Grado en Informática y Matemáticas.

Informática Gráfica. Curso 2014-15. Teoría.

Examen de la convocatoria ordinaria de febrero. Viernes, 6 de Febrero de 2015.

nombre:	*apellidos:*
e-mail:	

1. (2.5 puntos) Dado un cubo de lado 1, con una de sus esquinas en la posición (0,0,0) y otra en (1,1,1) –el resto las puede usted averiguar fácilmente-, dibuje cómo se vería el cubo (y el origen y los ejes del sistema de coordenadas del mundo) en un *viewport* cuadrado si la cámara tiene cada uno de los siguientes conjuntos de parámetros:

(1) O = (0,0,5); VPN = (0,0,1); VUP = (0,1,0)

(2) O = (0,0,5); VPN = (0,0,1); VUP = (0,-1,0)

(3) O = (5,5,5); VPN = (1,1,1); VUP = (0,1,0)

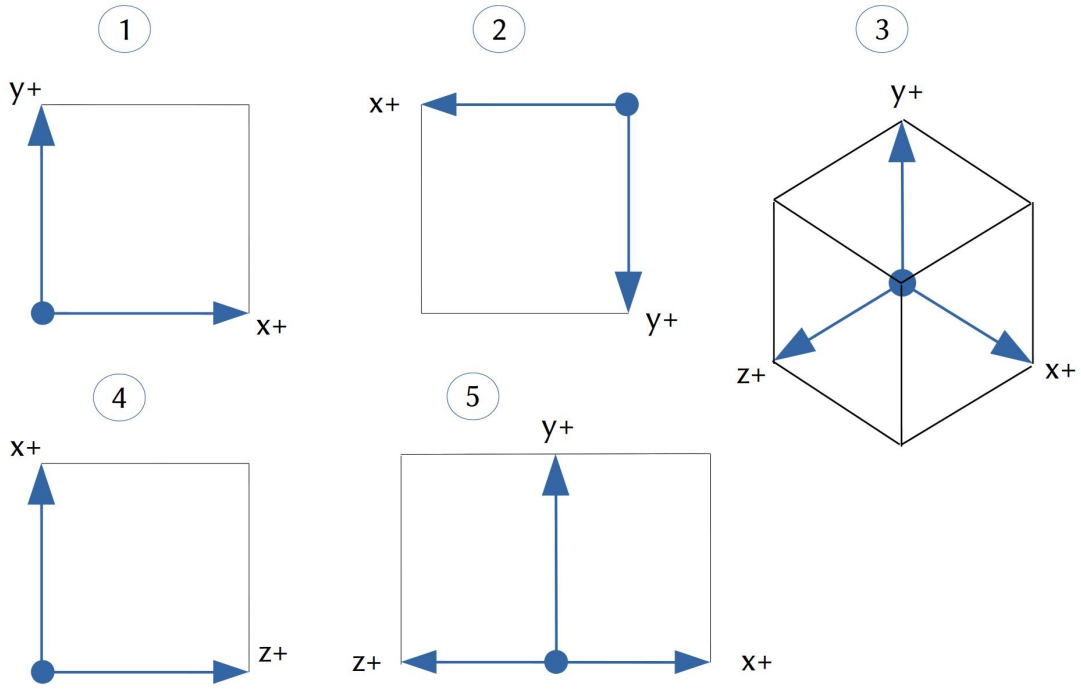
(4) O = (0,5,0); VPN = (0,1,0); VUP = (1,0,0)

(5) O = (5,0,5); VPN = (1,0,1); VUP = (0,1,0)

Suponemos en todos los casos que se usa una proyección paralela (ortográfica), y que el cubo aparece completamente dentro del viewport, sin recortarse ninguna parte del mismo.

