Ampliación de Informática Gráfica

Tema 1. Modelos de Representación de Objetos 3D

Índice del Tema.

- Modelos de Representación Poligonal
- Modelos de Superficies:
  - Superficies Implícitas.
  - Superficies Paramétricas.
- Modelo de Sólidos
  - Modelos de Barrido
  - Geometría Sólida Constructiva.
- Modelos de Partición espacial
  - Partición no jerárquica.
  - Partición jerárquica.

Objeto de los modelos de representación de objetos 3D

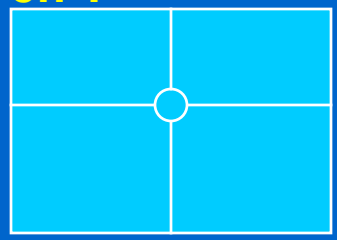
- Definir estructuras de datos 3D capaces de representar las propiedades geométricas y físicas de los objetos para su computación gráfica.
  - Que es el interior
  - Que es el exterior
  - Que pasa en la superficie.
- Aproximaciones al problema:
  - Definición de la superficie de separación. Modelos de superficie.
  - Definición de todo el espació ocupado por el objeto. Modelos de Sólidos.
- Requisitos:
  - Facilidad de generación de objetos
  - Aspecto final de la representación.
  - Requerimientos de computo de la aplicación

Modelos de Superficie

- Cuándo Utilizamos los modelos de superficie:
  - El objeto a representar es muy similar a una superficie: ejemplo una bandera, una lamina metálica (el capo de un coche).
  - Sólo nos interesa el aspecto externo del objeto.
  - El tipo de biblioteca gráfica sólo soporta este modelo.
- Propiedades exigibles a los modelos de superficies en gráficos: que sean variedades bidimensionales (*2-mainfold*)
  - No se intersectan consigo mismas.
- A veces tenemos superficies abiertas.
- Cuando la superficie es cerrada se denominan a los objetos modelos de frontera o B-rep (boundary representation). Posibilidad de cambio con modelos sólidos.
- Aproximaciones al modelado de superficies:
  - Discreta o aproximada: modelo poliédrico
  - Continua o exacta: representación matemática de superficies

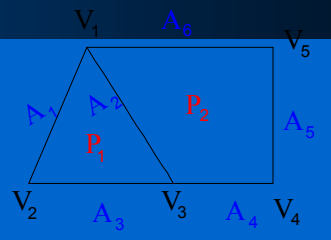
Modelos de Superficies Poligonales (1/7)

- Se basa en describir la superficie a partir de un conjunto de polígonos conectados. (Vértices + Aristas).
- Que significado tiene el concepto de variedades bidimensional en polígonos:
  - Dos polígonos no se intersectan entre sí, excepto en las artistas
  - En una arista no deben coincidir más de dos polígonos.
  - Un vértice debe pertenecer al menos a dos aristas.
- Otros problemas a considerar en la representación polígonal:
  - Polígonos degenerados.
  - En polígonos de más de tres vértices, falta de coplanariedad. Efecto *bowtie* o pajarita
  - Problemas de aspecto con vértices en T



- Ventajas del modelo:
  - Simplicidad.
  - El hardware esta con él.
- Limitaciones del modelo:
  - La naturaleza no es polígonal.
  - *Aliasing* de polígonos debido a la discretización.
  - No es sencillo modelar objetos complejos.

Modelos de Superficies Poligonales (2/7)
Estructuras de Datos.



- **Especificación del Modelo:**
  - **Representación Doblemente Indexada.**
    - **Facilita la consulta de vecindad. Modelos multiresolución.**
  - **Representación Indexada.**
    - **Facilita la determinación de vértices comunes**
    - **Problemas representación normales.**
  - **Representación Explicita.**
    - **Gran espacio de almacenamiento.**
    - **Rapidez de proceso.**

Tabla de Polígonos

P_1: $X_1\ Y_1\ Z_1$
$X_2\ Y_2\ Z_2$
$X_3\ Y_3\ Z_3$
P_2: ----------

(NO INDEXADA)

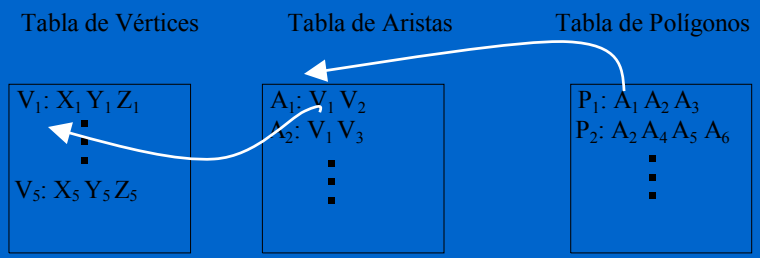


Tabla de Polígonos

P_1: $V_1\ V_2\ V_3$
P_2: $V_1\ V_3\ V_4\ V_5$

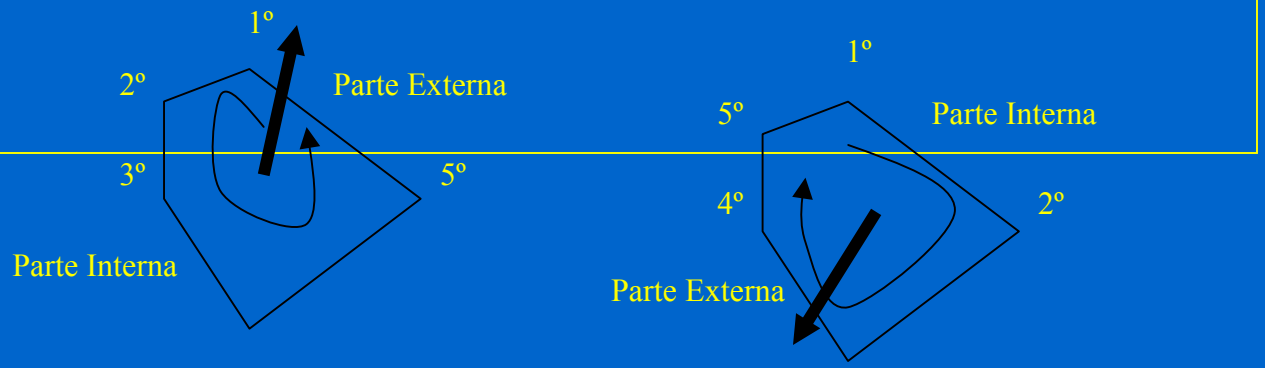
Tabla de Vértices

V_1: $X_1\ Y_1\ Z_1$
V_5: $X_5\ Y_5\ Z_5$

(INDEXADA)

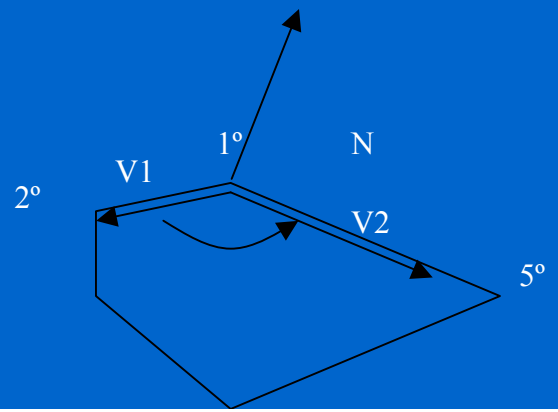
Modelos de Superficies Poligonales (3/7)
Consideraciones Adicionales Sobre Polígonos

- **Orden de indicación de los Vértices.**
  - **Problemas de degeneración.**
  - **Utilidad para determinar la parte delantera y trasera.**
  - **Útil en algoritmos de optimización y eliminación de parte nos visibles.**
  - **Regla del sacacorchos.**



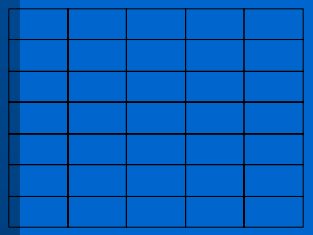
Modelos de Superficies Poligonales (4/7)
Consideraciones Adicionales Sobre Polígonos

- **Determinación de Coplanariedad.**
  - **Ecuación General del Plano: Ax+By+Cz-D=0**
  - **A,B,C, coordenadas del vector normal al plano.**
- **Cálculo de Normales.**
  - **Producto vectorial: N=V1xV2**
  - **Importancia del Orden, de nuevo sacacorchos.**

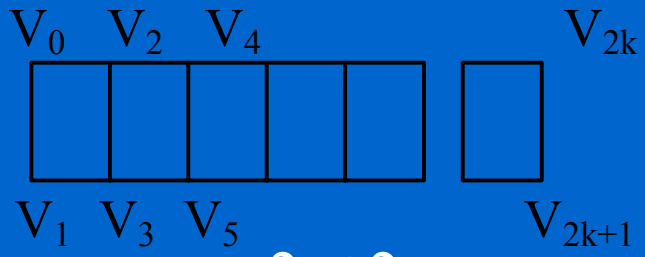


Modelos de Superficies Poligonales (5/7)
Primitivas Poligonales.

