Informática Gráfica
Tema 2. Modelado de objetos.

Domingo Martín

Dpto. Lenguajes y Sistemas Informáticos
ETSI Informática y de Telecomunicación
Universidad de Granada

Índice

Informática Gráfica
Tema 2. Modelado de objetos.

Introducción

- La generación de un gráfico mediante un ordenador es un proceso largo y dificultoso
- Hacen falta:
  - Objetos
  - Observador
  - Luz
- En este tema veremos cómo tratar los objetos.
- Tenemos que obtener una representación de los objetos que sea tratable por el ordenador: un modelo.

Introducción

- Un objeto real:
  - ocupa espacio
  - tiene un color
  - tiene un material
  - etc.
- Físicamente están formados por partículas que se agrupan a distintos niveles. En el nivel más básico están los quarks.
- Al bajar de nivel se aumenta el número de particulas pero éstas son más sencillas

Introducción

- ¿Cual puede ser la representación tratable por el ordenador, siendo éste una máquina de tratar información?

Partícula $\Longleftrightarrow$ **Punto**

  - $1D\ \Re \rightarrow p = (x)$
  - $2D\ \Re^2 \rightarrow p = (x, y)$
  - $3D\ \Re^3 \rightarrow p = (x, y, z)$

- El punto se debe referenciar con respecto a un sistema de coordenadas

Introducción

En general, se modelizan objetos de nuestra realidad tridimensional $\rightarrow$ volúmenes

- Obsérvese que se establece una relación entre una entidad en 3 dimensiones con otra de 0 dimensiones.
- La principal característica del punto es su posición, que en tres dimensiones vendría dada por $\mathbf{p} = (x, y, z)$ en coordenadas cartesianas, y $\mathbf{p} = (\alpha, \beta, d)$ en coordenadas polares.

Modelo de puntos

- El punto, entendido como coordenadas en el espacio, números, es tratable por el ordenador
- El conjunto de puntos o vértices conforman la geometría del objeto
- El problema que surge es cómo asociar a cada partícula un punto, dada la gran cantidad que hay.

Solución $\rightarrow$ usar los más representativos

	Vértices
P0	(x0,y0,z0)
P1	(x1,y1,z1)
P2	(x2,y2,z2)
P3	(x3,y3,z3)
P4	(x4,y4,z4)
P5	(x5,y5,z5)
P6	(x6,y6,z6)
P7	(x7,y7,z7)

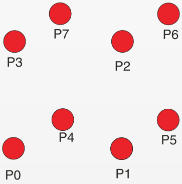


Figura : Un cubo representado con 8 puntos.

Modelo de alambres

- El modelo de puntos no permite una correcta visualización salvo que la densidad sea grande
- Una solución es mostrar relaciones: topología
- El **modelo de alambres** visualiza las aristas (conexión entre parejas de vértices)

Aristas

A0	(P0,P1)
A1	(P1,P2)
A2	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
A10	[illegible]
A11	(P4,P0)

Vértices

P0	(x0,y0,z0)
P1	(x1,y1,z1)
P2	(x2,y2,z2)
P3	(x3,y3,z3)
P4	(x4,y4,z4)
P5	(x5,y5,z5)
P6	(x6,y6,z6)
P7	(x7,y7,z7)

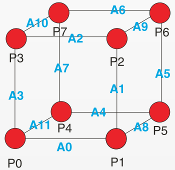


Figura : Un cubo representado con las aristas.

Modelo de superficie

- El modelo de alambres puede producir una solución ambigüa y además tampoco representa de forma realista.
- La solución consiste en mostrar la superficie: **modelos de fronteras**
- El objeto se representa como un conjunto de polígonos, por ejemplo, triángulos o cuadriláteros

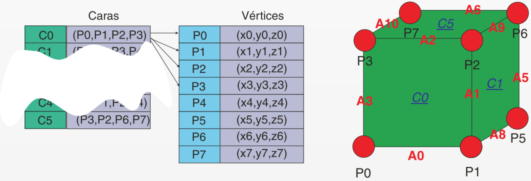


Figura : Un cubo representado con las caras.

Modelo de volumen

- Hay aplicaciones para las que es más importante conocer cual es el contenido y no la superficie.
  - aplicaciones médicas
  - exploración del subsuelo
  - etc.
- La unidad de representación es el vóxel, de forma cúbica

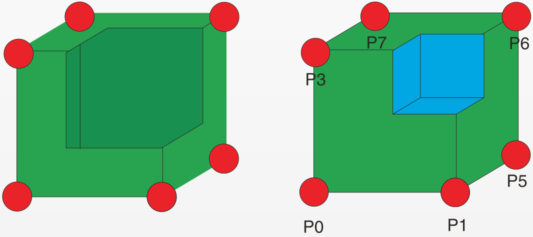


Figura : Un cubo representado como un volumen.

Modelo de puntos

```
float Vertices[10][3] ;
```

	Vértices
P0	(x0,y0,z0)
P1	(x1,y1,z1)
P2	(x2,y2,z2)
P3	(x3,y3,z3)
P4	(x4,y4,z4)
P5	(x5,y5,z5)
P6	(x6,y6,z6)
P7	(x7,y7,z7)

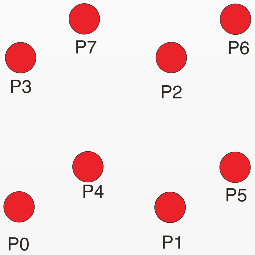


Figura : Un cubo representado con 8 puntos.

