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
Francisco J. Melero

Curso 2018/19

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. Modelado y visualización de objetos 3D sencillos

1.1. Objetivos

Con esta práctica se quiere que el alumno aprenda:

- A crear y utilizar estructuras de datos que permitan representar objetos 3D sencillos.
- A utilizar las primitivas de dibujo de OpenGL para dibujar los objetos.

1.2. Desarrollo

Para el desarrollo de esta práctica se entrega el esqueleto de una aplicación gráfica basada en eventos, mediante GLUT y Qt 5, y con la parte gráfica realizada por OpenGL. La aplicación no sólo contiene el código de inicialización de OpenGL y la captura de los eventos principales, sino que se ha implmentado una estructura de clases que permite representar objetos 3D, incluyendo unos ejes y un tetraedro. También está implementada una cámara que se mueve con las teclas de cursor, para moverse, y página adelante y página atrás para acercarse y aljarse.

El alumno deberá estudiar y comprender el código que se entrega. Hecho esto, deberá añadir la funciones que permiten dibujar en mode relleno y modo ajedres. Además deberá crear la clase cubo y utilizar una instancia que permita visualizarla cuando se apriete la tecla 2.

Por tanto, al final se dispondrá de los siguientes modos de dibujado:

- Puntos
- Líneas
- Relleno
- Ajedrez

Para poder visualizar en modo fill, sólo hay que usar como primitiva de dibujo los triángulos, `GL_TRIANGLES`, y cambiar la forma en la que se visualiza el mismo mediante la instrucción `glPolygonMode`, para que se dibuje el interior.

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

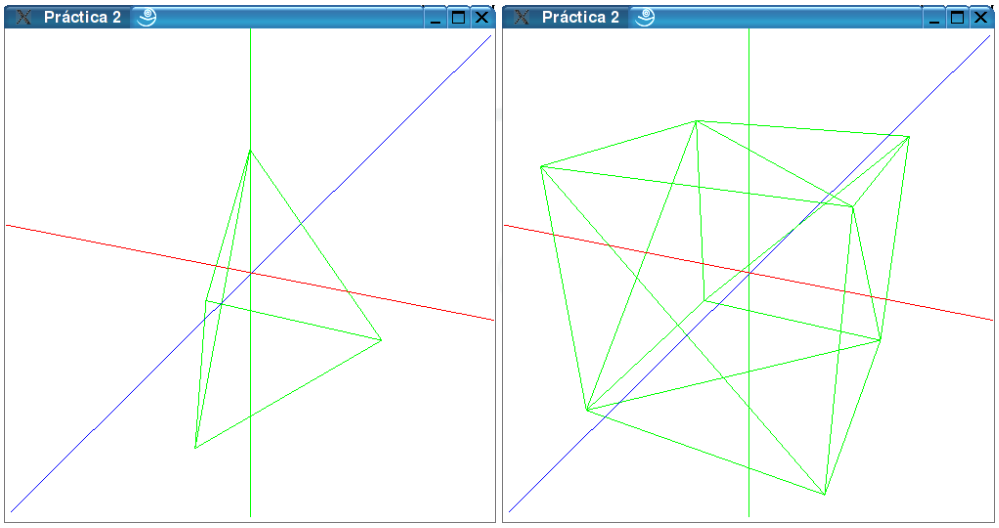
Las siguientes teclas para activar los distintos modos y objetos son las siguientes:

- Tecla p: Visualizar en modo puntos
- Tecla l: Visualizar en modo líneas/aristas
- Tecla f: Visualizar en modo relleno
- Tecla c: Visualizar en modo ajedrez
- Tecla 1: Activar tetraedo
- Tecla 2: Activar cubo

