DISEÑO ASISTIDO POR COMPUTADORA

Práctica final: diseñador de habitaciones

Alumno: Javier de Reyes Sorroche
4º Ingeniería Informática

Índice

Descripción de la práctica

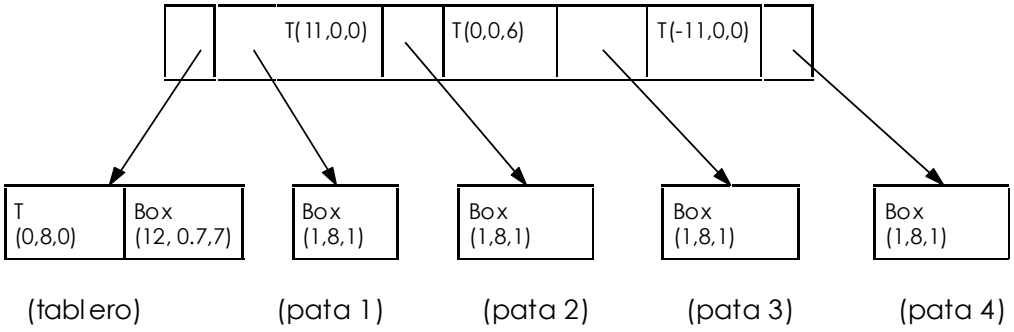
El contenido de esta práctica es un diseñador de habitaciones bastante simple. Su funcionamiento se detalla en el manual de usuario posterior, pero adelantemos que habrá varios tipos de muebles que podremos colocar sobre un escenario previamente definido, pudiendo también interactuar con algunos de los muebles. Pasemos pues a describir someramente la práctica en sí.

Lo primero que se hizo fue definir sobre el papel los modelos de los muebles que se iban a utilizar en la práctica. Se decidió usar 7 tipos de mueble diferentes: mesas, sillas, armarios, camas, estanterías, sofás y mesillas de noche (en principio tenemos un diseñador de dormitorios más que de habitaciones). Una vez estuvimos satisfechos de los bocetos, nos planteamos cómo pasarlos a OpenGL. En principio, la mayoría de los muebles podían modelarse exclusivamente con cubos de distintas dimensiones, así que decidimos aprovechar el procedimiento box de un archivo anterior para generar esos cubos. Pero con esto no sería suficiente, porque los cajones de los armarios y mesillas de noche debían tener un lado hueco. Así se usó el procedimiento box como base para implementar el procedimiento mibox, que nos crearía un cubo con un lado hueco. Usamos box como base porque nos interesaba que los lados tuvieran volumen (no queríamos caras planas). Una vez tuvimos las primitivas necesarias para construir los muebles, implementamos un procedimiento para cada tipo de mueble, que dibuja un objeto de ese tipo (con las dimensiones que habíamos definido sobre el papel) en el origen de coordenadas. Los muebles son de dimensiones fijas, para no tener problemas a la hora de evitar colisiones entre unos muebles y otros; tendremos una especie de burbuja envolvente sobre la cual no podemos meter muebles, y esa burbuja debe ser de dimensión conocida para que el programa funcione (volveremos sobre ello más adelante).

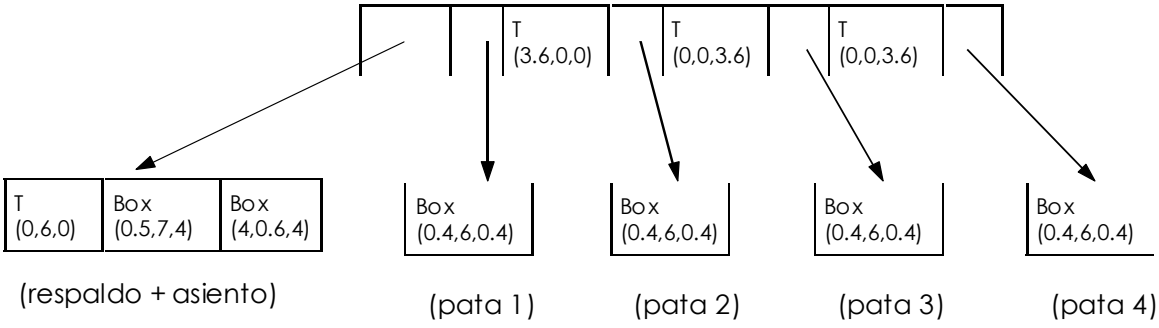
Modelo jerárquico de los objetos construidos

A continuación mostramos los modelos jerárquicos utilizados para cada modelo, donde T significa traslación, y Ra significa rotación sobre el eje a. En los casos en que el dibujo dependa de algún parámetro (por ejemplo, el grado de apertura de las puertas) se deja indicado.