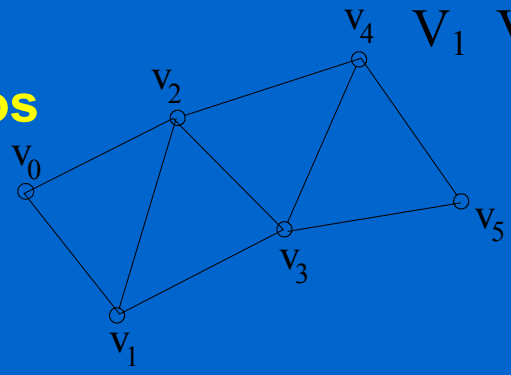
- **Normalmente se trabaja a partir de primitivas poligonales optimizadas.**
  - **Reducimos la transferencia de datos.**
  - **Disminuimos flexibilidad.**
- **Primitivas más comunes**
  - **Tira de cuadrados.**
  - **Tira de Triángulos.**
  - **Abanico de Triángulos**
  - **Matriz de cuadrados.**



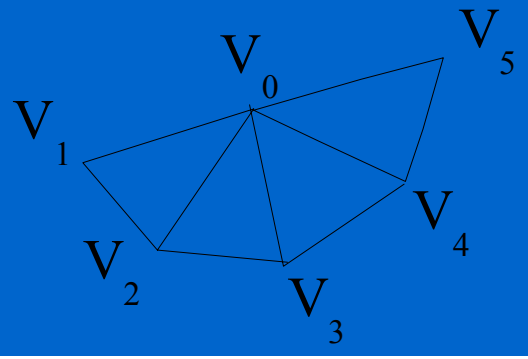
nxm



2n+2



n+2



Modelos de Superficies Poligonales (6/7)
Modelo Poligonal sobre OpenGL.(1/2)

- **OpenGL es una biblioteca gráfica de bajo nivel que implementa mas de 150 funciones gráficas.**
- **Utiliza primitivas de tipo poligonal como base para sus representaciones gráficas.**
- **Necesita representaciones poligonales en modo explicito.**
- **Tipos de Primitivas soportados:**

Valor de la Cte GL	Tipo de Primitiva Poligonal
GL_POINTS	Puntos aislados
GL_LINES	Líneas de dos vértices
GL_LINE_STRIP	Línea de cualquier numero de vértices
GL_LINE_LOOP	Línea Cerrada.
GL_POLYGON	Polígono de Cualquier tipo
GL_TRIANGLES	Polígonos de tres lados
GL_TRIANGLE_STRIP	Tira de Triangulos
GL_QUADS	Polígonos de cuatro vertices
GL_QUAD_STRIP	Tira de Cuadrilateros.
GL_TRIANGLE_FAN	Abanico de triangulos.

Modelos de Superficies Poligonales (7/7)
Modelo Poligonal sobre OpenGL.(2/2)

- **Funciones Básicas para trabajo con polígonos:**

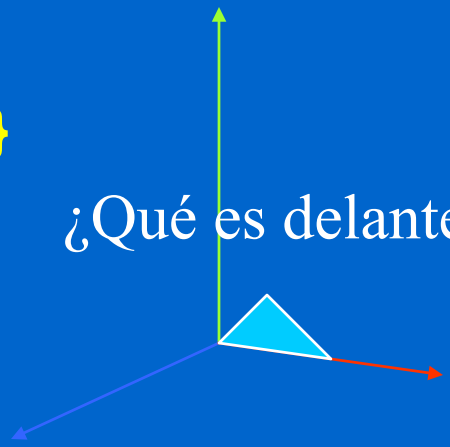
```
glBegin (uint tipo_de primitiva);
  lista ordenada de vertices de la primitiva
glEnd();
```

- **Para indicar los vertices:**
  - **glVertex→tiene diversas variantes:**

```
glVertex3f(float x, float y, float z)
glVertex3fv(float xyz[3]);
```

- **Triángulo de vértices {0,0,0},{2,0,0},{1,1,0}**

```
glBegin(GL_TRIANGLES);
  glVertex3f(0,0,0);
  glVertex3f(2,0,0);
  glVertex3f(1,1,0);
glEnd();
```



Modelo de Superficies Matemáticas

- Ecuaciones matemáticas de descripción de superficies:
  - Superficies Implícitas. F(x,y,z)=0
    - Cuádricas
    - Superficies equipotenciales.
  - Superficies Explicitas. Z=f(x,y);
  - Superficies Paramétricas.(x,y,z)=f(u,v)
    - Las Explicitas se tratan como paramétricas.
    - Bezier, BSplines, NURBS.
- Posibilidades de describir objetos a través de trozos de superficies adyacentes. Evaluar problemas de continuidad (C0,C1,C2)
- Más sencillas de modelar
- Objetos naturales y matemáticos.

$$\left.\begin{array}{l} x = u \\ y = v \end{array}\right\} z = f_z(u,v) \quad \Rightarrow$$

$$\Rightarrow (x,y,z) = (u,v,f_z(u,v)) = f(u,v)$$

Modelo de Superficies Matemáticas
Superficies Implícitas

- **Superficies Cuádricas. Se trata de primitivas matemáticas que responden a la ecuación:**

$$ax^2 + by^2 + cz^2 + 2dxy + 2eyz + 2fxz + 2gx + 2hy + 2jz + k = 0$$

- **Dependiendo de los valores de los coeficientes generamos: esferas, elipsoides, toros, etc.**

- **Son comunes en paquetes de modelado por la sencillez de uso. Se pueden extender para representación de sólidos de forma simple.**
- **El problema principal es lo limitado de las formas disponibles.**
- **Siguen sin servir para modelar elementos naturales.**
- **Al menos evitan el *aliasing*.**

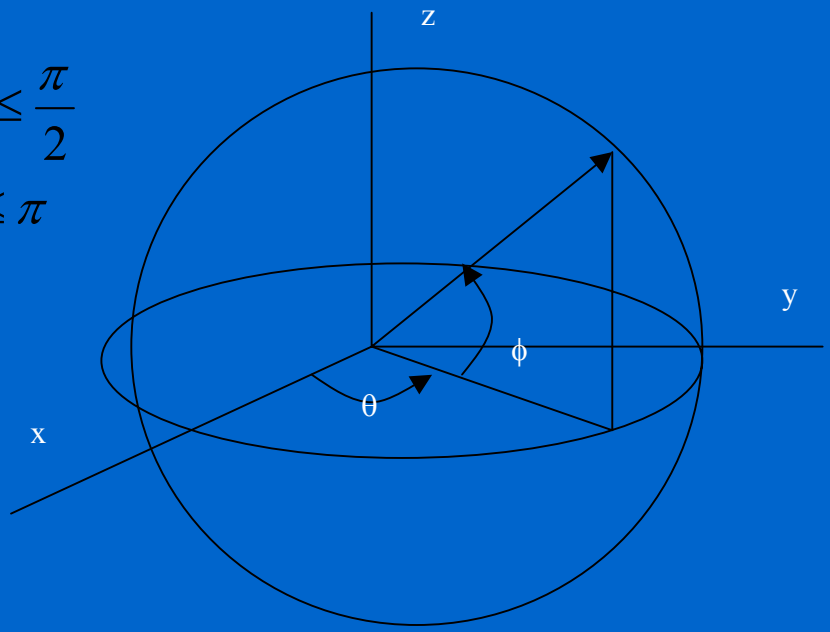
Modelo de Superficies Matemáticas
Superficies Implícitas

- **Esfera:**

$$ax^2 + by^2 + cz^2 - r^2 = 0$$

- **A veces se trabaja en representación polar**

$$x = r\cos\phi\cos\theta, -\frac{\pi}{2} \le \phi \le \frac{\pi}{2}$$
$$y = r\cos\phi\sin\theta, -\pi \le \theta \le \pi$$
$$z = r\sin\phi$$



Modelo de Superficies Matemáticas

Superficies Implícitas

- **Ejemplo función OpenGL para discretiza una esfera (t-strips en latitud)**

```
void Esfera_T_Strip(float radio, int nlatitud , int nlongitud)
{
 float inct,incf;
int i,j;
float vertice[3];
      inct=2*PI/nlatitud;
      incf=PI/nlatitud;
      for(i=0;i<nlatitud;i++)
      {
                  glBegin(GL_TRIANGLE_STRIP);

                  vertice[0]=vertice[1]=0;
                  vertice[2]=-radio;
                  glVertex3fv(vertice);
                  for(j=1;j<nlongitud-1;j++)
                  {
                              vertice[0]=radio*cos(i*inct)*cos(j*incf-0.5*PI);
                              vertice[1]=radio*sin(i*inct)*cos(j*incf-0.5*PI);
                              vertice[2]=radio*sin(j*incf-0.5*PI);
                              glVertex3fv(vertice);

                              vertice[0]=radio*cos((i+1)*inct)*cos(j*incf-0.5*PI);
                              vertice[1]=radio*sin((i+1)*inct)*cos(j*incf-0.5*PI);
                              glVertex3fv(vertice);

                  }
                  vertice[0]=vertice[1]=0;
                  vertice[2]=radio;
                  glVertex3fv(vertice);
      }
}
```

Modelo de Superficies Matemáticas
Superficies Implícitas.