1.3. Evaluación

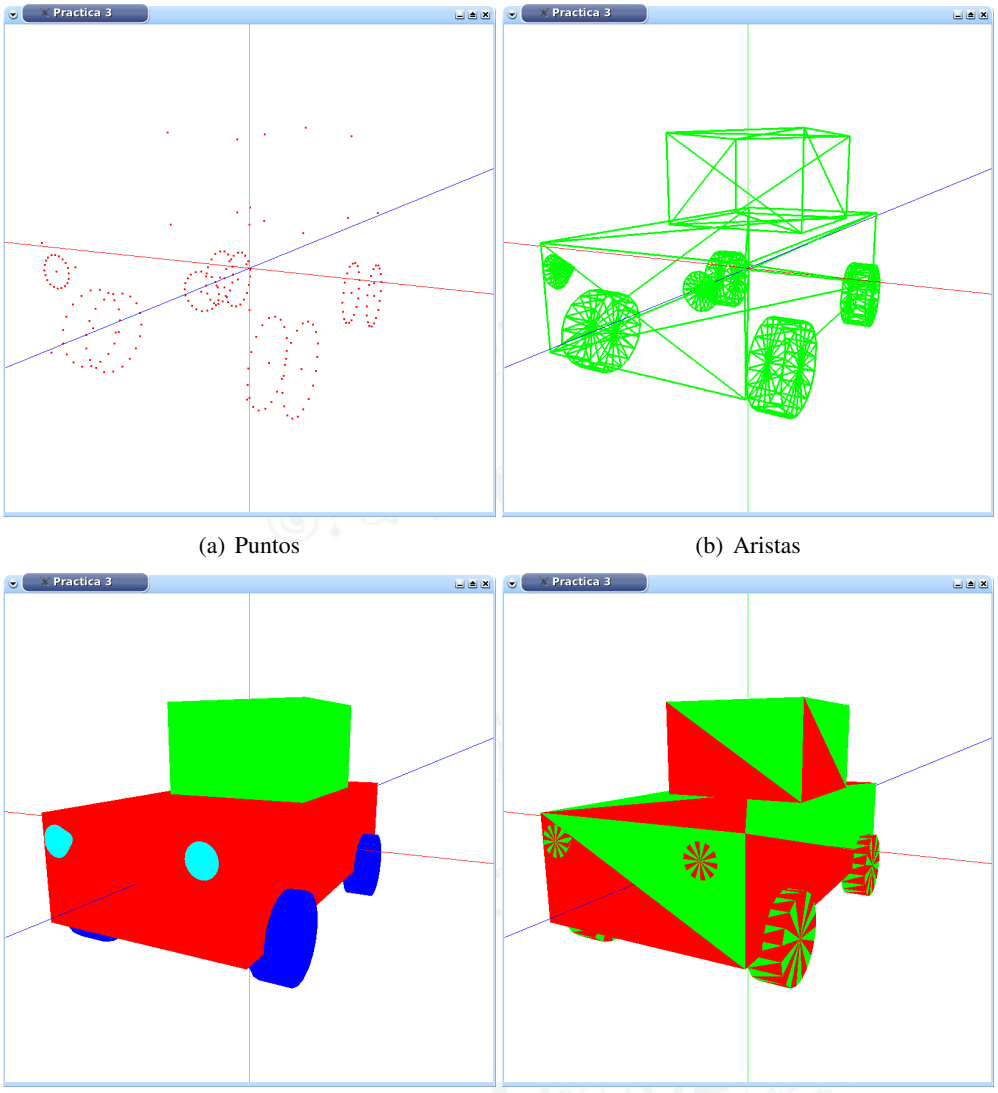
La evaluación de la práctica, sobre 10 puntos, se hará del modo siguiente:

- Creación de la clase cubo y si visualización (6 pt.)



(a) Tetraedro (b) Cubo

Figura 1.1: Tetraedro y cubo visualizados en modo alambre.



(a) Puntos (b) Aristas

(c) Sólido (d) Ajedrez

Figura 1.2: Coche mostrado con los distintos modos de visualización.

- Creación del código que permite visualizar en los modos rellenos y ajedrez. (4pt)

1.4. Duración

La práctica se desarrollará en 1 sesión

1.5. 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
- M. E. Mortenson; *Geometric Modeling*; John Wiley & Sons,1985

Práctica 2

Modelos PLY y Poligonales

2.1. Objetivos

Aprender a:

- A cargar modelos guardados en ficheros externos en formato PLY (Polygon File Format) y su visualización.
- 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.

2.2. Desarrollo

PLY es un formato para almacenar modelos gráficos mediante listas de vértices, caras poligonales y diversas propiedades (colores, normales, etc.) que fue desarrollado por Greg Turk en la universidad de Stanford durante los años 90. Para más información consultar:

`http://www.dcs.ed.ac.uk/teaching/cs4/www/graphics/Web/ply.html`

Para la realización de la práctica, en primer lugar, se visualizarán modelos de objetos guardados en formato PLY usando los modos de visualización implementados en la primera práctica. Para ello, se entregará el código de un lector básico de ficheros PLY para objetos únicamente compuestos por vértices y caras triangulares, que devuelve un vector de coordenadas de los vértices, flotantes, y un vector de los índices de vértices, enteros sin signo, que forman cada cara. A partir de los mismos se creará el objeto PLY usando las estructuras ya definidas.