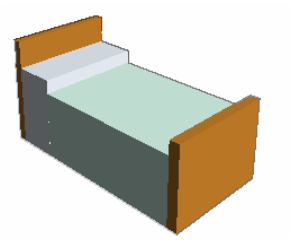
- Mesa:

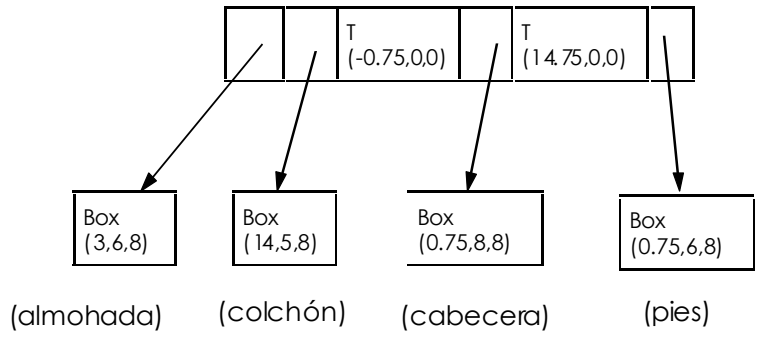


- Silla:

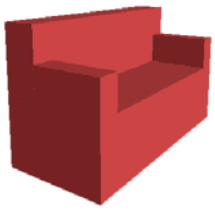


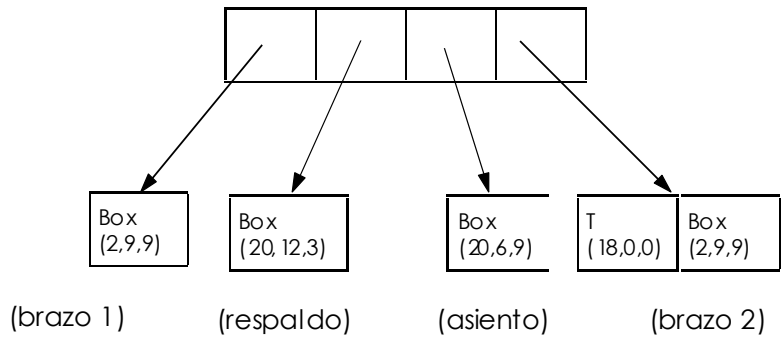
- Cama:



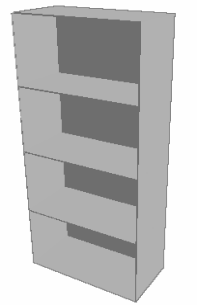

T
(-0.75,0,0)
T
(14.75,0,0)
Box
(3,6,8)
Box
(14,5,8)
Box
(0.75,8,8)
Box
(0.75,6,8)
(almohada)
(colchón)
(cabecera)
(pies)

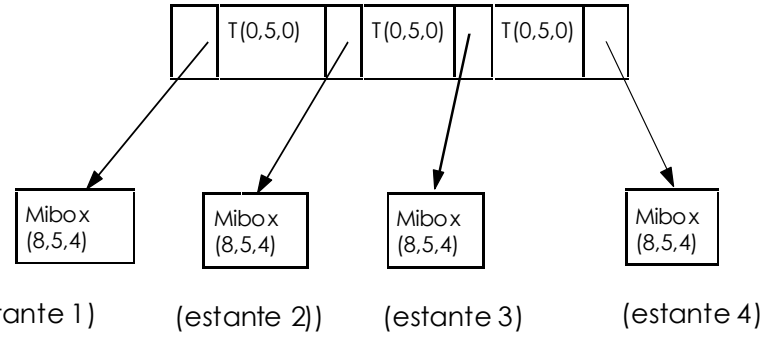
- Sofá:




Box
(2,9,9)
Box
(20, 12,3)
Box
(20,6,9)
T
(18,0,0)
Box
(2,9,9)
(brazo 1)
(respaldo)
(asiento)
(brazo 2)

