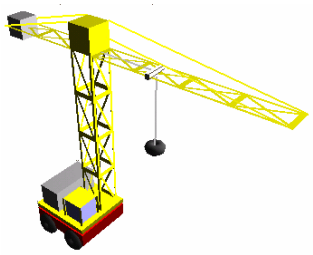
Proyecto DAC
"El Mundo de la Grúa"



Antonio Martín Gutiérrez
Diseño Asistido por Computador
4º Ing. Informática (Granada)

ÍNDICE

1. Descripción general del problema y planteamiento inicial de soluciones.

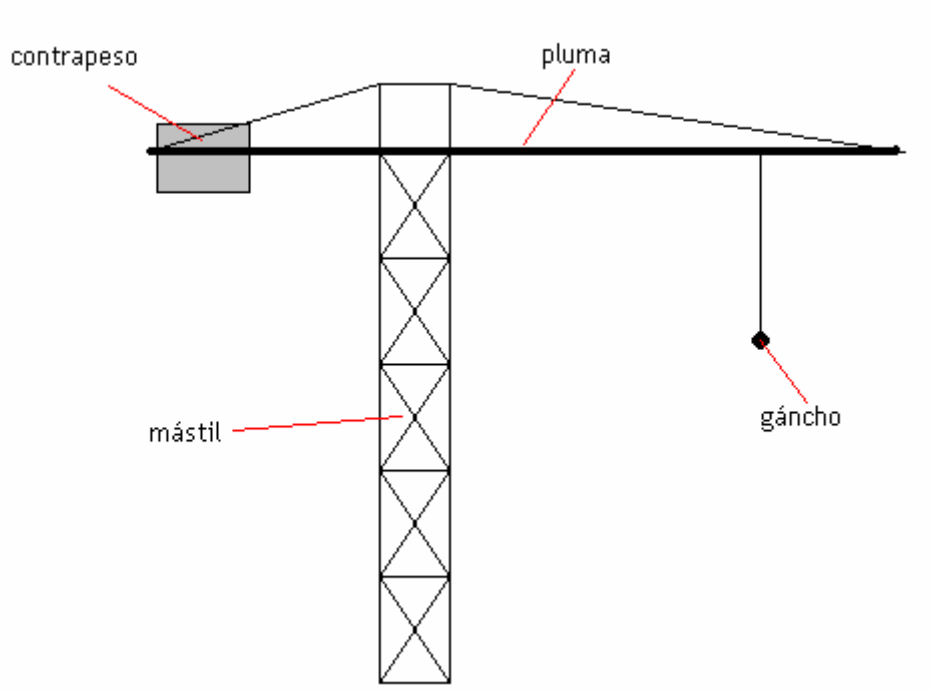
La idea era diseñar un programa que permitiera simular el control sobre una grúa, la cual se movería sobre un mundo 3D. En este mundo 3D existirían una serie de objetos con los que la grúa interaccionaría, permitiendo engancharlos, transportarlos y depositarlos en cualquier otra parte.

Partiendo de esta idea, surgen una serie de problemas como el almacenamiento de los modelos, su volumen para controlar los choques, la delimitación del espacio de actuación de la grúa, cómo controlar el movimiento de la grúa, cómo efectuar el control de la cámara sobre la escena...

Como último añadido, se pretendía también incorporar selección de objetos del mundo 3D mediante el ratón e insertar alguna pequeña animación. Para ello se intentan añadir dos nuevas funciones a la grúa: coger objetos automáticamente mediante selección de los mismos, y depositarlos también de forma automática en el lugar que le digamos. Esto trae consigo nuevos problemas que iremos comentando en adelante.