Modelo de alambres

```
int Edges[20][2];

Edges[0][0]=0;
Edges[0][1]=3;
...
```

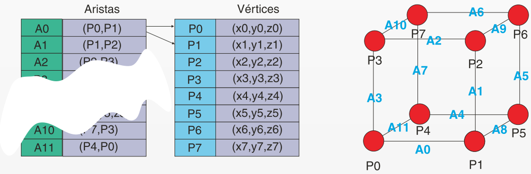


Figura : Un cubo representado con las aristas.

Modelo de superficies (cuadriláteros)

```
int Quads[30][4];

Quads[0][0]=0;
Quads[0][1]=3;
Quads[0][2]=8;
Quads[0][3]=9;
...
```

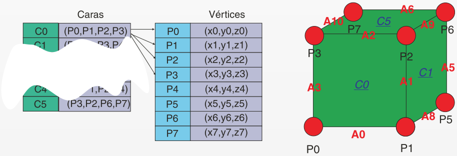


Figura : Un cubo representado con las caras.

Modelo de superficies

Una representación muy usada en gráficos se compone de los vértices, aristas y caras

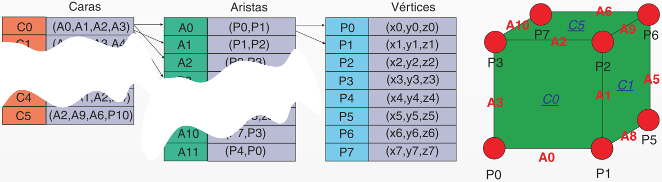


Figura : Un cubo representado con las caras, las aristas y los vértices.

Representación de los objetos

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

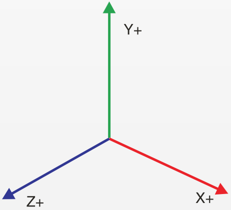
```
_vertex3f Vertex;

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

vector<_vertex3f> Vertices ;
vector<_vertex3i> Triangles ;
...
```

Sistema de coordenadas

- La definición de la geometría se tiene que hacer con respecto a un sistema de coordenadas
  - Coordenadas cartesianas
  - Coordenadas polares
  - Coordenadas cilíndricas
- El sistema de coordenadas en el que se definen los objetos se llama Sistema de Coordenadas de Mundo
- Se eligen las unidades que sean más apropiadas para el usuario



Visualización de los modelos

- Importante: crear el modelo es distinto de dibujar el modelo
- Crear el modelo: una sola vez, generar vectores de geometría y topología
- Dibujar el modelo: utilizar los vectores de geometría y topologia para dibujar

Visualización del modelo de puntos

```
for( i= 0 ; i < 10 ; i++ ){
   draw_vertex( Vertices[i][0], Vertices[i][1], Vertices[i][2] ) ;
}
```

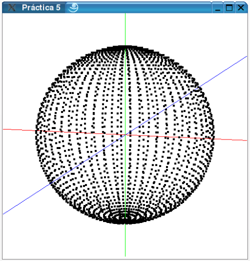


Figura : Esfera dibujada con puntos.

Visualización del modelo de alambres

```
for( i=0 ; i<20 ; i++ ) {
   Vertex_1 = Edges[i][0] ;
   Vertex_2 = Edges[i][1] ;
   draw_edge( Vertices[Vertex_1], Vertices[Vertex_2] ) ;
}
```

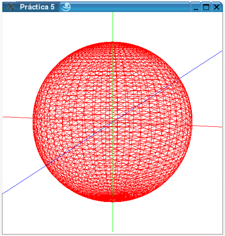


Figura : Esfera dibujada con aristas.

Visualización del modelo de superficies (triángulos)

```
for( i= 0 ; i < 30 ; i++ ){
   Vertex_1 = Triangles[i][0] ;
   Vertex_2 = Triangles[i][1] ;
   Vertex_3 = Triangles[i][2] ;
   draw_triangle( Vertices[Vertex_1], Vertices[Vertex_2] ,
       Vertices[Vertex_3] ) ;
}
```

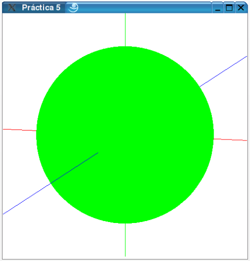


Figura : Esfera dibujada con triángulos.

Visualización de los modelos

- Se puede comprobar que con el color sólido se pierde la sensación de volumen.
- Solución
  - Dibujar las aristas
  - Incluir la iluminación
  - Dibujar en modo ajedrez (2 colores)

```
for( i=0 ; i<30 ; i++ ) {
   if ( i%2 == 0 ) set_color( Color1 ) ;
   else set_color(Color2);

   Vertex_1 = Triangles[i][0] ;
   Vertex_2 = Triangles[i][1] ;
   Vertex_3 = Triangles[i][2] ;
   draw_triangle( Vertices[Vertex_1], Vertices[Vertex_2],
        Vertices[Vertex_3]);
}
```

Visualización de los modelos con OpenGL

Veamos el código OpenGL para visualizar

Modelo de puntos

```
vector<_vertex3f> Vertices;
...
glBegin( GL_POINTS ) ;
for( i=0 ; i < Vertices.size() ; i++ ) {
    glVertex3f( Vertices[i].x, Vertices[i].y, Vertices[i].z );
}
glEnd() ;
```

Visualización de los modelos con OpenGL

Modelo de aristas:

```
vector<_vertex3f> Vertices;
vector<_vertex2i> Edges;
int Vertex_1,Vertex_2;
...
glBegin( GL_LINES ) ;
for( i= 0 ; i < Edges.size() ; i++ ){
   Vertex_1 = Edges[i]._0 ;
   Vertex_2 = Edges[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() ;
```