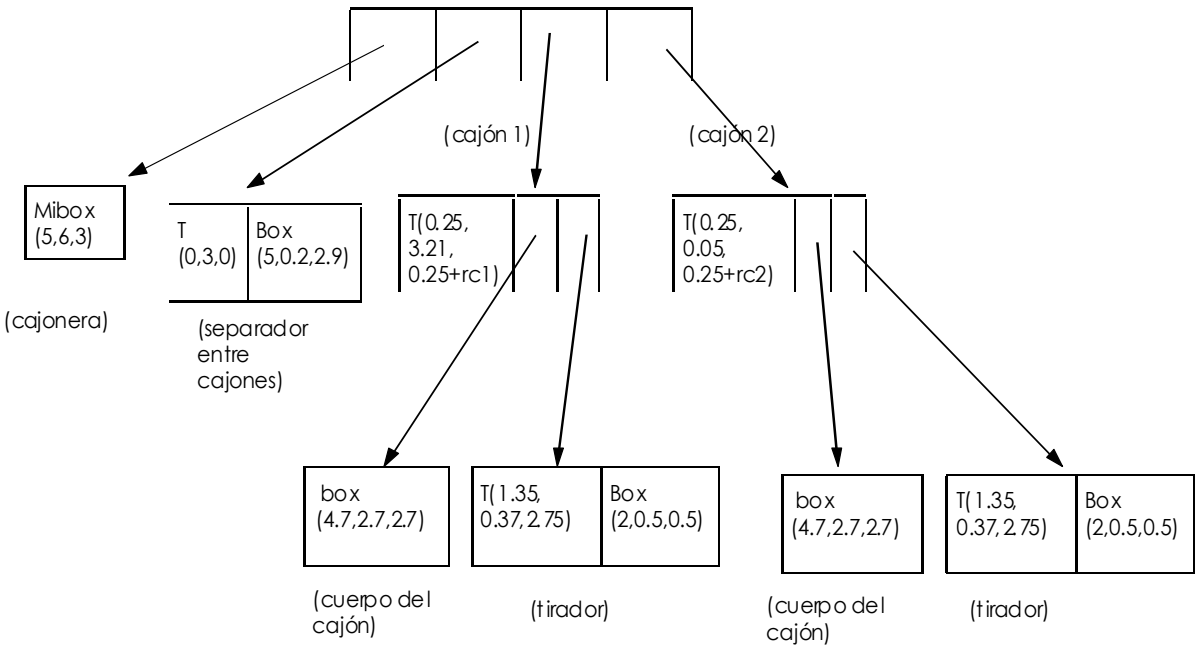
- Estantería:





(estante 1) (estante 2)) (estante 3) (estante 4)

- Mesilla de noche:



(cajonera)

(separador entre cajones)

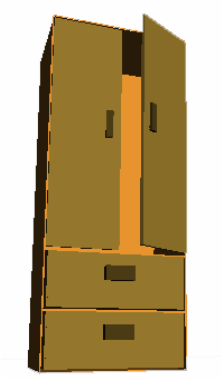
(cuerpo del cajón)

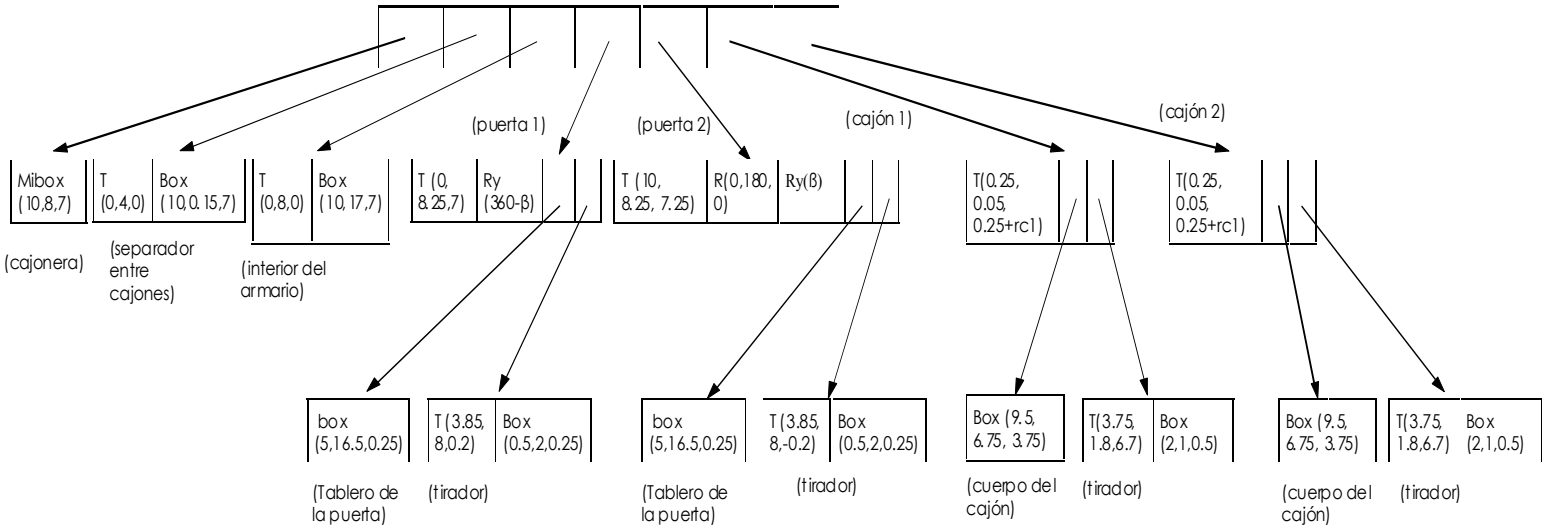
(tirador)