Para empezar el diseño, se toman las siguientes decisiones:

- La cámara (u observador) podremos controlarla completamente. Esto implica control sobre la posición de la misma y la dirección a la que se enfoca (se omite el parámetro de ángulo hacia arriba de la cámara "view up" que se considerará constante).
- El mundo 3D de la grúa será un plano centrado en el origen de la escena.
- La grúa estará constituida por una base sobre la que se encuentra la estructura propia de la grúa. Esta estructura podrá girar libremente respecto de la base. La estructura será la clásica de las grúas de construcción, que se componen de un mástil vertical y en lo alto uno horizontal (llamado "pluma") que también puede girar sobre el plano horizontal, respecto del mástil que lo sostiene. En la pluma se encuentra el sistema para enganchar los distintos objetos, que estará compuesto por un gancho que podremos alejar o acercar a lo largo de la pluma y bajar o subir gracias a el cable que lo une a la misma.
- Tendremos control sobre todos los parámetros de la grúa: posición, ángulo de la base, ángulo de la pluma, desplazamiento del gancho en la pluma y caída del mismo hacia abajo.
- Los objetos serán cajas que no se podrán mover salvo que los enganche la grúa y los desplace.
- Controlaremos colisiones entre: los propios objetos (que permitirá apilarlos), y entre éstos y la grúa (para evitar atravesarlos). Además, nada podrá sobrepasar los límites del mapa, ni atravesar el suelo. Por otro lado, la grúa la vamos a dividir en dos objetos distintos para controlar choques: la base y el gancho. Por último, las colisiones se controlarán mediante esferas (distancia euclídea) centradas en el modelo del objeto.



Con todas estas ideas y restricciones veamos ahora las cuestiones de diseño relacionadas.

2. Justificaciones de diseño

2.1. Estructuras de datos empleadas

En función de las restricciones que asumimos en el apartado anterior, las estructuras de datos empleadas son las siguientes:

- **La grúa.** Para ella deberemos almacenar su posición, orientación en la escena (dirección a la que apunta el frente), ángulo que forma la estructura respecto la base, ángulo que forma la pluma respecto al mástil, desplazamiento del gancho en la pluma y caída del gancho hacia abajo. Un aspecto adicional es que en un determinado momento tendrá que transportar objetos, con lo que habrá que incluir una referencia al objeto que porta. La estructura resultante es *('Punto' se define como tres reales)*:

```
typedef struct{
        Punto posic;
        Punto direc;
        float vel;
        float angBase;
        float angPluma;
        float desplGancho;
        float caidaGancho;
        int objEnganchado;
} Grua;
```

- **La cámara.** Para controlar este apartado, tendremos que almacenar la posición y la orientación. También se añade el vector de

"view up" aunque este parámetro no se modificará durante la ejecución y se inicializará apuntando hacia arriba (*{0, 1, 0}*). Adicionalmente, tanto para la cámara como para la grúa se almacena la *vel*ocidad, que será simplemente un factor que influirá en los desplazamientos de la posición.

```
typedef struct{
        Punto posic;
        Punto direc;
        Punto up;
        float vel;
} Vista;
```

- **Los objetos.** La idea era disponer de varios tipos de objetos cuyo volumen al final será el mismo. Para ellos almacenamos su tipo (*cubo, esfera, cono, dodecaedro o icosaedro*), su posición y su orientación.

```
typedef struct{
        Punto posic;
        Punto direc;
        int tipo;
} Objeto;
```

- **Otras estructuras.** Dado que tendremos que almacenar diferentes objetos, se define una estructura adicional que contendrá el número de objetos y un puntero al vector de los mismos. Asimismo, existe una última estructura que será la que permita el control de la animación cuando esta ocurra (las animaciones serán comentadas más adelante). Esta contiene el tipo de animación (dos posibilidades, animación para coger objeto y para dejarlo en el lugar indicado), una referencia al objeto implicado, el paso que transcurre en la animación, un punto de destino y la velocidad que tenía la grúa antes de iniciar la animación (la animación tiene una velocidad independiente).

