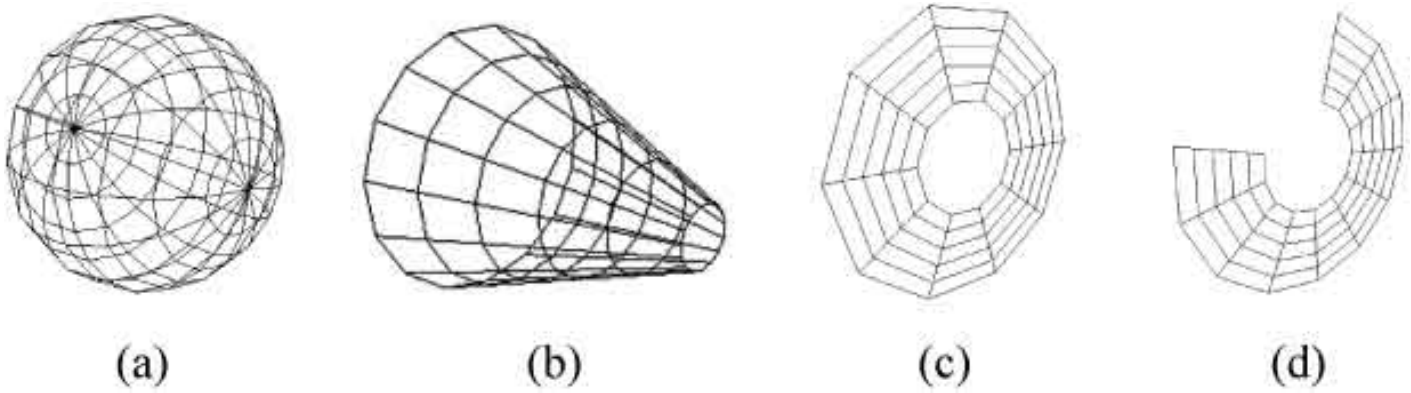
Entidades cuádricas

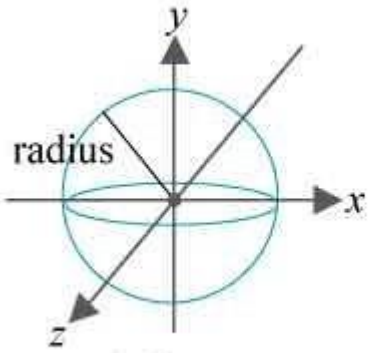
A. Gavilanes
Departamento de Sistemas Informáticos y Computación
Facultad de Informática
Universidad Complutense de Madrid

Entidades cuádricas

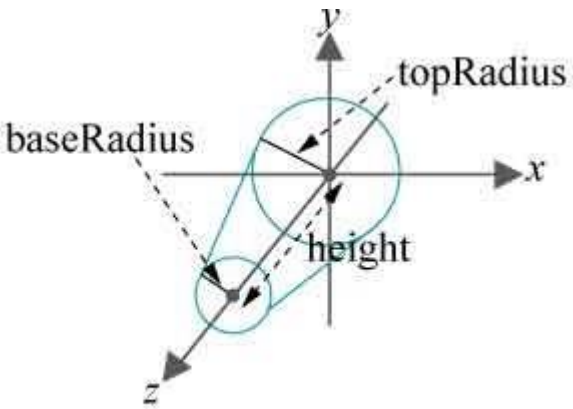


- Se proporcionan con la librería GLU.
- Se pueden crear cuatro tipos de entidades: (a) esferas, (b) cilindros, (c) discos y (d) discos parciales.
- Los entidades se declaran así: `GLUquadricObj* q;`
- Los entidades se construyen así: `q=gluNewQuadric();`
- Los entidades se destruyen así: `gluDeleteQuadric(q);`

