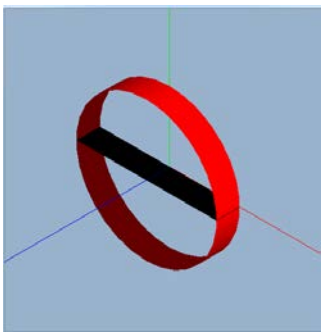
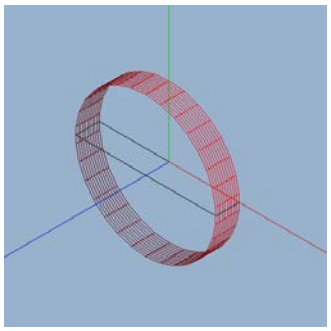
Gráficos por computador
Máster en Ingeniería Informática
Curso 18-19. Práctica 2

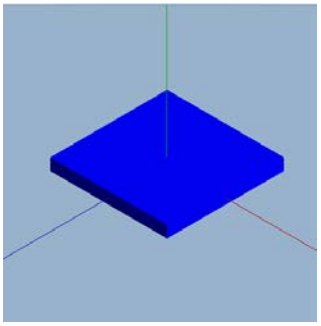
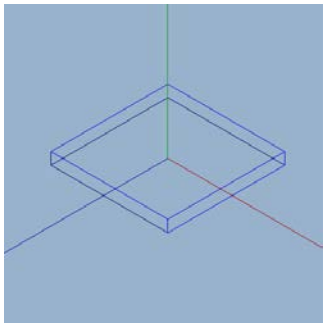
1. Define una nueva entidad **`Rotor`** cuyos elementos se renderizan al "rotor de un dron" como el que se muestra en las siguientes capturas:

Se trata de un cilindro (entidad cuádrica) rojo con una paleta que es un rectángulo negro. Cuida que la paleta no se salga del cilindro.

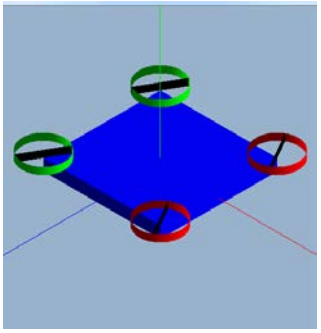
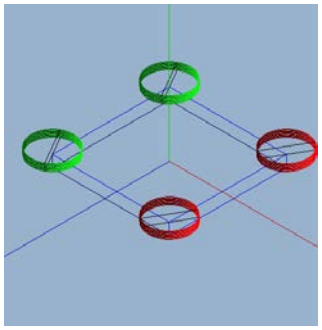
2. Añade animación a la clase **`Rotor`** de forma que la paleta gire. Haz depender el giro de un booleano de forma que, cuando el booleano es `true/false`, la paleta gira en sentido horario/antihorario. Solamente debe girar la paleta, no el cilindro. Se ve bien que el cilindro no gira cuando el rotor se dibuja en modo armazón (figura de arriba a la derecha).

3. Define una nueva entidad **`Chasis`** cuyos elementos se renderizan al "chasis de un dron" como el que se muestra en las siguientes capturas:

Se trata de un cubo azul, con tapas por arriba y por abajo, convenientemente escalado.

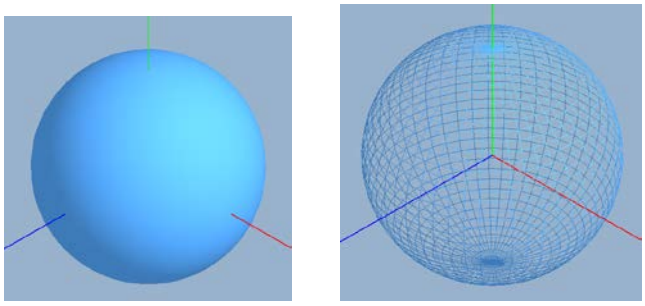
4. Define una nueva entidad **`Dron`** cuyos elementos se renderizan a "un dron" como el que se muestra en las siguientes capturas:

El dron tiene dos rotores verdes y dos rojos y un chasis azul.

5. Añade animación a la clase **`Dron`** teniendo en cuenta que, como en todos los drones de cuatro rotores (que se precien), de los rotores de un mismo lado, uno gira en sentido horario y otro en antihorario.

