PRÁCTICA 3 – Documentación

Asignatura: Informática Gráfica

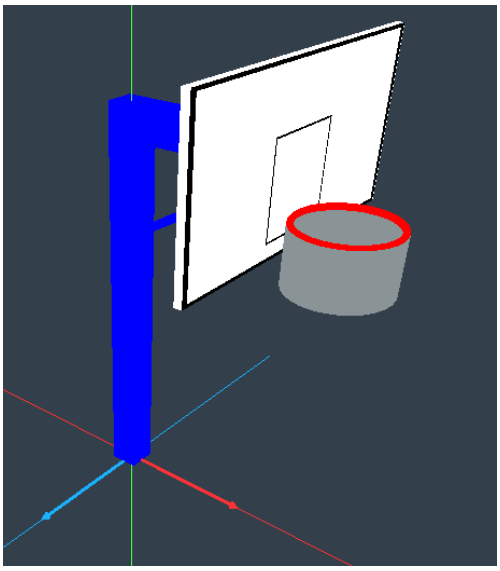
Curso académico: 2023/2024

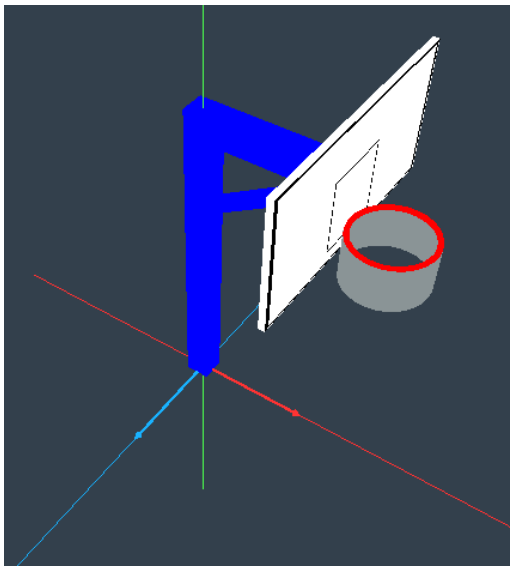
Nombre y Apellidos: Elena Torres Fernández

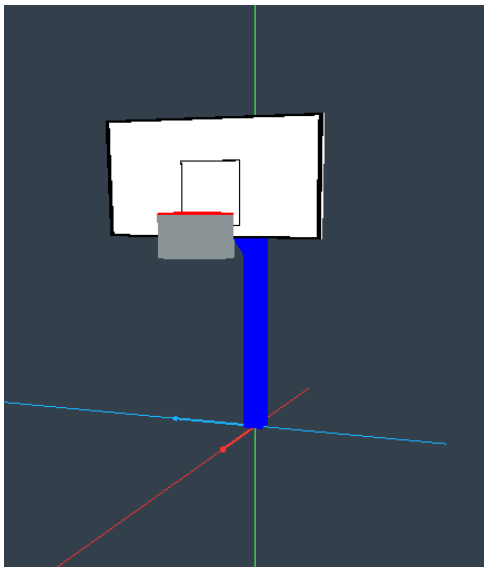
Titulación: Doble Grado Ingeniería Informática y Matemáticas

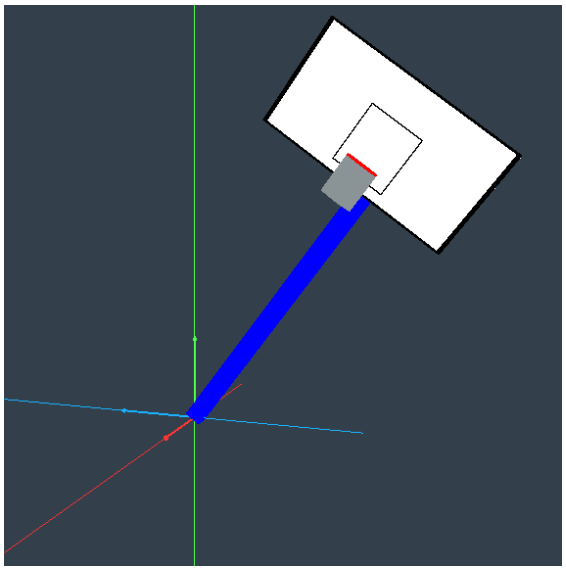
Índice

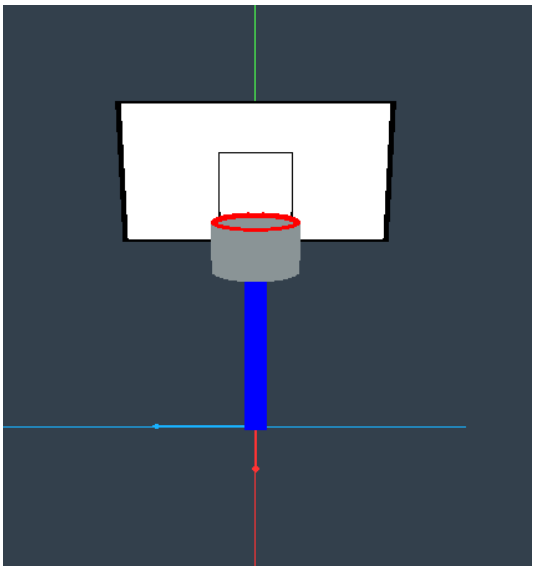
1. Capturas de pantalla del modelo

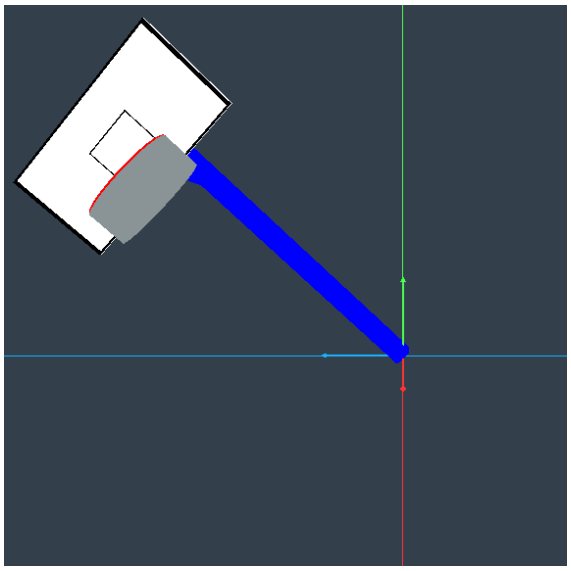




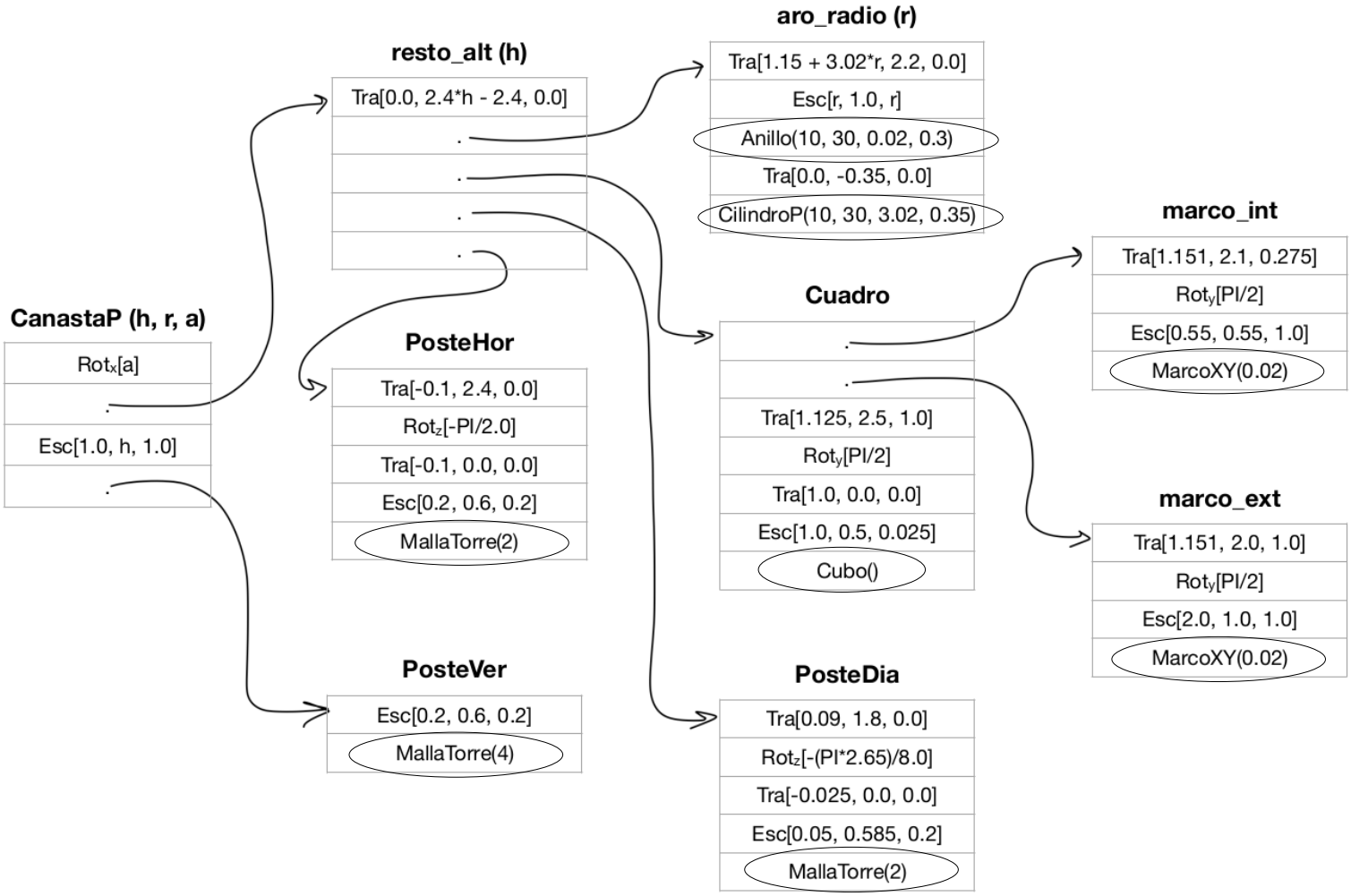








2. Grafo de escena tipo PHIGS



3. Lista nodos del grafo

CanastaP (h, r, a)

- **CanastaP** es una clase derivada de NodoGrafoEscena.
- Tiene asociados 3 parámetros o grados de libertad: h (paltura), r (pradio), a (palpha).
- No tiene un color específico.
- Está declarada en *modelo-jer.h* en el rango de líneas [119-132].
- Está implementada en *modelo-jer.cpp* en el rango de líneas [220-255].

resto_alt (h)

- **resto_alt** es un objeto NodoGrafoEscena que se crea dentro de la clase CanastaP.
- Tiene asociado 1 grado de libertad: h (paltura).
- No tiene un color específico.
- Está construido en *modelo-jer.cpp* en el rango de líneas [237-242].

aro_radio (r)

- **aro_radio** es un objeto de NodoGrafoEscena que se crea dentro de la clase CanastaP.
- Tiene asociado 1 grado de libertad: r (pradio).
- No tiene un color específico.
- Está construido en *modelo-jer.cpp* en el rango de líneas [223, 234].

Cuadro

- **Cuadro** es una clase derivada de NodoGrafoEscena.
- No tiene asociado ningún grado de libertad.
- No tiene un color específico.
- Está declarada en *modelo-jer.h* en el rango de líneas [101-105].
- Está implementada en *modelo-jer.cpp* en el rango de líneas [172-197].

marco_int

- **marco_int** es un objeto de NodoGrafoEscena que se crea dentro de la clase Cuadro.
- No tiene asociado ningún grado de libertad.
- No tiene un color específico.
- Está construido en *modelo-jer.cpp* en el rango de líneas [175, 181].

marco_ext

- **marco_ext** es un objeto de NodoGrafoEscena que se crea dentro de la clase Cuadro.
- No tiene asociado ningún grado de libertad.
- No tiene un color específico.
- Está construido en *modelo-jer.cpp* en el rango de líneas [184, 188].

