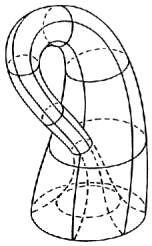
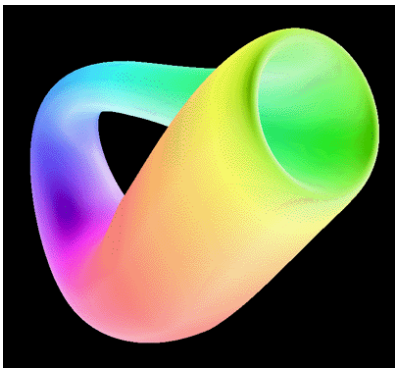
MODELADO DE OBJETOS

"El modelado geométrico se basa en la no diferenciación del espacio físico del espacio geométrico euclídeo pero la geometría euclidiana no siempre se corresponde con el espacio físico que nos rodea"



Superficie de Klein



1

MODELADO DE OBJETOS

1. Introducción

2. Modelos de fronteras

3. Transformaciones geométricas

4. Generación de modelos de fronteras

5. Modelos jerárquicos

Anexos.

I. Modelado de objetos naturales

II. Rotoscoping 3D

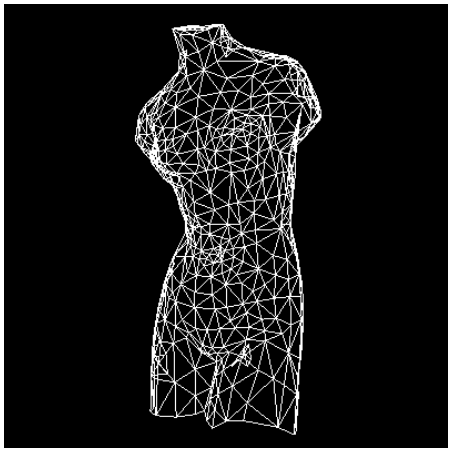
2

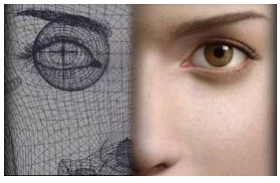
1. Introducción

- El modelado geométrico consiste en la construcción de un modelo matemático de la forma de un objeto físico tanto bidimensional como tridimensional
- Un modelo geométrico computacional será aproximado ya que no se puede representar en un ordenador toda la complejidad de un objeto real.
- Modelos geométricos computacionales de:
  - Curvas
  - Superficies
  - Sólidos

1. Introducción

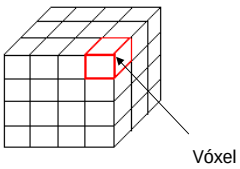
- Un sólido modelado por las fronteras (B-rep). Se representa por un conjunto de caras planas (polígonos) que delimitan su superficie.

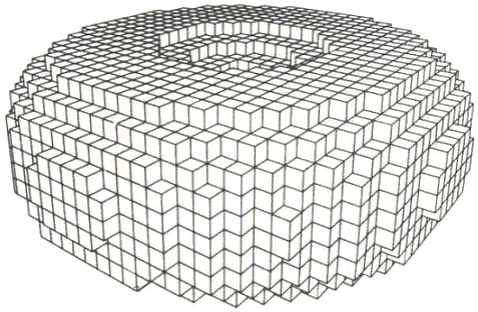




1. Introducción

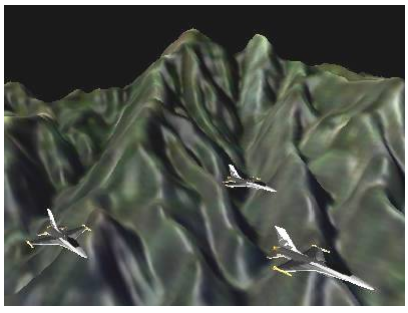
- Un sólido modelado por enumeración espacial se representa por un conjunto de volúmenes (vóxeles) que permiten modelar su interior.
  - La representación de un sólido con vóxeles es mediante una matriz tridimensional





1. Introducción

- Enumeración espacial

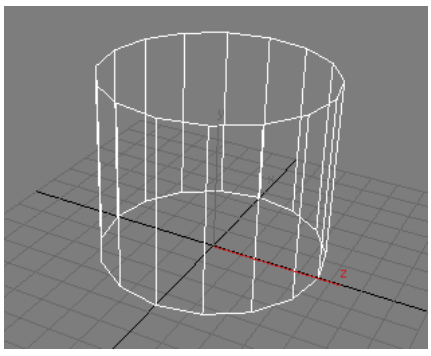


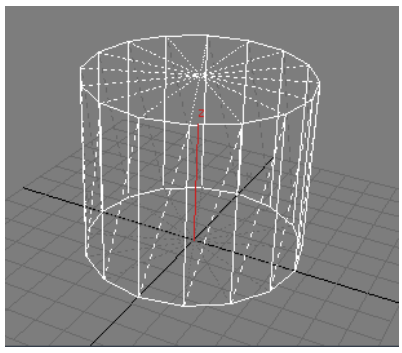
1. Introducción

- Enumeración espacial: escultura digital (digital sculpting, volume sculpting, voxel sculpting)
  - Zbrush (http://www.pixologic.com/zbrush/)
  - Autodesk Mudbox
  - 3D-Coat (https://3dcoat.com)

2. Modelos de fronteras (B-rep)

- Se representan usando mallas de polígonos (caras planas)
- El número de vértices de las caras puede ser cualquiera
- Lo normal es considerar mallas formadas por triángulos





2. Modelos de fronteras (B-rep)

2.1. Lista de triángulos aislados

- La malla es una lista de triángulos

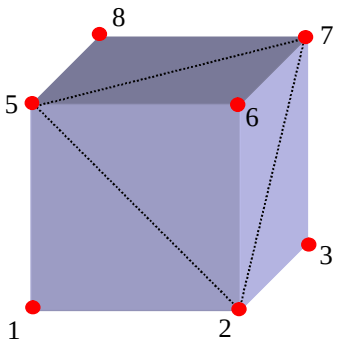
$x_1, y_1, z_1,\quad x_2, y_2, z_2,\quad x_5, y_5, z_5$

$x_7, y_7, z_7,\quad x_8, y_8, z_8,\quad x_5, y_5, z_5$

$x_2, y_2, z_2,\quad x_3, y_3, z_3,\quad x_7, y_7, z_7$

$\ldots$

Cada entrada de la lista contiene los tres vértices de cada triángulo



2. Modelos de fronteras (B-rep)

2.2. Tiras de triángulos

- Cada triángulo de la tira comparte una arista y dos vértices, que "en principio" no hay que repetir

x_0, y_0, z_0

x_1, y_1, z_1

x_2, y_2, z_2

Las tres primeras entradas la lista de vértices definen el primer triángulo

x_3, y_3, z_3

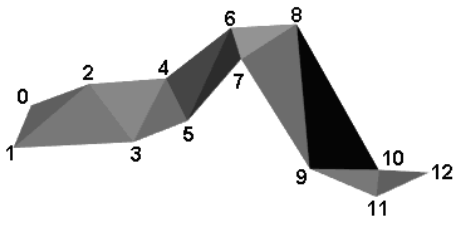
x_4, y_4, z_5

x_5, y_5, z_5

Resto de vértices

x_6, y_6, z_6

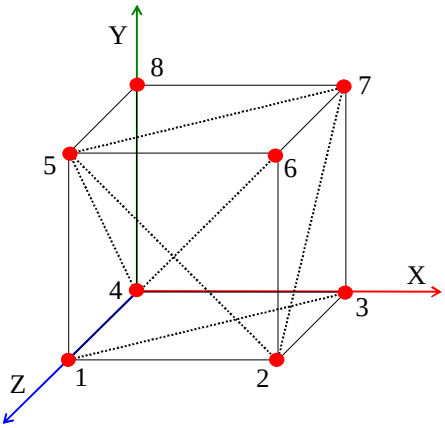
$\ldots$

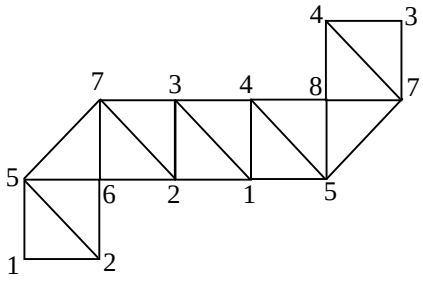


2. Modelos de fronteras (B-rep)

2.2. Tiras de triángulos

- Ejemplo. Cubo construido con una tira de triángulos

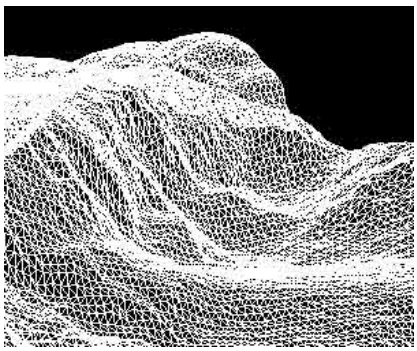


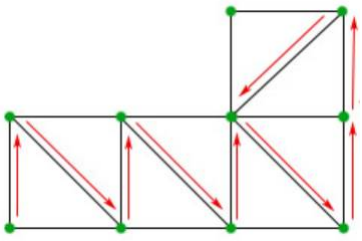


2. Modelos de fronteras (B-rep)

2.2. Tiras de triángulos

- Ejemplo. Plano con una tira de triángulos





2. Modelos de fronteras (B-rep)

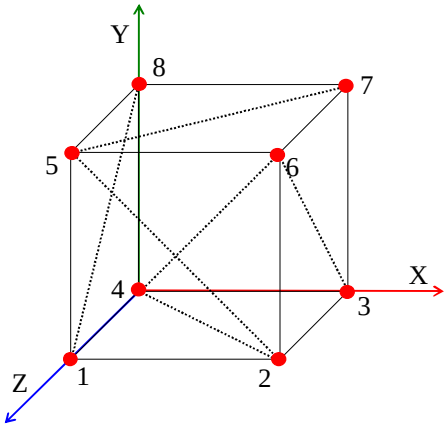
2.3. Listas de vértices y triángulos

- Información por separado de la geometría (vértices) y de topología (conectividad entre vértices)
  - Lista de vértices: una entrada por cada vértice, por tanto no se repiten
  - Lista de caras: una entrada por triángulo que contiene los índices a los vértices correspondientes de la lista de vértices