En segundo lugar, dado el código que permite generar objetos por revolución de manera simple pero no óptima, habrá que modificarlo para obtener la versión que no contiene triángulos degenerados ni vértices repetidos. Para crear la versión mejorada hay que seguir el siguiente proceso:

Los pasos para crear un sólido por revolución serían:

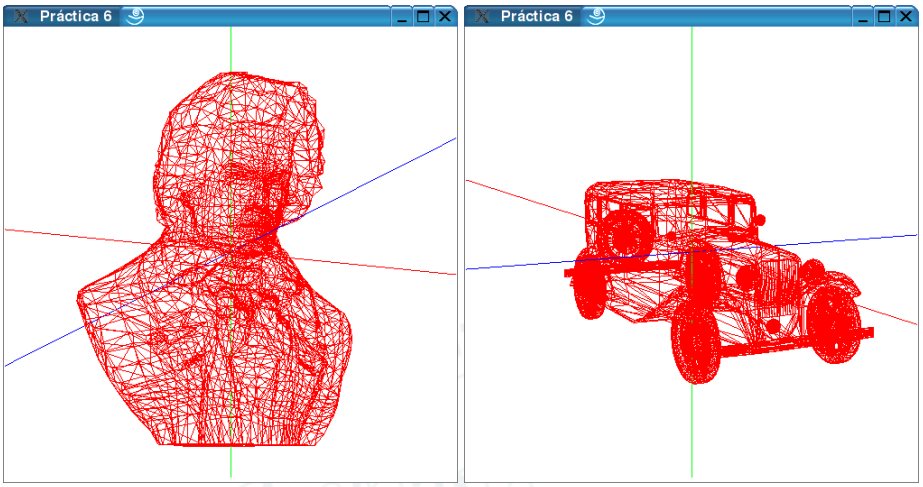
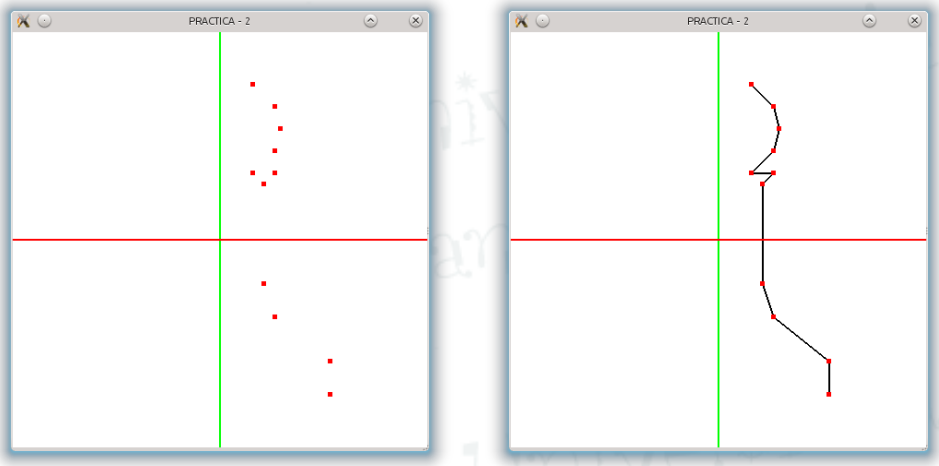


Figura 2.1: Objetos PLY.



(a) Puntos (b) Polilínea

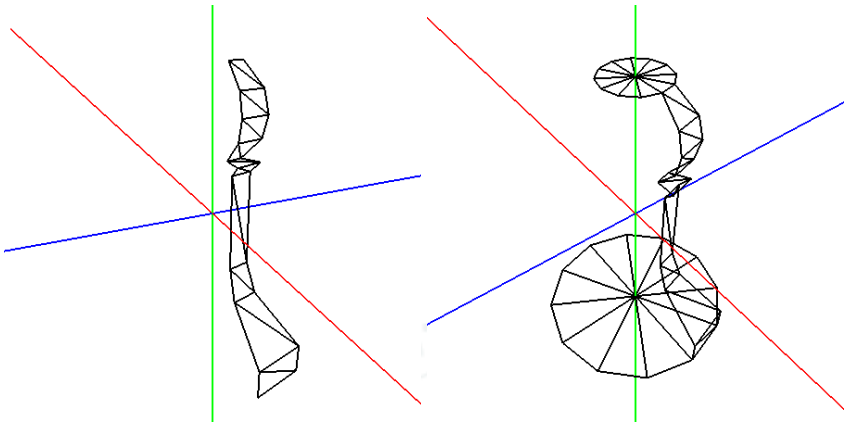
Figura 2.2: Perfil inicial.

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

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

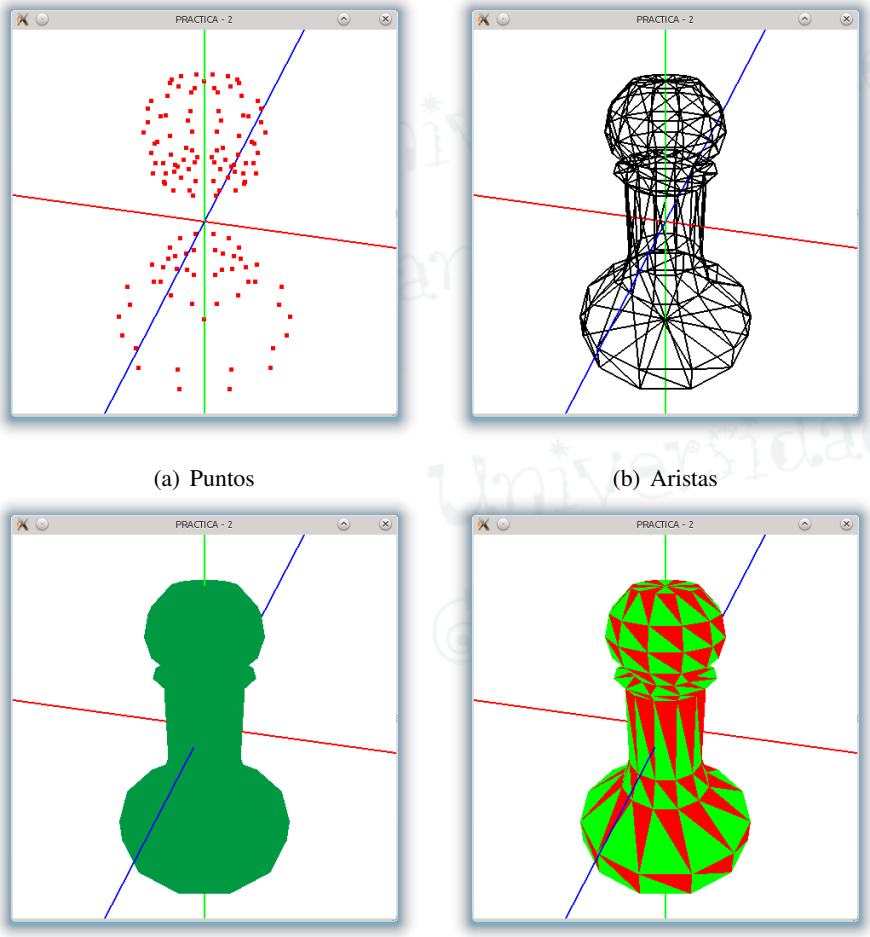
siendo $p_i(x_i, y_i, 0)$ con $i = 1, ...M$ los puntos que definen el perfil (ver figura 2.2).

- 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



(a) (b)

Figura 2.3: 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 2.4: Sólido generado por revolución con distintos modos de visualización.

conjunto de $N \times M$ vértices agrupados en N perfiles Q_j, siendo:

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

- Se guardan $N \times M$ los vértices obtenidos en un vector de vértices según la estructura de datos definida 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 2.3(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 2.3(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 también se podrá visualizar usando cualquiera de los distintos modos de visualización implementados para la primera práctica (ver figura 2.4).

Dos consideraciones sobre la implementación del código para el objeto por revolución: 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, donde hay 11 vértices y una sola cara que no se utilizará)

```
ply
format ascii 1.0
element vertex 11
property float32 x
property float32 y
property float32 z
element face 1
property list uchar uint vertex_indices
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
3 0 1 2
```

Se añadirán las siguientes teclas:

- Tecla 3: Activar cono
- Tecla 4: Activar cilindro
- Tecla 5: Activar esfera
- Tecla 6: Activar objeto PLY cargado

2.3. Evaluación

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

- Creación de una clase para los objetos PLY (3 pts.).
- Mejora del código para el modelado de objetos por revolución de tal manera que no haya triángulos degenerados ni vértices repetidos (4 pts.).
- Creación de las clases para dibujar un cono, un cilindro y una esfera (3 pts.).