2.2. Funcionamiento general. Los "callbacks".

Toda la interacción con la aplicación se gestiona mediante los "callbacks", procedimientos del programa que se lanzan en respuesta a un evento recibido. Así, tendremos callbacks para los eventos de teclado y los eventos de ratón con los que controlaremos el funcionamiento del programa. Mediante el ratón (haciendo click), podremos indicar a la grúa que enganche un objeto o bien que lo deje en un lugar concreto. El teclado nos permitirá gestionar todo el movimiento y acción de la grúa y el observador.

Con idea de no tener casi todo el teclado ocupado con funciones asignadas a teclas, existen dos estados del programa: modo control de cámara y modo control de grúa. Se puede pasar de uno a otro mediante el *tabulador* o la opción correspondiente en el menú (botón derecho del ratón). En cada modo se puede hacer uso de unas determinadas teclas con funciones asignadas.

En el modo control de cámara, podremos cambiar los parámetros que definen el observador. Así se podrá desplazar la cámara adelante, atrás, a izquierda y a derecha. Además se podrá también cambiar la dirección de enfoque permitiendo mirar hacia arriba, abajo, a izquierda y a derecha. Tanto la posición como la dirección en la que enfocar, tal y como se vio en las estructuras empleadas, son vectores. De este modo, alterar cualquier parámetro será un operación sobre vectores. Por ejemplo, cambiar la dirección de enfoque supondrá una rotación para el vector de dirección, y cambiar la posición, se reducirá a un desplazamiento del punto sobre un vector horizontal o vertical con un incremento que dependerá de la velocidad de cámara que se tenga en ese momento.

Cuando pasamos a modo control de grúa, será cuando podremos manejarla cambiando cualquiera de sus parámetros. Al igual que en el caso de la cámara, los desplazamientos de la grúa son sencillas operaciones de movimiento del punto posición en una determinada dirección sobre el plano horizontal. Lógicamente, también dispondremos de teclas asignadas para cambiar los ángulos de la base, la pluma, el desplazamiento del gancho y su caída. Con todo ello podremos girar y desplazar el gancho hasta colocarlo cerca de un objeto para engancharlo. Una vez que acercamos el gancho lo suficiente sobre un objeto, tendremos la posibilidad de engancharlo utilizando la tecla *espacio*. La posición y dirección de los objetos sólo tienen la posibilidad de cambiar en este momento: cuando están enganchados y la grúa se desplaza.

Hay que mencionar, que siempre que se realice un movimiento de la grúa, se comprueban las posibles colisiones, de modo que si existen, se deshace el movimiento. La implementación dada para la comprobación de colisiones sigue una estructura jerárquica, es decir: primero comprueba que la grúa no colisiona con nada (ni se sale del mapa), luego que tampoco lo hace el gancho y, por último, que tampoco colisiona el objeto enganchado, de haberlo.

Respecto al ratón, y al margen del menú que aparece al pulsar el botón derecho, éste nos permitirá tener cierto control sobre los dos tipos de animaciones. Cuando la grúa no tiene ningún elemento enganchado y hacemos click con el botón izquierdo, se pasa a modo render y se comprueba si se pinchó sobre algún objeto. Si fue así, se lanza la animación para intentar enganchar dicho objeto. La otra acción que puede tener lugar, sucede cuando elegimos la opción del menú *"Grua Automática -> Dejar objeto"* y la grúa tiene un objeto enganchado. Entonces, pasaremos a una proyección paralela con una vista aérea de la escena. El click en una posición del escenario lanzará la animación para dejar el objeto en el lugar elegido. Conocer este lugar es sencillo gracias a dos variables que nos indican la dimensión del *viewport* y que nos permiten llevar a cabo la proporción sobre las coordenadas del punto imagen en el que se pinchó. Ambas variables (ancho y alto del *viewport*) se mantendrán siempre actualizadas gracias al callback que tendremos para el evento de redimensionado de la ventana.