2. Modelos de fronteras (B-rep)

2.3. Listas de vértices y triángulos

- Ejemplo



Lista de triángulos	Lista de vértices	
1,2,5		
2,6,5	1	x_1, y_1, z_1
2,3,6	2	x_2, y_2, z_2
3,7,6	3	x_3, y_3, z_3
3,4,7	4	x_4, y_4, z_5
4,8,7	5	x_5, y_5, z_5
4,1,8	6	x_6, y_6, z_6
1,5,8	7	x_7, y_7, z_7
5,6,7	8	x_8, y_8, z_8
5,7,8		
4,2,1		
4,3,1		

2. Modelos de fronteras (B-rep)

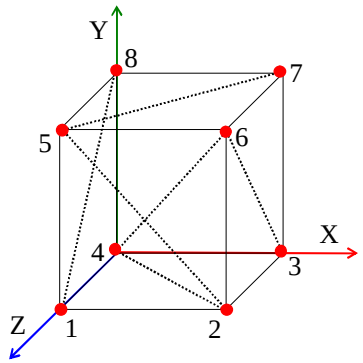
2.4. Listas de vértices, aristas y triángulos

- Es una de las representaciones mas usadas para una malla
- Información por separado de la geometría (vértices) y de topología (caras y aristas)
  - Lista de vértices: una entrada por cada vértice, por tanto no se repiten
  - Lista de aristas: una entrada por arista que contiene los índices a los vértices extremos de la arista
  - Lista de caras: una entrada por triángulo que contiene los índices a los vértices correspondientes de la lista de vértices
- Permite visualizar en modo de alambre de forma eficiente

2. Modelos de fronteras (B-rep)

2.4. Listas de vértices, aristas y triángulos

- Ejemplo

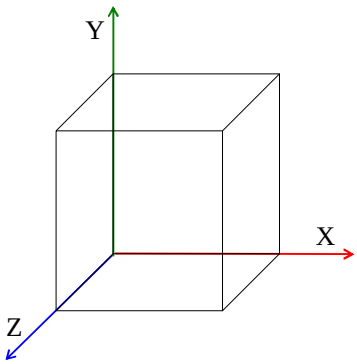


Lista de triángulos	Lista de vértices	Lista de aristas
1,2,5		1,2
2,6,5	1 x_1, y_1, z_1	2,3
2,3,6	2 x_2, y_2, z_2	3,4
3,7,6	3 x_3, y_3, z_3	4,1
3,4,7	4 x_4, y_4, z_5	1,5
4,8,7	5 x_5, y_5, z_5	2,6
4,1,8	6 x_6, y_6, z_6	3,7
1,5,8	7 x_7, y_7, z_7	4,8
5,6,7	8 x_8, y_8, z_8	5,6
5,7,8		6,7
4,2,1		7,8
4,3,1		8,5

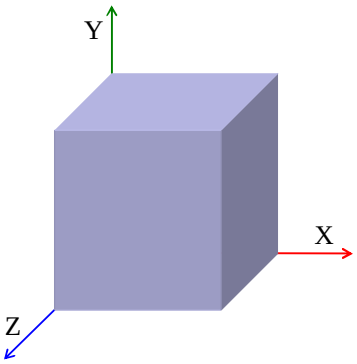
2. Modelos de fronteras (B-rep)

2.4. Listas de vértices, aristas y triángulos

- Ejemplo



Visualización con la lista de aristas



Visualización con la lista de caras

2. Modelos de fronteras (B-rep)

2.5. Implementación y visualización de las mallas

- La construcción de los modelos ha de estar separada de la visualización

- Tipos de datos básicos

```
const unsigned int X=0, Y=1, Z= 2 ;          // índices de los ejes cartesianos X,Y,Z

struct vertex
{                                            // tres valores reales que representan un vértice
  float coord[3];                            // acceder con: coord[0], coord[1], coord[2]
};                                           // o con: coord[X], coord[Y], coo[Z]

struct face
{
  int  ind_ver[3];                           // tres enteros para almacenar los índices de
};                                           // los vértices que forman un triángulo
```

2. Modelos de fronteras (B-rep)

2.5. Implementación y visualización de las mallas

- Visualizar en modo punto. Válido en todos los casos menos para listas de triángulos aislados

```
void visualizar_points (vertex *puntos, int n, float, r, float g, float b, int size)
{
Int i;
glColor3f(r,g,b);
glPointSize(size);
glBegin(GL_POINTS);
    for (i=0;i<n;i++)  glVertex3f( puntos[i].coord[0], puntos[i].coord[1],
puntos[i].coord[2]);
glEnd();
}
```


2. Modelos de fronteras (B-rep)

2.5. Implementación y visualización de las mallas

- Malla de triángulos aislados

```
struct vertex
{
  float coord[3];
};
```

```
struct triangle
{
  vertex ver[3];          // vértices de un triángulo 1, 2, 3
};

struct meshTA
{
  int  n;                 // número de triángulos
  triangle *tri;          // lista de triángulos de la malla
};

...
meshTA *objeto;
...
objeto= (meshTA*) malloc (sizeof(meshTA));
objeto->n=puntos;
objeto->tri=(triangle*)malloc (puntos*sizeof(triangle));
```

2. Modelos de fronteras (B-rep)

2.5. Implementación y visualización de las mallas

- Malla de triángulos aislados

```
void visualizar_meshTA (meshTA *malla, float, r, float g, float b, GLenum modo)
{
Int i;
n=malla->n;

if (modo==GL_LINE) glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);
     else if (modo==GL_FILL) glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);
          else glPolygonMode(GL_FRONT_AND_BACK, GL_POINT);

  glColor3f(r,g,b);
  glBegin(GL_TRIANGLES);
    for (i=0;i<n;i++)
           {glVertex3fv(malla->tri[i].ver[0].coord);
              glVertex3fv(malla->tri[i].ver[1].coord);
            glVertex3fv(malla->tri[i].ver[2].coord);
           }
  glEnd();
}
```


2. Modelos de fronteras (B-rep)

2.5. Implementación y visualización de las mallas

- Malla de tiras de triángulos

```
struct triangle_strip
{
  int n;                    // número de vértices de la tira
  vertex *ver;              // vértices de la tira de triángulos
};

struct meshTT
{
  int  n_s;                 // número de tiras de triángulos
  triangle_strip *strip;    // tira de triángulos
};
...
meshTT *objeto;
...
objeto= (meshTT*) malloc (sizeof(meshTT));
objeto->n_s=1;                                          // una sola tira de triángulos
objeto->strip[0]=(triangle_strip*)malloc(1sizeof(triangle_strip));
objeto->strip[0].n=puntos;
objeto->strip[0].ver=(vertex*)malloc(puntos*sizeof(vertex));
```

```
struct vertex
{
  float coord[3];
};
```

2. Modelos de fronteras (B-rep)

2.5. Implementación y visualización de las mallas

- Malla de tiras de triángulos

```
void visualizar_meshTT(meshTT *malla, float, r, float g, float b, GLenum modo)
{
int n_s, n, i, j;
n_s=malla->n_s;

if (modo==GL_LINE) glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);
     else if (modo==GL_FILL) glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);

glColor3f(r,g,b);
for (j=0;j<n_s;j++)
  {n=malla->strip[j].n;
   glBegin(GL_TRIANGLE_STRIP);
      for (i=0;i<n;i++)
          glVertex3f(malla->strip[j].ver[i].coord[0], malla->strip[j].ver[i].coord[1],
                     malla->strip[j].ver[i].coord[2]);
    glEnd();
   }
}
```


2. Modelos de fronteras (B-rep)

2.5. Implementación y visualización de las mallas

- Malla de vértices y triángulos

```
struct meshVT
{
  int  n_v;          // número de vértices
  int  n_c;          // número de triángulos
  vertex *ver;       // lista de vértices
  face  *car;        // lista de triángulos
};

...
meshVT *objeto;
...
objeto=(meshVT*) malloc(sizeof(meshVT));
objeto->n_v=puntos;
objeto->ver=(vertex*) malloc(puntos*sizeof(vertex));
objeto->n_c=caras;
objeto->car=(face*) malloc(caras*sizeof(face));
```

```
struct vertex
{
  float coord[3];
};

struct face
{
 int  ind_ver[3];
};
```

2. Modelos de fronteras (B-rep)

2.5. Implementación y visualización de las mallas

- Malla de vértices y triángulos

```
void visualizar_meshVT (meshVT *malla, float, r, float g, float b, GLenum modo)
{
int n_c,n_v,i;
n_c=malla->n_c;
if (modo==GL_LINE) glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);
     else if (modo==GL_FILL) glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);

glColor3f(r,g,b);
glBegin(GL_TRIANGLES);
  for (i=0;i<n_c;i++)
     { n_v=malla->car[i].ind_ver[0];
       glVertex3fv(malla->ver[n_v].coord);
       n_v=malla->car[i].ind_ver[1];
       glVertex3fv(malla->ver[n_v].coord);
       n_v=malla->car[i].ind_ver[2];
       glVertex3fv(malla->ver[n_v].coord);
    }
   glEnd();
}
```


