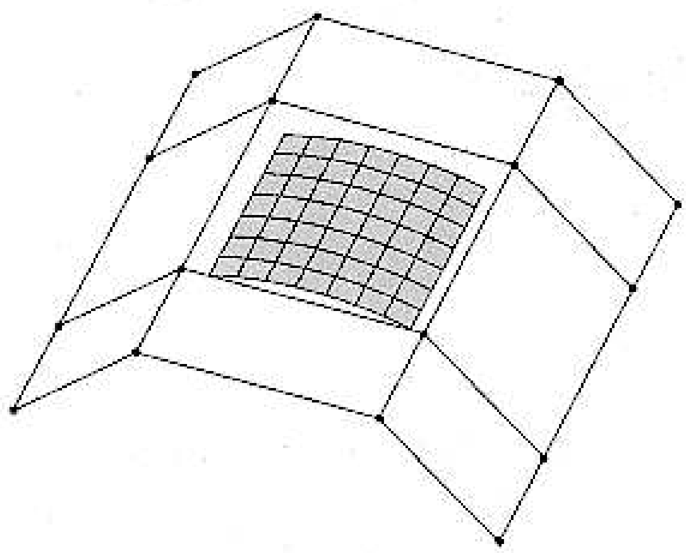
Parches de B-splines

- La ecuación:

$$P(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{m} P_{ij} B_i(u) B_j(v)$$

- 4x4 puntos de control concretan un parche.

- necesitan 8x8 nudos.



- Podemos repetir puntos de control: