Dpto. de Lenguajes y Sistemas Informáticos
Escuela Técnica Superior de Ingenierías Informática y Telecomunicación

Prácticas de Informática Gráfica

Autores:
Pedro Cano
Antonio López
Domingo Martín
Juan carlos Torres
Carlos Ureña

Curso 2013/14

La Informática Gráfica

La gran ventaja de los gráficos por ordenador, la posibilidad de crear mundos virtuales sin ningún tipo de límite, excepto los propios de las capacidades humanas, es a su vez su gran inconveniente, ya que es necesario crear toda una serie de modelos o representaciones de todas las cosas que se pretenden obtener que sean tratables por el ordenador.

Así, es necesario crear modelos de los objetos, de la cámara, de la interacción de la luz (virtual) con los objetos, del movimiento, etc. A pesar de la dificultad y complejidad, los resultados obtenidos suelen compensar el esfuerzo.

Ese es el objetivo de estas prácticas: convertir la generación de gráficos mediante ordenador en una tarea satisfactoria, en el sentido de que sea algo que se hace "con ganas".

Con todo, hemos intentado que la dificultad vaya apareciendo de una forma gradual y natural. Siguiendo una estructura incremental, en la cual cada práctica se basará en la realizada anteriormente, planteamos partir desde la primera práctica, que servirá para tomar un contacto inicial, y terminar generando un sistema de partículas con animación y detección de colisiones.

Esperamos que las prácticas propuestas alcancen los objetivos y que sirvan para enseñar los conceptos básicos de la Informática Gráfica, y si puede ser entreteniendo, mejor.

Índice general

Práctica 1

Introducción. Visualización de modelos PLY

1.1. Objetivos

Con esta práctica se quiere que el alumno aprenda:

- A utilizar las primitivas de dibujo de OpenGL
- A distinguir la diferencia entre definir un modelo en código y definirlo mediante datos
- A crear estructuras de datos apropiadas para su uso en visualización de modelos gráficos
- A leer modelos guardados en ficheros externos y su visualización, en concreto en formato PLY

1.2. Desarrollo

Para el desarrollo de esta práctica se entrega el esqueleto de una aplicación gráfica basada en eventos, mediante glut, y con la parte gráfica realizada por OpenGL. Para facilitar su uso, la aplicación permite abrir una ventana, mostrar unos ejes y mover una cámara básica. Así mismo, se incluye el código básico para dibujar los vértices de un cubo unidad. Se entrega también el código de un lector básicos de ficheros PLY compuestos únicamente por triángulos.

El alumno estudiará el código. En particular debe fijarse en el código que permite dibujar los 8 vértices. Se debe comprobar la rigidez que impone el definir el modelo mediante código (¿qué pasa si en vez de un cubo es un dodecaedro?). Se debe entender que la solución pasa por independizar la visualización de la definición de modelo. Reflexionar sobre esto pues es un concepto muy importante.

El alumno debe encontrar una solución que permita una mayor flexibilidad. Una vez entendido e implementado, se hará uso del código que permite leer un fichero PLY, visual-

izándolo. Es probable que se vea la necesidad de crear estructuras de datos que faciliten el manejo de los modelos y sus datos. Por ejemplo, en vez de tener 3 flotantes para definir las coordenadas de un vértice, utilizar una estructura que los convierta en una entidad (ver el código vertex.h como ejemplo).

Finalmente se creará el código que permita visualizar el modelo con los siguiente modos:

- Alambre
- Sólido
- Ajedrez

Para poder visualizar en modo alambre (también para el modo sólido y ajedréz) lo que se hace es mandar a OpenGL como primitiva los triángulos, `GL_TRIANGLES`, y cambiar la forma en la que se visualiza el mismo mediante la instrucción `glPolygonMode`, permitiendo el dibujar los vértices, las aristas o la parte sólida.

Para el modo ajedrez basta con dibujar en modo sólido pero cambiando alternativamente el color de relleno.

