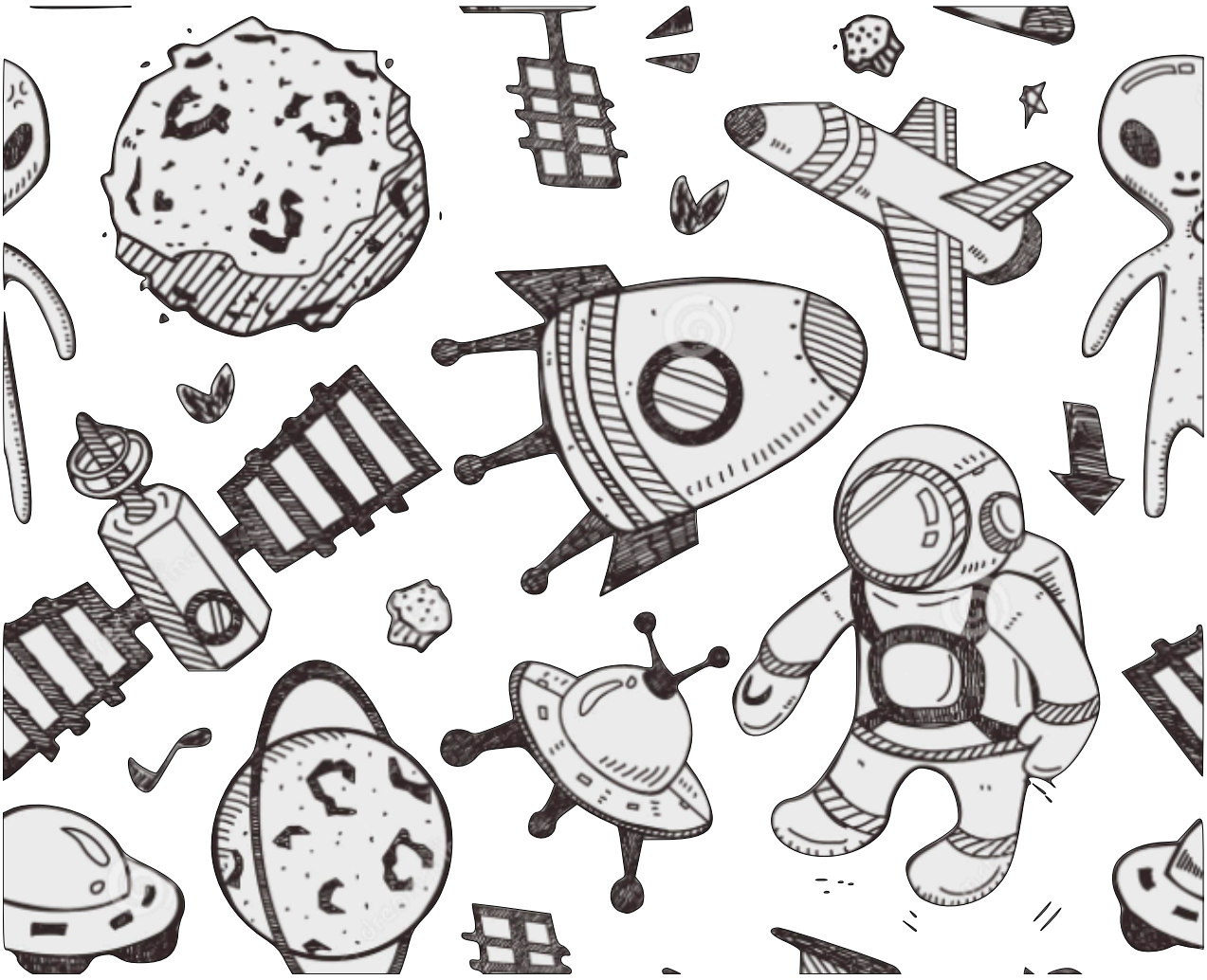


Proyecto Final de Prácticas.
Destrucción de Planetas.
Diseño.