(cuerpo del cajón)

(tirador)

- Armario:




Mibox (10,8,7)
T (0,4,0)
Box (10,0.15,7)
T (0,8,0)
Box (10,17,7)
T (0, 8.25,7)
Ry (360-β)
T (10, 8.25, 7.25)
R(0,180, 0)
Ry(β)
T(0.25, 0.05, 0.25+rc1)
T(0.25, 0.05, 0.25+rc1)
(puerta 1)
(puerta 2)
(cajón 1)
(cajón 2)
(cajonera)
(separador entre cajones)
(interior del armario)
box (5,16.5,0.25)
T (3.85, 8,0.2)
Box (0.5,2,0.25)
box (5,16.5,0.25)
T (3.85, 8,-0.2)
Box (0.5,2,0.25)
Box (9.5, 6.75, 3.75)
T(3.75, 1.8,6.7)
Box (2,1,0.5)
Box (9.5, 6.75, 3.75)
T(3.75, 1.8,6.7)
Box (2,1,0.5)
(Tablero de la puerta)
(tirador)
(Tablero de la puerta)
(tirador)
(cuerpo del cajón)
(tirador)
(cuerpo del cajón)
(tirador)

Estructuras de datos y funciones utilizadas

Tras haber probado que los modelos se dibujaban como debían, pasamos a plantearnos las estructuras de datos que íbamos a necesitar para un buen funcionamiento del programa. Estaba claro que necesitaríamos una estructura que almacenara la posición espacial de cada muebla, que después seria recorrida a la hora de dibujar, la creamos con el nombre de "posición". Almacenamos en ella las coordenadas x,z del mueble (la y será 0 siempre, porque insertamos muebles a nivel del suelo) en la estructura. Usamos un vector de este tipo de estructuras para cada tipo de muebles (7 vectores en total: mesas[], sillas[], armarios[], mesillas[], camas[], estanterías[], sofas[]). También era necesario saber cuantos objetos de cada tipo había situados sobre el escenario; esto lo controlamos con una serie de variables (nmesas, nsillas, narmarios, ncamas, nsofas, nmesillas, nestanterias) inicializadas a 0, que incrementarían su valor conforme fuéramos añadiendo muebles. El numero máximo de muebles de un determinado tipo es de 50 (mas que suficiente para el tamaño de la habitación). Se hizo estático para no perder tiempo en reserva y liberación de memoria dinámica y poder centrarnos en la parte de OpenGL, que al fin y al cabo es lo que interesa en esta asignatura.

Necesitamos también saber en que estado del programa nos encontramos: si estamos visualizando o poniendo muebles. Lo controlamos con sendas variables booleanas (visualizando y poniendomueble). Además, cuando estamos poniendo muebles necesitamos saber qué tipo de mueble estamos colocando. Usamos otras variables (poniendomesas, poniendosilas, poniendoarmarios, poniendocamas, poniendomesillas, poniendoestanterias, poniendosofas) para este fin.

Pero sin duda, lo que mas carga de estructuras de datos produjo fue la selección. Cuando hagamos clic sobre la puerta de un armario o sobre los cajones de mesillas y armarios, se abrirán. Para esto lo primero es tener asociados identificadores a los objetos relevantes, cosa que hicimos en la función de dibujado de los mismos. También era necesario controlar el estado de las puertas de los armarios (abriéndose o cerrándose) y los cajones, así como el grado de apertura de los mismos. Para esto se usaron los correspondientes conjuntos de variables (habrá tantas como muebles del tipo, por lo que se declaro un vector de 50 componentes para cada una de ellas): abriendoc1a, cerrandoc1a, abriendoc1m, cerrandoc1m, abriendoc2a, cerrandoc2a, abriendoc2m, cerrandoc2m, abriendop1a y cerrandop1a para los estados de puertas y armarios, y rotp1, rotp2, rotc1a, rotc2a, rotc1m, rotc2m para el grado de apertura.

Cuando insertábamos muebles nos dimos cuenta de que podrían colisionar los unos con los otros, por lo que al visualizar el resultado no era nada agradable. Para no permitir las colisiones, usamos una serie de variables booleanas, que cambiarán de valor si es necesario en el momento que soltemos el ratón, y si están a 1 no permiten la inserción. Sus nombres son prohibidomesas, prohibidosillas, prohibidoarmarios, prohibidocamas, prohibidosofas, prohibidoestanterias, prohibidomesillas.