Existe una última función por comentar: la *función de desocupado*. Esta es una función especial que se dispara cuando OpenGL no está realizando ninguna operación. Esta es la función que se suele usar para las animaciones y con ese fin se ha empleado. Cuando se dispara, comprueba si se está llevando a cabo alguna animación y si es así, realiza un paso de la misma. Las animaciones se comentan con más detalle en el siguiente punto.

2.3. Descripción de los algoritmos principales

Al margen de los algoritmos para realizar operaciones de desplazamientos y rotaciones de vectores (que no tienen más interés que el matemático), y de los algoritmos de iniciación y salvado de las posiciones de los modelos, merecen especial descripción las animaciones y el algoritmo del cálculo de colisiones.

Recordemos que una animación venía determinada por su tipo, una referencia al objeto involucrado, el paso en el que se encontraba, el punto de destino y la velocidad que poseía la grúa antes de iniciarse la animación.

```
typedef struct{
        int tipo;
        int obj;
        int paso;
        Punto dest;
        float initVel;
} Animacion;
```

Cuando se dispara la función de desocupado, comprueba el tipo de la animación que se ejecuta y llama a la función para seguir el siguiente paso en la misma.

Ambas animaciones siguen prácticamente los mismos pasos:

- Primero recoger el gancho hasta la máxima altura con el fin de evitar en todo lo posible el choque con otro obstáculo.
- Luego se endereza la base, haciéndola girar hasta que el ángulo que tiene asociada sea cero.
- Posteriormente se aleja el gancho hasta situarlo a la máxima distancia en la pluma.
- Luego se hace lo mismo que para la base con la pluma, haciéndola que mire justo hacia el frente. Con este y los anteriores pasos, tendremos perfectamente localizado el gancho en la punta de la pluma, arriba.
- Ahora se trata de que la dirección de avance de la grúa coincida con la dirección en la que se encuentra el objetivo. Para ello se la hace girar hasta conseguirlo.
- Seguidamente avanzamos hasta colocarnos justo en frente del destino.
- Una vez llegamos, descendemos el gancho hasta notar colisión.
- Accionamos el gancho. Si se trataba de coger un objeto, este será enganchado. Si por el contrario se trataba de dejarlo, este se depositará sobre aquello con lo que detectamos la colisión (el suelo u otro objeto).
- Por último, recogemos de nuevo el gancho hasta arriba.

Si en cualquiera de los pasos se detecta colisión con algún objeto, la animación se detendrá y si se desea, tendrá que ser esquivado manualmente.

Este no es un algoritmo óptimo de planificación (ni es lo que se pretende). Tan sólo es un algoritmo sencillo que resulta efectivo.

Respecto al algoritmo del cálculo de colisiones, se le llama justo después de realizar un movimiento, para deshacerlo en caso de determinar la existencia de colisiones. Los pasos que sigue son los siguientes:

- Primero se determina el objeto para el que se busca colisión: la grúa, el gancho o un objeto.
- A continuación se comprueba la existencia de colisión con el límite del mapa, y si la hay, termina el algoritmo indicando este hecho.
- En este paso, se realizan comprobaciones distintas en función del objeto para el que se pregunta:
  - Si es la grúa, se pregunta si hay colisión para el gancho. Si la hay se detiene el algoritmo y lo notifica.

- Si es el gancho, comprueba recursivamente si colisiona el objeto que tiene enganchado (si tiene). Si colisiona termina, y si no, prueba la distancia con la propia base de la grúa, notificándolo de haber colisión.
- Si se llamó al algoritmo para un objeto se prueba que no atraviese el suelo. Si no lo hace, comprueba la distancia a la base de la grúa y si hay colisión lo indica y termina.

o Si hasta ahora no se ha detectado colisión alguna, se comprueba la distancia con el resto de objetos inmóviles. Si hay colisión se indica y termina.
o Si se llega a este punto, se notifica que no existen colisiones y finaliza el algoritmo.