2.4. Duración

La práctica se desarrollará en 2 sesiones.

2.5. 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.

Práctica 3

Modelos jerárquicos

3.1. Objetivos

Con esta práctica el alumno aprenderá a:

- Diseñar modelos jerárquicos de objetos articulados.
- Controlar los parámetros de animación de los grados de libertad de modelos jerárquicos usando OpenGL.
- Gestionar y usar la pila de transformaciones de OpenGL.
- Modificar los valores de las transformaciones automáticamente creando una animación

3.2. Desarrollo

Para realizar un modelo jerárquico es importante seguir un proceso sistemático, tal y como se ha estudiado en teoría, poniendo especial interés en la definición correcta de los grados de libertad que presente el modelo.

Para modificar los parámetros asociados a los grados de libertad del modelo utilizaremos el teclado. Para ello tendremos que escribir código para modificar los parámetros como respuesta a la pulsación de teclas.

Las acciones a realizar en esta práctica son:

1. Diseñar el grafo del modelo jerárquico del objeto diseñado, determinando el tamaño de las piezas y las transformaciones geométricas a aplicar. El mismo debe tener al menos 5 niveles y 3 grados de libertad distintos (al menos deben aparecer giros y desplazamientos). Puedes tomar como ejemplo el diseño de una grúa semejante a las del ejemplo (ver figura 3.1). En el ejemplo, estas gruas tienen al menos tres grados de libertad: ángulo de giro de la torre, giro del brazo y altura del gancho.

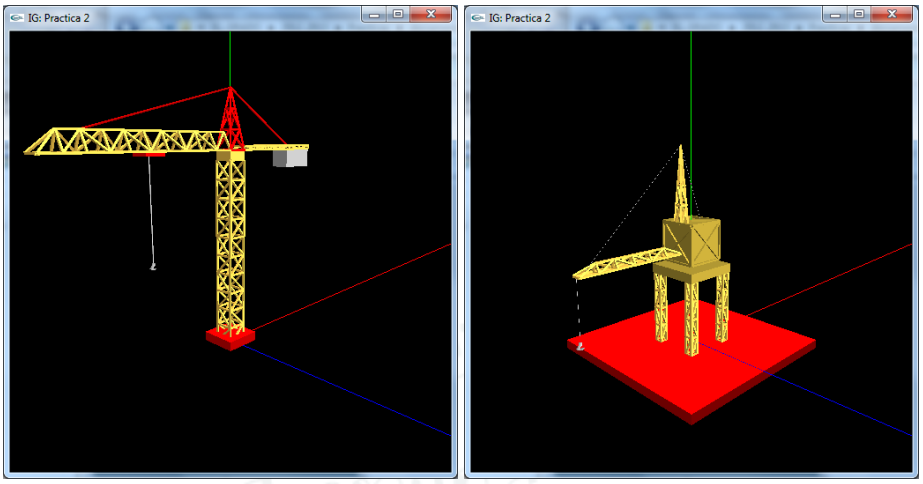


Figura 3.1: Ejemplos del resultado de la práctica 3.

2. Crear las clases necesarias para implementar el modelo jerárquico. Cada clase permitirá definir un tipo de objeto, incluyendo las transformaciones que permiten modelarlo así como las transformaciones y variables que permiten transformarlo, en su caso. La relación jeráquica, en la versión entregada, se implementa mediante la inclusión de las instancias de los objetos hijos en los objetos padres (existen otras posibilidades).

   a) Crear primero la versión estática del modelo, esto es, aquella que sólo muestra al modelo en una posición inicial pero con las dimensiones correctas.

   b) Modificar el modelo para incluir las transformaciones que permiten el movimiento. Esto implica que se deberán poder modificar los parámetros cuando se pulsen las teclas asociadas.

3. Ejecutar el programa y comprobar que los movimientos son correctos.

Para construir los modelos jerárquicos se deben utilizar otros elementos más sencillos que al combinarse mediante instanciación utilizando las transformaciones geométricas necesarias, nos permitirán construir modelos mucho más complejos. Se deben usar los elementos creados en las prácticas anteriores o nuevos que se creen para la actual.