Por ultimo, añadimos los parámetros necesarios para la visualización, como las rotaciones de la cámara (view_rotx, view_roty, view_rotz), entre otras (Dist es la distancia de la cámara; ViewX y ViewY controlan el tamaño de la ventana). Estas variables también se fueron añadiendo al principio del programa.

Una vez teníamos las estructuras de datos, pasamos a la implementación de las funciones. Aprovechamos algunas de programas anteriores (como la de redimensionado de la ventana, las funciones de transformación de visualización o las de movimiento de cámara mediante la pulsación de una tecla), pero la mayoría tendríamos que implementarlas desde cero. A continuación las comentamos brevemente:

- seleccionMenu: se usa cuando el usuario ha escogido alguna de las opciones del menú. Lo único que hace es poner al día las variables que se vean afectadas por la elección del usuario y redibujar (si el modo es de visualizado dibujará en 3D; caso contrario usará una perspectiva paralela para ver la habitación desde arriba y poner muebles), o salir del programa si esa era la opción.
- setViewTrans: Inicializa la matriz de proyección y llama a viewTrans para la transformación de visualización.
- ViewTrans: Inicializa la transformación de visualización, llamando a glFrustrum y colocando la cámara en el lugar apropiado, siempre en perspectiva 3D. No inicializa la matriz de proyección, porque en la selección la necesitaremos.
- setViewTransOrtho: hace lo mismo que setViewTrans, pero empleando perspectiva paralela (hace glOrtho en lugar de glFrustrum). La usaremos cuando estemos colocando muebles, para tener una buena visión del escenario.
- creaMenu: como su nombre indica, define los elementos del menú del que antes hablamos. Desde el programa principal la llamaremos para que glut una las opciones a dicho menú.
- Box: dibuja un cubo con las dimensiones que se le pasan como parámetros. Cuatro de las caras las crea como una tira de cuadriláteros, y las laterales las añade como cuadriláteros simples
- Mibox: Basándose en la anterior, crea un conjunto de cajas de grosor 0.1 que definen un cubo con una cara lateral hueca. Todas las piezas de este cajón se crean con el procedimiento box anterior
- Mesa: Dibuja, basándose en el procedimiento box, una mesa .
- Silla: Dibuja, basándose en el procedimiento box, una silla.
- Cama: Dibuja, basándose en el procedimiento box, una cama.
- Sofa: Dibuja, basándose en el procedimiento box, un sofá.
- Estantería: Dibuja, basándose sólo en el procedimiento mibox, una estantería compuesta de 3 cajones.
- Cajon: crea un cajón basándose en el procedimiento mibox (usando además box para el tirador del cajón), asignando además un identificador al objeto, necesario para la selección.
- Armario: basándose en los procedimientos mibox, box (para los tiradores de las puertas) y cajon, crea un armario con un identificador base pasado

como parámetro, excepto en las puertas, donde los identificadores serán diferentes (en la función pick hablaremos más del tema).

- Draw: dibuja la escena completa en el instante actual. Primero dibuja el suelo (desplazado ligeramente hacia abajo para que al colocar muebles encima no perdamos parte de ellos) como un cubo simple. A continuación recorre los vectores de muebles, y para cada uno de ellos, si hay mueble colocado, trasladamos a la posición donde se encuentre y lo dibujamos con la función correspondiente. En los casos de armarios y mesillas, al haber interacción, debe hacer más cosas. Así, compruebo la variable id, que me dice si el usuario ha hecho clic sobre uno de los elementos interactivos. De ser así actualizamos el estado del objeto (abriéndose o cerrándose). Finalmente intercambia los buffers y actualiza redibujando.
- Idle: función de animación de los objetos interactivos (puertas y cajones). Simplemente consiste en avanzar o retroceder la variable correspondiente según sea el estado del objeto (abriéndose o cerrándose). También actualizaremos el estado si el objeto queda completamente cerrado, poniendo sus variables abriendo y cerrando a 0.
- Special: función de movimiento de cámara de acuerdo a la pulsación de las teclas de cursor. Simplemente actualiza una variable de visualización si es necesario y redibuja.
- Key: función para acercar o alejar la cámara, o para rotarla sobre Z, que hace lo mismo que special, pero manipulando otras variables de visualización.
- Reshape: Redimensiona la ventana, obteniendo el viewport actual y recalculando el glFrustrum
- Init: Inicializa algunos parámetros del programa, como la iluminación.
- RatonMovido: Esta función es invocada cada vez que está moviéndose el ratón. Si el ratón no está pulsado no se hace nada. En caso contrario, si estamos poniendo un mueble, significa que debemos arrastrar el modelo por el escenario para que el usuario pueda ponerlo (antes de soltar el ratón). Para ello comprobamos la variable poniendomueble, y si está activa, averiguamos cuál es el tipo concreto de mueble que se está poniendo, actualizando su posición y redibujando la escena completa para recoger el arrastre del mueble.
- Pick: Esta función se encarga de devolver el identificador del objeto sobre el que se ha hecho clic. Adicionalmente, introducimos en la variable global id2 el identificador correspondiente a la instancia de objeto sobre la que se hizo clic (en los casos de armarios o mesillas, con los que hay interacción). No se han querido usar identificadores anidados, así que hubo que montar algunos artificios para conseguir que el conjunto funcionara. Lo primero que se hace es almacenar en i (lo que devolveremos) el valor del identificador más cercano a la posición donde se hizo clic. Para ello se reutilizó un fragmento de código de uno de los ejemplos de clase, lo cual no supuso mayor problema. Expliquemos un poco la idea del segundo identificador. Como queremos que si hacemos clic sobre la puerta de un armario sólo se abra la puerta de ese armario y no la de todos, es necesario que cada armario tenga un identificador diferente. Adicionalmente es necesario que el identificador i que acabamos de