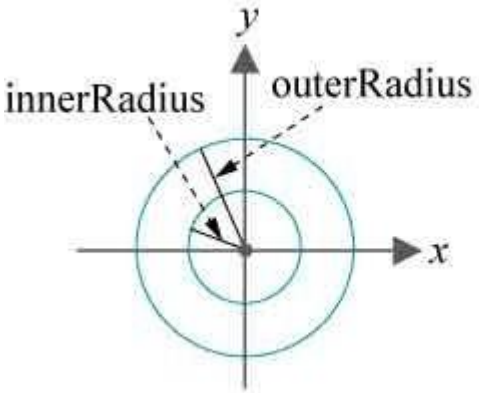
Entidades cuádricas



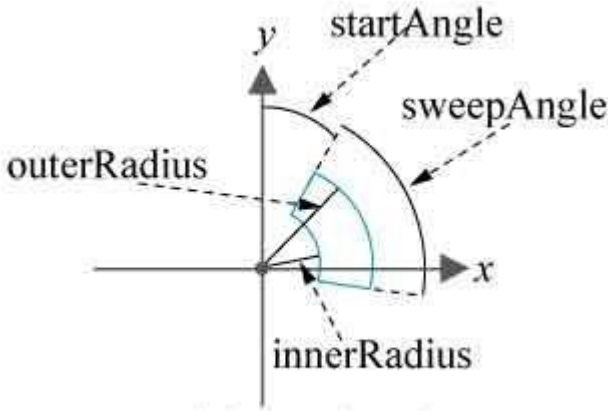
Sphere



Tapered Cylinder



Annular Disc



Partial Annular Disc

Entidades cuádricas

- Los comandos para dibujar las entidades cuádricas son:
  - **`gluSphere(q, radius, slices, stacks);`**
    - `slices=nº de meridianos; stacks=nº de paralelos`
    - `radius es GLdouble, y slices, stacks son int`
  - **`gluCylinder(q, baseRadius, topRadius, height, slices, stacks);`**
    - `Se construyen sobre el eje Z`
    - `slices=nº de lados, stacks=nº de rodajas`
    - `baseRadius, topRadius, height son GLdouble, y slices, stacks son int`
    - `Cuando cualquiera de los radios es 0 se obtienen conos`

Entidades cuádricas

- **gluDisk(q, innerRadius, outerRadius, slices, rings);**
  - Se construyen en el plano XY
  - slices=nº de lados, rings=nº de anillos
  - innerRadius, outerRadius son Gldouble, y slices, rings son int
- **gluPartialDisk(q, innerRadius, outerRadius, slices, rings, startAngle, sweepAngle);**
  - Se construyen en el plano XY
  - innerRadius, outerRadius son Gldouble; slices, rings son int, y startAngle, sweepAngle son ángulos en grados
  - Los ángulos se miden en sentido horario, mirando desde la parte positiva del eje Z, sobre el plano XY, empezando en el eje Y.