6. Define una nueva entidad **`Esfera`** cuyos elementos se renderizan a una esfera **obtenida por revolución** a partir de un perfil semicircular. Tanto el número de puntos del perfil (`m`), que marcará el número de paralelos de la esfera, como el número de muestras que se tomen (`n`), que marcará el número de meridianos, así como el radio de la esfera, serán atributos de la clase **`Esfera`**.



7. Modifica la clase **Camara** a fin de que cuente con los 21 atributos que aparecen en las transparencias. Algunos de ellos es posible que ya estuvieran. Define la funcionalidad que aparece en las transparencias, a saber:

- **void setAxes()**: calcula **u**, **v**, **front** usando **row()**
- **void setVM()**: invoca **lookAt()**, lo deja en **viewMat** y llama a **setAxes()**
- **void set3D()**: hace **eye**=**(frente, frente, frente)**, **frente = radio*cos(ang)** y **ang=-45°**, y llama a **setVM()**
- **void setCenital()**: cambia **eye/up** para tener una vista cenital y llama a **setVM()**

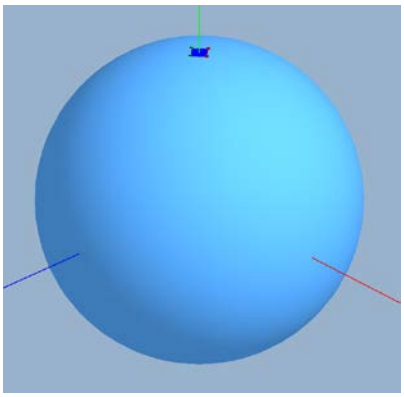
Incorpora la siguiente funcionalidad que aparece definida en las transparencias:

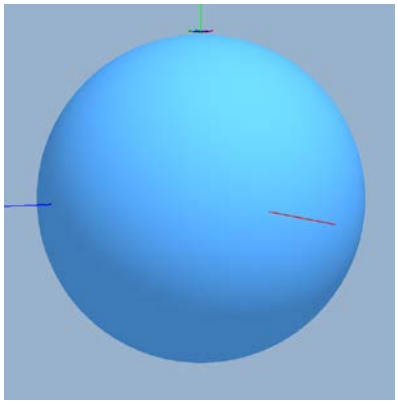
- **void moveLR(GLdouble cs)**: mueve **eye/look** una distancia **right*cs**. Análogamente **moveFB(cs)**, usando **front*cs**; y **moveUD(),** usando **upward*cs**
- **void lookLR(GLdouble cs)**: mueve **look** una distancia **right*cs**. Similarmente **lookUD(cs)** con **upward*cs**
- **void orbit(GLdouble ax, Gldouble ay)**: orbita la cámara alrededor de **look**
- **void uploadPM():** fija **projMat**, según el valor del booleano **orto**
- **void uploadScale(Gldouble s):** modifica **factScale**

8. Incorpora a **main** los eventos de ratón **mouse()** y **motion()** tal como aparecen definidos en las transparencias. Recuerda que el botón derecho del ratón debe desplazar la cámara y el botón izquierdo la debe hacer orbitar alrededor de **look**.

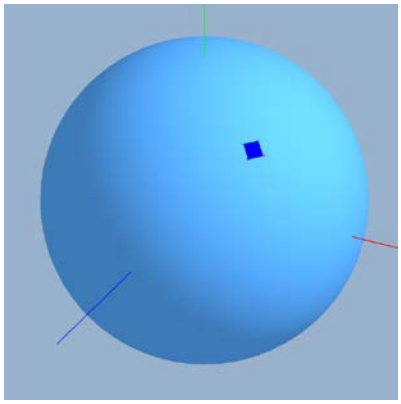
9. Incorpora a **main** el evento de ratón **mouseWheele()** tal como aparece definido en las transparencias. Recuerda que con la tecla `CONTROL` pulsada, la cámara debe hacer un zoom mientras que sin estar pulsada debe desplazarla en la dirección de vista.

10. En **`init()`** de la clase **`Scene`**, define una escena que contenga una esfera grande, con su malla definida por revolución, y un dron pequeño situado en el polo norte de la esfera, muy cercano a ella, pero sin tocarla, tal como aparece en las capturas de abajo.