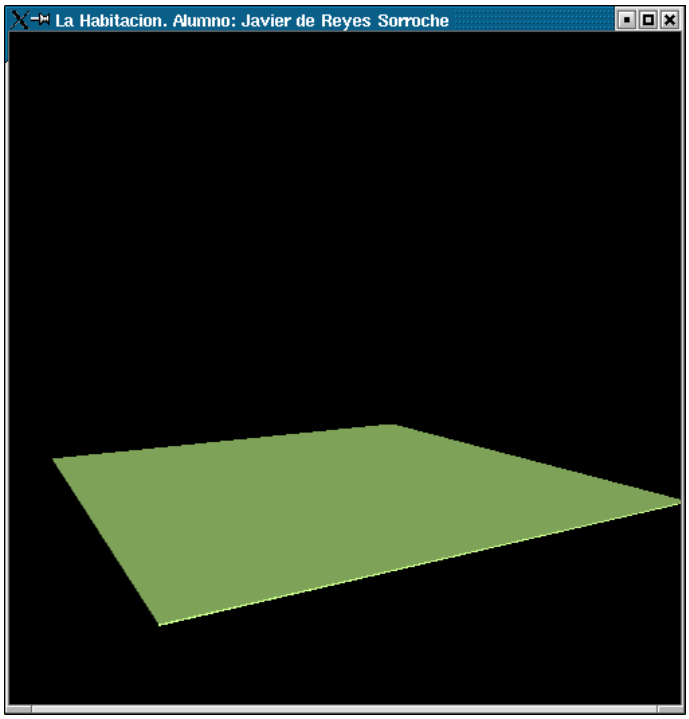
calcular sea el de una puerta o cajón, porque si no, no querremos interactuar. Por eso el resto del código está orientado a recoger el valor del identificador variable del objeto armario o mesilla para que la función draw pueda usarlo y a su vez pasarlo a la función idle que se encargará de animarlo. Esto lo lograremos haciendo varias consultas sobre el buffer de selección para asegurarnos de tener el identificador que nos interesa. En las pruebas que hicimos, funcionó bien en todos los casos, lo cual hace pensar que el efecto es el mismo que anidar identificadores.

- ClickRaton: Esta función se llama cada vez que el usuario hace clic con el ratón. El botón derecho no lo consideramos porque ya está unido al menú contextual del que hemos hablado antes. Si se trata del botón izquierdo y su estado es de pulsado, actuaremos según el estado del programa. Si estamos visualizando, llamaremos a la función pick para ver si es necesario interactuar con los objetos. Si no estamos visualizando es que estamos poniendo un mueble. Averiguaremos qué tipo de mueble estamos poniendo con la variable correspondiente, y una vez lo sepamos incrementaremos el número de muebles de ese tipo, colocando en su estructura posición asociada el campo correspondiente a la posición de la pantalla en que se hizo clic (esta posición podrá cambiar con la función ratónMovido si no se suelta el botón izquierdo mientras se mueve). Por otro lado, si estamos poniendo muebles y se suelta el ratón , el significado es que queremos dejar el mueble fijo en la posición del ratón. Pero nos surge un problema: si en esa posición ya había un mueble, no se debe permitir la inserción del mueble. Para ello usaremos la variable prohibido asociada al tipo de mueble en cuestión; recorreremos los vectores de muebles y si la envolvente de alguno de ellos interfiere en la envolvente que hemos definido para el mueble que queremos colocar (cuadrilátero porque todos los muebles pueden ser descritos sin dificultad en esta figura), la variable prohibido se pondrá a 1. Al final de la función, si esa variable se comprueba activada, eliminamos el objeto, lo que causará que no se dibuje. A efectos prácticos veremos que cuando soltamos en una posición prohibida, el objeto nuevo desaparece.