- **Elipsoide**

$$\left(\frac{x}{r_x}\right)^2+\left(\frac{y}{r_y}\right)^2+\left(\frac{z}{r_z}\right)^2-1=0$$

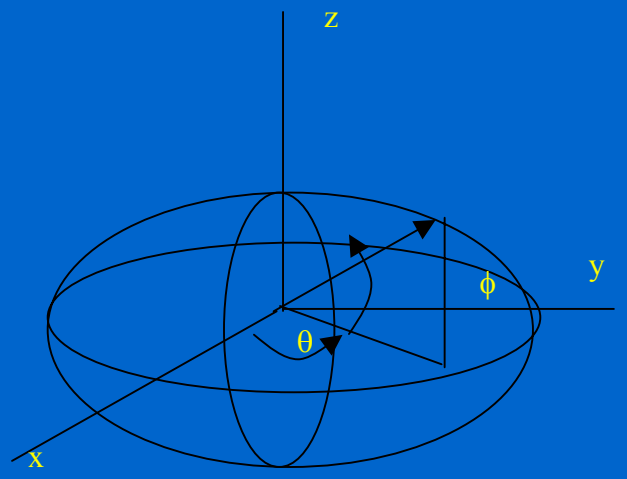
$$x = r_x \cos\phi\cos\theta, -\frac{\pi}{2}\le\phi\le\frac{\pi}{2}$$

$$y = r_y \sin\phi\sin\theta, -\pi\le\theta\le\pi$$

$$z = r_z \sin\phi$$

- **Supercuádricas**

$$\left[\left(\frac{x}{r_x}\right)^{2/s_2}+\left(\frac{y}{r_y}\right)^{2/s_2}\right]^{s_2/s_1}+\left(\frac{z}{r_z}\right)^{2/s_1}-1=0$$



Modelo de Superficies Matemáticas
Superficies Implícitas. Isosuperficies

- **Isosuperficies o Superficies equipotenciales.**
  - **Útiles para la representación de formas suaves.**
  - **Formas orgánicas y moleculares.**
  - **Se definen campos de valor escalar, o sea a cada punto del espacio le corresponde un valor (campos de temperatura, campos de potencial,etc.)**
  - **La superficie queda definida por el conjunto de puntos con un determinado valor del campo.(superficies equipotenciales).**

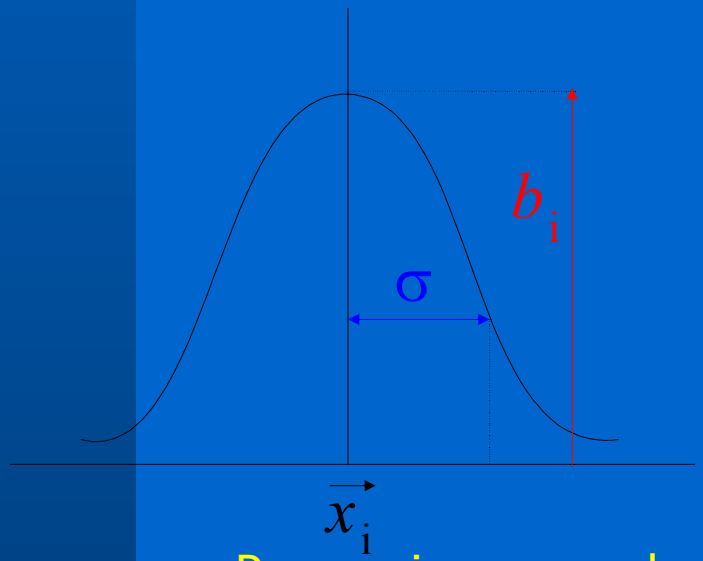
$$g(x,y,z)=T \quad (umbral\ o\ threshold)$$

$$f(x,y,z)=g(x,y,z)-T=0$$

- **Modelado basado en mover los puntos o elementos generadores del campo. Normalmente se trabaja con campos esféricos o elipsoidales.**
- **Resultado objetos globulares de formas abultadas (*blobs o blobbly objects*)**

Modelo de Superficies Matemáticas
Superficies Implícitas. Isosuperficies

- **Tipos de superficies, basadas en las funciones generadoras.**
  - **Funciones basadas en Campos Gaussianos. *Blobs* típicos.**



$$\vec{x}_i \rightarrow posicion$$
$$b_i \rightarrow fuerza = altura$$
$$r_i(\sigma) \rightarrow desviacion\ tipica$$

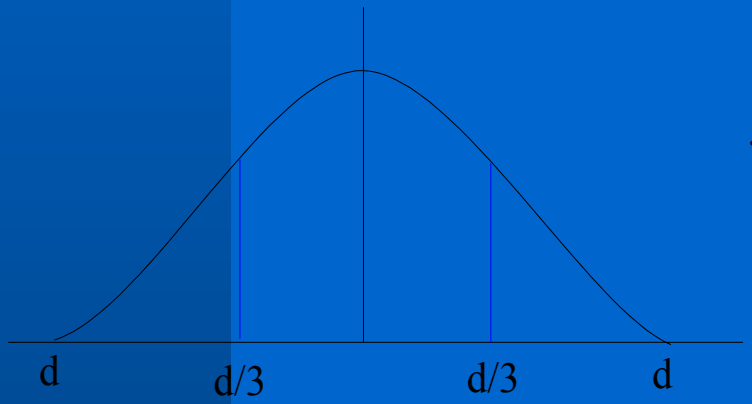
$$f_i(\vec{x}_i, b_i, r_i) = b_i e^{-\frac{1}{r_i^2}(\vec{x} - \vec{x}_i)^2}$$

Para varios generadores la superficie queda definida como

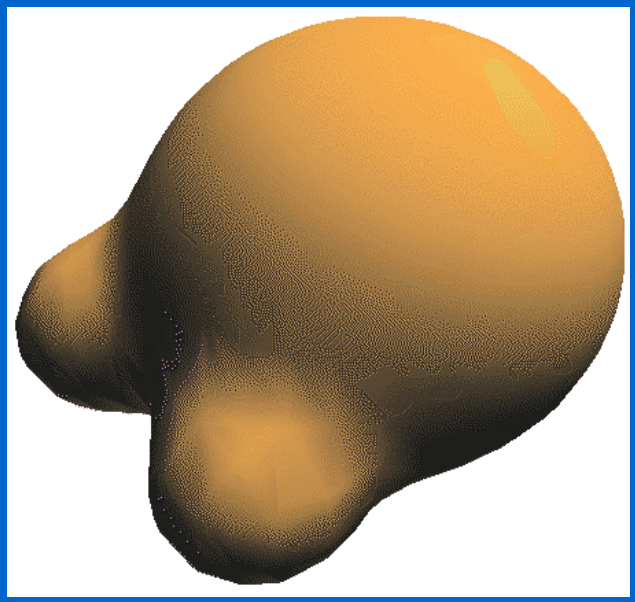
$$f(x,y,z) = \sum_i b_i e^{-\frac{1}{r_i^2}(\vec{x} - \vec{x}_i)^2} = T$$

Modelo de Superficies Matemáticas
Superficies Implícitas. Isosuperficies

- **Tipos de superficies, basadas en las funciones generadoras.**
  - **Funciones basadas en Poligonales a trozos. *Metaballs***



$$f(r)=\begin{cases} b\left(1-3r^2/d^2\right) & si\ r\in[0,\mathrm{d}/3] \\ \frac{3}{2}b(1-r/d)^2 & si\ r\in[\mathrm{d}/3,\mathrm{d}] \\ 0 & si\ r>\mathrm{d} \end{cases}$$



  - *Soft Objects*.

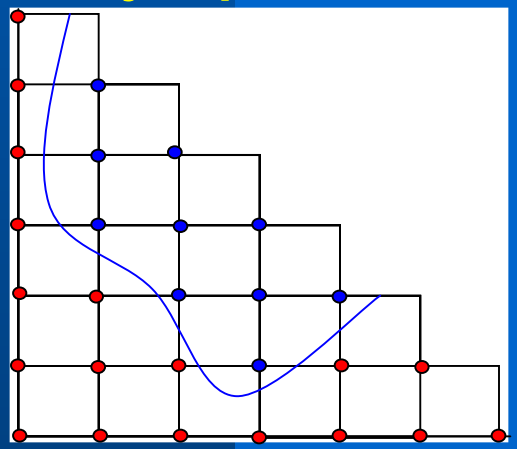
$$f(r)=\begin{cases} 1-\frac{22r^2}{9\mathrm{d}^2}+\frac{17r^4}{9\mathrm{d}^4}-\frac{4r^6}{9\mathrm{d}^6} & si\ r\in[0,\mathrm{d}] \\ 0 & si\ r>\mathrm{d} \end{cases}$$

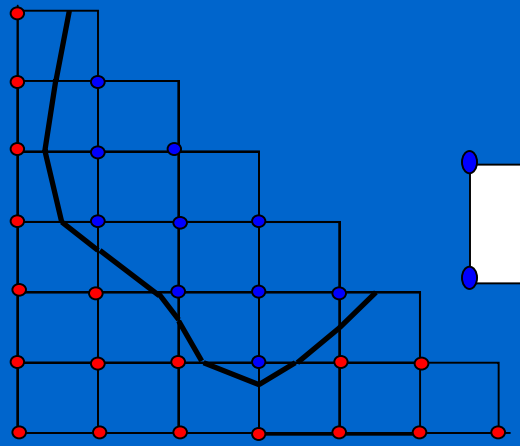
Modelo de Superficies Matemáticas
Superficies Implícitas. Isosuperficies