Entidades cuádricas

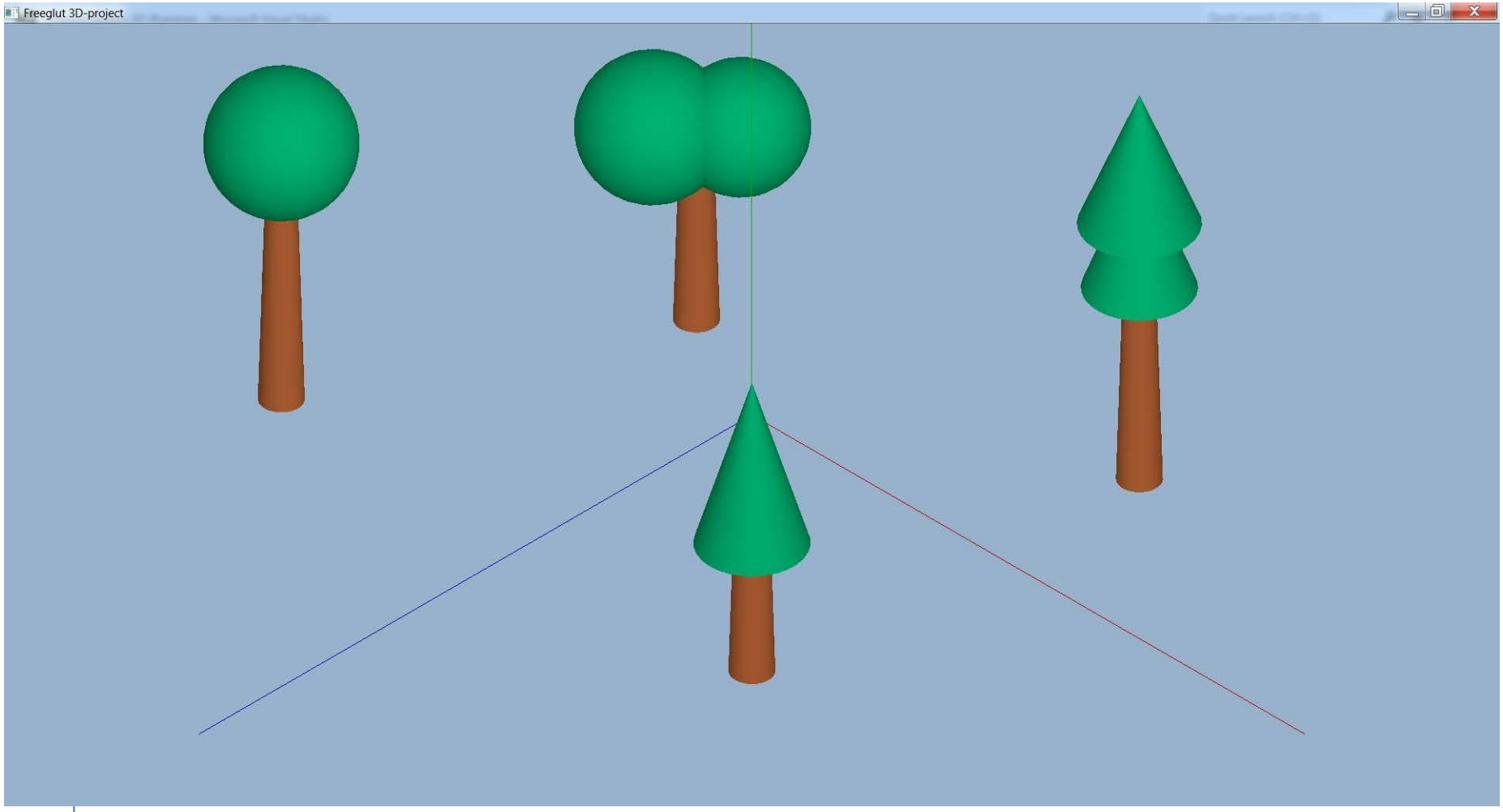
- Los comandos para especificar el modo en que se dibujan las entidades cuádricas son:

```
gluQuadricDrawStyle(q, tipoDeRecubrimiento);
```

- Los tipos de recubrimiento son:
  - **GLU_POINT**: Solamente se muestran los puntos del armazón del objeto cuádrico
  - **GLU_LINE**: Solamente se muestran las líneas del armazón del objeto cuádrico
  - **GLU_FILL**: Rellena cada cara del armazón del objeto cuádrico, teniendo en cuenta la iluminación