Por último, comentar el funcionamiento del programa principal. Simplemente inicializa parámetros, crea el menú, une a eventos de glut funciones que acabamos de describir y lanza el gestor de eventos de glut.

Manual de usuario

El diseñador de habitaciones construido es bastante simple de utilizar; al lanzar el ejecutable aparecerá una ventana, en principio con un escenario vacío:



Estamos en el modo de visualización, que puede controlarse con las siguientes teclas:

- Cursor izquierda: girar la cámara hacia la izquierda sobre el eje y
- Cursor derecha: girar la cámara hacia la derecha sobre el eje y
- Cursor arriba: rotar la cámara hacia arriba en el sentido del eje x
- Cursor abajo: rotar la cámara hacia abajo en el sentido del eje x
- z: rotar la cámara hacia la izquierda en el sentido del eje z
- Z: rotar la cámara hacia la derecha en el sentido del eje z
- + : Hacer zoom hacia adelante
- - : Hacer zoom hacia atrás

Si pulsamos el botón derecho del ratón aparecerá un menú contextual como el que se muestra en la siguiente figura:

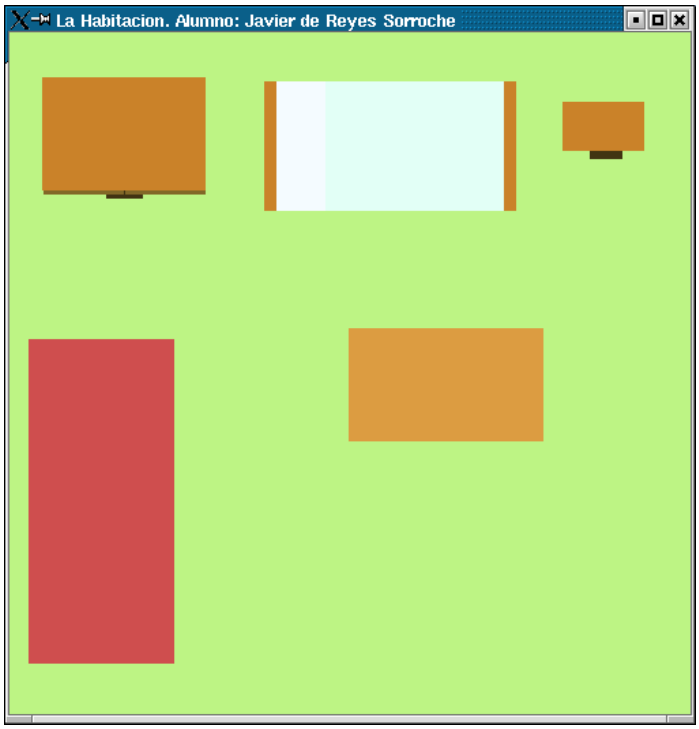
Las opciones disponibles en el menú son las siguientes:

- *Visualizar*: Pasar al modo 3D para ver la habitación en perspectiva
- *Mesa:* Entrar en el modo de poner muebles (perspectiva paralela) y colocar una mesa
- *Silla:* Entrar en el modo de poner muebles (perspectiva paralela) y colocar una silla
- *Armario*: Entrar en el modo de poner muebles (perspectiva paralela) y colocar un armario
- *Mesilla*: Entrar en el modo de poner muebles (perspectiva paralela) y colocar una mesilla de noche
- *Estantería:* Entrar en el modo de poner muebles (perspectiva paralela) y colocar una estantería
- *Sofá*: Entrar en el modo de poner muebles (perspectiva paralela) y colocar un sofá
- *Cama*: Entrar en el modo de poner muebles (perspectiva paralela) y colocar una cama
- *Salir:* Terminar la ejecución del programa