Visualización de los modelos con OpenGL

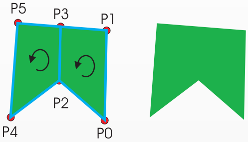
Modelo de superficies (triángulos)

```
vector<_vertex3f> Vertices;
vector<_vertex3i> Triangles;
int Vertex_1,Vertex_2,Vertex_3;
...
glBegin( GL_TRIANGLES );
for ( i= 0 ; i < Triangles.size() ; i++ ){
   Vertex_1 = Triangles[i]._0 ;
   Vertex_2 = Triangles[i]._1 ;
   Vertex_3 = Triangles[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() ;
```

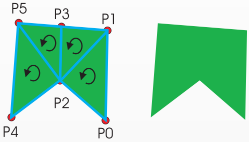
Optimización de los modelos

- Los procedimientos de dibujar puntos, aristas y caras no son eficientes: generan una llamada por primitiva o no reutilizan la información
- Solución → usar primitivas especializadas
  - Tira de cuadrilateros
  - Tira de triángulos
  - Abanico de triángulos

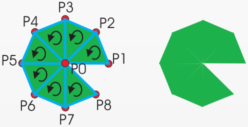
Optimización de los modelos



(a) Tira de cuadriláteros



(b) Tira de triángulos



(c) Abanico de triángulos

Figura : Primitivas gráficas especializadas.

Métodos de generación de objetos

Generación de objetos

- Escanéo
- Geometría Constructiva de Sólidos (C.S.G.)
- Parches de superficies
- Definición paramétrica
- Superficies implícitas
- Barrido
    - Dada una curva o superficie, se desplaza siguiendo una trayectoria
        - Circular
        - Lineal
        - Curva 3D

Generación por barrido

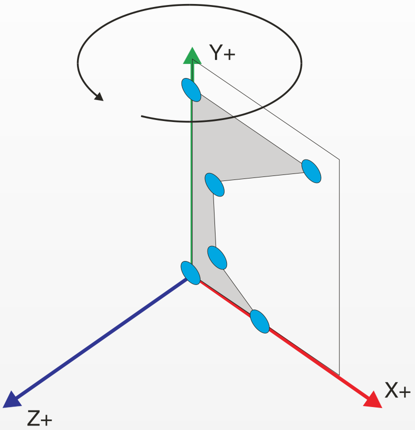


Figura : Barrido circular.

Generación por barrido

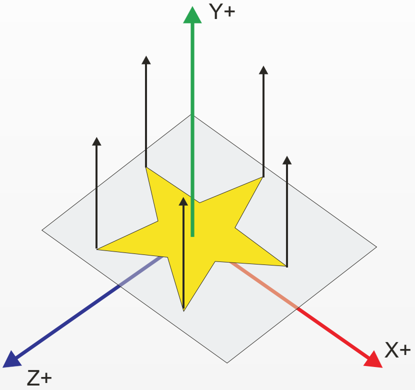


Figura : Barrido lineal.

Generación por barrido

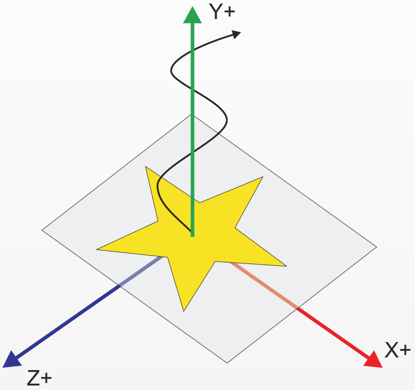


Figura : Barrido siguiendo una curva 3D.

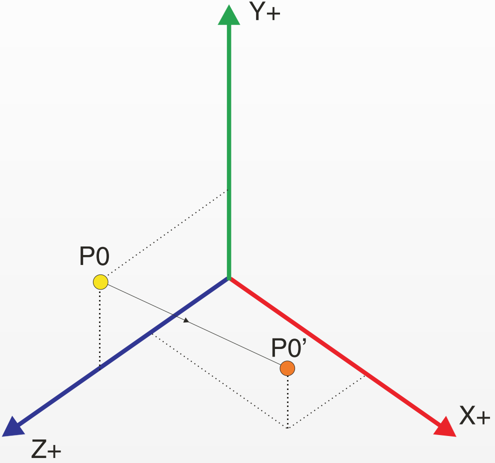
Transformaciones geométricas

Los objetos pueden ser modificados mediante transformaciones geométricas

- Traslaciones
- Escalados
- Rotaciones
- Deslizamientos
- Reflejos...