2. Modelos de fronteras (B-rep)

2.5. Implementación y visualización de las mallas

- Malla de vértices, aristas y triángulos

```
struct edge
{
int ind_lado[2];        // índices de los vértices de una arista
};

struct meshVAT
{ int  n_v;             // número de vértices
  int  n_a;             // número de aristas
  int  n_c;             // número de triángulos
  vertex  *ver;         // lista de vértices
  edge  *ari;           // lista de aristas
  face  *car;           // lista de triángulos
};
...
meshVAT *objeto;
...
objeto=(meshVAT*) malloc(sizeof(meshVAT));
objeto->n_v=puntos;  objeto->ver=(vertex*) malloc(puntos*sizeof(vertex));
objeto->n_c=caras;   objeto->car=(face*) malloc(caras*sizeof(face));
objeto->n_a=lados;   objeto->ari=(edge*) malloc(lados*sizeof(edge));
```

```
struct vertex
{
 float coord[3];
};

struct face
{
 int  ind_ver[3];
};
```

2. Modelos de fronteras (B-rep)

2.5. Implementación y visualización de las mallas

- Malla de vértices, aristas y triángulos

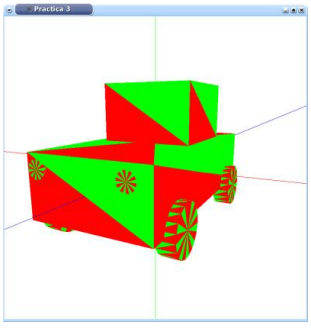
```
void visualizar_meshVAT_aristas (meshVAT *malla, float r, float g, float b)
{
int n_v,i;
glColor3f(r,g,b);
glBegin(GL_LINES);
for (i=0;i<malla->n_a;i++)
      { n_v=malla->ari[i].ind_lado[0];
        glVertex3fv(malla->ver[n_v].coord);
        n_v=malla->ari[i].ind_lado[1];
        glVertex3fv(malla->ver[n_v].coord);
      }
glEnd();
}

void visualizar_meshVAT_triangulos (meshVAT *malla, float, r, float g, float b)
{
...
glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);
...
}
```


2. Modelos de fronteras (B-rep)

2.5. Implementación y visualización de las mallas

- Otros modos de visualización: ajedrez

```
void visualizar_meshVAT_ajedrez (meshVAT *malla, float, color_a[3], float
color_b[3])
{
int n_c,n_v,i;
n_c=malla->n_c;
glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);

glBegin(GL_TRIANGLES);
  for (i=0;i<n_c;i++)
     {if ( i %2 == 0 ) glColor3f(color_a[0], color_a[1], color_a[1]);
              else glColor3f(color_b[0], color_b[1], color_b[1]);
        n_v=malla->car[i].ind_ver[0];
        glVertex3fv(malla->ver[n_v].coord);
        ...
    }
   glEnd();
}
```



2. Modelos de fronteras (B-rep)

2.5. Implementación y visualización de las mallas

- Para facilitar la codificación con C++ se pueden usar:
  - Plantillas (templates)
  - STL (Standard Template Library)

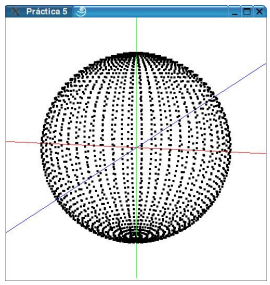
```
_vertex3f Vertex;
Vertex.x = 0 ;
Vertex.y = 5 ;
Vertex.z = 10 ;

// estructuras para una malla de vértices, aristas y puntos

vector<_vertex3f>  Vertices ;          // lista de vértices
vector<_vertex3i>  Triangulos ;        // lista de triángulos
vector<_vertex2i>  Aristas;            // lista de aristas
```


2. Modelos de fronteras (B-rep)

2.5. Implementación y visualización de las mallas

- STL vector

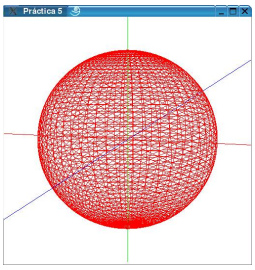
```
// Visualizando en modo puntos

...
int i
...
glBegin( GL_POINTS ) ;
for( i=0 ; i < Vertices.size() ; i++ )
    glVertex3f( Vertices[i].x, Vertices[i].y, Vertices[i].z
);
glEnd() ;
```



2. Modelos de fronteras (B-rep)

2.5. Implementación y visualización de las mallas

- STL vector

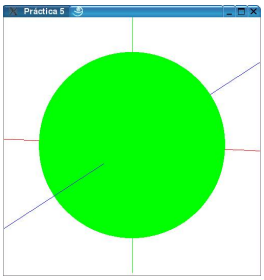



```
// Visualizando en modo alambre
...
int Vertex_1,Vertex_2, i;
...
glBegin( GL_LINES ) ;
for( i= 0 ; i < Aristas.size() ; i++ ){
Vertex_1 = Aristas[i]._0 ;
Vertex_2 = Aristas[i]._1 ;
glVertex3f( Vertices[Vertex_1]x, Vertices[Vertex_1].y,Vertices[Vertex_1].z ) ;
glVertex3f( Vertices[Vertex_2]x, Vertices[Vertex_2].y,Vertices[Vertex_2].z ) ;
}
glEnd() ;
```


2. Modelos de fronteras (B-rep)

2.5. Implementación y visualización de las mallas

- STL vector



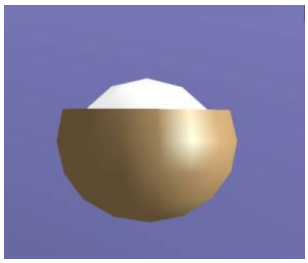

```
// Visualizando en modo sólido
...
int Vertex_1,Vertex_2,Vertex_3;
...
glBegin( GL_TRIANGLES );
for ( i= 0 ; i < Triangulos.size() ; i++ ){
Vertex_1 = Triangulos[i]._0 ;
Vertex_2 = Triangulos[i]._1 ;
Vertex_3 = Triangulos[i]._2 ;
glVertex3f( Vertices[Vertex_1]x, Vertices[Vertex_1].y, Vertices[Vertex_1].z);
glVertex3f( Vertices[Vertex_2]x, Vertices[Vertex_2].y, Vertices[Vertex_2].z);
glVertex3f( Vertices[Vertex_3]x, Vertices[Vertex_3].y, Vertices[Vertex_3].z);
}
glEnd() ;
```

2. Modelos de fronteras (B-rep)

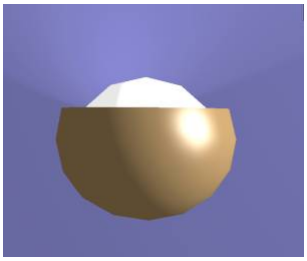
2.6. Atributos de las caras

- Cálculo de normales, a utilizar en iluminación
  - Normal a las caras: suavizado plano
  - Normal a los vértices: suavizado de Gouraud o Phong

Suavizado plano



Suavizado Gouraud

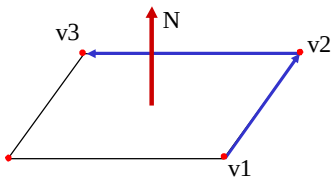


Suavizado Phong

2. Modelos de fronteras (B-rep)

2.6. Atributos de las caras

- Normal a las caras

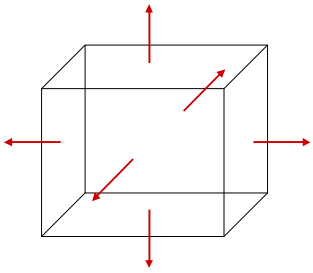


$$\vec{N} = \frac{\overrightarrow{v_2v_1} \otimes \overrightarrow{v_3v_2}}{\left\| \overrightarrow{v_2v_1} \otimes \overrightarrow{v_3v_2} \right\|}$$

2. Modelos de fronteras (B-rep)

2.6. Atributos de las caras

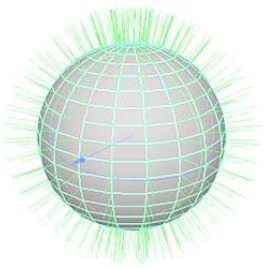
- La iluminación plana es verosímil para objetos que si se puedan modelar con caras planas pero no para objetos suaves (curvos)

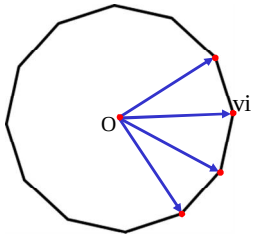


2. Modelos de fronteras (B-rep)

2.6. Atributos de las caras

- Normal a los vértices. Dos formas de calcular esta normal:
  - A partir de la forma del objeto





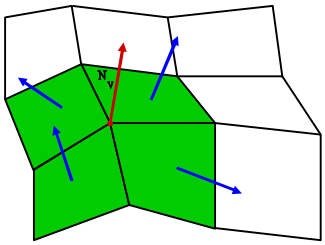
$$\vec{N}_{vi} = \frac{\overrightarrow{v_i O}}{\left\| \overrightarrow{v_i O} \right\|}$$

Normal a vértices en una esfera

2. Modelos de fronteras (B-rep)

2.6. Atributos de las caras

- Normal a los vértices.
  - Una normal promedio de las normales de las caras que se reúnen en un vértice dado