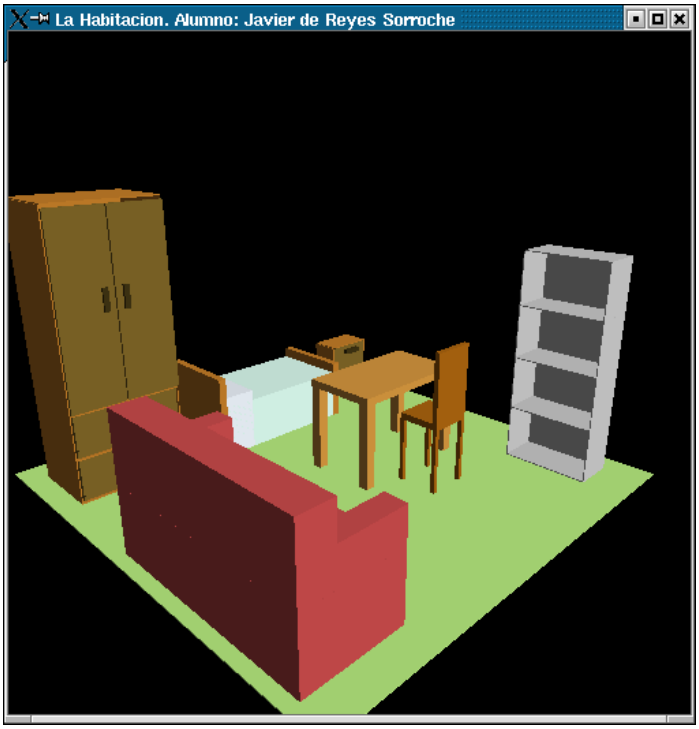
Comentemos un poco más sobre el modo de posicionamiento de muebles. Cuando seleccionemos alguna de las opciones para colocar mobiliario, la perspectiva pasará a ser paralela (desde arriba), y podremos, con el ratón, situar un mueble sobre el escenario. Haremos clic para colocar el mueble, pudiendo arrastrar el ratón para colocarlo donde deseemos. El mueble quedará fijo al soltar el ratón, excepto en el caso de que colisione con alguno de los muebles colocados previamente. En ese caso el mueble no se situará en el escenario al soltar el ratón.

Una vez colocado el mueble, su posición es inalterable; se podría haber hecho el programa de forma que se pudieran recolocar posteriormente, pero simplemente no hubo tiempo.

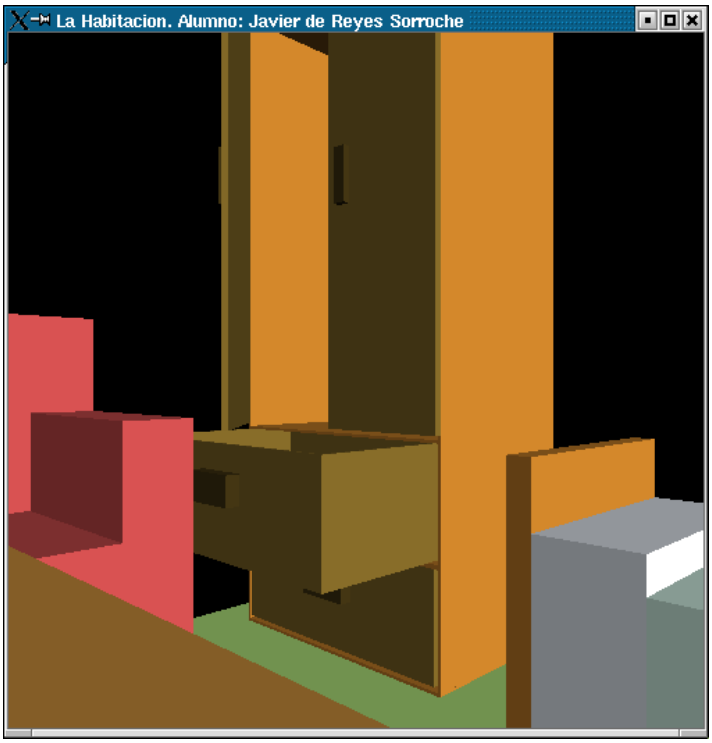
En la siguiente figura se muestra un ejemplo de la habitación mientras se están situando los muebles.



En cualquier momento durante la ejecución podremos pasar al modo de visualización simplemente seleccionando la opción correspondiente en el menú contextual. Podremos navegar por la habitación a nuestro antojo y además podremos interactuar con algunos elementos, como veremos enseguida. La siguiente figura muestra un ejemplo de habitación en el modo de visualizado.



Adicionalmente, se quiso añadir algo de interactividad al programa. Así, cuando estemos en el modo de visualización, podremos hacer clic sobre las puertas y cajones de los armarios, que se abrirán si hacemos clic una vez y se cerrarán si hacemos clic cuando se están abriendo. Si no hacemos clic, se cerrarán solas al llegar a abrirse del todo. Lo mismo ocurre con los cajones de las mesillas de noche. En esta figura mostramos un ejemplo de interactividad en acción:



En principio, esta es toda la funcionalidad del programa. Con más tiempo el resultado habría sido mejorable, pero en todo caso es un ejemplo de lo que se puede llegar a hacer con unos pocos conocimientos sobre OpenGL.