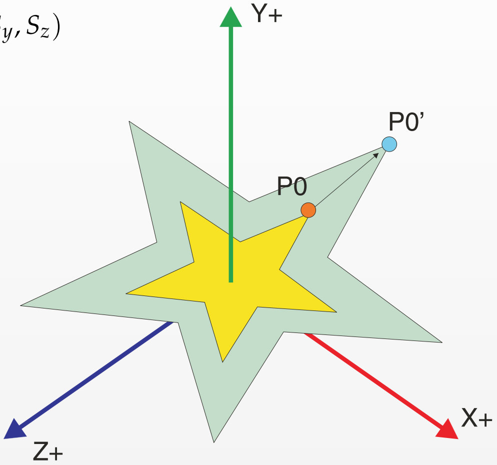
Translaciones

- $P' = T(P)$
- $x' = x + \delta x$
- $y' = y + \delta y$
- $z' = z + \delta z$



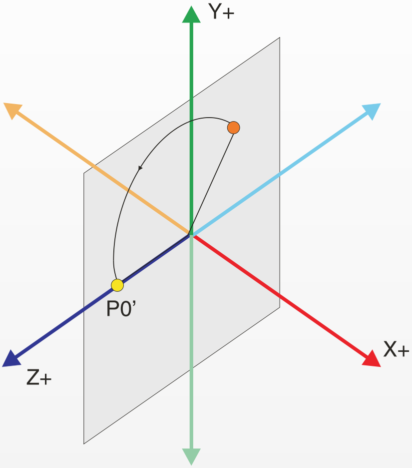
Escalados

- $P' = S(P)$ con $S = (S_x, S_y, S_z)$
- $x' = x * S_x$
- $y' = y * S_y$
- $z' = z * S_z$



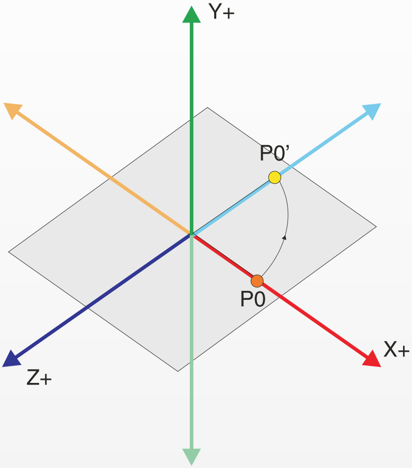
Rotación eje X

- $P' = R(P)$ con $R = \alpha$
- $x' = x$
- $y' = y * \cos(\alpha) - z * \sin(\alpha)$
- $z' = y * \sin(\alpha) + z * \cos(\alpha)$



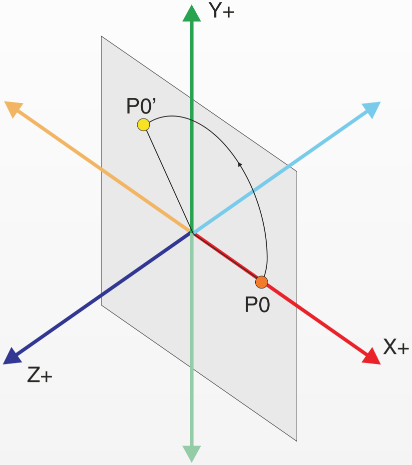
Rotación eje Y

- $P' = R(P)$ con $R = \alpha$
- $x' = x * \cos(\alpha) + z * \sin(\alpha)$
- $y' = y$
- $z' = -x * \sin(\alpha) + z * \cos(\alpha)$



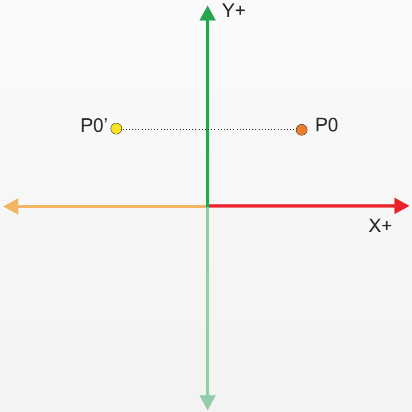
Rotación eje Z

- $P' = R(P)$ con $R = \alpha$
- $x' = x * \cos(\alpha) - y * \sin(\alpha)$
- $y' = x * \sin(\alpha) + z * \cos(\alpha)$
- $z' = z$