3. El apartado gráfico

3.1. Aspectos generales

Entre los aspectos generales se encuentran cuestiones como la transformación de visualización, la iluminación, la selección de objetos y algunos otros parámetros de OpenGL.

Respecto a la inicialización de OpenGL, se emplean parámetros que permanecerán constantes a lo largo de toda la ejecución:

- Representación de las dos caras en las superficies. Esto es necesario por la grúa. Toda la estructura está construida a base de *GL_QUADS*, cuadrados que sólo se verían por una cara si no activamos esta opción de OpenGL.
- Habilitamos la luz 0 para la iluminación.
- El último parámetro que habilitamos es el test de profundidad.

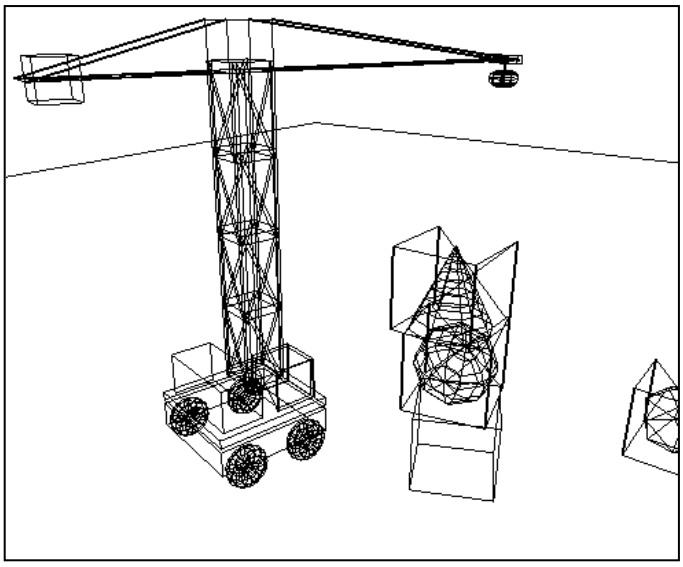
En cuanto a la iluminación, como se ha mencionado se emplea la luz 0 de OpenGL, pero es una iluminación sencilla. La luz permanece en una posición fija respecto a la escena.

La transformación de visualización tiene en cuenta varios aspectos. En primer lugar, se separa por un lado una función que realiza la transformación sin inicializar la matriz al principio. Esta es necesaria para la selección de objetos con el ratón. Otra cuestión a tener en cuenta es que la proyección es siempre en perspectiva, salvo cuando se elige la opción de depositar un objeto en un determinado lugar del mapa. Entonces se necesita una proyección paralela que ajuste la escena al tamaño de la ventana.

Respecto a la proyección perspectiva, una vez cargada la matriz de proyección deseada y pasando a la de modelado, la colocación de la cámara se hace gracias a los parámetros que almacenábamos (posición, dirección y *view up* del observador). Se emplea para ello la función *gluLookAt* a la que se le pasan dichos parámetros.

Finalmente, para la selección de objetos se emplean exactamente los pasos aprendidos en la práctica 4 (interacción) de la asignatura.

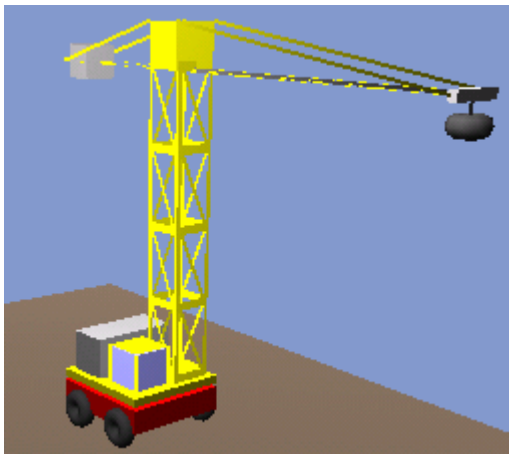
Un último aspecto a destacar, es la posibilidad de alternar entre el modelo de alambres de la escena y el modelo sólido, con sólo seleccionar la opción correspondiente en el menú.