- O es la posición del observador (origen del sistema de coordenadas de la cámara),
- VPN es el vector perpendicular al plano de visión (eje Z+ del sistema de coordenadas de la cámara),
- VUP es el vector que indica el sentido "hacia arriba" (eje Y+ del sistema de coordenadas de cámara)

Respuesta



2. (2.5 puntos) Partiendo de un objeto definido por su tabla de vértices y caras (triángulos), escribir en código C o C++ una función que devuelva el área total del objeto. Usar la fórmula de Herón para el área de un triángulo:

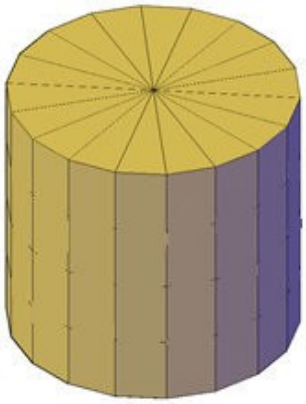
$$\text{área} = \sqrt{s(s-a)(s-b)(s-c)} \qquad \text{donde:} \quad s = \frac{a+b+c}{2}$$

(aquí ***a***, ***b,*** y ***c*** son las longitudes de cada una de las tres aristas del triángulo)

Respuesta

```
// se hace como una función que devuelve el área:

float Area( MallaTVT * m )
{
   float area = 0.0 ;
   // recorremos todas las caras
   for( unsigned i = 0 ; i < m->caras.size() ; i++ )
   {
      // recuperamos las posiciones en WC de los tres vértices
      Tupla3f v0 = m->vertices[m->caras[i][0]],
              v1 = m->vertices[m->caras[i][1]],
              v2 = m->vertices[m->caras[i][2]];

      // calculamos a,b,c,s
      float   a = (v1-v0).longitud(),
              b = (v2-v0).longitud(),
              b = (v1-v2).longitud(),
              s = 0.5*(a+b+c) ;

      // sumamos área del triángulo al área total
      area += sqrt( s*(s-a)*(s-b)*(s-c) ) ;
   }
   return area ;
}
```

3. (2.5 puntos) Se quiere texturizar una lata formada por un cilindro con una tapa superior (figura de la derecha) usando una sola imagen (a la izquierda de la figura). Explicar mediante un código en C cómo se han de asignar las coordenadas de textura a los vértices del objeto (superficie lateral y tapa superior) durante la generación de la tabla de vértices por revolución, usando N copias de un perfil formado por dos puntos: $P1=(r,0,0)$ y $P2=(r,h,0)$ (siendo r el radio de la lata, y h su altura). Las coordenadas de textura están en el rango 0-1. Se ha visualizado una cuadrícula de tamaño 4x4 sobre ella. No es necesario indicar como se crean las caras, solo los vértices.



Respuesta

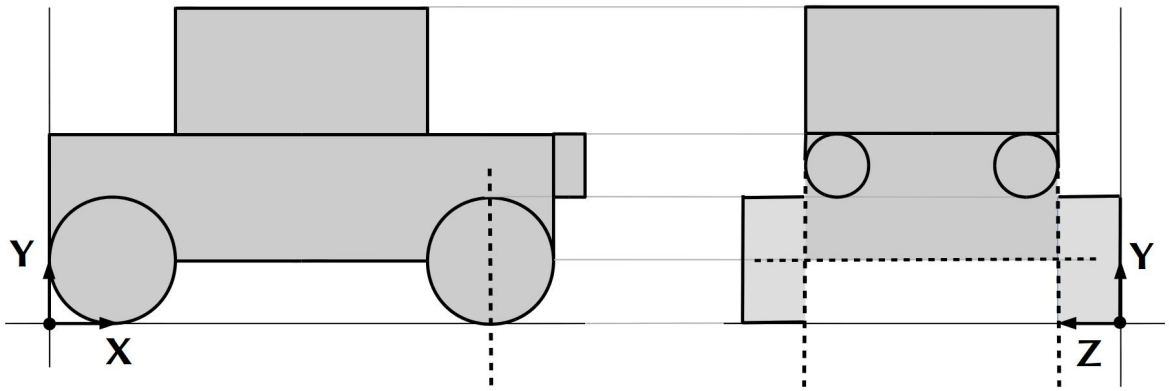
```
// asumimos que 'n' es una constante con el número de copias del perfil
// que queremos crear (sin incluir la ultima). Se crea solo la tapa de arriba.
// asumimos que 'h' y 'r' son constantes reales predefinidas

std::vector<Tupla3f> ver ;                // tabla de posiciones de vértices
std::vector<Tupla2f> cct ;                // tabla de coordenadas de textura

for( unsigned i = 0 ; i <= n ; i++ )
{
    float f = float(i)/float(n),          // fracción de perfiles ya calculados
          a = 2.0*M_PI*f,                 // ángulo del perfil (radianes)
          c = cos(a), s = sin(alpha),     // seno y coseno del 'a'
          x = r*c, z = r*s ;              // coordenada x,z de los tres vértices

    // añadir los tres vértices a 'ver'
    ver.push_back( Tupla3f( x, 0, z ) );  // vértice inferior del cilindro
    ver.push_back( Tupla3f( x, h, z ) );  // vértice superior del cilindro
    ver.push_back( Tupla3f( x, h, z ) );  // vértice en la tapa (misma pos.)

    // insertar las coords. de textura a 'cct'
    cct.push_back( Tupla2f( f, 0.0 ) );   // vértice inferior del cilindro
    cct.push_back( Tupla2f( f, 0.5 ) );   // vértice superior del cilindro
    cct.push_back( Tupla2f( 0.25+0.25*c, 0.75+0.25*s ) );  // vértice en la tapa
}

// insertamos vértice en el centro de la tapa superior al final
verts.push_back( Tupla3f( 0.0, h, 0.0 ) );
cct.push_back( Tupla2f( 0.25, 0.75 ) );
```