1.3. Evaluación

Para evaluación de la práctica se tendrán en cuentan los siguientes items que deberán cumplirse (sobre 10):

- Creación del código que permite visualizar un vector de flotantes en modo puntos (2pt)

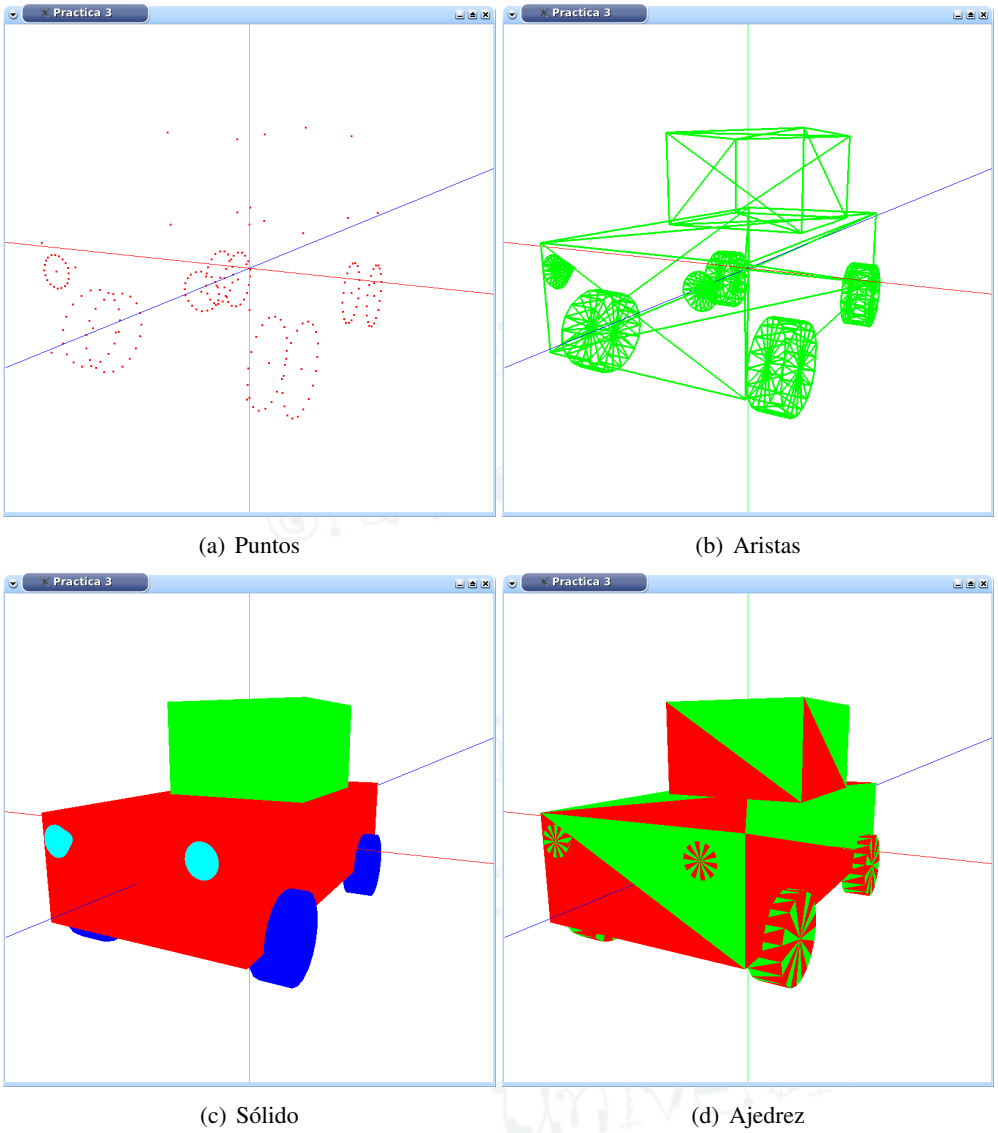
- Lectura de un fichero PLY y visualización en modo puntos (4 pt.)
  Se leerar el fichero PLY y dados los vectores de coordenadas de los vértices y el vector de los indices de vértices, se creará una estructura de datos que guarde los vértices y las caras (se recomienda usar STL y el fichero auxiliar vertex.h).

- Creación del código que permite visualizar en los modos alambre, sólido y ajedrez. (4pt)
  Creación de los procedimientos para visualizar en los distintos modos.

1.4. Extensiones

1.5. Duración

La práctica se desarrollará en 2 sesiones



(a) Puntos (b) Aristas

(c) Sólido (d) Ajedrez

Figura 1.1: Coche mostrado con los distintos modos de visualización.