3.2. La Grúa

La grúa es el modelo "complejo" del programa, cuyo dibujado se realiza de un modo jerárquico. La descomposición que se hace del modelo para su reconstrucción es la siguiente:

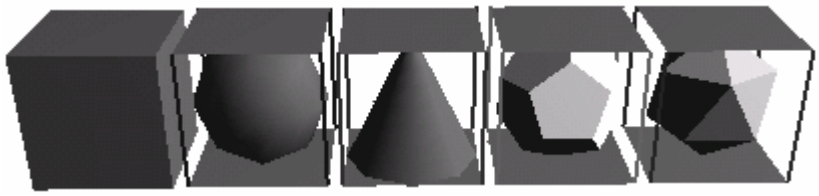
- Por un lado se dibuja la base y por otro la estructura. Esto nos permite aplicar en medio el ángulo de giro que posee la estructura respecto la base.
- La base se compone de un cubo y cuatro *toros* que constituyen las ruedas.
- La estructura se compone de: otra base, el contrapeso, la cabina (son todos cubos), el mástil y la pluma. Entre el dibujado del mástil y la pluma se aplica la rotación que existe entre ambos.
- El mástil se dibuja completamente a partir de una figura básica, compuesta por un marco de láminas y otra que atraviesa en diagonal. Con cuatro de estas figuras básicas (por rotación) se obtiene un fragmento del tronco del mástil. Cuatro de estos fragmentos, uno sobre otro, componen el mástil.
- Por último, la pluma se subdivide en:
- un cubo que hace de contrapeso,
- otro cubo que simula el lugar donde se almacena el motor del gancho y el cable,
- los cables soporte (que son simples líneas),
- la estructura horizontal que da soporte a todos estos elementos
- y por último el gancho. Pero el dibujado de éste cuenta a su vez con el dibujo de las dos líneas que simulan el cable que lo sustentan; un pequeño cubo, desde el que el gancho cuelga; y por último el propio gancho (un imán), que está representado por otro *toro*.



3.3. Los objetos

Los objetos son sencillos modelos geométricos obtenidos de llamadas a funciones de la biblioteca *glut*. Entre ellos podemos encontrar un cubo, una esfera, un cono, un dodecaedro y un icosaedro. Para que tengan un aspecto de caja, se introducen entre un esqueleto de esta, conformado por dos tapas y cuatro alambres que las unen.

Para dibujar un objeto, sólo hay que trasladar hasta la posición que tiene y rotar según el ángulo que forma su vector de dirección en el plano horizontal.



4. Manual de usuario

Tras ejecutar el programa:

```
$> grua
```

nos aparecerá directamente la escena inicial.

La escena inicial es generada a partir de los ficheros *vista.ini* y *objetos.ini*. El fichero *vista.ini* tiene la siguiente estructura:

```
- Posición del observador
- Dirección a donde mira
- Velocidad de mvto de cámara (entorno a 5 es un valor aceptable)
```

Los dos primeros parámetros constan de tres reales, correspondientes a las coordenadas (x, y, z) de los mismos. La velocidad de movimiento es un solo real. Un ejemplo de fichero *vista.ini* podría ser:

```
12.263 7.050 6.617
-1.336 -0.821 -0.740
9.000
```

El fichero *objetos.ini* tiene la siguiente estructura:

```
GRÚA
- Posición
- Dirección
- Velocidad
- Ángulo de la base
- Ángulo de la pluma
- Desplazamiento del gancho
- Caida del gancho
- Objeto enganchado

OBJETOS
- Posición
- Dirección
- Tipo de objeto:
      0 -> Cubo
      1 -> Esfera
      2 -> Cono
      3 -> Dodecaedro
      4 -> Icosaedro
```