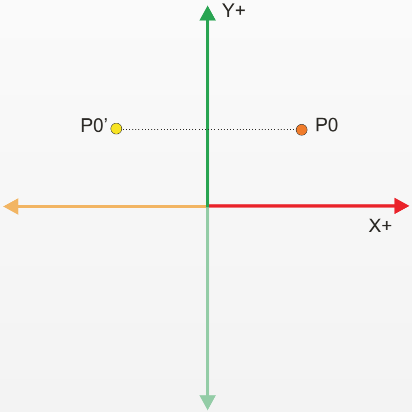
Reflejo eje X

- $x' = -x$
- $y' = y$



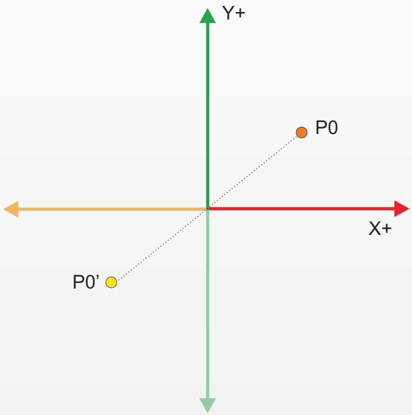
Reflejo eje Y

- $x' = x$
- $y' = -y$



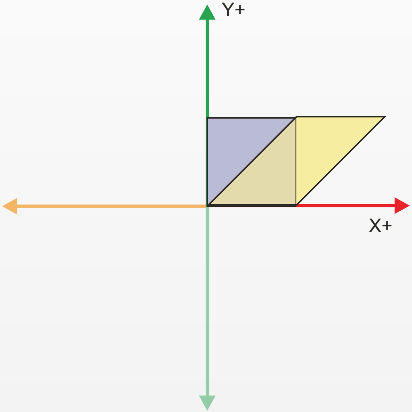
Reflejo bisectriz

- $x' = -x$
- $y' = -y$



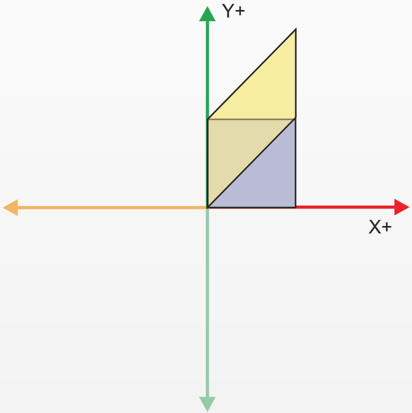
Deslizamiento eje X

- $x' = x + y * F_x$
- $y' = y$



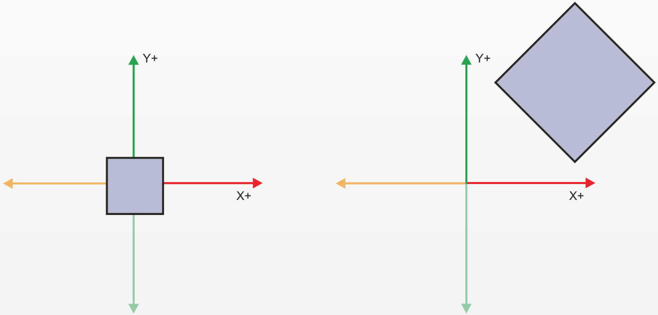
Deslizamiento eje Y

- $x' = x$
- $y' = y + x * F_y$



Transformación de modelado

- Al conjunto de transformaciones geométricas que permiten posicionar, orientar y cambiar de tamaño a un objeto se le denomina transformación de modelado



Transformación de modelado

- Normalmente se aplica más de una transformación geométrica
- Si se aplica paso a paso cada transformación $\rightarrow$ Ineficiente

$$V' = T_1(V) \rightarrow V'' = T_2(V') \rightarrow V''' = T_3(V'')$$

- ¿Existe la posibilidad de obtener una trasformación que sea igual a la combinación de las transformaciones?
- Sí $\rightarrow$ Ineficiente

$$T = T_1 \otimes T_2 \otimes T_3$$
$$V''' = T(V)$$

Transformación mediante matrices

- Con la utilización de la formulación explícita para las transformaciones, la codificación de la combinación es engorrosa
- Solución $\rightarrow$ escritura matricial
- Escribir las transformaciones como matrices $\rightarrow$ ventajas
  - Álgebra de matrices
  - Composición de transformaciones muy fácil
  - Transformación inversa

Transformación mediante matrices

- Transformaciones

$$T = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$$

- Vértices

$$V = \begin{bmatrix} x & y & z \end{bmatrix} \quad o \quad V = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Transformación mediante matrices

- Transformar = multiplicar el vector que representa al vértice por la matriz que representa la transformacion
- $V' = V * T$

$$\begin{bmatrix} x' & y' & z' \end{bmatrix} = \begin{bmatrix} x & y & z \end{bmatrix} * \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$$

- Operando

$$\begin{aligned} x' &= x * a + y * d + z * g \\ y' &= x * b + y * e + z * h \\ z' &= x * c + y * f + z * i \end{aligned}$$

Transformación mediante matrices

- El resultado de la transformación mediante matrices tiene que ser igual a la formulación explicita
- Escalado

$$\begin{array}{l} x' = x * S_x \\ y' = y * S_y \\ z' = z * S_z \end{array} \quad = \quad \begin{array}{l} x' = x * a + y * d + z * g \\ y' = x * b + y * e + z * h \\ z' = x * c + y * f + z * i \end{array}$$

$$\downarrow$$

$$\begin{array}{l} x * S_x = x * a + y * d + z * g \\ y * S_y = x * b + y * e + z * h \\ z * S_z = x * c + y * f + z * i \end{array} \quad \rightarrow \quad \begin{array}{lll} a = S_x & b = 0 & c = 0 \\ d = 0 & e = S_y & f = 0 \\ g = 0 & h = 0 & i = S_z \end{array}$$

Transformación mediante matrices

Rotación eje X
$$\begin{bmatrix} a=1 & b=0 & c=0 \\ d=0 & e=\cos(\alpha) & f=\sin(\alpha) \\ g=0 & h=-\sin(\alpha) & i=\cos(\alpha) \end{bmatrix}$$

Rotación eje Y
$$\begin{bmatrix} a=\cos(\alpha) & b=0 & c=-\sin(\alpha) \\ d=0 & e=1 & f=0 \\ g=\sin(\alpha) & h=0 & i=\cos(\alpha) \end{bmatrix}$$

Rotación eje Z
$$\begin{bmatrix} a=\cos(\alpha) & b=\sin(\alpha) & c=0 \\ d=-\sin(\alpha) & e=\cos(\alpha) & f=0 \\ g=0 & h=0 & i=1 \end{bmatrix}$$

Transformación mediante matrices

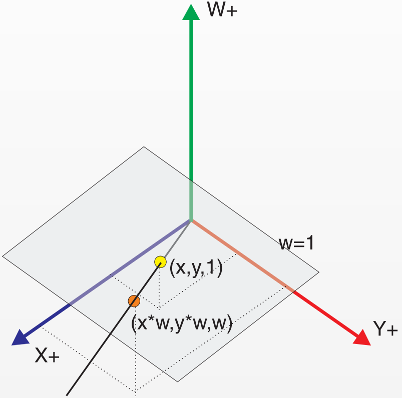
$$\text{Translación} \quad \begin{bmatrix} a = 1 & b = 0 & c = 0 \\ d = 0 & e = 1 & f = 0 \\ g = \frac{\delta_x}{z} & h = \frac{\delta_y}{z} & i = 1 + \frac{\delta_z}{z} \end{bmatrix}$$