1.6. Bibliografía

- Mark Segal y Kurt Akeley;*The OpenGL Graphics System: A Specification (version 4.1)*; http://www.opengl.org/
- Edward Angel;*Interactive Computer Graphics. A top-down approach with OpenGl*; Addison-Wesley, 2000
- J. Foley, A. van Dam, S. Feiner y J. F. Hughes; *Computer Graphics: Principles And Practice, 2 Edition*; Addison-Wesley,1992

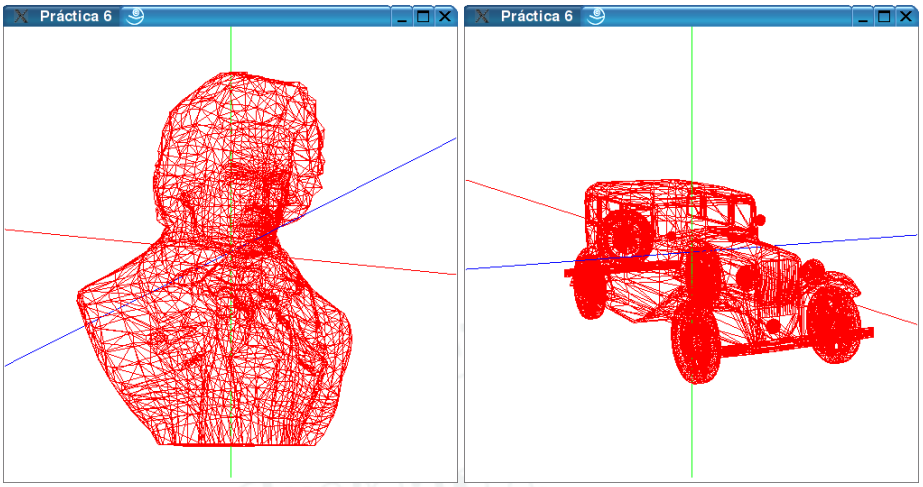


Figura 1.2: Objetos PLY.

- M. E. Mortenson; *Geometric Modeling*; John Wiley & Sons,1985

Práctica 2

Modelos Poligonales

2.1. Objetivos

Aprender a:

- Modelar objetos sólidos poligonales mediante técnicas sencillas. En este caso se usará la técnica de modelado por revolución de un perfil alrededor de un eje de rotación
- Realizar cálculos geométricos sobre las caras y vértices de un modelo poligonal

2.2. Desarrollo

En primer lugar, se ha de crear un código que a partir del conjunto de puntos que representen un perfil respecto a un plano principal ($X = 0$, $Y = 0$, $Z = 0$), de un parámetro que indique el número de lados longitudinales del objeto a generar por revolución y de un eje de rotación, calcule el conjunto de vértices y el conjunto el caras que representan el sólido obtenido.

Veamos los pasos para crear un sólido por revolución.

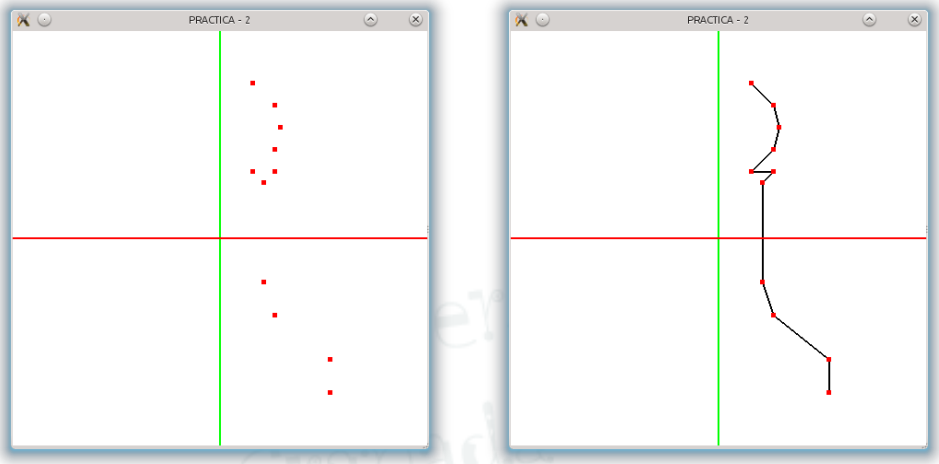
- Sea, por ejemplo, un perfil inicial Q_1 en el plano $Z = 0$ definido como:

$$Q_1\left(p_1(x_1, y_1, 0), ..., p_M(x_M, y_M, 0)\right),$$

  siendo $p_i(x_i, y_i, 0)$ con $i = 1, ...M$ los puntos que definen el perfil (ver figura 2.1).

- Se toma como eje de rotación el eje Y y si N es número lados longitudinales, se obtienen los puntos o vértices del sólido poligonal a construir multiplicando Q_1 por N sucesivas transformaciones de rotación con respecto al eje Y, a las que notamos por $R_Y(\alpha_j)$ siendo α_j los valores de N ángulos de rotación equiespaciados. Se obtiene un conjunto de $N \times M$ vértices agrupados en N perfiles Q_j, siendo:

$$Q_j = Q_1 R_Y(\alpha_j), \quad \text{con } j = 1, \ldots, N$$



(a) Puntos (b) Polilínea

Figura 2.1: Perfil inicial.