4. (2.5 puntos). Supongamos que disponemos de los siguientes dos primitivas geométricas 3D:

- **Cilindro**, sin tapas, de altura y radio unidad, con la base apoyada en el plano XZ y con el centro de la base en el origen de coordenadas.
- **Disco**: disco plano de radio unidad, perpendicular al eje Y y con el centro en el origen.
- **Cubo**: cubo de lado 2 unidades con centro en el origen.

Con estas tres primitivas queremos construir un modelo de coche como en la figura (vemos el perfil, a la izquierda, y el alzado, a la derecha). Tiene cuatro ruedas de radio y ancho unidad, así como dos faros delanteros, de radio y ancho 1/2. El cuerpo está hecho con dos paralelepípedos (el inferior mide 8 x 2 x 4 unidades en X,Y, y Z, respectivamente, y el superior mide 4 x 2 x 4 unidades).

Como producto de los movimientos del volante, las dos ruedas delanteras giran cada una en torno a un eje vertical (líneas verticales a rayas de la figura), de forma que el ángulo de rotación será cero en reposo (como en la figura). A este ángulo lo llamamos *ang_giro*, y es igual para ambas ruedas. Además, para que el coche pueda rodar, las cuatro ruedas giran sobre los ejes delantero y trasero del coche (línea a rayas horizontal), de forma solidaria (todas por igual), según un ángulo (que llamamos *ang_rodar.*)