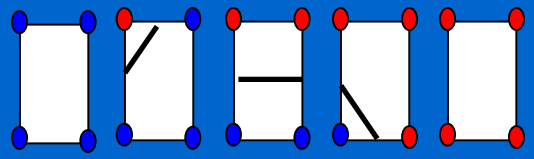
- **¿Cómo Evaluar las isosuperfices?**
  - **Métodos de trazado de rayos, revisaremos en el futuro.**
  - **Métodos aproximados. Durante la Fase diseño de la forma y para sistemas poligonales.**
- **Evaluación en fase de diseño.**
  - **Situamos esferas o elipsoides del tamaño correspondiente al campo del valor umbral.**
  - **La superficie se obtiene a partir de una discretización del espacio.**
- **Evaluación para modelos poligonales.**
  - **Método de *Marching Cubes*.**

Modelo de Superficies Matemáticas
Superficies Implícitas. Isosuperficies

- **Método de *Marching cubes***
  - **Discritización del espacio en Voxels.**
  - **Estudio del la situación del voxel con respecto a la superficie.**
    - **Completamente dentro.**
    - **Completamente fuera.**
    - **Intersectado. Caso a evaluar.**
  - **Ejemplo bidimensional:**

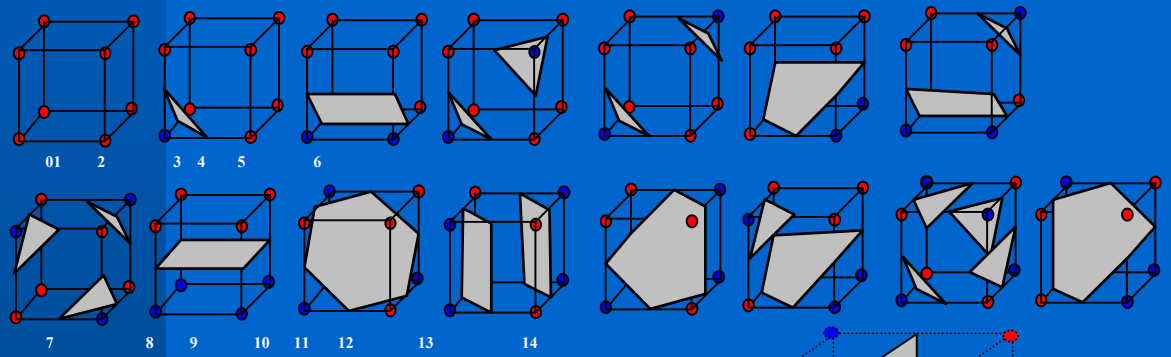


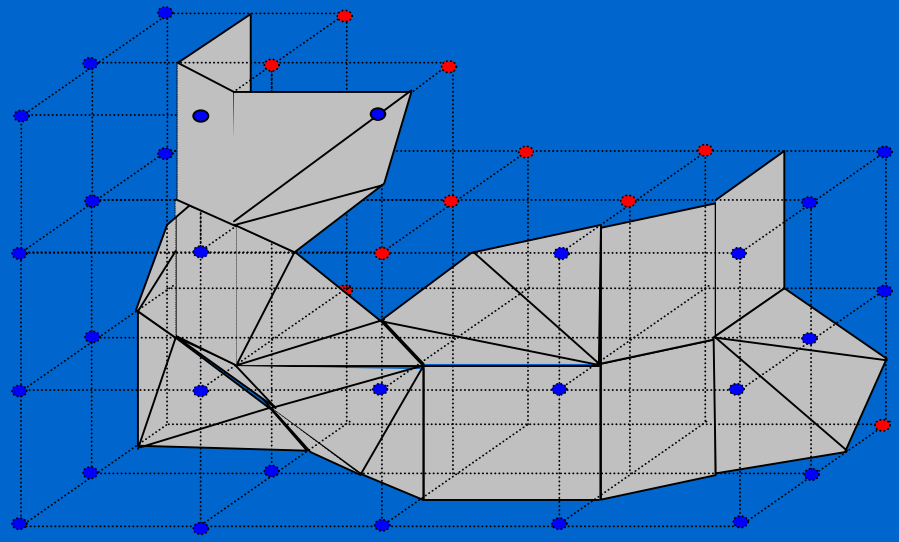




Modelo de Superficies Matemáticas
Superficies Implícitas. Isosuperficies

- **Método de *Marching cubes***
  - **El caso tridimensional es bastante más complejo**

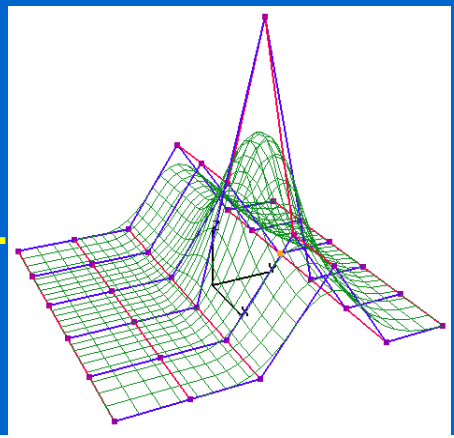


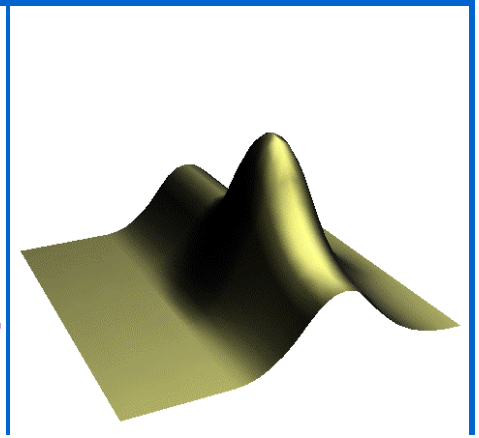


Modelo de Superficies Matemáticas

Superficies Paramétricas.

- **Problemas Superficies Implicitas.**
  - **Número de Formas limitado.**
  - **Problemas para conseguir representación poligonal.**
- **Que buscamos con la Sup. Paramétricas:**
  - **Facilidad de modelado de formas libres.**
  - **Control local de deformaciones.**
  - **Buenas prestaciones en transformaciones a modelos poligonales.**
  - **Posibilidades de calculo de tangencias y curvatura.**
  - **Buenas prestaciones en cuanto a almacenamiento.**
- **Dos aproximaciones Principales:**
  - **Parto de un conjunto de puntos y quiero que la superficie pase por ellos. Superficies de Interpolación.**
  - **Tengo un conjunto de puntos que controlan la forma de la superficie, que no pasa necesariamente por ellos. Superficies de aproximación.**





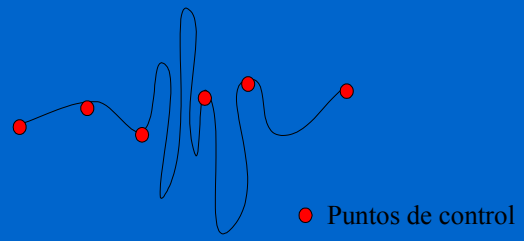
Modelo de Superficies Matemáticas

Superficies Paramétricas.

- Vamos a hablar de curvas, las superficies simplemente consisten en utilizar dos parámetros.
- La aproximación Básica se basa en el uso de funciones polinómicas.

$$C(u) = \sum_{i=0}^{n} a_i u^i$$

- Por tanto la curva depende del grado del polinomio, si disponemos de n puntos para definir la curva necesitamos un polinomio de grado n. Esto es un problema.

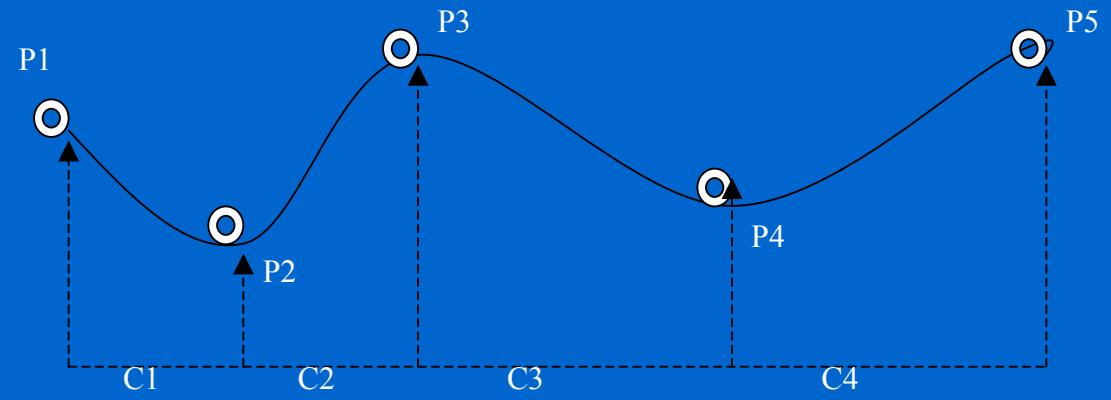


- Otra posibilidad consiste en definir la curva como combinación de trozos de curvas. En este caso tendremos que tener cuidado con los grados de continuidad. Este tipo de curvas se suelen denominar *splines*. Nombre que viene de unos dispositivos utilizados en dibujo técnico.