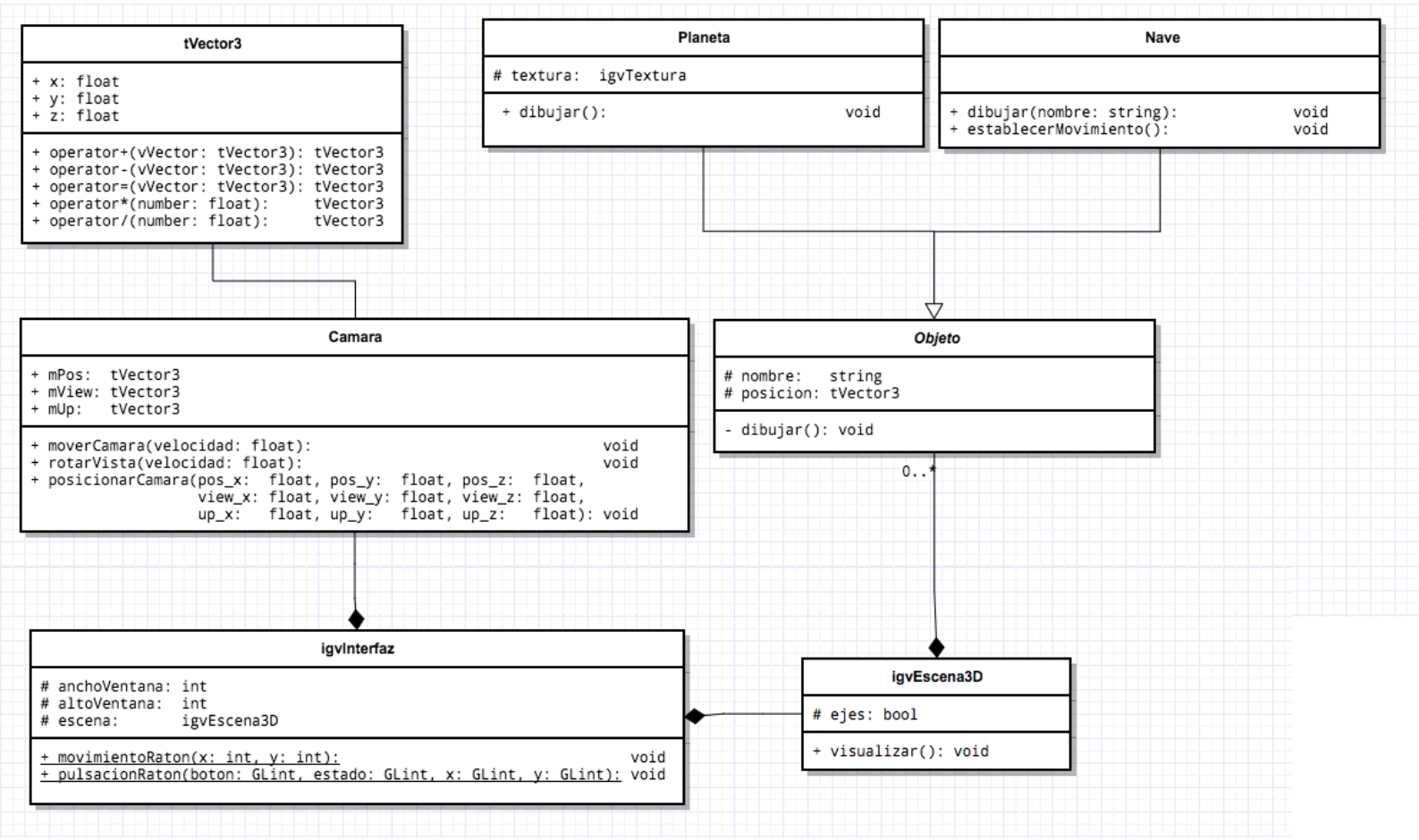
Daniel Moya Leiva. 77360609V.

DIAGRAMA UML

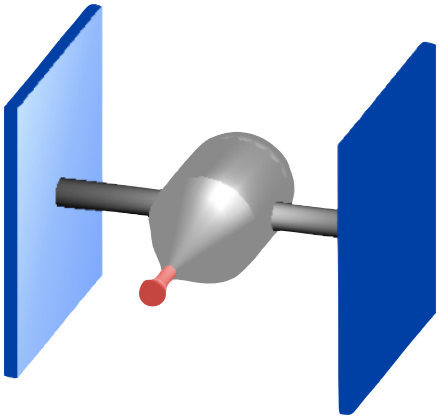
Usaremos una clase llamada *tVector3* en lugar de *igvPunto3D* para almacenar posiciones, pues hemos sobrecargado los operadores **+**, **-**, **=**, *****, y **/**. Esto nos da mucha libertad para trabajar con la cámara y las posiciones, y básicamente su funcionalidad es la misma.