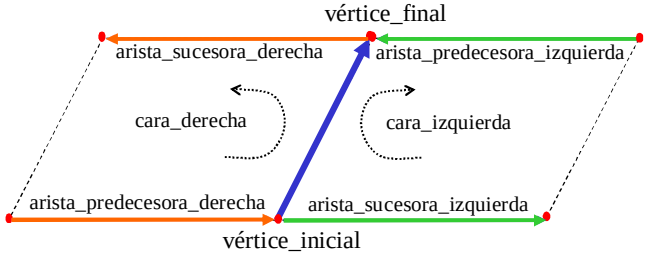
$$\vec{N}_{v,i} = \frac{\sum_{j=1}^{n} \vec{N}_{j,i}}{\left\| \sum_{j=1}^{n} \vec{N}_{j,i} \right\|}$$

$N_{j,i}$ $(j = 1,..,n)$ *nórmales de las caras que comparten el vértice i*

2. Modelos de fronteras (B-rep)

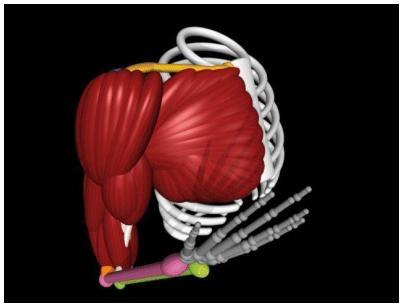
2.7. Aristas aladas (winged-edges , Baumgart72)

- Esta estructura pone el énfasis en las aristas en lugar de las caras
- Para cada arista tenemos:
  - sus dos vértices
  - las dos caras adyacentes a la arista (cara derecha e izquierda)
  - Las aristas predecesoras y sucesoras en las caras adyacentes



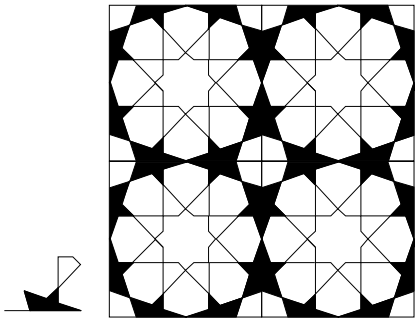
3. Transformaciones geométricas

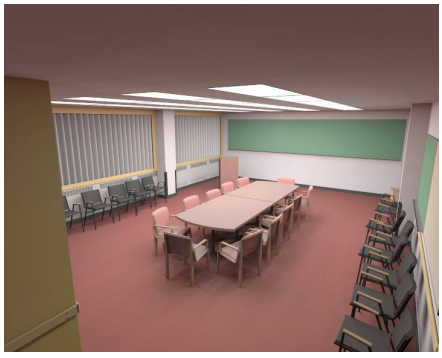
- El movimiento de cualquier objeto se expresa con transformaciones geométricas



3. Transformaciones geométricas

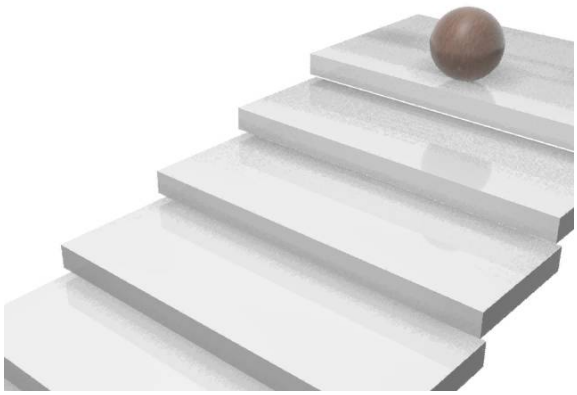
- En el manejo de instancias de objetos gráficos desempeñan un papel muy importante. Una instancia es un objeto gráfico definido en su propio sistema de coordenadas cartesianas





3. Transformaciones geométricas

- Un comentario sobre animación



3. Transformaciones geométricas

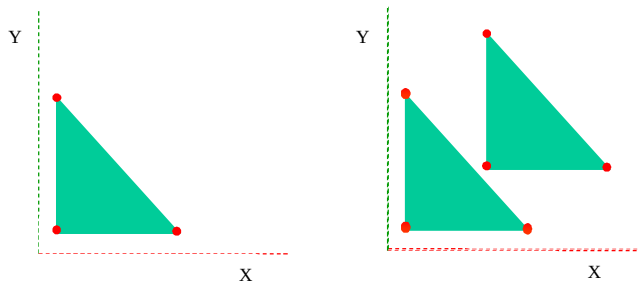
3.1. Tipos de transformaciones

- Un objeto se puede considerar como un conjunto de puntos. Las transformaciones geométricas serán operaciones para calcular nuevas posiciones de los puntos de un objeto
  - Translación
  - Rotación
  - Escalado
  - Deformación (cizalla)
  - Reflexión (caso particular del escalado)

3. Transformaciones geométricas

3.1. Tipos de transformaciones

- Translación

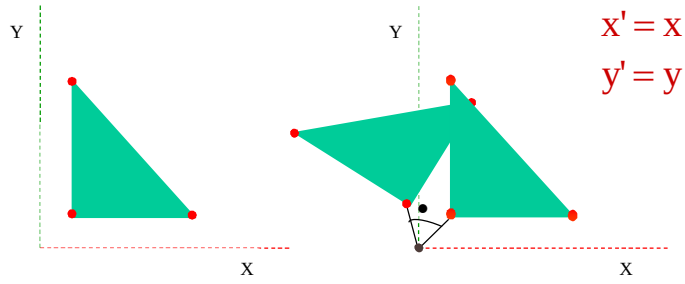


$$x' = x + t_x$$
$$y' = y + t_y$$

3. Transformaciones geométricas

3.1. Tipos de transformaciones

- Rotación con ángulo en el sentido contrario a las agujas del reloj

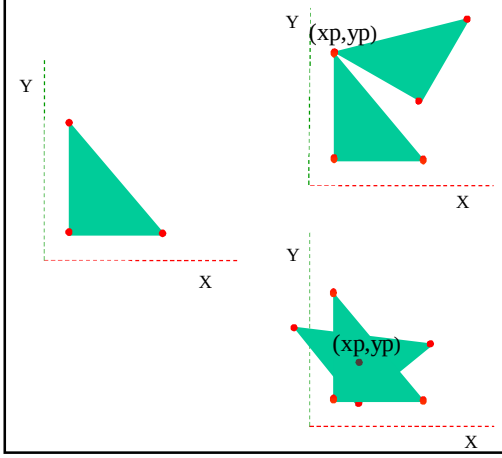


$$x' = x \cos\theta - y \operatorname{sen}\theta$$
$$y' = y \cos\theta + x \operatorname{sen}\theta$$

3. Transformaciones geométricas

3.1. Tipos de transformaciones

- Rotación con ángulo en el sentido contrario a las agujas del reloj y con respecto a un punto pivote

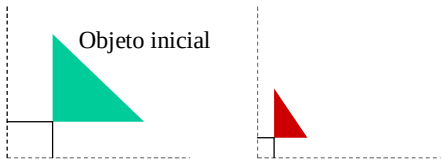


$$x' = xp + (x - xp)\ \cos\theta - (y - yp)\ \text{sen}\ \theta$$
$$y' = yp + (y - yp)\ \cos\theta + (x - xp)\ \text{sen}\ \theta$$

3. Transformaciones geométricas

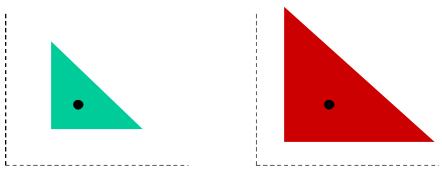
3.1. Tipos de transformaciones

- Escalado con respecto al origen



$$x' = x\ Sx$$
$$y' = y\ Sy$$

- Escalado con respecto a un punto pivote



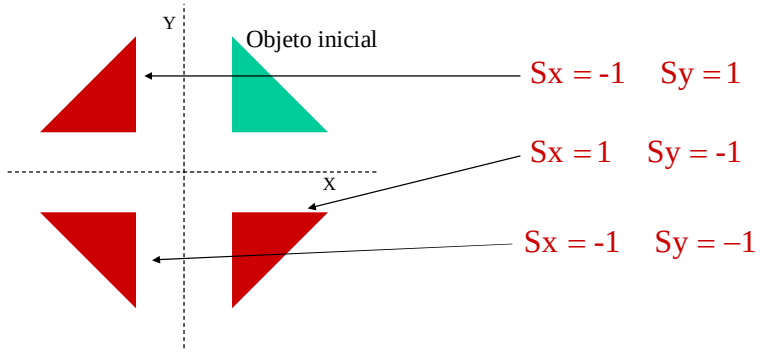
$$x' = xp + (x - xp)\ Sx$$
$$y' = yp + (y - yp)\ Sy$$

3. Transformaciones geométricas

3.1. Tipos de transformaciones

- Reflexión (Flip). Caso particular del escalado

$$x' = x \; Sx$$
$$y' = y \; Sy$$



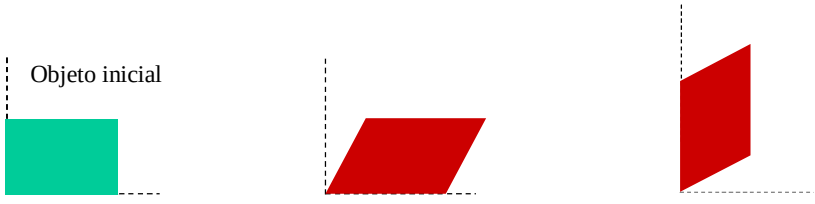
3. Transformaciones geométricas

3.1. Tipos de transformaciones

- Deformación o cizalla (Shear) con respecto al origen

$$x' = x + y\,SHx$$
$$y' = y$$

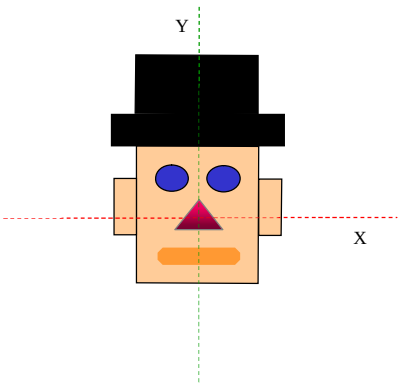
$$x' = x$$
$$y' = x\,SHy + y$$

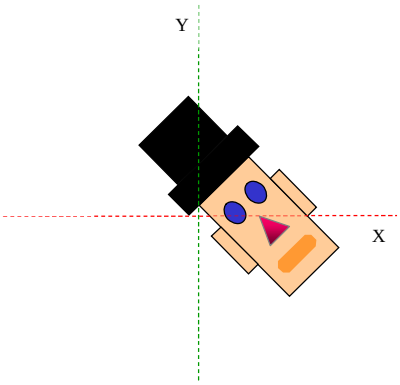


3. Transformaciones geométricas

3.1. Tipos de transformaciones

- Al conjunto de transformaciones geométricas que se aplican a un objeto se le denomina transformación de modelado