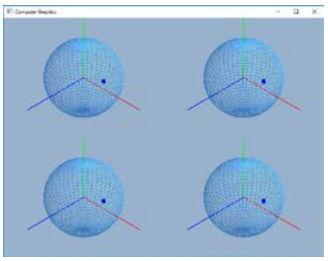


11. Pon en comentario la funcionalidad actual (`roll`, `yaw`, `pitch`) de las teclas especiales. Define un método **`move()`** en la clase **`Scene`** de forma que, invocado con las teclas **`GLUT_KEY_UP/GLUT_KEY_DOWN`**, mueva el dron arriba y abajo, sin perder su pequeña separación de la esfera, a lo largo del meridiano de longitud 45º oeste, tal como se intuye que hace en la captura de más abajo. Se supone que las paletas de los rotores del dron se mueven, según se desplaza este.



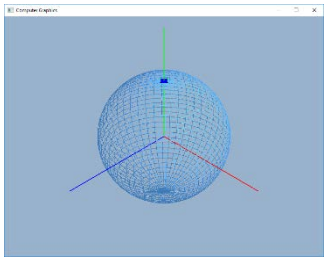
12. **(Opcional)** Haz que las teclas especiales permitan desplazar el dron por cualquier lugar de la esfera. Para ello, las teclas **`GLUT_KEY_LEFT/ GLUT_KEY_RIGHT`** mueven el dron por el paralelo en el que se encuentra, mientras que **`GLUT_KEY_UP/GLUT_KEY_DOWN`** lo mueven por su meridiano.

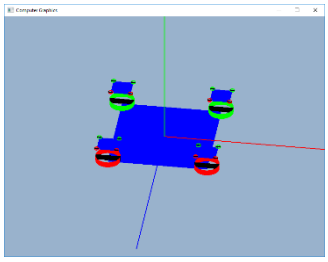
13. Programa el embaldosado explicado en clase de forma que se pueda activar mediante la tecla '**`h`**' (**y desactivar**, es decir, regresar a la escena tal como se muestra inicialmente) la renderización de la escena en cuatro puertos de vista iguales. Los eventos de ratón y el movimiento del dron por la esfera se tienen que mantener en todos los puertos de vista por igual, como cuando solo hay uno.



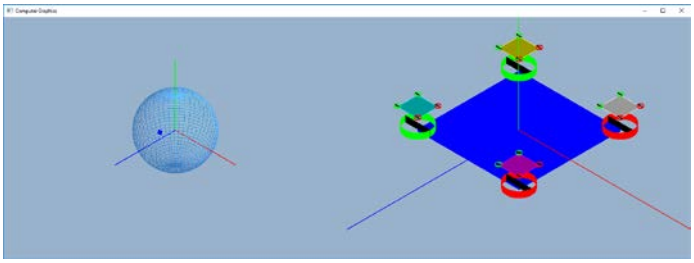
14. Para hacer este apartado debes declarar variables en **`main`** para manejar dos cámaras, dos puertos de vista y dos escenas diferentes.

Construye dos escenas. Una es la que se viene manejando hasta ahora, con una gran esfera, cuya malla se ha hecho por revolución, y un pequeño dron en su polo norte. La otra es una escena auxiliar que tiene un dron junto con un pequeño dron encima de cada uno de sus rotores. Mira las figuras adjuntas.





Habrá dos modos de renderización. Uno es el normal, y se muestra la escena de la izquierda, y el otro es el modo con dos puertos de vista. El paso a este modo de renderización se realiza con la tecla **`'j'`** y se usa una variable global booleana **`twoPorts`** para controlarlo. Esta variable te permitirá, por ejemplo, en **`display()`**, saber si estás en modo "normal" o si tienes que renderizar las dos escenas, una en cada puerto de vista. Los dos puertos de vista, cuando se muestren, lo harán el segundo a la derecha del primero, cada uno de ellos ocupando la mitad de la ventana cliente.



Debes modificar la programación de los eventos de ratón de forma que este pueda actuar independientemente en cada puerto de vista, según se halle en uno u otro. También debes reprogramar **`resize()`** y **`display()`** a fin de que, al redimensionar la ventana cliente, las nuevas dimensiones de los puertos de vista no deformen su escena. Para ello usa el comando **`glutGet(GLUT_WINDOW_WIDTH)`** que proporciona el ancho de la ventana cliente. El alto se calcula a partir de este, tal como se hace en el método **`embaldosar()`**.

Por último, debes programar que la cámara del puerto de vista de la derecha gire alrededor del eje `Y` de la escena (movimiento `yaw` de la cámara, cuyo código es deseable que no hayas eliminado), al pulsar la tecla **`'k'`**.

Como es obvio, y como en el apartado **13**, la tecla **`'u'`** hace **`update()`**, de una o de las dos escenas, según el modo en que estemos.