Por otro lado, cambiaremos el uso de *igvCamara* por el de *Camara* puesto que tiene una implementación más sencilla de una cámara en primera persona (FPP en adelante). Esta clase tendrá tres métodos básicos: *moverCamara* para cambiar la posición en la que nos encontramos; *rotarVista* para cambiar el campo de visión; y *posicionarCamara* para establecer el inicio de la cámara.

En cuanto a los objetos que habrá por la escena, usaremos una clase abstracta como es *Objeto* de la que heredan las clases *Planeta* y *Satelite* que serán los encargados de mover y dibujar los elementos que están en la escena.

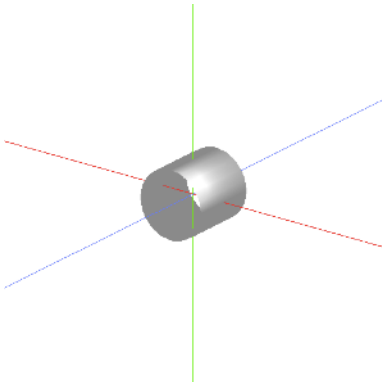


GRAFO DE ESCENA.



Satélite.

```
GLfloat gris[]      = {0.5,   0.5,   0.5};
GLfloat negro[]     = {0.0,   0.0,   0.0};
GLfloat azul[]      = {0.0,   0.15,  0.61};
GLfloat rojo[]      = {0.8,   0.15,  0.15};
```

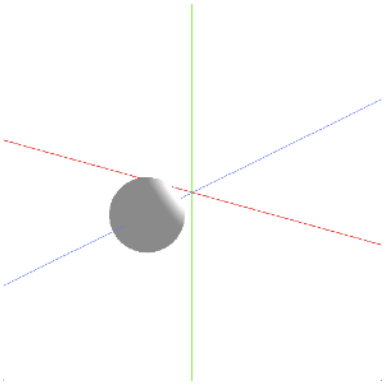


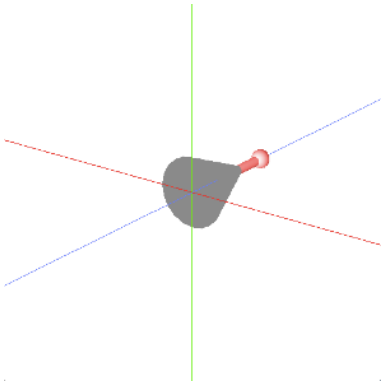
```
// Cuerpo

glTranslatef(0, 0, -2);

GLUquadricObj *cuerpo;
cuerpo = gluNewQuadric();

gluQuadricDrawStyle(cuerpo, GLU_FILL);
gluCylinder(cuerpo, 2, 2, 4, 20, 20);
```

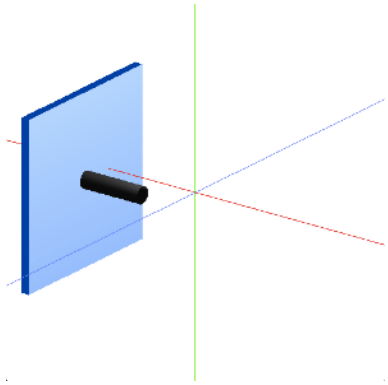
```
// Parte trasera

glPushMatrix();
    glTranslatef(0, 0, 4);

    glutSolidSphere(2, 50, 50);
glPopMatrix();
```

```
// Parte delantera

glPushMatrix();

  // Cabeza

  glTranslatef(0, 0, -4);

  GLUquadricObj *cabeza;
  cabeza = gluNewQuadric();

  gluQuadricDrawStyle(cabeza, GLU_FILL);
  gluCylinder(cabeza, 0.25, 2, 4, 20, 20);

  // Antena

  glTranslatef(0, 0, -2);
  GLUquadricObj *antena;
  antena = gluNewQuadric();

  glMaterialfv(GL_FRONT, GL_EMISSION, rojo);
  gluQuadricDrawStyle(antena, GLU_FILL);
  gluCylinder(antena, 0.25, 0.25, 2, 20, 20);

  glutSolidSphere(0.5, 50, 50);
glPopMatrix();
```

```
// Brazo Derecho

glMaterialfv(GL_FRONT, GL_EMISSION, negro);
glPushMatrix();

  // Brazo

  glTranslatef(6, 0, 2);
  glRotatef(-90, 0, 1, 0);

  GLUquadricObj *brazoder;
  brazoder = gluNewQuadric();

  gluQuadricDrawStyle(brazoder, GLU_FILL);
  gluCylinder(brazoder, 0.5, 0.5, 4, 20, 20);

  // Panel

  glMaterialfv(GL_FRONT, GL_EMISSION, azul);
  glScalef(2, 2, 0.1);
  glutSolidCube(5);

glPopMatrix();
```