Modelo de Superficies Matemáticas
Superficies Paramétricas. Interpolación

SPLINES NATURALES

- **Normalmente se basan en polinomios de grado cúbico. Donde se exige continuidad C^2**



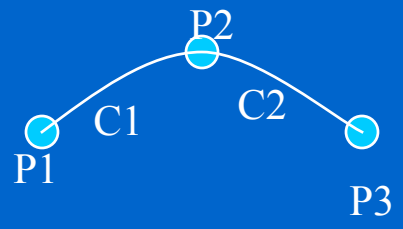
- **Por tanto la curva consiste en n-1 trozos de grado 3, donde n es el numero de puntos por los que pasa la curva. El parámetro u se evalúa en el intervalo [0,1]**

$$C_j(u) = \sum_{i=0}^{3} a_{ji} u^i$$

Modelo de Superficies Matemáticas

SPLINES NATURALES

- El problema para calcular estas curvas es determinar los valores de los coeficientes, para ello nos basamos en los puntos de paso y la condición de continuidad C^2 y desarrollamos un sistema de ecuaciones, donde la incógnita son los coeficientes. El numero de incógnitas para n puntos de paso es de 4(n-1) coeficientes y por tanto las mismas ecuaciones.
- Por ejemplo para una curva de tres puntos, tenemos dos curvas y 8 coeficientes. Veamos como planteamos las ecuaciones:



1. $C_1(0) = P1$ *Ecuaciones de punto de paso 2*
   $C_1(1)=P2$
   $C_2(0)=P2$
   $C_2(1)=P3$
2. $C_1'(1)=C_2'(0)$ *Ecuaciones de primera y segunda derivada*
   $C_1''(1)=C_2''(0)$

Las dos ecuaciones que nos faltan las sacamos como hemos comentado anteriormente de asignar un valor de tangencia en los extremos o sea un P1' y un P3'

3. $C_1'(0)=P1'$
   $C_2'(1)=P3'$

La forma final de las ecuaciones segun la forma de las curvas vista anteriormente sería:

1. $a_{10} = P1$
2. $a_{10}+a_{11}+a_{12}+a_{13}=P2$
3. $a_{20}=P2$
4. $a_{20}+a_{21}+a_{22}+a_{23}=P3$
5. $a_{11}+2a_{12}+3a_{13}=a_{21}$
6. $2a_{12}+6a_{13} = 2\,a_{22}$
7. $a_{11}=P1'$
8. $a_{21}+2a_{22}+ 3a_{23} =P3'$

$$C_j(u) = \sum_{i=0}^{3} a_{ji} u^i$$

Modelo de Superficies Matemáticas

SPLINES NATURALES

- Pasemos a ver ahora un ejemplo práctico de trabajo con estas curvas una vez determinados los valores de los coeficiente.
- Para optimizar el cálculo y evitar uso de potencias, se utiliza el método de HORNER. Utiliza la factorización. Con sólo 3 productos y 4 sumas se resuelve todo. El método normal llevaría 8 productos y cuatro sumas.

$$C_j(u) = \sum_{i=0}^{3} a_{ji} u^i \Rightarrow C_j(u) = a_{j0} + a_{j1}u + a_{j2}u^2 + a_{j3}u^3$$

$$Horner: C_j(u) = a_{j0} + u(a_{j1} + u(a_{j2} + a_{j3}u))$$

```
void Evalua_Spline_Natural(float nptos, float *param[3][4], float ndiv)
{
int i,j;
Float aux[3],u;
                glBegin(GL_LINE_STRIP);
                for(i=0;i<nptos-1;i++)
                          for(j=0;j<=ndiv;j++)
                          {
                                    u=j/ndiv;
                                    aux[0]=param[i][0][0]+u*(param[i][0][1]+
                                              u*(param[i][0][2]+u* param[i][0][3]));
                                    aux[1]=param[i][1][0]+u*(param[i][1][1]+
                                              u*(param[i][1][2]+u* param[i][1][3]));
                                    aux[2]=param[i][2][0]+u*(param[i][2][1]+
                                              u*(param[i][2][2]+u* param[i][2][3]));
                          }
                glEnd();
}
```

Modelo de Superficies Matemáticas
Superficies Paramétricas. Aproximación

- Las curvas de Bézier solamente pasan por los puntos extremos.
- La curvas de Bézier cumplen la propiedad de cierre convexo del polígono de control.
- Las curvas de bézier se definen a partir de la combinación de una familia especial de polinomios: los polinomios de Berstein.
- Dos aproximaciones principales:
  - Curvas de Bezier basadas en polinomios de grado N.
  - Curvas de Bezier definidas a trozos.

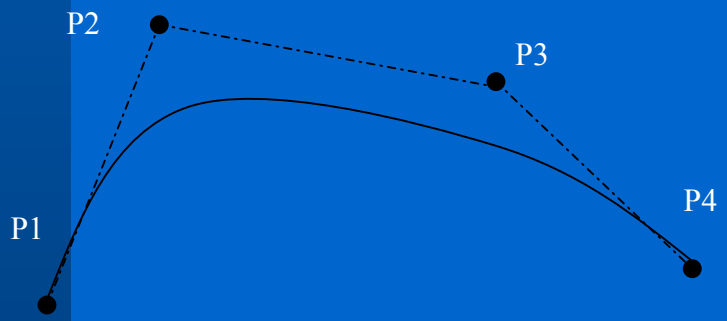
Modelo de Superficies Matemáticas
Superficies Paramétricas. Aproximación

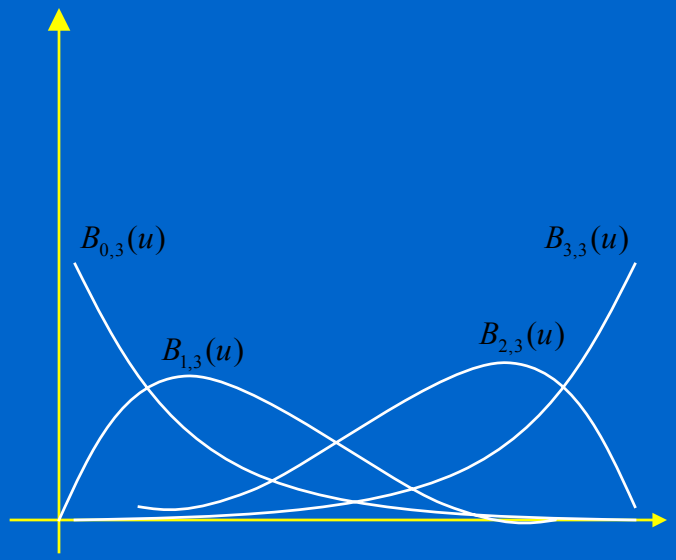
- **Forma de las Curvas de Bezier generales:**

$$C(u) = \sum_{i=0}^{n} B_{i,n}(u) P_i \qquad 0 \le u \le 1$$

con

$$B_{i,n}(u) = \frac{n!}{i!(n-1)!} u^i (1-u)^{n-i} \; con \quad \sum_{i=0}^{n} B_{i,n}(u) = 1 \forall u$$