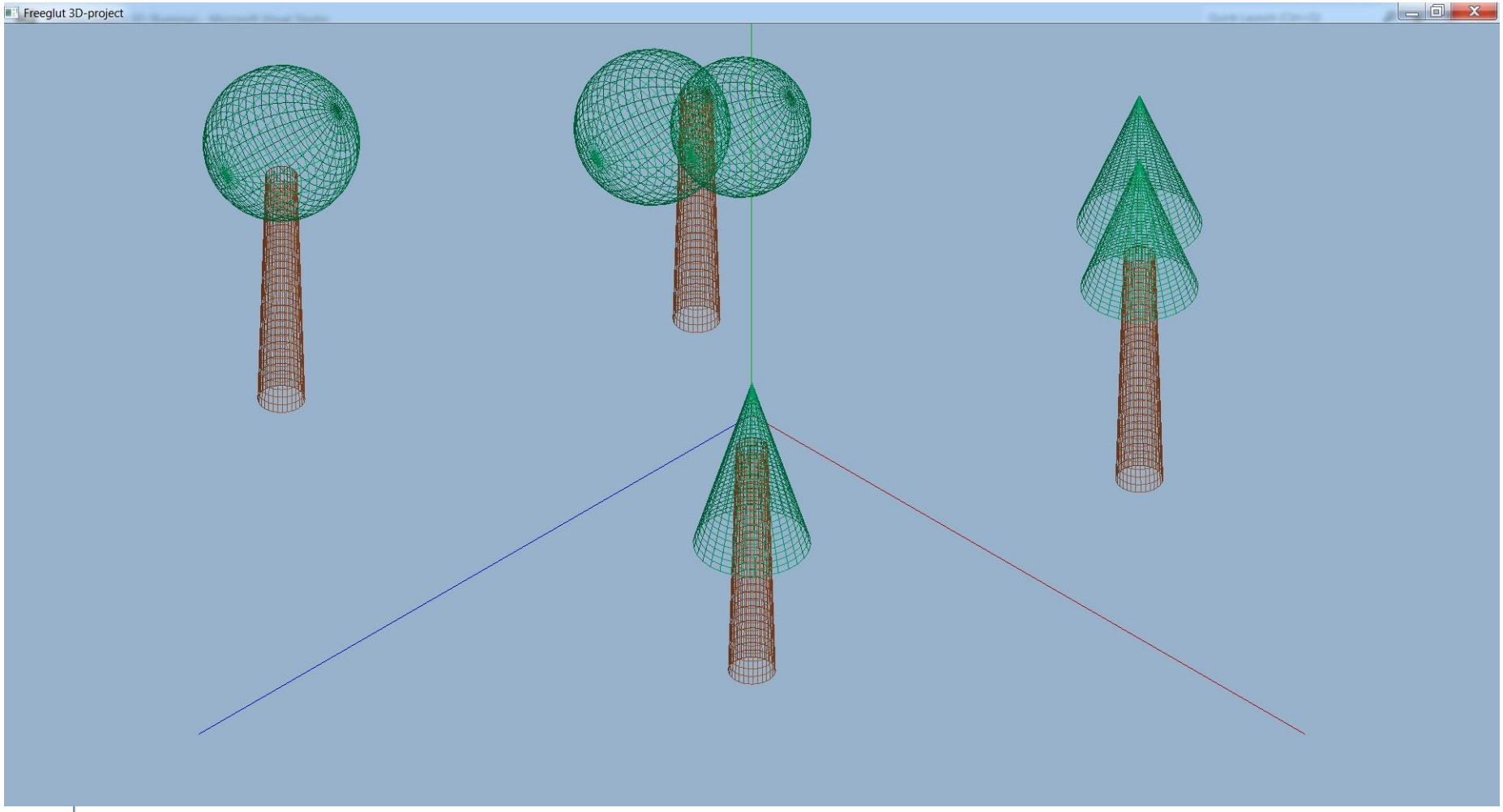
Entidades cuádricas

- `gluQuadricDrawStyle(q, GLU_FILL);`



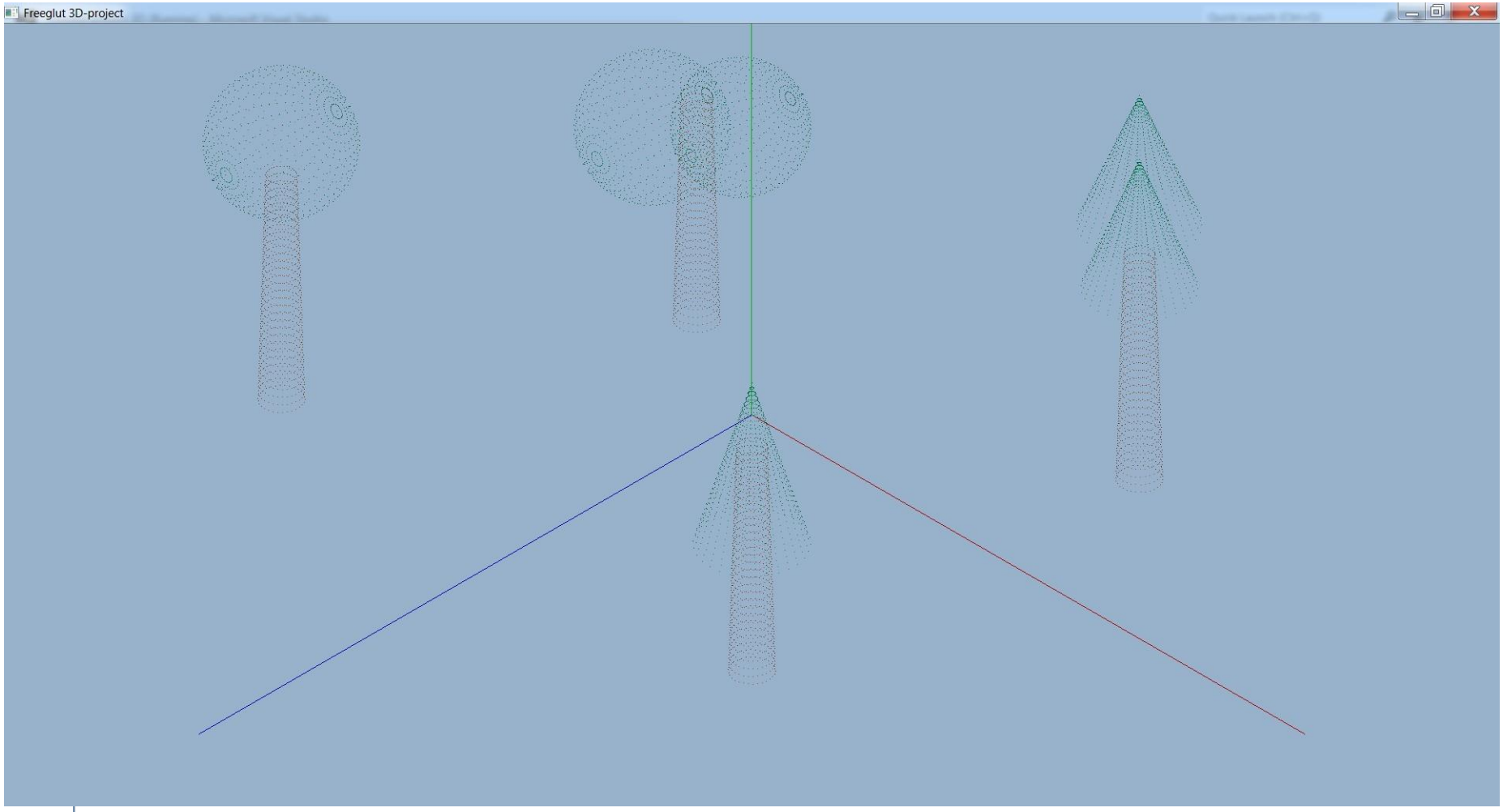
Entidades cuádricas

- `gluQuadricDrawStyle(q, GLU_LINE);`



Entidades cuádricas

- `gluQuadricDrawStyle(q, GLU_POINT);`



Entidades cuádricas en el proyecto

- La clase **QuadricEntity**

```
class QuadricEntity : public Abs_Entity {
    public:
        QuadricEntity();
        ~QuadricEntity() { gluDeleteQuadric(q); };
    protected:
        GLUquadricObj* q;
};

QuadricEntity::QuadricEntity() {
    q = gluNewQuadric();
}
```

Entidades cuádricas en el proyecto

- La clase **Sphere**

```
class Sphere : public QuadricEntity {
    public:
        Sphere(GLdouble r);   // r es el radio de la esfera
        void render(glm::dmat4 const& modelViewMat) const;
    protected:
        GLdouble r;
};
```

- Análogamente se definen las clases **Cylinder**, **Disk**, **PartialDisk**

Entidades cuádricas en el proyecto

```
Sphere::Sphere(GLdouble rr) { r = rr; }
void Sphere::render(glm::dmat4 const& modelViewMat) const {
    dmat4 aMat = modelViewMat * mModelMat;
    upload(aMat);
    // Aquí se puede fijar el color de la esfera así:
        // glEnable(GL_COLOR_MATERIAL);
        // glColor3f(...);
    // Aquí se puede fijar el modo de dibujar la esfera:
        // gluQuadricDrawStyle(q, ...);
    gluSphere(q, r, 50, 50);
    // Aquí se debe recuperar el color:
        // glColor3f(1.0, 1.0, 1.0);
}
```

Ejemplo

```
Sphere* esfera = new Sphere(100.0);
gObjects.push_back(esfera);

Cylinder* cono = new Cylinder(50.0, 0, 100.0);
glm::dmat4 mAux = cono->modelMat();
mAux = translate(mAux, dvec3(0, 85, 0));
mAux = rotate(mAux, radians(-90.0), dvec3(1.0, 0, 0));
cono->setModelMat(mAux);
gObjects.push_back(cono);
```