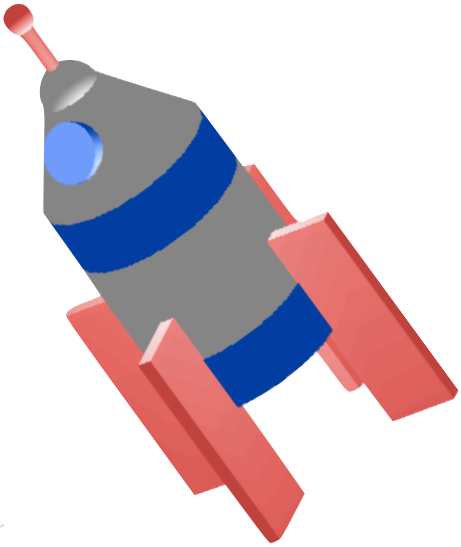


Implementación Satélite.

```
glPushMatrix();

  glMaterialfv(GL_FRONT, GL_EMISSION, gris);
  // Cuerpo
  glTranslatef(0, 0, -2);
  GLUquadricObj *cuerpo;
  cuerpo = gluNewQuadric();
  gluQuadricDrawStyle(cuerpo, GLU_FILL);
  gluCylinder(cuerpo, 2, 2, 4, 20, 20);


  // Parte trasera
  glPushMatrix();
    glTranslatef(0, 0, 4);
    glutSolidSphere(2, 50, 50);
  glPopMatrix();


  // Parte delantera
  glPushMatrix();
    // Cabeza
    glTranslatef(0, 0, -4);
    GLUquadricObj *cabeza;
    cabeza = gluNewQuadric();
    gluQuadricDrawStyle(cabeza, GLU_FILL);
    gluCylinder(cabeza, 0.25, 2, 4, 20, 20);
    // Antena
    glTranslatef(0, 0, -2);
    GLUquadricObj *antena;
    antena = gluNewQuadric();
    glMaterialfv(GL_FRONT, GL_EMISSION, rojo);
    gluQuadricDrawStyle(antena, GLU_FILL);
    gluCylinder(antena, 0.25, 0.25, 2, 20, 20);
    glutSolidSphere(0.5, 50, 50);
  glPopMatrix();
```

```
  // Brazo Izquierdo
  glMaterialfv(GL_FRONT, GL_EMISSION, negro);
  glPushMatrix();
    // Brazo
    glTranslatef(-6, 0, 2);
    glRotatef(90, 0, 1, 0);
    GLUquadricObj *brazoizq;
    brazoizq = gluNewQuadric();
    gluQuadricDrawStyle(brazoizq, GLU_FILL);
    gluCylinder(brazoizq, 0.5, 0.5, 4, 20, 20);
    // Panel
    glMaterialfv(GL_FRONT, GL_EMISSION, azul);
    glScalef(2, 2, 0.1);
    glutSolidCube(5);
  glPopMatrix();


  // Brazo Derecho
  glMaterialfv(GL_FRONT, GL_EMISSION, negro);
  glPushMatrix();
    // Brazo
    glTranslatef(6, 0, 2);
    glRotatef(-90, 0, 1, 0);
    GLUquadricObj *brazoder;
    brazoder = gluNewQuadric();
    gluQuadricDrawStyle(brazoder, GLU_FILL);
    gluCylinder(brazoder, 0.5, 0.5, 4, 20, 20);
    // Panel
    glMaterialfv(GL_FRONT, GL_EMISSION, azul);
    glScalef(2, 2, 0.1);
    glutSolidCube(5);
  glPopMatrix();

glPopMatrix();
```