3. Transformaciones geométricas

3.2. Representación matricial

- Las reglas para componer ecuaciones son complejas si se usan funciones.
- Se usa la representación matricial para las transformaciones. Componer transformaciones equivale a multiplicar matrices.
- Se emplean las coordenadas homogéneas, en las que un punto se representa por el vector (x',y',h) (2D) o por (x',y',z',h) (3D) y se usan matrices 3x3 (2D) o 4x4 (3D) para representar las trasformaciones
- Para no realizar normalizaciones se toma h=1 y la última columna de las matrices 3x3 como (0,0,1) o bien (0,0,0,1) para las matrices 4x4.

3. Transformaciones geométricas

3.2. Representación matricial

- Hay dos representaciones para las matrices:
  - A) si un punto se considera como vector fila (x,y,1) se multiplica por la izquierda de la matriz
  - B) si un punto se considera como columna $(x,y,1)^t$ se multiplica por la derecha

$$T_{tx,ty} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{pmatrix} \quad R_\theta = \begin{pmatrix} \cos\theta & \text{sen}\,\theta & 0 \\ -\text{sen}\,\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$S_{sx,sy} = \begin{pmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad SH_{shx,shy} = \begin{pmatrix} 1 & sh_y & 0 \\ sh_x & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Forma A

$$T^t_{tx,ty} = \begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix} \quad R^t_\theta = \begin{pmatrix} \cos\theta & -\text{sen}\,\theta & 0 \\ \text{sen}\,\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$S^t_{sx,sy} = \begin{pmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad SH^t_{shx,shy} = \begin{pmatrix} 1 & sh_x & 0 \\ sh_y & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

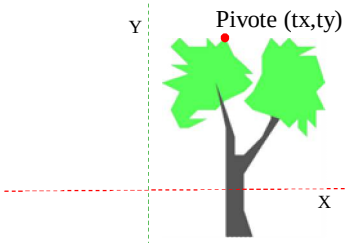
Forma B

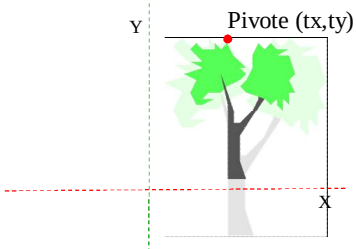
3. Transformaciones geométricas

3.2. Representación matricial

- Ejemplo de escalado con respecto a un punto pivote, con la representación A)

$$\begin{pmatrix} x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots \\ x_n & y_n & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -t_x & -t_y & 1 \end{pmatrix} \begin{pmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{pmatrix} = \text{Objeto} \times T_{-tx,-ty} \times S_{sx,sy} \times T_{tx,ty} \equiv \text{Objeto} \times M$$



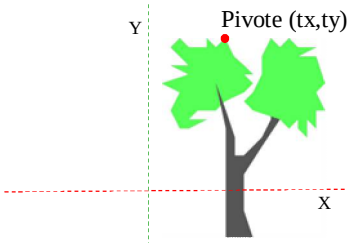


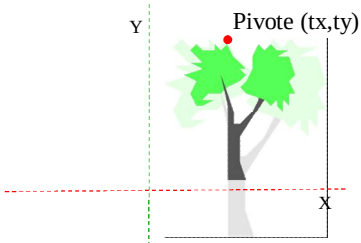
3. Transformaciones geométricas

3.2. Representación matricial

- Ejemplo de escalado con respecto a un punto pivote, con la representación B)

$$\begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & 0 & -t_x \\ 0 & 1 & -t_y \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} x_1 & \cdots & x_n \\ y_1 & \cdots & y_n \\ 1 & \cdots & 1 \end{pmatrix} = T^t_{tx,ty} \times S^t_{sx,sy} \times T^t_{-tx,-ty} \times \text{Objeto} \equiv M \times \text{Objeto}$$





53

3. Transformaciones geométricas

3.2. Representación matricial

- Toda transformación geométrica tiene inversa. Fácil de calcular con matrices: con la notación A):

$$T^{-1}_{tx,ty} = T_{-tx,-ty} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -t_x & -t_y & 1 \end{pmatrix} \qquad R^{-1}_{\theta} = R_{-\theta} = \begin{pmatrix} \cos\theta & -\text{sen}\,\theta & 0 \\ \text{sen}\,\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$S^{-1}_{sx,sy} = S_{1/sx,1/sy} = \begin{pmatrix} 1/s_x & 0 & 0 \\ 0 & 1/s_y & 0 \\ 0 & 0 & 1 \end{pmatrix} \qquad SH^{-1}_{shx,shy} = \begin{pmatrix} \frac{1}{1-sh_x sh_y} & \frac{-sh_y}{1-sh_x sh_y} & 0 \\ \frac{-sh_x}{1-sh_x sh_y} & \frac{1}{1-sh_x sh_y} & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

54

3. Transformaciones geométricas

3.2. Representación matricial

- Matrices 3D (translación, escalado y defomación), adoptando la notación de puntos como vectores fila, (x,y,z,1)

$$T_{tx,ty,tz} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ t_x & t_y & t_z & 1 \end{pmatrix} \qquad S_{sx,sy,sz} = \begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$SH_x = \begin{pmatrix} 1 & sh_{xy} & sh_{xz} & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad SH_y = \begin{pmatrix} 1 & 0 & 0 & 0 \\ sh_{yx} & 1 & sh_{yz} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad SH_z = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ sh_{zx} & sh_{zy} & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

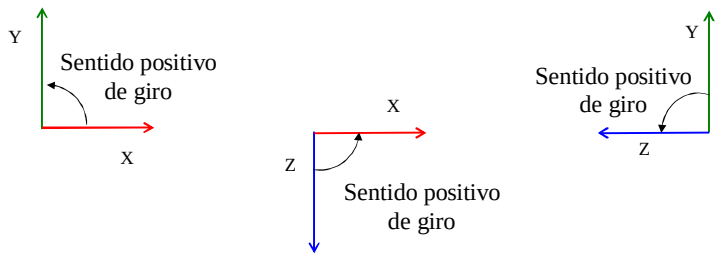
55

3. Transformaciones geométricas

3.2. Representación matricial

- Matrices 3D para rotación, adoptando la notación de puntos como vectores fila, (x,y,z,1)

$$R_{z,\theta} = \begin{pmatrix} \cos\theta & \text{sen}\,\theta & 0 & 0 \\ -\text{sen}\,\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad R_{y,\theta} = \begin{pmatrix} \cos\theta & 0 & -\text{sen}\,\theta & 0 \\ 0 & 1 & 0 & 0 \\ \text{sen}\,\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad R_{x,\theta} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & \text{sen}\,\theta & 0 \\ 0 & -\text{sen}\,\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

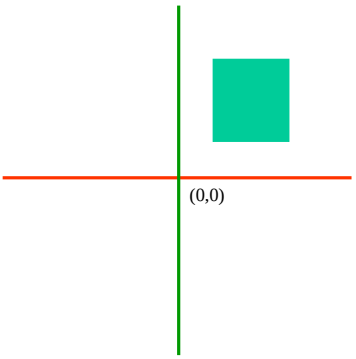


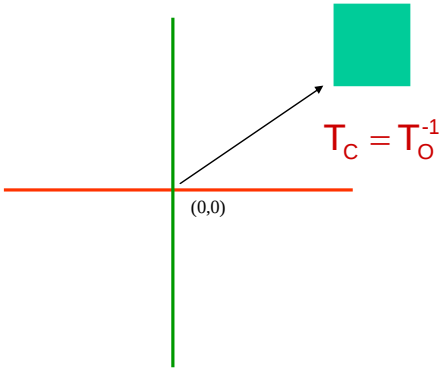
56

3. Transformaciones geométricas

3.2. Representación matricial

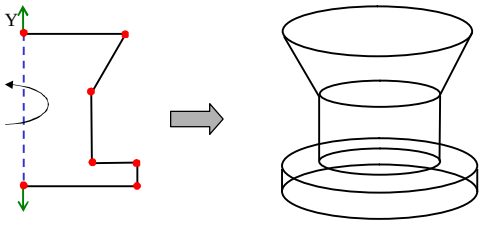
- Transformaciones de coordenadas. Son inversas a las transformaciones geométricas.