- No se puede: hay que calcular una matriz para cada vértice
- Solución $\rightarrow$ Coordenadas Homogéneas

Coordenadas homogéneas

- Consiste en añadir una coordenada más
- Se pasa de un espacion n dimensional a otro $n+1$ dimensional
- Nos interesa 3D $\rightarrow$ 4D
- Más fácil de visualizar el paso de 2D a 3D
- Procedimiento para pasar de 2D a 3D:
  - $(x, y) \rightarrow (x', y', w)$ con $w \neq 0$
    $x' = x * w$
    $y' = y * w$
  - Ejemplo: $(2, 3) \rightarrow (4, 6, 2), (8, 12, 4), (20, 30, 10), \ldots$
  - Si $w = 1$ entonces $(x, y) \rightarrow (x, y, 1)$
- Procedimiento para pasar de 3D a 2D:
  - Si $w \neq 0$ $(x', y', w) \rightarrow (x, y, 1) \rightarrow (x, y)$
    $x = x'/w$
    $y = y'/w$
  - Si $w = 0$ entonces el punto está en el infinito

Coordenadas homogéneas



Coordenadas homogéneas

Rotación eje X $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha) & \sin(\alpha) & 0 \\ 0 & -\sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Rotación eje Y $\begin{bmatrix} \cos(\alpha) & 0 & -\sin(\alpha) & 0 \\ 0 & 1 & 0 & 0 \\ \sin(\alpha) & 0 & \cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Rotación eje Z $\begin{bmatrix} \cos(\alpha) & \sin(\alpha) & 0 & 0 \\ -\sin(\alpha) & \cos(\alpha) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Coordenadas homogéneas

Escalado
$$\begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

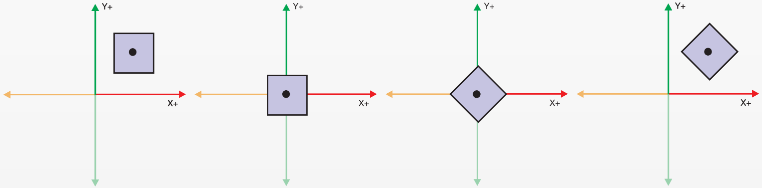
Translación
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \delta_x & \delta_y & \delta_z & 1 \end{bmatrix}$$

Coordenadas homogéneas

- Mediante la notación matricial se regulariza la composición de transformaciones
- Componer=Multiplicar
  - $T = T_1 \otimes T_2 \otimes T_3$
  - $T = M_1 * M_2 * M_3$ con $T_1 = M_1 \quad T_2 = M_2 \quad T_3 = M_3$
- Transformación inversa = inversa de la matriz de transformación
  - $T^{-1} = M^{-1}$

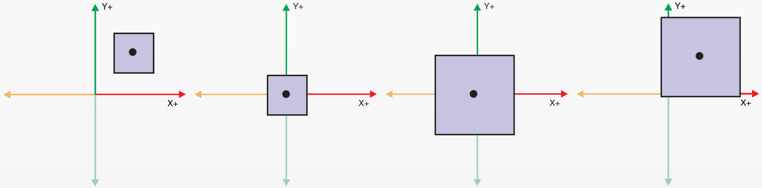
Composición de transformaciones

- Rotación con respecto a un punto pivote
  - La rotación se hace respecto al origen
  - ¿Y si se quiere rotar alrededor de otro punto P ?
  - Solución: convertir el punto P en el origen
  - $T = T(-P) * R(\alpha) * T(P)$



Composición de transformaciones

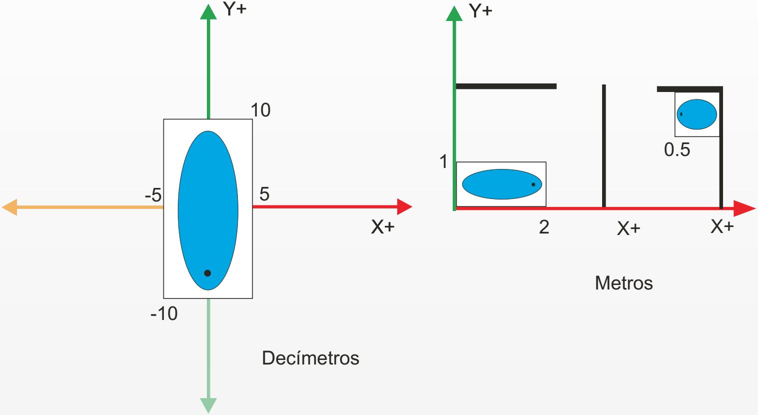
- Escalado con respecto a un punto arbitrario
  - El escalado se hace respecto al origen
  - ¿Y si se quiere escalar con respecto a otro punto P ?
  - Solución: convertir el punto P en el origen
  - $T = T(-P) * S(F_s) * T(P)$



Transformación de instancia

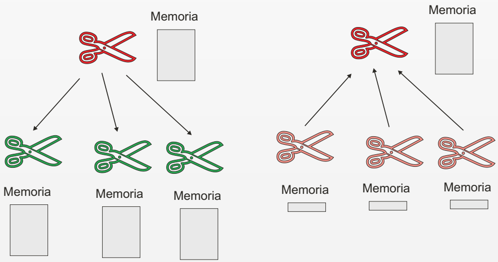
- Dado un objeto (símbolo) definido en su propio sistema de coordenadas, se desea obtener una copia en otro sistema de coordenadas (instancia)
- El objeto se define en el Sistema de Coordenadas Maestras Se eligen unidades apropiadas para el diseño del objeto
- Transformación de instancia
  - Escalado $\rightarrow$ cambio de escala
  - Rotación
  - Translación