Modelo de Superficies Matemáticas
Superficies Paramétricas. Aproximación

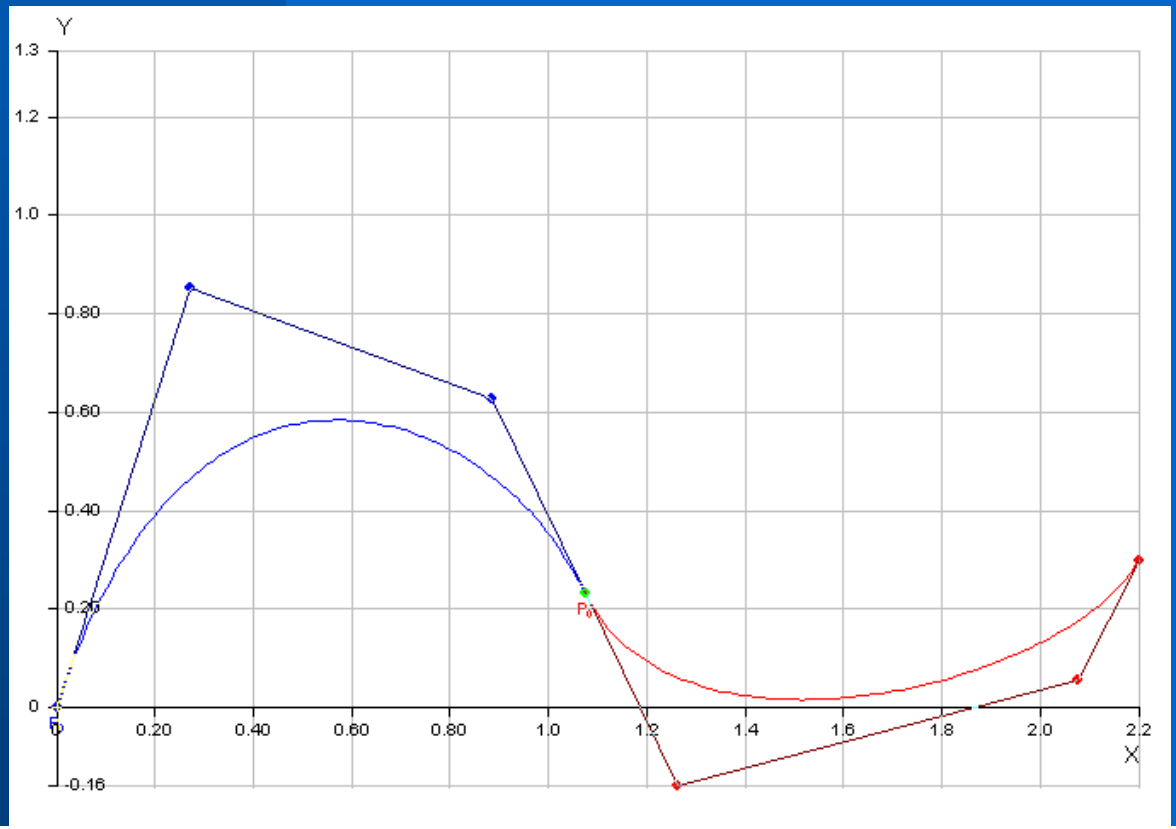
- **Combinación de Curvas de Bezier:**

$$\alpha:[0,1] \Rightarrow \Re^3 \;\|\; \alpha(0) = P_0 \;;\; \alpha(1) = P_n$$

$$\beta:[0,1] \Rightarrow \Re^3 \;\|\; \beta(0) = Q_0 \;;\; \beta(1) = Q_n$$

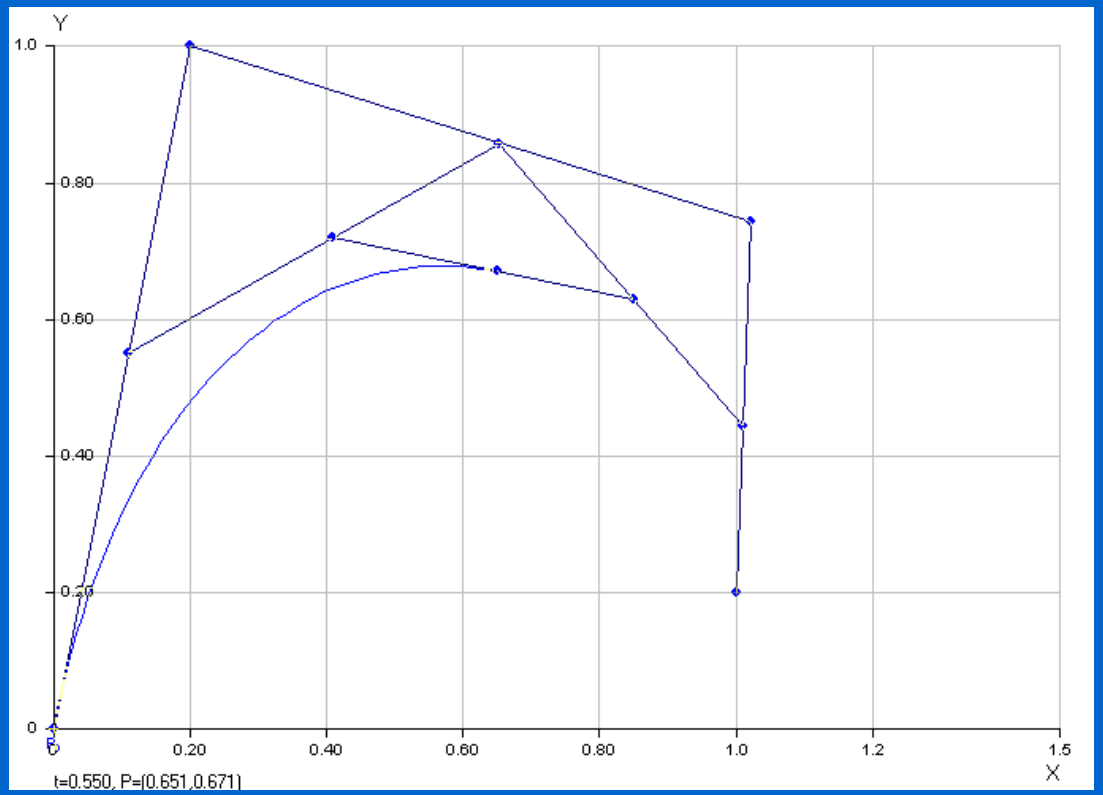
$$\alpha \circ \beta:[0,2] \Rightarrow \Re^3$$

$$t \Rightarrow \begin{cases} \alpha(t) & 0 \le t \le 1 \\ \beta(t-1) & 1 \le t \le 2 \end{cases}$$



Modelo de Superficies Matemáticas
Superficies Paramétricas. Aproximación

- **Método Geométrico de cálculo: Casteljau**
  - **Basado en interpolación lineal recursiva.**



- **Superficies de Bezier.**

$$S_{bezier}(u,v) = \sum_{i=0}^{n}\sum_{j=0}^{m} B_{i,n}(u)B_{j,m}(v)P_{ij} \qquad 0 \le u \le 1, 0 \le v \le 1$$

Modelo de Superficies Matemáticas
Superficies Paramétricas. Aproximación

- **Curvas y Superficies de Bezier en OpenGL**

```
void dibujar_curvas_bezier()
{
int i,j;

      /** Definicion del Mapeo */
      glMap1f(GL_MAP1_VERTEX_3,0.0,1.0,3,n_pc_c_bezier,&pc_c_bezier[i]);
      /*Habilitacion del mapeo a vertices 3D */
      glEnable(GL_MAP1_VERTEX_3);
/* Especificación del numero de evaluaciones del mapeo (100 en los valores 0 y 1 del parámetro*/
      glMapGrid1d(100,0,1.0);
/*Dibujado de los puntos de la curva de bezier */
      glEvalMesh1(GL_LINE,0,100);
/*Tambien se podria hacer asi
      glBegin(GL_LINE_STRIP);
      for(j=0;j<100;j++)
                  glEvalCoord1f(j/100.0);
      glEnd(); */
/* Dibujado de la polilinea de control*/
                  glBegin(GL_LINE_STRIP);
                  for(j=0;j<n_pc_c_bezier;j++)
                              glVertex3fv(pc_c_bezier[j]);
                  glEnd();
/* Dibujado de los puntos de control */
                  glPointSize(5.0);
                  glBegin(GL_POINTS);
                  for(j=0;j<n_pc_c_bezier;j++)
                              glVertex3fv(pc_c_bezier[j]);
                  glEnd();

}
```

Modelo de Superficies Matemáticas

Superficies Paramétricas. Aproximación

- **Curvas y Superficies de Bezier en OpenGL**

```
void dibuja_s_bezier()
{
int i,j,k;
        glDisable(GL_LIGHTING);
        glMap2f(GL_MAP2_VERTEX_3,0,1,3,n_pc_s_bezier[0],0,1,30,n_pc_s_bezier[1],&pc_s_bezier[i]);
        glEnable(GL_MAP2_VERTEX_3);
        /** Evaluacion de la Mesh con 25 divisiones en cada parametro de 0 a 25 */
        glMapGrid2f(25,0,1,25,0,1);
/*Dibujado de la superficie de bezier */
        glEvalMesh2(GL_LINE,0,25,0,25);
 /*Dibujado del poligono de control y de los puntos de control*/
        glLineWidth(2.0);
        glColor3f(0,1,0);
        for(j=0;j<n_pc_s_bezier[0];j++)
        {
                        glBegin(GL_LINE_STRIP);
                        for(k=0;k<n_pc_s_bezier[1];k++)
                                        glVertex3fv(pc_s_bezier[j][k]);
                        glEnd();
        }
          glColor3f(0,0,1);
        for(j=0;j<n_pc_s_bezier[1];j++)
        {
                        glBegin(GL_LINE_STRIP);
                        for(k=0;k<n_pc_s_bezier[i][0];k++)
                                        glVertex3fv(pc_s_bezier[k][j]);
                        glEnd();
        }
        glPointSize(5.0);
        glColor3f(1,1,0);
        glBegin(GL_POINTS);
        for(j=0;j<n_pc_s_bezier[0];j++)
                        for(k=0;k<n_pc_s_bezier[1];k++)
                                        glVertex3fv(pc_s_bezier[j][k]);
        glEnd();
        }
}
```

Modelo de Superficies Matemáticas
Superficies Paramétricas. Aproximación

Curvas B-Splines

- Las Bezier tienen un problema grave, el numero de puntos de control dictan el grado de los polinomios de Berstien.
- Otro problema adicional es que todos los polinomios de Berstein están activos para todo el rango del parámetro. Conlleva pérdidas de Control local.
- ¿Qué necesitamos para mejorar estos aspectos?
  - Buscar una funciones base que permitan generar curvas que pasen por un número cualquiera de puntos y manteniendo un grado determinado.
  - Poder controlar el rango del parámetro sobre el cual actúa cada uno de los polinomios base que combinemos.
- La respuesta a la pregunta anterior son las B-Splines.