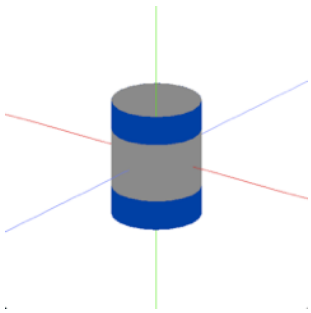
GRAFO DE ESCENA.



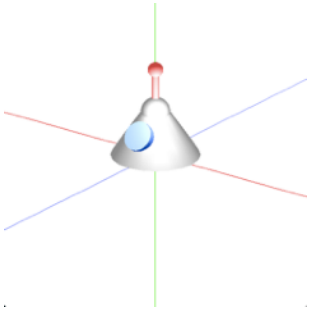
Cohete.

```
GLfloat gris[]      = {0.5,   0.5,  0.5};
GLfloat rojo[]      = {0.8,   0.15,  0.15};
GLfloat azul[]      = {0.0,   0.15,  0.61};
GLfloat azulClaro[] = {0.4,   0.55,  1};
```

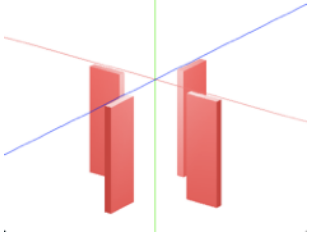
```
// Cuerpo
glMaterialfv(GL_FRONT, GL_EMISSION, gris);
glTranslatef(0, 0, -4);

GLUquadricObj *cuerpo;
cuerpo = gluNewQuadric();

gluQuadricDrawStyle(cuerpo, GLU_FILL);
gluCylinder(cuerpo, 3, 3, 8, 20, 20);

glPushMatrix();
  glMaterialfv(GL_FRONT, GL_EMISSION, azul);

  gluCylinder(cuerpo, 3.01, 3.01, 2, 20, 20);

  glTranslatef(0, 0, 6);
  gluCylinder(cuerpo, 3.01, 3.01, 2, 20, 20);
glPopMatrix();
```



```
// Cabeza
glMaterialfv(GL_FRONT, GL_EMISSION, gris);
glPushMatrix();
  glTranslatef(0, 0, -3);

  GLUquadricObj *cabeza;
  cabeza = gluNewQuadric();

  gluQuadricDrawStyle(cabeza, GLU_FILL);
  gluCylinder(cabeza, 1, 3, 3, 20, 20);

  // Escotilla
  glPushMatrix();
    glMaterialfv(GL_FRONT, GL_EMISSION, azul);
    glRotatef(-125, 1, 0, 0);
    glTranslatef(0, -2, 0);
    GLUquadricObj *escotilla;
    escotilla = gluNewQuadric();

    gluQuadricDrawStyle(escotilla, GLU_FILL);
    gluCylinder(escotilla, 1, 1, 1, 20, 20);

  // Ventana
    glMaterialfv(GL_FRONT, GL_EMISSION, azulClaro);
    glTranslatef(0, 0, 1);

    GLUquadricObj *ventana;
    ventana = gluNewQuadric();

    gluQuadricDrawStyle(ventana, GLU_FILL);
    gluDisk(ventana, 0, 1, 50, 50);
  glPopMatrix();

  // Antena
  glPushMatrix();
    glMaterialfv(GL_FRONT, GL_EMISSION, gris);
    glutSolidSphere(1, 50, 50);

    glTranslatef(0, 0, -3);

    glMaterialfv(GL_FRONT, GL_EMISSION, rojo);
    GLUquadricObj *antena;
    antena = gluNewQuadric();

    gluQuadricDrawStyle(antena, GLU_FILL);
    gluCylinder(antena, 0.25, 0.25, 2, 20, 20);

    glutSolidSphere(0.5, 50, 50);
  glPopMatrix();
glPopMatrix();
```



```
// Alas laterales
glPushMatrix();
  glMaterialfv(GL_FRONT, GL_EMISSION, rojo);
  glRotatef(90, 1, 0, 0);

  for (int i = -4; i <= 4; i+=8) {
    glPushMatrix();
      glTranslatef(i, 8, 0);
      glScalef(0.5, 1.5, 0.1);
      glutSolidCube(5);
    glPopMatrix();
  }

  // Alas frontales
  glMaterialfv(GL_FRONT, GL_EMISSION, rojo);
  glRotatef(90, 0, 1, 0);

  for (int i = 4; i >= -4; i-=8) {
    glPushMatrix();
      glTranslatef(i, 8, 0);
      glScalef(0.5, 1.5, 0.1);
      glutSolidCube(5);
    glPopMatrix();
  }

glPopMatrix();
```