Transformación de instancia



Transformación de instancia

- Es importante distinguir entre:
  - Hacer copias
    - Los cambios sobre el símbolo no afectan a las instancias
    - Ocupa mucha memoria
  - Hacer referencias
    - La instancia existe temporalmente
    - Los cambios sobre el símbolo afectan a las instancias
    - Ocupa poca memoria



Jerarquización

- Normalmente los modelos presentan una estructura jerárquica de dependencias
- La jerarquización consiste en construir objetos complejos a partir de otros sencillos.
- Para ello se establecen relaciones de dependencia:
  - P.E.: Abuelos $\rightarrow$ padres $\rightarrow$ hijos
  - P.E.: Mover el brazo $\rightarrow$ mover la mano $\rightarrow$ mover los dedos
- Un objeto de un nivel n es construido con elementos del nivel n-1
- Se implementa mediante la transformaciones de instancia y la herencia
- Se realiza mediante el mecanismo de la pila
- Este tipo de estructuras se modelizan mediante grafos orientados

Jerarquización

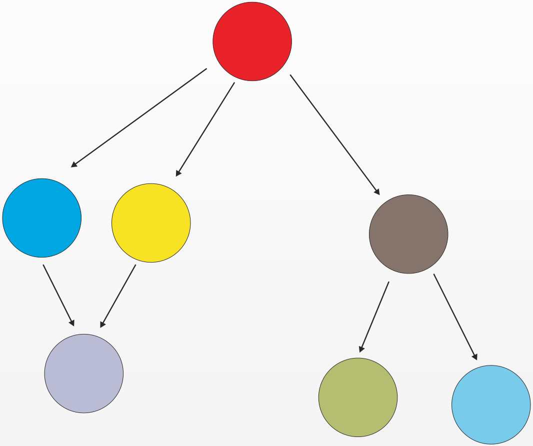


Figura : Estructura de grafo.

Jerarquización

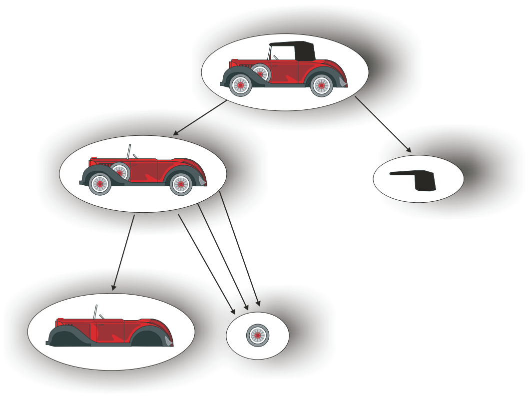


Figura : Construcción de un objeto complejo a partir de otros más sencillos.

Jerarquización

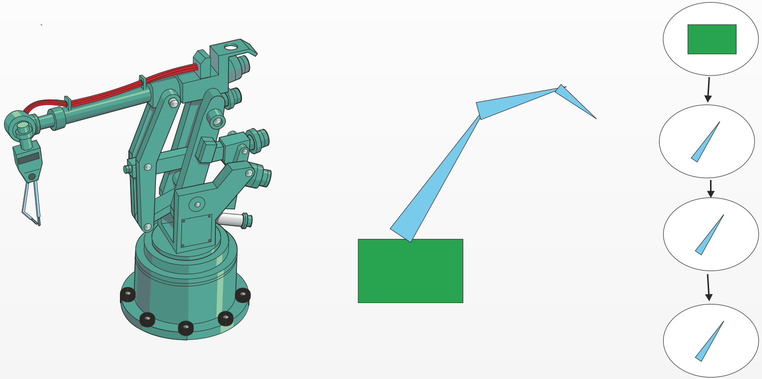


Figura : Jerarquía de movimiento.

Jerarquización

```
dibujar_ciudad()
{
   transformacion(T1)
   dibujar_cubo();
   transformacion(T2)
   dibujar_cubo();
   transformacion(T3)
   dibujar_cubo();
   ...
}
```

```
dibujar_rueda()
{
   transformacion(T1)
   dibujar_cilindro();
}
```

Jerarquización

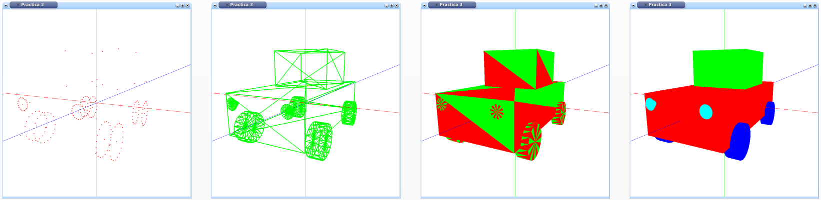


Figura : Coche construido a partir de un cubo, un clindro y un cono, en los distintos modos de visualización.