15. Migra tu proyecto al modelo jerárquico. Para ello debes crear la clase **`CompoundEntity`** con su atributo nuevo:

```
std::vector<Entity*> grObjects;
```

y su accesora nueva a la **`i`**-ésima entidad:

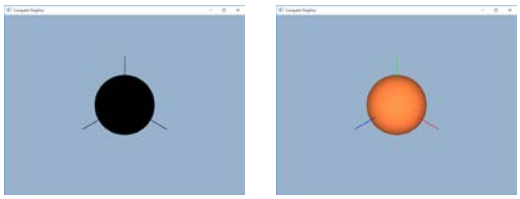
```
Entity* getEntity(int i);
```

Define, en esta clase, **`~CompoundEntity()`**, **`render()`** y **`update()`** como se explica en las transparencias.

16. Haz que la clase **`Scene`** herede de **`CompoundEntity`**. Elimina los métodos **`render()`** y **`update()`** de **`Scene`**. La ejecución del proyecto debe seguir siendo la que era.

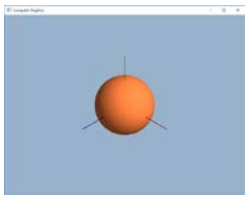
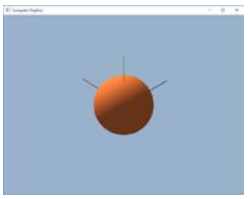
17. Haz que las clases **`Rotor`** y **`Dron`** hereden de **`CompoundEntity`**. Elimina el método **`render()`** de ambas clases. La ejecución del proyecto debe seguir siendo la que era.

18. Activa el modo **`ColorMaterial`** en **`init()`** de **`Scene`**. Considera una escena que tiene solamente unos ejes y una esfera por revolución. La esfera no tiene material sino color **`(0.8, 0.4, 0.2)`**. Añade luz ambiente global dada por la terna RGB **`(0.5, 0.5, 0.5)`**. Haz que esta luz se apague (terna negra)/encienda (terna gris) con las teclas **`n/m`**. En ausencia de cualquier otra luz, la escena se verá negra (abajo a la izquierda). Con la luz ambiente global activada y con la luz direccional del apartado **19** activada, la escena se verá como abajo a la derecha.

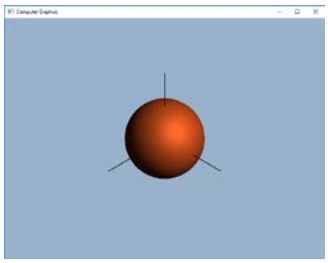
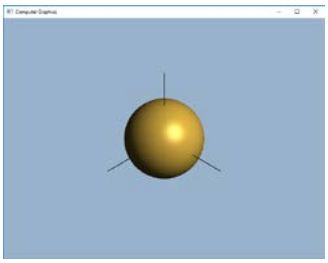
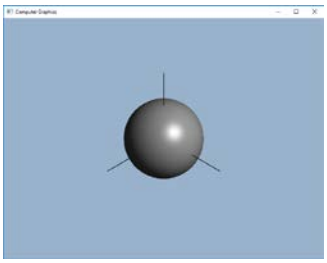


19. Se continúa con la escena del apartado anterior. Se mantiene activado el modo **`ColorMaterial`**. Añade a **`init()`** de **`Scene`** una luz direccional **fija** cuyo vector de incidencia es **`(-1, -1, 0)`**. Acuérdate de normalizar este vector añadiendo **`glEnable(GL_NORMALIZE)`** al **`init()`**. Recuerda que todas las luces que definas actúan desde su creación pues así aparece en el código de las transparencias. Las componentes difusa, ambiente y especular de esta luz direccional son **`(1.0, 1.0, 1.0)`**, **`(0.2, 0.2, 0.2)`** y **`(0.5, 0.5, 0.5)`**, respectivamente.

El resultado de la activación de esta luz debería ser una esfera como la de abajo a la izquierda. Recuerda que cuando vayas con la cámara al lado opuesto de la escena debes ver algo como lo de abajo a la derecha, pues la luz direccional es fija. Observa que la esfera de abajo a la izquierda no está iluminada igual que la de arriba a la derecha.