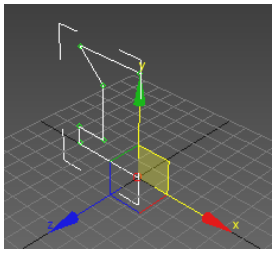


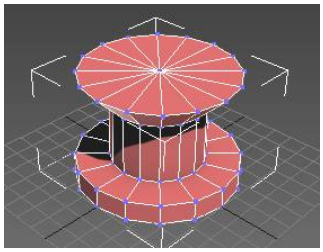
4. Generación de modelos de frontera

4.1. Revolución



$$p'(x,y,z) = R_{y,\alpha} p(x,y,z)$$



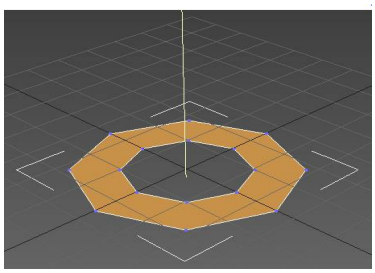


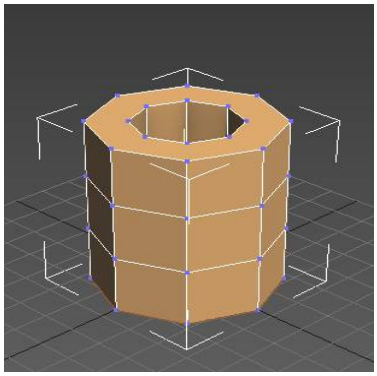
4. Generación de modelos de frontera

4.2. Barrido o extrusión

- Extrusión según una recta

$$p'(x,y,z) = T_{tx,ty,tz}\,p(x,y,z)$$

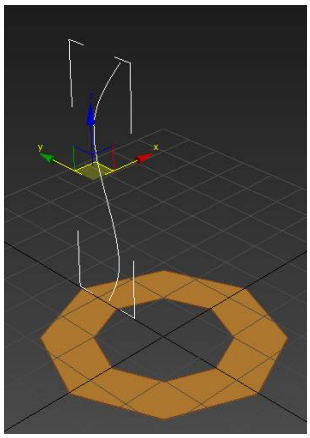


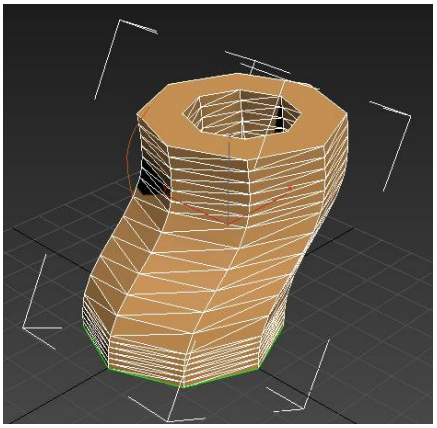


4. Generación de modelos de frontera

4.2. Barrido o extrusión

- Extrusión según una curva

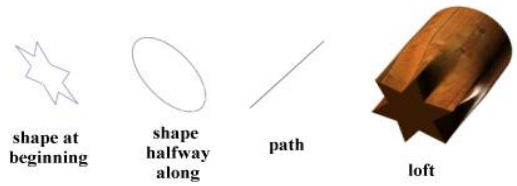




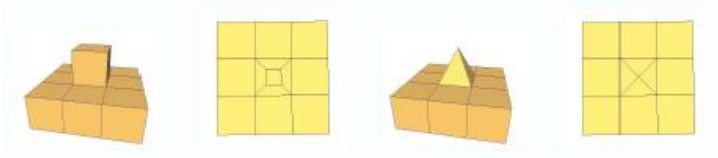
4. Generación de modelos de frontera

4.2. Barrido o extrusión

- Combinación de perfiles

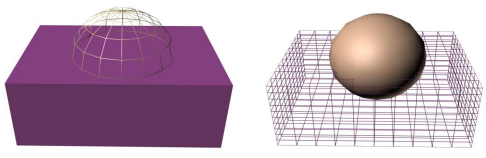


- Extrusión local



4. Generación de modelos de frontera

4.3. Operaciones booleanas

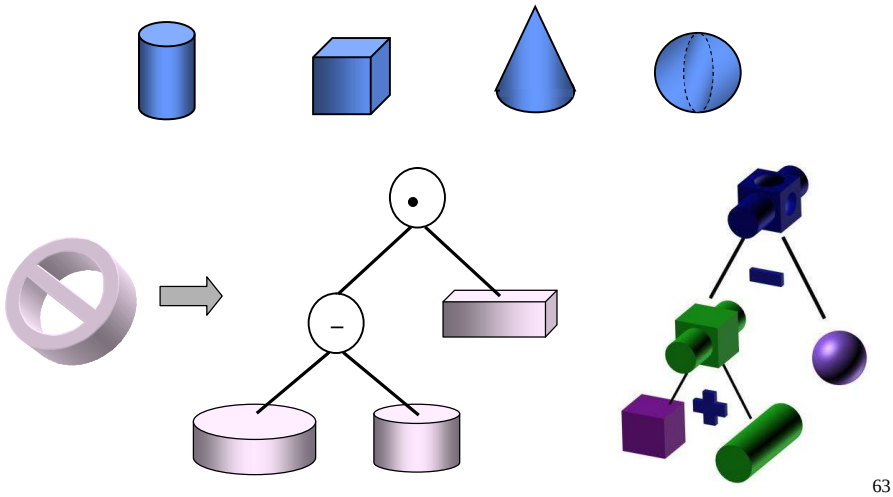


Objeto 1 Objeto 2

Unión Intersección Diferencia

4. Generación de modelos de frontera

4.4. Geometría constructiva de sólidos (CSG)

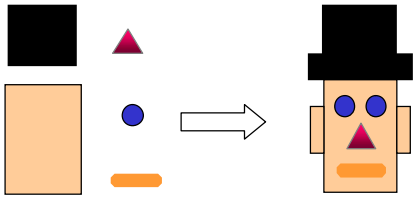


5. Modelos jerárquicos

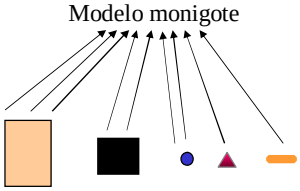
- Permiten la creación de modelos complejos a partir de modelos más simples
- Tipos de modelos jerárquicos:
  - Objetos compuestos
  - Objetos articulados
  - Escenas formadas por varios objetos
- Se representan mediante grafos dirigidos acíclicos (Directed Acyclic Graph o DAG). No hay un tipo de grafo normalizado
  - En las hojas tenemos objetos simples.
  - En los nodos transformaciones geométricas e instancias o bien un objeto y en los arcos transformaciones geométricas

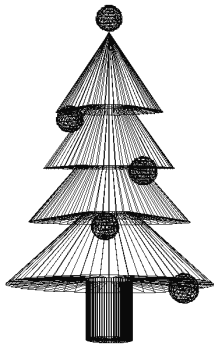
5. Modelos jerárquicos

5.1. Tipos de modelos jerárquicos. Compuestos



Modelo monigote

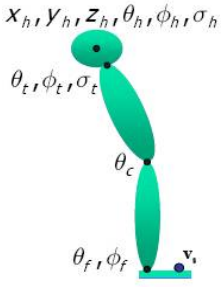


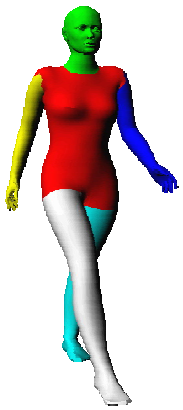


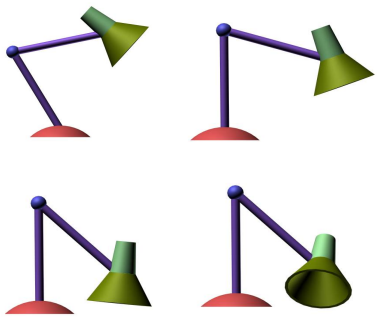
5. Modelos jerárquicos

5.1. Tipos de modelos jerárquicos. Objetos articulados

- Herencia de transformaciones

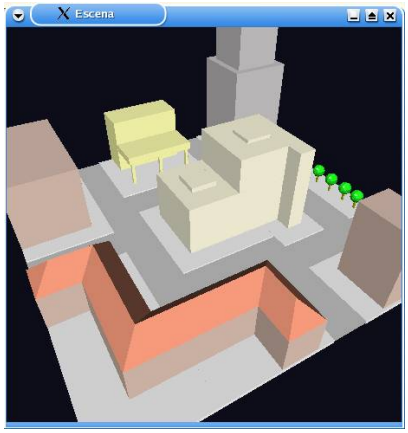






5. Modelos jerárquicos

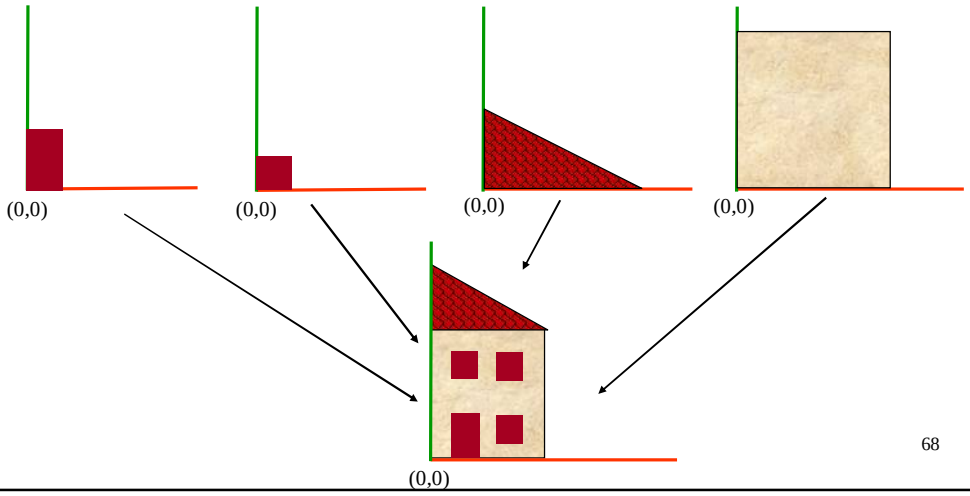
5.1. Tipos de modelos jerárquicos. Escena