3.2.1. Animación

Una vez se han implementado las transformaciones que aplican el movimiento, se puede modificar el modelo simplemente pulsando las teclas correspondientes. Pero también se puede conseguir que el movimiento sea automático, sin tener que estar pulsando las teclas. Para ello sólo hay que hacer que los parámetros cambien con el tiempo sin nuestra intervención. También tendremos que ajustar la velocidad a la que se producen los cambios.

Para poder automatizar el cambio de los parámetros, vamos a hacer uso del evento *idle* de GLUT (para Qt hay que usar un QTimer con un tiempo 0). Este evento, si se activa, hace

que se llame a la función que se haya indicado cuando en gestor de eventos está desocupado, idle. En general, este suele ser el estado en el que se encuentra el gestor pues los eventos, si están bien implementados, se deben resolver rápidamente. Esto significa que la función que da servicio al evento idle se estará llamando numerosas veces, y lo mejor, no interrumpirá que se produzcan otros eventos.

Para activar el evento idle hay que hacer lo siguiente:

```
glutIdleFunc(funcion_idle);
```

siendo `void función_idle()` el nombre de la función que se llama. Lo que se va hace en esta función es actualizar el valor de los parámetros que controlan las transformaciones asociadas al movimiento. Por ejemplo, el ángulo para un rotación.

Hay que tener en cuenta que lo que estamos haciendo es lo siguiente $P' = f(P)$ o $P' = f(t)$. Esto es, calculamos un nuevo valor para el parámetro P usando el valor anterior, o calculamos el nuevo valor como función del tiempo.

Por ejemplo, podemos calcular el siguiente valor del ángulo de una rotación de la siguiente manera: $\alpha' = \alpha + \delta$. δ es el incremento de ángulo que se aplica en cada paso. Por ejemplo podría ser $1°$ o $5°$. En el primer caso el cambio se produce más lentamente que en el segundo. Por tanto, al cambiar el valor de δ estaríamos cambiando la velocidad de giro. Para las traslaciones sería similar.

Hay recordar que una vez se han actualizado las variables hay que indicar que se redibuje la escena con `glutPostRedisplay`.

3.2.2. Resultados entregables

El alumno entregará un programa que represente y dibuje un modelo jerárquico con al menos una jerarquía de 5 niveles y con 3 grados de libertad, cuyos parámetros se podrán modificar por teclado.

Para esta práctica se deberán incorporarlas siguientes teclas:

- Tecla 7: Activar objeto jerárquico
- Tecla A: Activar/desactivar la animación
- Teclas Q/W: modifica primer grado de libertad del modelo jerárquico (aumenta/disminuye)
- Teclas S/D: modifica segundo grado de libertad del modelo jerárquico (aumenta/disminuye)
- Teclas Z/X: modifica tercer grado de libertad del modelo jerárquico (aumenta/disminuye)
- Tecla E/R: incrementar/decrementar la velocidad de modificación del primer grado de libertad del modelo jerárquico
- Tecla T/Y: incrementar/decrementar la velocidad de modificación del segundo grado de libertad del modelo jerárquico

- Tecla U/I: incrementar/decrementar la velocidad de modificación del tercer grado de libertad del modelo jerárquico

3.3. Evaluación

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

- Creación del modelo jerárquico y las clases correspondientes con al menos una jerarquía de 5 niveles (5 puntos)
- Inclusión de las transformaciones para que el modelo tenga 3 grados de libertad y se pueda mover con las teclas indicadas (3 puntos)
- Animación (2 puntos)

3.4. Duración

La práctica se desarrollará en 3 sesiones.

3.5. 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
- P. Shirley y S. Marschner; *Fundamentals of Computer Graphics, 3rd Edition*; A K Peters Ltd. 2009.

Apéndice

En las figuras 3.2 y 3.3 podéis ver algunos ejemplos de modelos jerárquicos que se pueden construir para la práctica (simplificando todo lo que se quiera los distintos elementos que los componen).

Estudiar con detalle cada uno y seleccionar el que os interese, o diseñar otro que tenga al menos 3 grados de libertad similares a los de las gruas que tenéis en el ejemplo.

Figura 3.2: Ejemplos de posibles modelos jerárquicos.

Figura 3.3: Ejemplos de posibles modelos jerárquicos.