La grúa es opcional, pero si se opta por especificarla, la primera línea del fichero debe contener la palabra *GRÚA*. Las posiciones y direcciones sólo constan de las coordenadas (x, z), ya que la altura es generada automáticamente por el programa. Ejemplos válidos de fichero *objetos* podrían ser:

```
GRÚA
1.859 3.111
-0.257 -0.966
5.050
356.000
-5.000
7.900
0.500
3

OBJETOS
-0.884 0.920
-0.612 -1.275
0

-0.955 0.914
0.240 0.971
1
```

```
OBJETOS
-1.000 1.000
0.000 1.000
2

1.092 -1.105
0.180 0.984
3

1.000 -1.000
0.000 1.000
4
```

Si no se encuentran los ficheros mencionados en el directorio, la escena inicial se generará sin objetos y con la grúa en el centro.

Una vez que se inicia el programa, habremos entrado en modo *movimiento de cámara*. Para cambiar al modo *movimiento de grúa* podemos utilizar el *tabulador* o la opción correspondiente en el menú. Las funciones asignadas a las teclas en modo *movimiento de cámara* son las siguientes:

A : girar cámara a la izquierda.
D : girar cámara a la derecha.
W : avanzar cámara adelante.
S : retroceder cámara hacia atrás.
Q : mover cámara horizontalmente a la izquierda.
E : mover cámara horizontalmente a la derecha.
U : girar cámara hacia arriba.
J : girar cámara hacia abajo.
I : subir la cámara.
K : bajar la cámara.
+ : aumentar velocidad de cámara.
- : disminuir velocidad de cámara.

En el modo *movimiento de grúa*, el movimiento se controla con:

A : Avanzar grúa girando a la izquierda.
D : Avanzar grúa girando a la derecha.
W : Avanzar grúa.
S : Retroceder grúa.
G : Girar la estructura de la grúa a la izquierda.
H : Girar la estructura de la grúa a la derecha.
T : Girar la pluma a la izquierda.
Y : Girar la pluma a la derecha.
U : Subir el gancho.
J : Bajar el gancho.
I : Alejar el gancho en la pluma.
K : Acercar el gancho en la pluma.
Espacio : Enganchar/Soltar objeto con el imán del gancho.
+ : Incrementar velocidad de la grúa.
- : Disminuir velocidad de la grúa.

Además de estas, existen otras acciones que estarán disponibles en los dos modos de manejo. Estas son las acciones que aparecen en el menú al pulsar con el botón derecho:

Grua Automática -> Coger Objeto : Esta función no es de gran utilidad ya que está siempre habilitada por defecto. Estando en vista perspectiva, si la grúa no tiene objeto enganchado y pinchamos uno con el ratón, comenzará la animación. La tecla alternativa es **F9**.

Grua Automática -> Dejar Objeto : Si la grúa tiene objeto enganchado, obtendremos una vista cenital de la escena. Pinchando en el lugar deseado del mapa, la grúa intentará depositar el objeto allí. La tecla alternativa para esta acción es **F10**.

Mostrar/Ocultar -> Caras/Alambres : Alterna el modo de dibujo entre modelo de alambre y modelo sólido. Tecla alternativa **F5**.
Mostrar/Ocultar -> Ejes : Muestra / oculta los ejes coordenados. La parte positiva de los ejes se dibuja con lína de 2 puntos, mientras que la parte negativa con línea de 1 punto de grosos. Tecla alternativa **F6**.
Modo Movimiento : Alterna entre *movimiento de cámara* y *movimiento de grúa*. Tecla alternativa el **tabulador**.
Fijar vista en Grúa : Cuando se mueve la grúa, la cámara la sigue, enfocándola en todo momento. Tecla alternativa **F1.**
Guardar Posiciones : Actualiza los ficheros *vista.ini* y *objetos.ini* con los valores actuales de la cámara y los modelos. Tecla alternativa **F2**.
Salir : Abandona la aplicación. Tecla alternativa **escape**. Esta tecla también permite detener una animación cuando esta se está ejecutando.