5. Modelos jerárquicos

5.2. Grafos dirigidos acíclicos

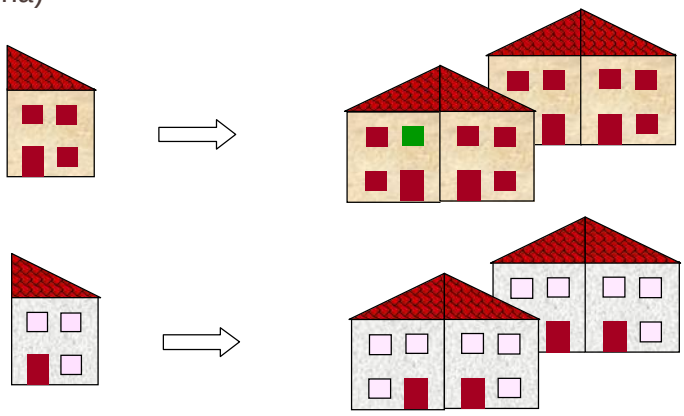
- Es conveniente utilizar coordenadas maestras



5. Modelos jerárquicos

5.2. Grafos dirigidos acíclicos

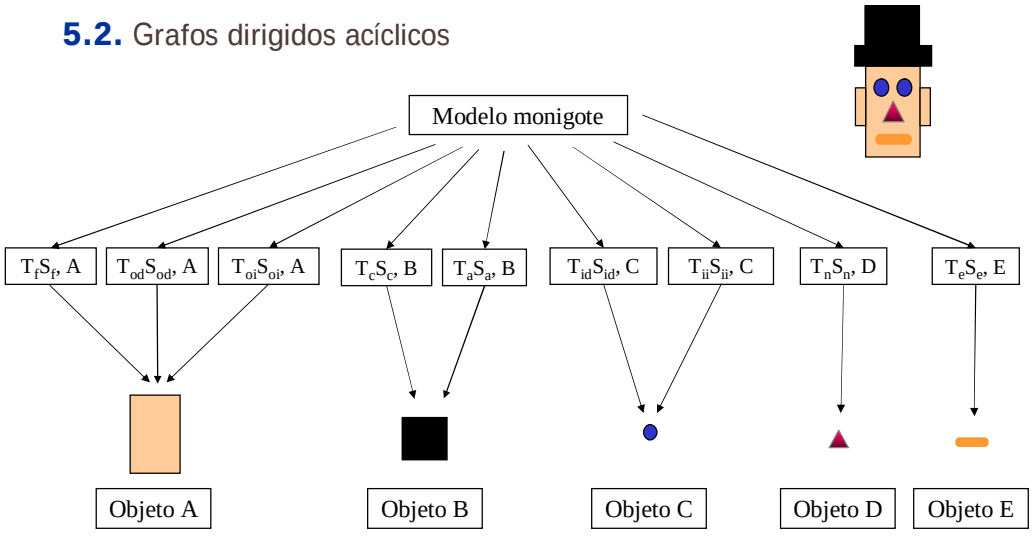
- Un objeto definido en coordenadas maestras se puede instanciar por copia (uso de mucha memoria) o por referencia (poca memoria)



69

5. Modelos jerárquicos

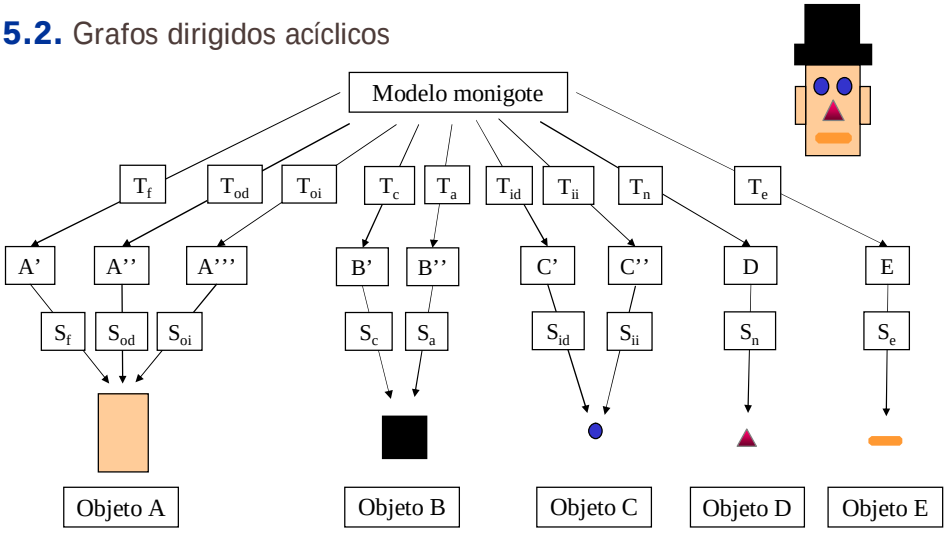
5.2. Grafos dirigidos acíclicos



70

5. Modelos jerárquicos

5.2. Grafos dirigidos acíclicos

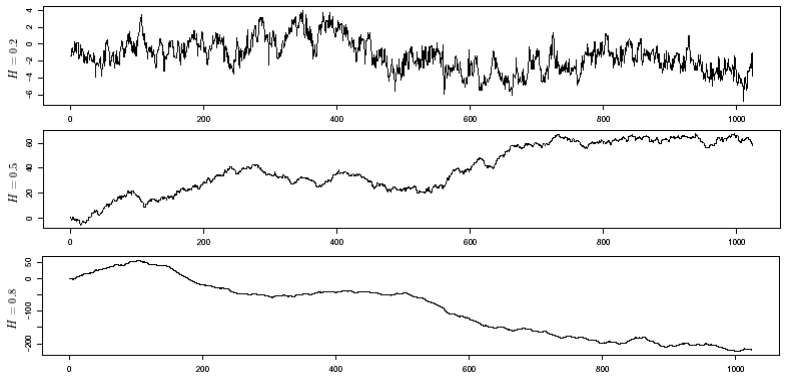


Anexo I. Objetos naturales

Movimiento Browniano fraccional en una dimensión

$$\mathrm{var}\left(Y(t_i) - Y(t_j)\right) \propto \left|t_i - t_j\right|^{2H}$$

$$Y(t), \text{ e } \left(1/r^H\right)Y(rt) \quad \text{son} \quad \text{indistinguibles}$$

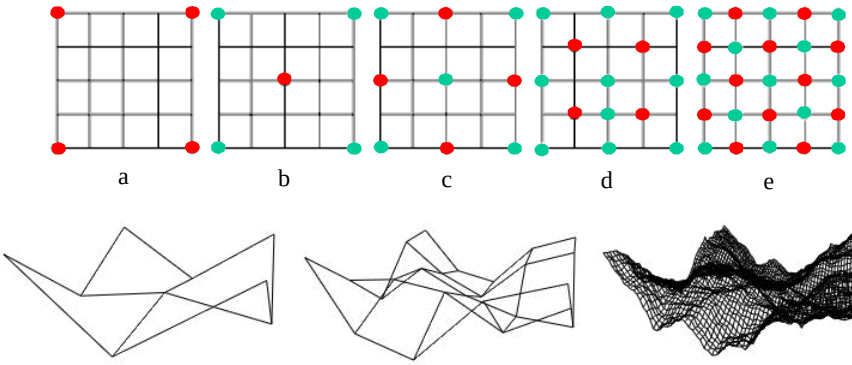


Anexo I. Objetos naturales

Modelado de montañas: Movimiento Browniano fraccional en dos dimensiones

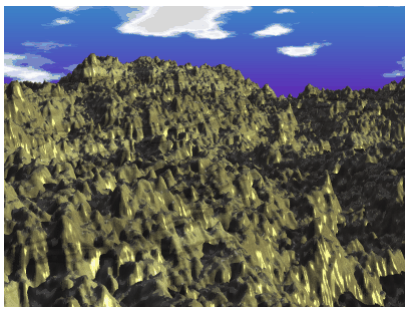
$$\mathrm{var}\left(Y(t_i,s_k)-Y(t_j,s_l)\right)\propto\left((t_i-t_j)^2+(s_k-s_l)^2\right)^H$$

- Algoritmo de desplazamiento por el punto medio



Anexo I. Objetos naturales

Modelado de montañas: Movimiento Browniano fraccional en dos dimensiones



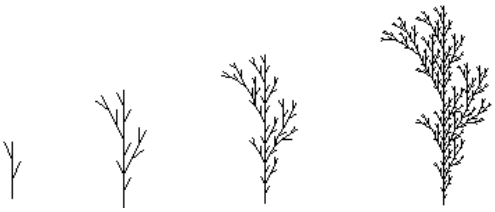
Anexo I. Objetos naturales

Modelado de Plantas: Sistemas L (Lindenmayer) (http://algorithmicbotany.org/)

$w : a$
$p_1 : a \rightarrow ab$
$p_2 : b \rightarrow a$

a, ab, aba, abaab abaababa

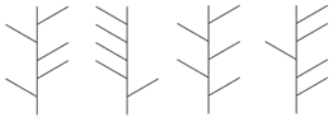
a, b factores de escala
$\delta = 20^{\circ}$
$w : F$
$p : [F] \rightarrow F[-bF]aF[+F][bF]$