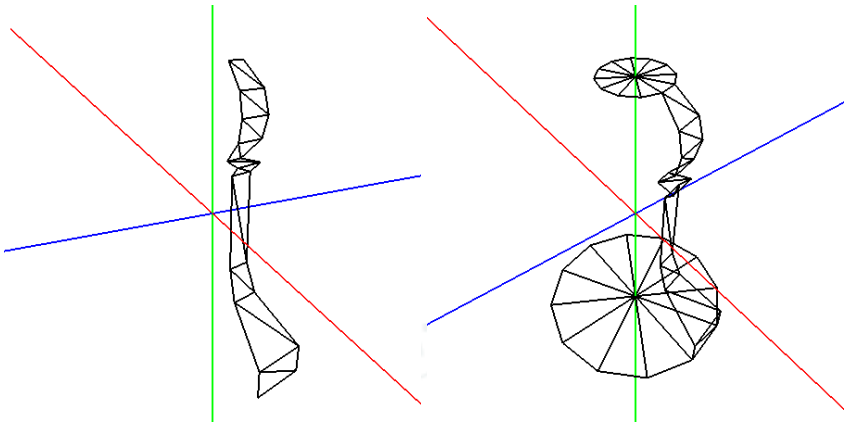
- Se guardan $N \times M$ los vértices obtenidos en un vector de vértices según la estructura de datos creada en la práctica anterior.
- Las caras longitudinales del sólido (triángulos) se crean a partir de los vértices de dos perfiles consecutivos Q_j y Q_{j+1}. Tomando dos puntos adyacentes en cada uno de los dos perfiles Q_j y Q_{j+1} y estando dichos puntos a la misma altura, se pueden crear dos triángulos. En la figura 2.2(a) se muestran los triángulos así obtenidos solamente para un lado longitudinal para una mejor visualización. Los vértices de los triángulos tienen que estar ordenados en el sentido contrario a las agujas del reloj.
- A continuación creamos las tapas del sólido tanto inferior como superior (ver figura 2.2(b)). Para ello se han de añadir dos puntos al vector de vértices que se obtienen por la proyección sobre el eje de rotación del primer y último punto del perfil inicial. Estos dos vértices serán compartidos por todas las caras de las tapas superior e inferior.
- Todas las caras, tanto las longitudinales como las tapas superior e inferior, se almacenan en la estructura de datos creada para las caras en la práctica anterior.

El modelo poligonal finalmente obtenido se podrá visualizar usando cualquiera de los distintos modos de visualización implementados para la primera práctica (ver figura 2.3).

Dos consideraciones sobre la implementación del código: primera, se puede hacer un tratamiento diferenciado cuando uno o ambos puntos extremos del perfil inicial están situados sobre el eje de rotación y segunda, el perfil inicial se puede leer de un fichero PLY cuyo contenido sólo ha de tener las coordenadas de los puntos de éste (no es difícil crear manualmente un perfil con un fichero PLY, véase el siguiente ejemplo).