Modelo de Superficies Matemáticas
Superficies Paramétricas. Aproximación

Curvas B-Splines

- Idea de partida similar a las splines naturales pero sin interpolación.
- Se une el espacio paramétrico de los diferentes trozos. Esto se realiza definiendo un vector de *knots.*
- Se puede seleccionar el grado de los polinomios base para controlar la suavidad de la curva: el grado es d, mientras que la base. Polinomios de Coox-deBoor

$$C(u)=\sum_{i=0}^{n} N_{i,d}(u)P_i \qquad u_{\min}<u<u_{\max}, 2\le d\le n+1$$

$$con$$

$$N_{i,1}(u)=\begin{cases} 1 si u_i\le u\le u_{i+1} \\ 0 en otro caso \end{cases} \qquad N_{i,d}(u)=\frac{u-u_i}{u_{i+d-1}-u_i}N_{i,d-1}(u)+\frac{u_{i+d}-u}{u_{i+d}-u_{i+1}}N_{i+1,d-1}(u)$$

- Ejemplo vector knots {0,1,2,3,4,5,6}, el numero de *knots*, n+d
- Dependiendo de la distribución del vector de knots tenemos distintas familias de BSplines

Modelo de Superficies Matemáticas
Superficies Paramétricas. Aproximación

Curvas B-Splines

- B-Splines con knots multiples. Disminuye el control local pero aumenta la aproximación de la curva a esos puntos de control.
- De hecho si el numero de repeticiones en los extremos es igual al grado de los polinomios base conseguimos que la curva pase por los extremos.
  - {0,0,0,0,2,2,2,2} Si tenemos cuatro puntos y queremos aproximarlo con una Bspline de grado 3 que pase por los extremos este puede ser un ejemplo de valores del vector de knots

Superficies B-Splines.

$$S_{BSpline}(u,v) = \sum_{i=0}^{n}\sum_{j=0}^{m} N_{j,l}(v)N_{i,d}(u)P_{ij}$$

Modelo de Superficies Matemáticas
Superficies Paramétricas. Aproximación

B-Splines Racionales.

- **Se definen a partir de la razón o promedio de los polinomios base. Esto permite asignar un peso wi a cada punto de control. Mejorando notablemente el grado de control local.**
- **Con este se pueden representar cuádricas y otras superficies de manera exacta con estas estructuras.**

$$C(u) = \frac{\sum_{i=0}^{n} w_i N_{i,d}(u) P_i}{\sum_{i=0}^{n} w_i N_{i,d}(u)_i}$$

- **Cuando podemos tener cualquier distribución del vector de knots se denominan NUBS-Non Uniform Rational B-Splines.**

Modelo de Superficies Matemáticas
Superficies Paramétricas. NURBS OpenGL

```
void dibujanurbs()
{
int i,j;
float ctrlptos[4][4][3],knots[8]={0,0,0,0,1,1,1,1};
GLUnurbsObj *pnurb=NULL; // Declaración de la variable de objeto NURB
/* Rellenado de unos puntos de control simples*/
      for(i=0;i<4;i++)
          for(j=0;j<4;j++)
          {         ctrlptos[i][j][0]=i;ctrlptos[i][j][1]=j;ctrlptos[i][j][2]=0;}
          for(i=0;i<4;i++)ctrlptos[2][i][2]=1;
/* Creacion del Objeto NURB */
      pnurb=gluNewNurbsRenderer();
      glEnable(GL_AUTO_NORMAL);
      //glShadeModel(GL_SMOOTH);
/* Cambio de las propiedaddes del objeto, en este caso la toleracia de error se pone en 20 pixel,
      esto implica mas o menos poligonización en la visualización*/
      gluNurbsProperty(pnurb,GLU_SAMPLING_TOLERANCE, 20.0);
/* Establecemos que quermos ver la NURB en modo alambre*/
      gluNurbsProperty(pnurb, GLU_DISPLAY_MODE,GLU_OUTLINE_POLYGON);
/* Iniciamos el dibujado de la superficie NURB*/
     gluBeginSurface(pnurb);
 /*Damos los valores de descripcion de la NURB, en este caso 8 knots y grado de los polinomios4*/
                    gluNurbsSurface(pnurb,8,knots,8,knots,4*3,3,&ctrlptos,4,4,GL_MAP2_VERTEX_3);
      gluEndSurface(pnurb);
 /* Eliminamos el objeto nurb que creamos*/
      gluDeleteNurbsRenderer(pnurb);
}
```

Representación de Modelos de Sólidos

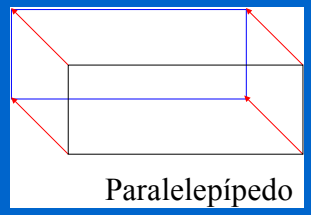
- Necesitamos Información de las Propiedades Interiores.
- Determinar cortes con otros objetos
- Los sólidos definen regiones del espacio completamente limitadas por superficies.
- Definición de Sólido regular
  - No intersecciones consigo mismo.
  - No se permite el sólido nulo.
  - Permitir combinaciones boolenas.
  - Sigan siendo sólidos ante transformaciones afines

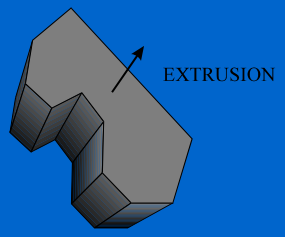
Representación de Modelos de Sólidos

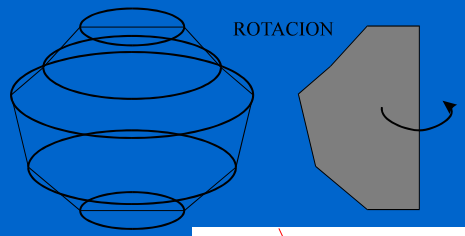
- **Tipos de representaciones de Sólidos.**
  - **Modelos de Barrido**
  - **Modelado CSG.**
  - **Partición Espacial:**
    - **No Jerarquica: Celdas, Voxels.**
    - **Jerárquica**
      - **Ortogonal: Octrees.**
      - **No Ortogonal: BSP-Trees**

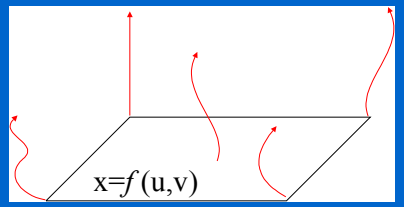
Representación de Modelos de Sólidos

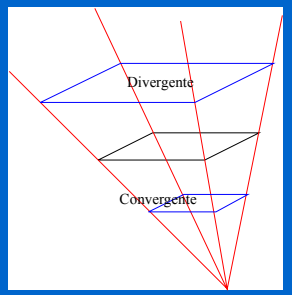
- **Primitivas Sólidas**
  - **Basadas en inecuaciones:**
    - **Cubo.** $X \geq -1, X \leq 1; Y \geq -1, Y \leq 1; Z \geq -1, Z \leq 1$
    - **Esfera.** $X^2 + Y^2 + Z^2 \leq 2$
- **Modelos de Barrido**
  - **Barrido Traslacional.**
  - **Barrido Rotacional.**
  - **Barrido Circular.**
  - **Barrido General**
    - **Extrusión Curva.**
    - **Extrusión Divergente.**





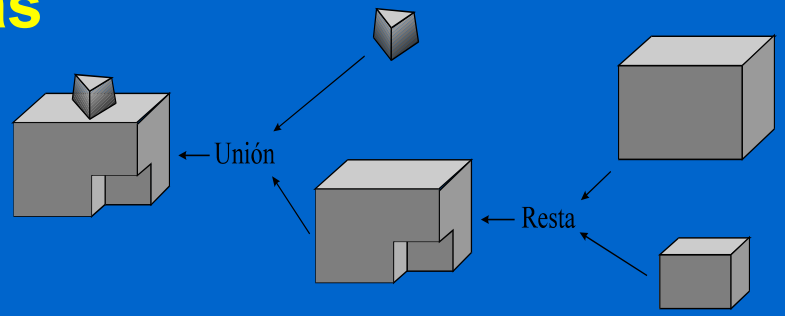






Representación de Modelos de Sólidos

- **Modelado Basado en Geometría sólida Constructiva (CSG)**
  - **Elementos Base. Primitivas Sólidas.**
  - **Operaciones booleanas**
    - **Unión**
    - **Intersección**
    - **Resta o diferencia**
    - **Negación**



  - **El objeto se define a partir de un árbol CSG, que suele recorrerse en profundidad:**
    - **Nodos Primitiva Sólida**
    - **Nodos Transformación.**
    - **Nodos Operación.**