Jerarquización

- Mediante los simbolos, las transformaciones y la jerarquización estamos modelando
- ¿Cómo se consigue que los objetos se muevan y cambien?
  $\rightarrow$ Animación $\rightarrow$ cambios con el tiempo
- Los cambios se realizan mediante transformaciones cuyo parámetro o parámetros se modifican con el tiempo

```
dibujar_rueda_girando()
{
   transformacion(Rotacion(tiempo))
   dibujar_rueda();
}
```

Jerarquización con OpenGL

- Las transformaciones se combinan en la pila GL_MODELVIEW
- Las transformaciones de un objeto pueden afectar a otro

```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity() ;
... // las transformaciones de la camara
glTranslatef(1,1,1) ; // T1
glRotatef(45,1,0,0) ; //T2
Objeto1() ;
glTranslatef(5,5,5) ; //T3
glScalef(2,2,2) ; // T4
Objeto2() ;
...
```

Jerarquización con OpenGL

- No se puede solucionar con la elimación de las transformaciones
- Solución → usar las características de la pila
- Dos instrucciones: glPushMatrix y glPopMatrix

```
glPushMatrix
  (TOP+1)=(TOP) // se copia el contenido
  TOP=TOP+1 // se actualiza el TOP

glPopMatrix
  TOP=TOP-1 // se actualiza el TOP
```

Jerarquización con OpenGL

- La pila permite el "paso" de transformaciones de un objeto a otro
- La pila aisla las transformaciones privadas de las que se pasan
- La pila permite la jerarquización

```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity() ;
... // las transformaciones de la camara
glPushMatrix
  glTranslatef(1,1,1) ; // T1
  glRotatef(45,1,0,0) ; //T2
  Objeto1() ;
glPopMatrix
glPushMatrix
  glTranslatef(5,5,5) ; //T3
  glScalef(2,2,2) ; // T4
  Objeto2() ;
glPopMatrix
...
```

Jerarquización con OpenGL

```
wheel(){
glPushMatrix
  glRotatef(90,0,0,1) ; //T1
  glScalef(1,0.1,1) ; // T2
  cilynder() ;
glPopMatrix
}
```

```
car(){
glPushMatrix
  glTranslatef(1,0,1) ;
  wheel() ; // rueda 1
glPopMatrix
glPushMatrix
  glTranslatef(-1,0,1) ;
  wheel() ; // rueda 2
glPopMatrix
...
}
```

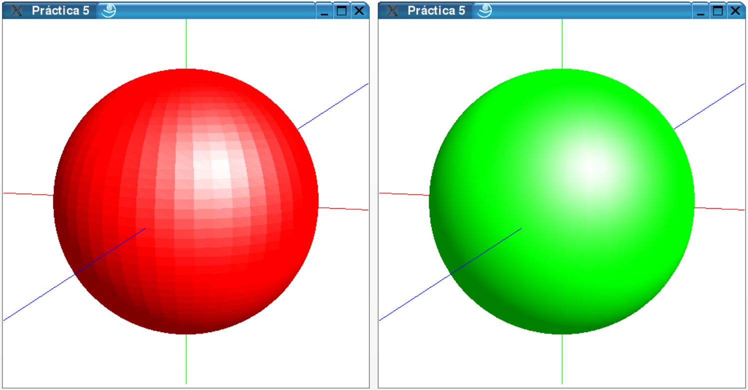
Cálculos y formatos

- Operaciones sobre modelos:
  - medir su superficie
  - medir su volumen
  - calcular su peso
  - encontrar el camino más corto entre dos puntos recorriendo su superficie
  - ...
- Solución $\rightarrow$ utilizar la geometría y la topología
- Crear modelos más complejos. Por ejemplo el de aristas aladas (**winged edged**)

Cálculos y formatos

- Estamos interesados en calcular las normales de las caras y de los vértices
- La normales se utilizan para la iluminación
- Normales de las caras $\rightarrow$ suavizado plano
- Normales de los vértices $\rightarrow$ suavizado de Gouraud

Cálculos y formatos



(a) Suavizado plano (b) Suavizado Gouraud

Figura : Suavizado plano y de Gouraud sobre una esfera.

Cálculos y formatos

Cálculo de las normales de las caras

- Dados $\mathbf{v}_0$, $\mathbf{v}_1$ y $\mathbf{v}_2$
- se obtienen $\mathbf{a} = \mathbf{v}_1 - \mathbf{v}_0$ y $\mathbf{b} = \mathbf{v}_2 - \mathbf{v}_0$
- $\mathbf{n} = \mathbf{a} \times \mathbf{b}$

- El vector $\mathbf{n}$ apunta según la regla de la mano derecha

Cálculos y formatos

Cálculo de las normales de los vértices

- El suavizado plano produce volumen pero no es realista para superficies curvas
- Solucion:
  - aumentar el número de caras
  - Interpolar → suavizado de Gouraud → cálculo de las normales en los vértices

$$\mathbf{n} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{n}_i$$

Cálculos y formatos

- Modelar suele ser una tarea lenta y dificultosa
- Solucion $\rightarrow$ usar programas especializados $\rightarrow$ exportar el modelo
- Formato: qué datos se almacenan y cómo se almacenan
- Muchos formatos, la mayoría propietarios y cerrados
- PLY formato abierto muy sencillo

Cálculos y formatos

Formato básico de un fichero PLY

```
Identificacion
Cabecera con descripcion de los datos almacenados
Datos
```

Cálculos y formatos

```
ply
format ascii 1.0
element vertex 8
property float x
property float y
property float z
element face 12
property list uchar vertex_indices
end_header
0 0 0
1 0 0
1 1 0
0 1 0
0 0 1
1 0 1
1 1 1
0 1 1
```

```
3 1 0 2
3 0 3 2
3 0 4 3
3 4 7 3
3 4 5 7
3 5 6 7
3 5 1 6
3 1 2 6
3 2 3 6
3 3 7 6
3 0 1 4
3 1 5 4
```