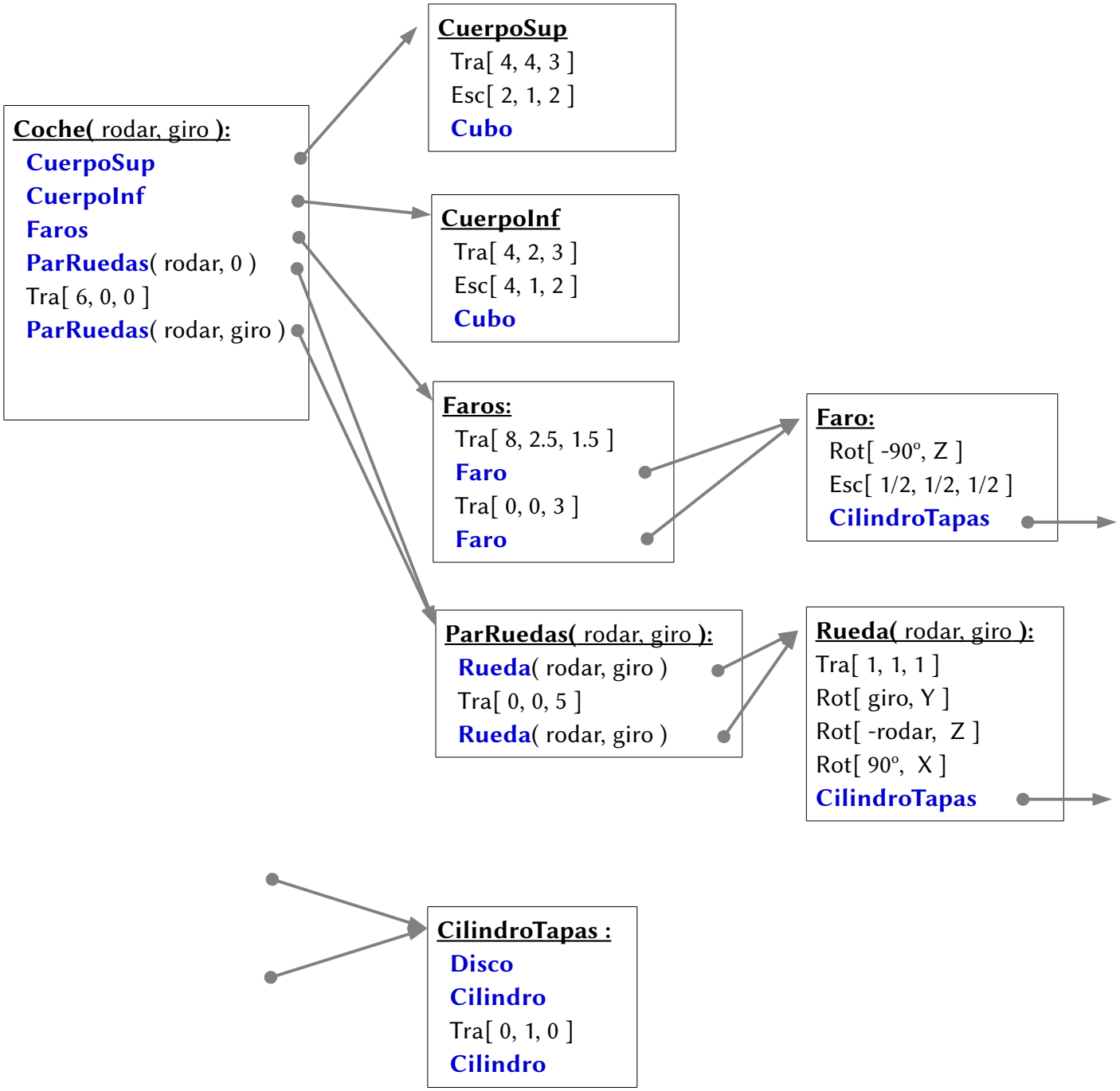


4.a. (1.25 punto) dibuja el grafo de escena parametrizado correspondiente al modelo jerárquico del coche (hay dos grados de libertad*:* los valores de *ang_giro* y de *ang_rodar*).

4.b. (1.25 punto) escribe la función que, invocando a las transformaciones geométricas adecuadas y las funciones de dibujado del cilindro, disco y cubo, pinte el coche. Dicha función o método se declara de esta forma:

```
void dibujaCoche( float ang_giro, float ang_rodar ) ;
```

Respuesta:


Coche(rodar, giro):
CuerpoSup
CuerpoInf
Faros
ParRuedas(rodar, 0)
Tra[6, 0, 0]
ParRuedas(rodar, giro)
CuerpoSup
Tra[4, 4, 3]
Esc[2, 1, 2]
Cubo
CuerpoInf
Tra[4, 2, 3]
Esc[4, 1, 2]
Cubo
Faros:
Tra[8, 2.5, 1.5]
Faro
Tra[0, 0, 3]
Faro
Faro:
Rot[-90º, Z]
Esc[1/2, 1/2, 1/2]
CilindroTapas
ParRuedas(rodar, giro):
Rueda(rodar, giro)
Tra[0, 0, 5]
Rueda(rodar, giro)
Rueda(rodar, giro):
Tra[1, 1, 1]
Rot[giro, Y]
Rot[-rodar, Z]
Rot[90º, X]
CilindroTapas
CilindroTapas :
Disco
Cilindro
Tra[0, 1, 0]
Cilindro