Implementación Cohete.

```
glPushMatrix();
  glScalef(0.5, 0.5, 0.5);
  glRotatef(90, 1, 0, 0);
 // Cuerpo
 glMaterialfv(GL_FRONT, GL_EMISSION, gris);
 glTranslatef(0, 0, -4);
 GLUquadricObj *cuerpo;
 cuerpo = gluNewQuadric();
 gluQuadricDrawStyle(cuerpo, GLU_FILL);
 gluCylinder(cuerpo, 3, 3, 8, 20, 20);
 glPushMatrix();
   glMaterialfv(GL_FRONT, GL_EMISSION, azul);
   gluCylinder(cuerpo, 3.01, 3.01, 2, 20, 20);
   glTranslatef(0, 0, 6);
   gluCylinder(cuerpo, 3.01, 3.01, 2, 20, 20);
 glPopMatrix();

 // Alas laterales
 glPushMatrix();
   glMaterialfv(GL_FRONT, GL_EMISSION, rojo);
   glRotatef(90, 1, 0, 0);
   for (int i = -4; i <= 4; i+=8) {
     glPushMatrix();
       glTranslatef(i, 8, 0);
       glScalef(0.5, 1.5, 0.1);
       glutSolidCube(5);
     glPopMatrix();
   }

   // Alas frontales
   glMaterialfv(GL_FRONT, GL_EMISSION, rojo);
   glRotatef(90, 0, 1, 0);
   for (int i = 4; i >= -4; i-=8) {
     glPushMatrix();
       glTranslatef(i, 8, 0);
       glScalef(0.5, 1.5, 0.1);
       glutSolidCube(5);
     glPopMatrix();
   }
 glPopMatrix();

 // Cabeza
 glMaterialfv(GL_FRONT, GL_EMISSION, gris);
 glPushMatrix();
   glTranslatef(0, 0, -3);
   GLUquadricObj *cabeza;
   cabeza = gluNewQuadric();
   gluQuadricDrawStyle(cabeza, GLU_FILL);
   gluCylinder(cabeza, 1, 3, 3, 20, 20);

   // Escotilla
   glPushMatrix();
     glMaterialfv(GL_FRONT, GL_EMISSION, azul);
     glRotatef(-125, 1, 0, 0);
     glTranslatef(0, -2, 0);
     GLUquadricObj *escotilla;
     escotilla = gluNewQuadric();
     gluQuadricDrawStyle(escotilla, GLU_FILL);
     gluCylinder(escotilla, 1, 1, 1, 20, 20);

     // Ventana
     glMaterialfv(GL_FRONT, GL_EMISSION, azulClaro);
     glTranslatef(0, 0, 1);
     GLUquadricObj *ventana;
     ventana = gluNewQuadric();
     gluQuadricDrawStyle(ventana, GLU_FILL);
     gluDisk(ventana, 0, 1, 50, 50);
   glPopMatrix();

   // Antena
   glPushMatrix();
     glMaterialfv(GL_FRONT, GL_EMISSION, gris);
     glutSolidSphere(1, 50, 50);
     glTranslatef(0, 0, -3);
     glMaterialfv(GL_FRONT, GL_EMISSION, rojo);
     GLUquadricObj *antena;
     antena = gluNewQuadric();
     gluQuadricDrawStyle(antena, GLU_FILL);
     gluCylinder(antena, 0.25, 0.25, 2, 20, 20);
     glutSolidSphere(0.5, 50, 50);
   glPopMatrix();
 glPopMatrix();
```