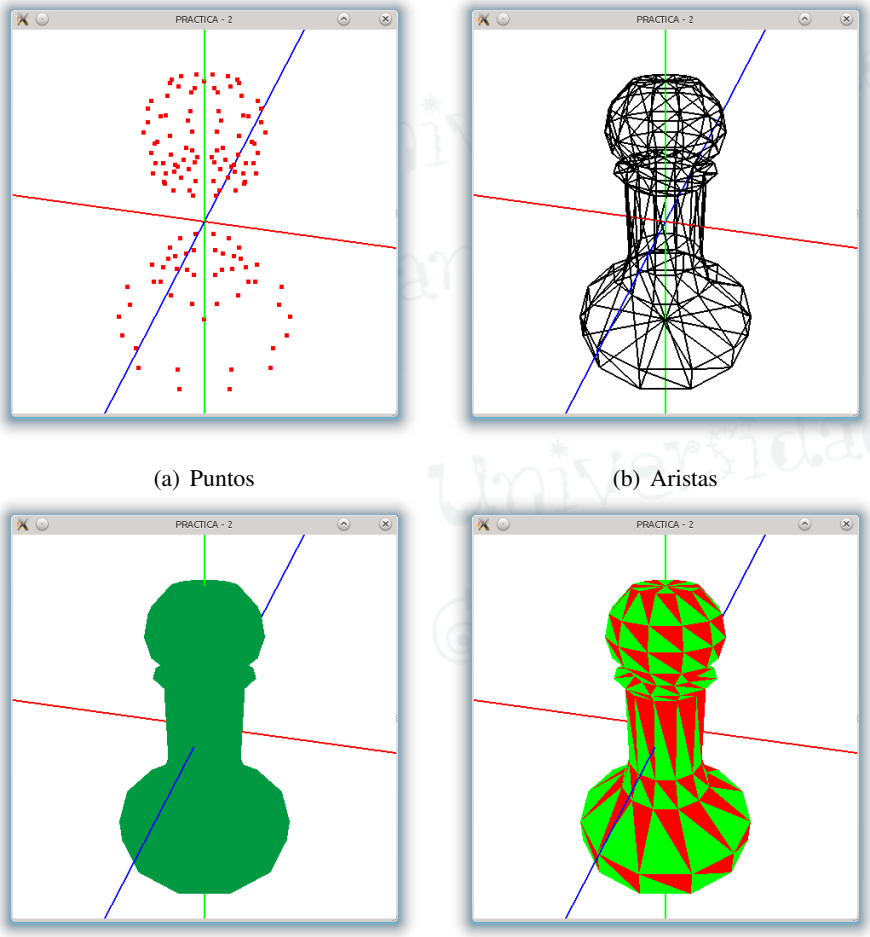
```
ply
format ascii 1.0

©
```



(a) (b)

Figura 2.2: Caras del sólido a construir: (a) logitudinales (solo un lado es mostrado) y (b) incluyendo las tapas superior e inferior.



(a) Puntos (b) Aristas

(c) Sólido (d) Ajedrez

Figura 2.3: Sólido generado por revolución con distintos modos de visualización.

```
element vertex 11
property float32 x
property float32 y
property float32 z
end_header
1.0 -1.4 0.0
1.0 -1.1 0.0
0.5 -0.7 0.0
0.4 -0.4 0.0
0.4 0.5 0.0
0.5 0.6 0.0
0.3 0.6 0.0
0.5 0.8 0.0
0.55 1.0 0.0
0.5 1.2 0.0
0.3 1.4 0.0
```

En segundo lugar, como cálculo geométrico sobre el modelo, se ha de realizar código para obtener los vectores normales a las caras y los vectores normales a los vértices. Estos vectores normales se utilizarán en la práctica dedicada a iluminación.

- A partir de la lista de vértices y de caras del objeto sólido, se calculan los vectores normales de todas las caras. Dada una cara triangular constituida por vértices A, B y C ordenados en el sentido contrario a las agujas del reloj, si se definen dos vectores $\vec{AB}$ y $\vec{BC}$ un vector normal al plano que contiene a la cara $\vec{N}$ se obtiene mediante la expresión $\vec{N} = \vec{AB} \otimes \vec{BC}$, donde $\otimes$ representa el producto vectorial de dos vectores. Un vector normal a una cara no tiene porque ser un vector normalizado o unitario. Así, los vectores normales $\vec{N}$, que han sido calculados previamente, tienen que ser normalizados.

  Se ha de crear una estructura de datos para almacenar los vectores normales a las caras.

- Los vectores normales a los vértices se calculan a partir de la lista de vectores normales a las caras que se obtiene en el paso anterior. El vector normal a un vértice v, que denotamos por $\vec{N}_v$ se obtiene como el promedio normalizado de los vectores normales de las caras que comparten dicho vértice:

$$\vec{N}_v = \frac{\sum_{i=1}^{n} \vec{N}_i}{|\sum_{i=1}^{n} \vec{N}_i|} \tag{2.1}$$

  donde n es el número de caras que se reúnen en el vértice v y $|\cdot|$ representa el módulo de un vector.

  Se ha de crear también, una estructura de datos para almacenar los vectores normales a los vértices.

2.3. Evaluación

La evaluación de la práctica, sobre 10 puntos, se hará del modo siguiente:

- Creación del código para el modelado de objetos por revolución (6 pts.).
- Creación del código para el cálculo de los vectores normales (4 pts.).

2.4. Extensiones

Se propone como extensión modelar sólidos por barrido a partir de un contorno cerrado.

2.5. Duración

La práctica se desarrollará en 3 sesiones

2.6. Bibliografía

- Mark Segal y Kurt Akeley; *The OpenGL Graphics System: A Specification (version 4.1)*; http://www.opengl.org/
- P. Shirley y S. Marschner; *Fundamentals of Computer Graphics, 3rd Edition*; A K Peters Ltd. 2009.
- J. Vince; *Mathematics for Computer Graphics*; Springer 2006.