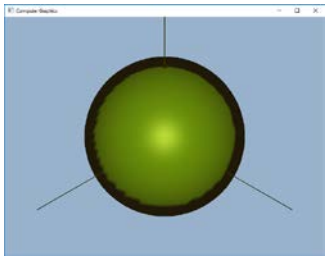
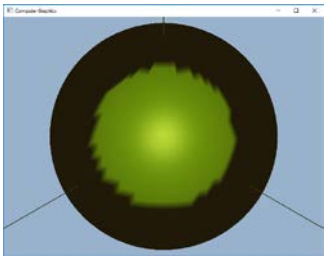
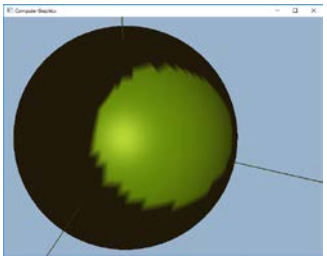



20. Se continúa con la escena del apartado anterior. Desactiva el modo **`ColorMaterial`**. Elimina el color de la esfera. Añade un atributo **`Material*`** a la clase **`Esfera`**. Dale valor por defecto y cámbialo con los métodos **`setCopper()`**, **`setGold()`** y **`setSilver()`** (estos dos últimos los tienes que definir de forma similar a como aparece definido el primero de ellos en las transparencias). Deberás obtener esferas como las de abajo, respectivamente, cuando la luz ambiente global está apagada y la luz direccional, activada.

21. Pon que la escena se muestre en proyección perspectiva. Haz que la esfera sea dorada invocando con ella **`setGold()`**, es decir, la esfera no tiene color, sino material. Añade una luz de minero a la cámara que sea además foco. Las componentes difusa y especular de la luz de este foco serán **`(0.4, 0.8, 0.0)`** y **`(0.5, 0.5, 0.5)`**, respectivamente. Recuerda cómo ha de ser la componente ambiente de esta luz de la cámara, teniendo en cuenta que es un foco. Define esa componente ambiente del foco de la cámara.

Apaga la luz ambiente global y la luz direccional fija de la escena. Aproxímate a la esfera con la rueda del ratón (sin pulsar la tecla **`CTRL`**). Como la proyección es perspectiva, la esfera aumentará de tamaño. Como la cámara se aproxima a la esfera, pasarás de una imagen como la de abajo a la izquierda, con una esfera verde manzana (date cuenta de que la esfera bajo esta luz ya no es dorada) y el cono de luz focal iluminando gran parte de ella, a una esfera como la de abajo en el centro, donde el cono ilumina solamente una parte de la esfera, o abajo a la derecha, en la que la cámara se ha desplazado a la izquierda pulsando el botón izquierdo del ratón.

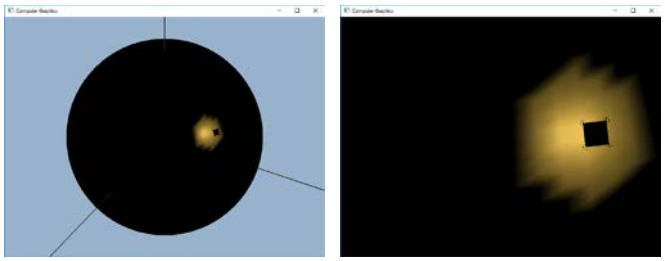




22. En este apartado la escena está compuesta por unos ejes, una esfera y un dron situado encima del polo norte de la esfera. Recuerda que estamos manejando un objeto de la clase **`Esfera`**, no de la clase **`Sphere`**, con lo que, si suponemos que

la esfera que hemos creado no la hemos movido, los polos norte y sur estarán en la parte positiva y negativa del eje **Y**, respectivamente.

Primero, desactiva el modo **ColorMaterial**. Haz que la esfera siga siendo dorada, como en el apartado anterior. Añade un foco al dron que debe acompañar, obviamente, a este en su movimiento alrededor de la esfera. El foco del dron es una luz que se declara como atributo protegido de la clase **Dron**. Las componentes difusa y especular de la luz de este foco serán **(1.0, 1.0, 1.0)** y **(0.5, 0.5, 0.5)**, respectivamente.

Cuando se apague la luz direccional, la de la cámara y el ambiente global se ponga a negro, el foco del dron encendido iluminará parte de la esfera tal como se muestra en la captura de abajo a la izquierda (a la derecha se ve un detalle).



Ahora activa el modo **ColorMaterial** y pon, como color de la esfera, la terna del coeficiente de reflexión difusa del oro que has usado en **setGold()**. Enciende la luz direccional y resultará la siguiente imagen. Más abajo se ve un detalle.