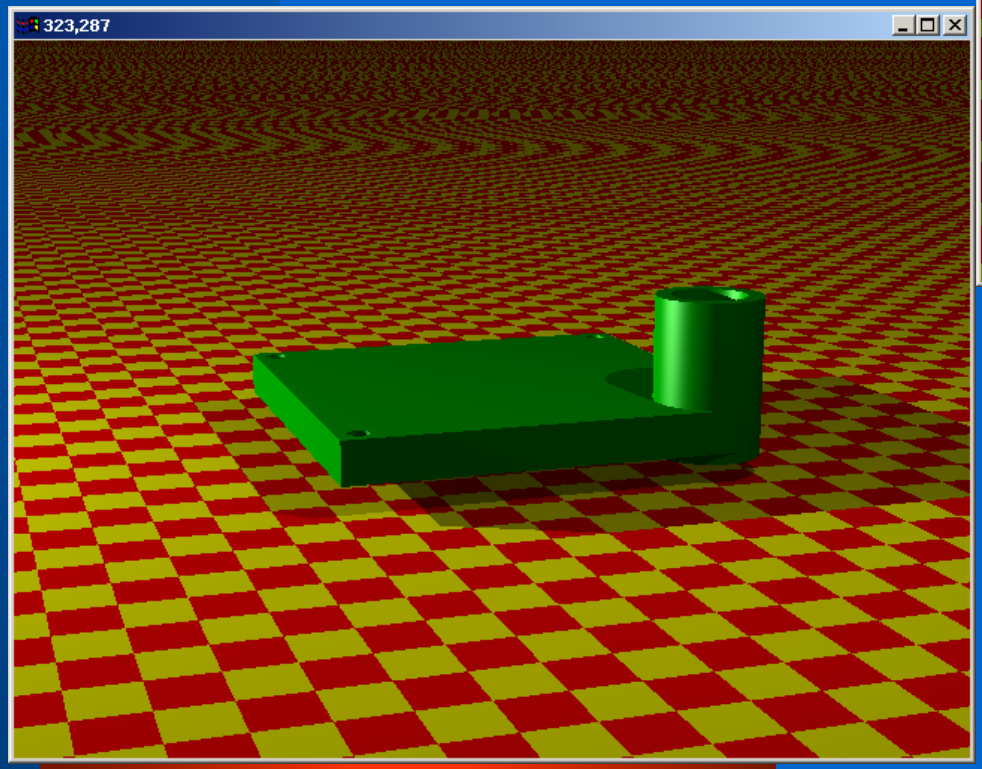
Representación de Modelos de Sólidos

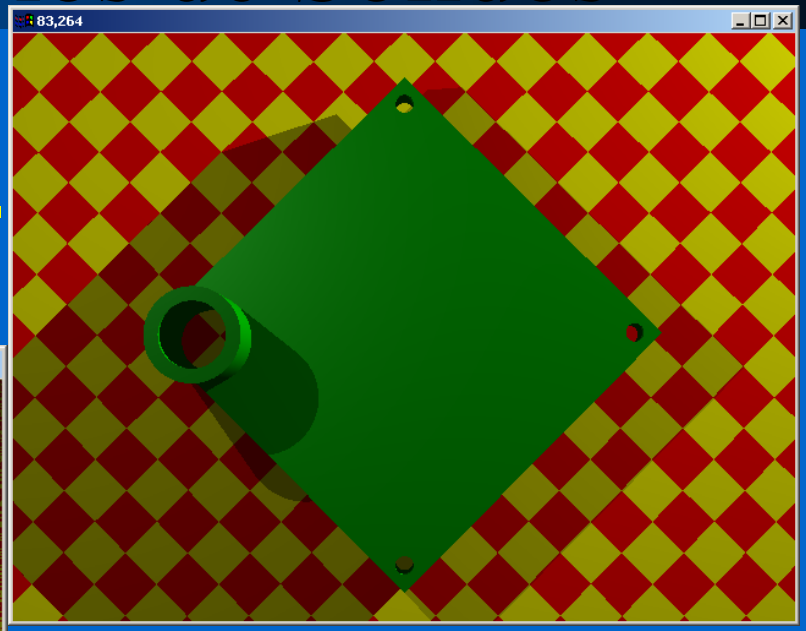
- **Algoritmo de deteminación de pertenencia a CSG:**

```
esta_dentro (nodo, punto)
      {
      switch (nodo_tipo) {
      caso UNION :      esta_dentro ← esta_dentro (nodo.hijo1) OR esta_dentro(nodo.hijo2)
      caso INTERSEC :esta_dentro ← esta_dentro(nodo.hijo1) AND esta_dentro(nodo.hijo2)
      caso RESTA:    esta_dentro ← esta_dentro (nodo.hijo1) AND NO esta_dentro(nodo.hijo2)
      .
      .
      .
      caso CUBO :       esta_dentro ←  comprobar_cubo(punto)
      .
      .
      .
      }
      }
```

Representación de Modelos de Sólidos

- **Ejemplo objeto CSG.**





Representación de Modelos de Sólidos

Modelos de Partición Espacial

- **Definición:Conjunto de volúmenes { Vi } tales que:**

$$\forall_i \quad V_i \neq \varnothing$$

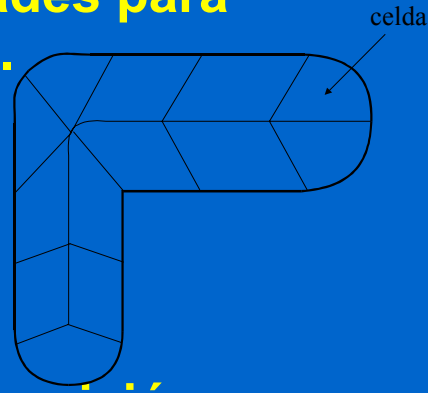
$$\forall_{i,j} \quad V_i \cap V_j = \varnothing \quad \bigcup_i \{V_i\} = \text{Espacio } (R^3)$$

- **Utilidades:**
  - **Clasificación de puntos.**
  - **Información particular de cada zona del espacio**
  - **Clasificación de objetos en una escena. Optimización de la visualización.**

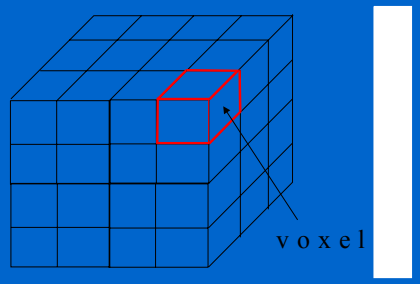
Representación de Modelos de Sólidos

Modelos de Partición Espacial

- **Particiónes no Jerárquicas:**
  - **Descomposición en celdas. Se utilizan cuando tenemos ese tipo de estructura o queremos añadir propiedades para análisis mecánicos de elementos finitos (FEA).**



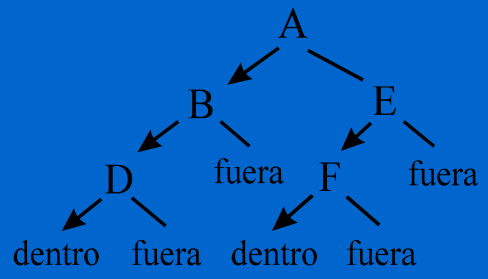
  - **Enumeración de ocupación espacial: descomposición en voxels, indicamos los que pertenecen al objeto y cuales no. Asignación de propiedades. Imagen medica.**

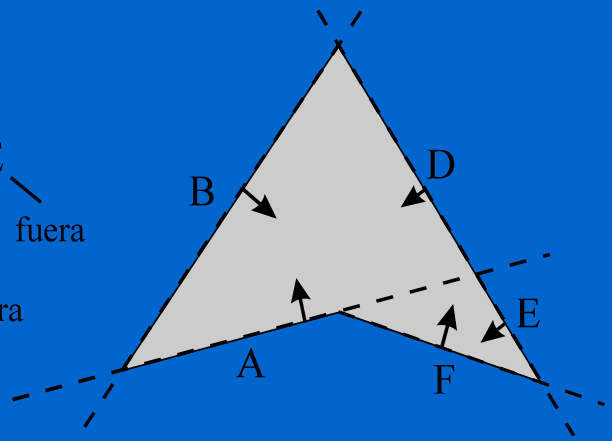


Representación de Modelos de Sólidos

Modelos de Partición Espacial

- **Particiones Jerárquicas no Ortogonales:**
  - **Arboles Binarios (BSP-tree): división del espacio en semiespacios utilizando árboles de orientación arbitraria.**



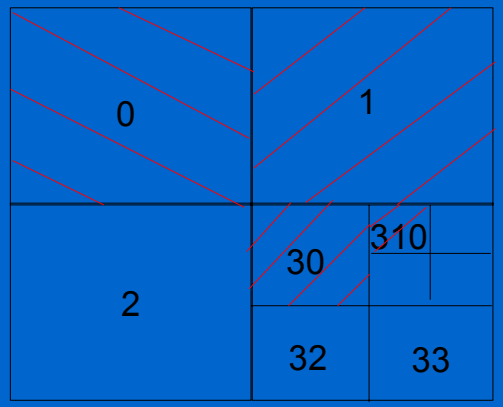



```
esta_dentro (nodo, punto)
{
si ( nodo = SI )              esta_dentro ← VERDAD /* Nodo terminal "dentro" = SI */
si ( nodo = NO )              esta_dentro ← FALSO   /* Nodo terminal "fuera" = NO */
si ( comprobar_punto_plano ( punto, nodo.plano) = DENTRO )
     esta_dentro ← esta_dentro (nodo.nodo_dentro, punto)  /* Pasar a hijo izquierdo */
     sino
      esta_dentro ← esta_dentro (nodo.nodo_fuera, punto)   /* Pasar a hijo derecho */
}
```

Representación de Modelos de Sólidos

Modelos de Partición Espacial

- **Particiones Jerárquicas Ortogonales:**
  - **Normalmente se realizan utilizando planos en la dirección de los ejes principales. Los espacios producidos pueden estar:**
    - **Completamente llenos**
    - **Completamente vacíos.**
    - **Ocupación Parcial**
  - **Ejemplo sobre elementos 2D. Árboles Cuaternarios o Quadtree:**



Objeto = 0+1+30+310

Representación de Modelos de Sólidos

Particiones Jeráquicas 3D, árboles Octarios u Octrees.

```
poner_nodo (nodo_padre, real lado, real lado_minimo){
        si lado > lado_minimo{
                para cada octante o cuadrante
si (octante ∈ objeto){
hijo ← crear_nodo (SI)
conectar_nodo (padre, hijo)
}
sino  si ( octante ∉ objeto){
hijo ← crear_nodo (NO)
conectar_nodo (padre, hijo)
}
sino{
hijo ← crear_nodo (INDETERMINADO)
conectar_nodo (padre, hijo)
poner_nodo ( hijo, lado/2, lado_minimo)
}
}
}
```