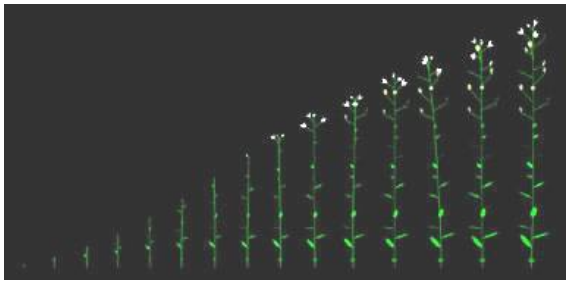
Anexo I. Objetos naturales

- Sistemas L paramétricos: una regla de producción por cada símbolo
- Sistemas L estocásticos: más de una regla de producción por símbolo

$$p- > \begin{cases} 0.5 : [+FF]FA \\ 0.5 : [-FF]FA \end{cases}$$

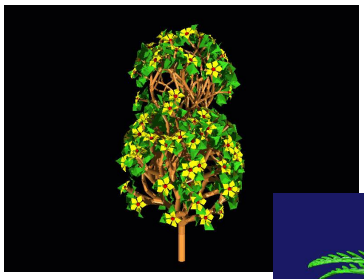


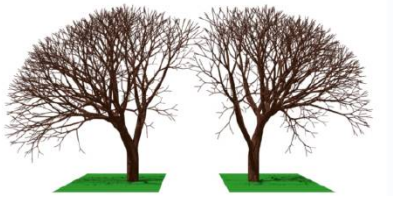
- Sistemas L temporales



Anexo I. Objetos naturales

Modelado de Plantas: Sistemas L (Lindenmayer)



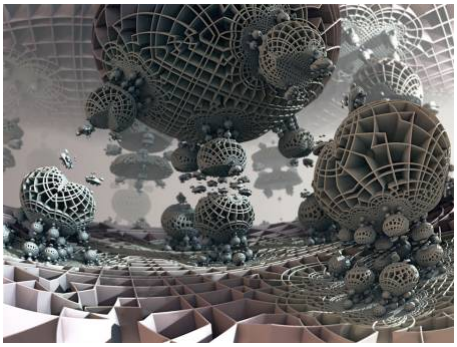


77

Anexo I. Objetos naturales

Software sobre fractales

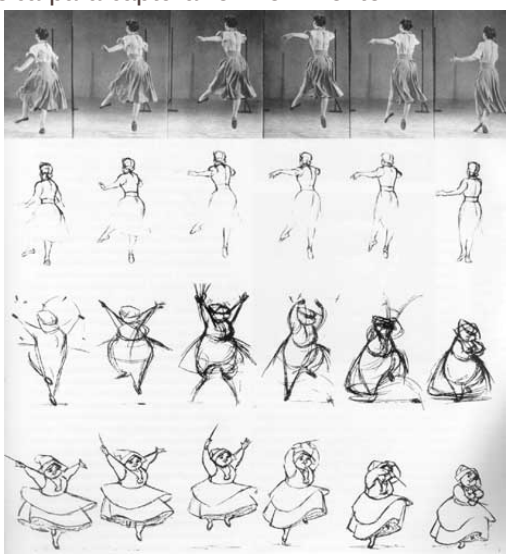
- http://mandelbulb.com/
- http://algorithmicbotany.org/virtual_laboratory/
- http://planetside.co.uk/whats-new-in-terragen-4/



78

Anexo II. Rotoscoping 3D

- Rotoscoping 2D: técnica clásica para capturar el movimiento (Max Fleischer 1914)



pp1

Anexo II. Rotoscoping 3D

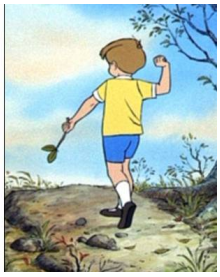
- Ejemplos de películas 2D

Diapositiva 80

pp1 "A Scanner Darkly " Una mirada oscura 2006
El señor de los anillos 1978
pp; 21/04/2008

Anexo II. Rotoscoping 3D

- "How Disney recycled its classic cartoons"

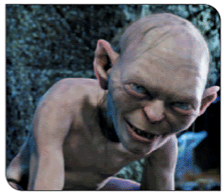



81

Anexo II. Rotoscoping 3D

- Dispositivos para capturar el movimiento
  - Electromagnéticos
  - Electromecánicos
  - Fibra óptica
  - Ópticos

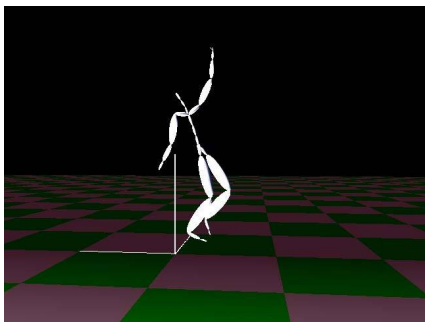


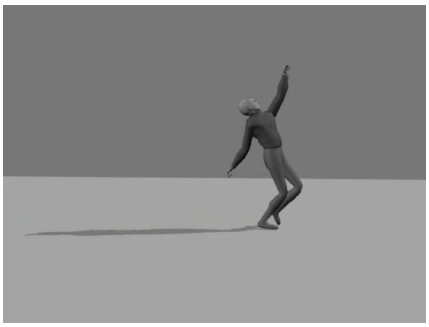



82

Anexo II. Rotoscoping 3D

- Dos tipos de ficheros para describir el movimiento
  - Esqueleto (.asf). Contiene las características biométricas y la jerarquía de las articulaciones del modelo
  - Datos angulares (.amc). Contiene los valores de los ángulos de rotación de las articulaciones para cada frame





Anexo II. Rotoscoping 3D

- Esqueleto (.asf)
  - Cabecera, huesos, jerarquía

```
:version 1.10
:name BioSkeleton
:units
mass 1.0
length 1.0
angle deg
:documentation
Example of an ASF file
:root
axis XYZ
order TX TY TZ RZ RY RX
position 0.0 0.0 0.0
orientation 0.0 0.0 0.0
:bonedata
```

Cabecera

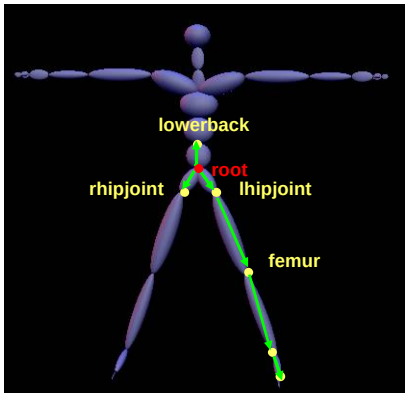
```
begin
id bone_id              // identificador del hueso
name bone_name          // nombre del hueso
direction dX dY dZ // vector de dirección en
                        // coordenadas mundiales
length 7.01722          // longitud del hueso
axis 0 0 20 XYZ         // rotaciones iniciales
dof rx ry rz            // grados de libertad
limits (-160.0 20.0)
       (-70.0 70.0)
       (-60.0 70.0)   // límites de rotaciones
end
```

Huesos

Anexo II. Rotoscoping 3D

- Esqueleto (.asf)
  - Cabecera, huesos, jerarquía

```
begin
  root lhipjoint rhipjoint lowerback
   lhipjoint lfemur
   lfemur ltibia
   ltibia lfoot
   lfoot ltoes
   rhipjoint rfemur
   rfemur rtibia
   rtibia rfoot
   rfoot rtoes
   lowerback upperback
   upperback thorax
   thorax lowerneck lclavicle rclavicle
   ...
  end
```

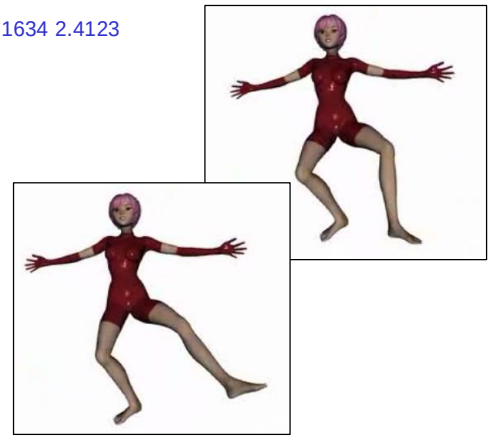


Jerarquía

Anexo II. Rotoscoping 3D

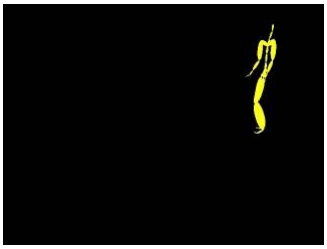
- Datos angulares (.amc)

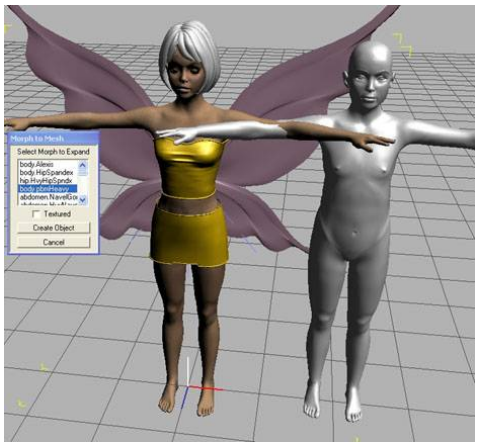
```
n                          // número de frame
root 18.0963 17.7682 -7.76048 0.944532 -46.1634 2.4123
lowerback 4.28681 -0.118085 -2.68161
upperback -0.0248133 -0.290989 0.0933187
thorax -2.41568 -0.11332 1.61874
lowerneck -22.4034 -5.47014 -9.7586
upperneck 25.4448 -7.16285 9.12669
head 12.9471 -2.96253 3.88887
rclavicle -8.95214e-014 1.76918e-014
rhumerus -38.9866 -9.5565 -84.0352
rradius 33.5224
rwrist 3.10481
rhand -14.7847 9.11536
rfingers 7.12502
...
```



Anexo II. Rotoscoping 3D

- Bibliotecas: http://mocap.cs.cmu.edu/tools.php
  http://www.mocapclub.com/Pages/Library.htm
  https://sites.google.com/a/cgspeed.com/cgspeed/motion-capture





87