PosteDia

- **PosteDia** es una clase derivada de NodoGrafoEscena.
- No tiene asociado ningún grado de libertad.
- No tiene un color específico.
- Está declarada en *modelo-jer.h* en el rango de líneas [95-99].
- Está implementada en *modelo-jer.cpp* en el rango de líneas [155-163].

PosteHor

- **PosteHor** es una clase derivada de NodoGrafoEscena.
- No tiene asociado ningún grado de libertad.
- No tiene un color específico.
- Está declarada en *modelo-jer.h* en el rango de líneas [89-93].
- Está implementada en *modelo-jer.cpp* en el rango de líneas [145-153].

PosteVer

- **PosteVer** es una clase derivada de NodoGrafoEscena.
- No tiene asociado ningún grado de libertad.
- No tiene un color específico.
- Está declarada en *modelo-jer.h* en el rango de líneas [83-87].
- Está implementada en *modelo-jer.cpp* en el rango de líneas [139-143].

Lista objetos de malla-ind y malla-revol usados:

MallaTorre(n) → Hecho en el ejercicio adicional número 3 de la práctica 2.

- **MallaTorre** es una clase derivada de MallaInd.
- No tiene asociado ningún grado de libertad.
- Tiene el color **azul** (0.0, 0.0, 1.0).
- Está declarada en *malla-ind.h* en el rango de líneas [253-262].
- Está implementada en *malla-ind.cpp* en el rango de líneas [727-758].

Cubo() → Hecho en la práctica 1.

- **Cubo** es una clase derivada de MallaInd.
- No tiene asociado ningún grado de libertad.
- No tiene un color específico (blanco por defecto).
- Está declarada en *malla-ind.h* en el rango de líneas [115-120].
- Está implementada en *malla-ind.cpp* en el rango de líneas [264-286].

MarcoXY(grosor) → Hecho nuevo para esta práctica.

- **MarcoXY** es una clase derivada de MallaInd.
- No tiene asociado ningún grado de libertad.
- Tiene el color **negro** (0.0, 0.0, 0.0).
- Está declarada en *malla-ind.h* en el rango de líneas [297-301].
- Está implementada en *malla-ind.cpp* en el rango de líneas [810-835].

Anillo(num_verts_per, nperfiles, r, R) → Hecho nuevo para esta práctica.

- **Anillo** es una clase derivada de MallaRevol.
- No tiene asociado ningún grado de libertad.
- Tiene el color **rojo** (1.0, 0.0, 0.0).
- Está declarada en *malla-revol.h* en el rango de líneas [140-153].
- Está implementada en *malla-revol.cpp* en el rango de líneas [217-239].

CilindroP(num_verts_per, nperfiles, radio, altura) → Hecho nuevo para esta práctica, dando parámetros para el radio y la altura al ya construido en la práctica 2.

- **CilindroP** es una clase derivada de MallaRevol.
- No tiene asociado ningún grado de libertad.
- Tiene el color **gris** (0.54, 0.58, 0.59).
- Está declarada en *malla-revol.h* en el rango de líneas [86-99].
- Está implementada en *malla-revol.cpp* en el rango de líneas [131-146].

4. Lista parámetros o grados de libertad del grafo

h (paltura) **1/3**

Nodos donde están las matrices que dependen del parámetro paltura:

NODO *MATRICES*
CanastaP → **pm_alt_poste = glm::scale(glm::vec3(1.0, paltura, 1.0)) ;*
resto_alt → **pm_alt_resto = glm::translate(glm::vec3(0.0, 2.4*paltura – 2.4, 0.0));*

En ambas matrices, el parámetro paltura oscila entre 1.0 y 3.0 con 1 oscilación por segundo (n = 1). Así, la canasta pasa de su altura inicial al triple y vuelve a la inicial en 1 segundo.

```
case (0):
   vmin = 1.0;        // altura poste
   vmax = 3.0;
   v = (vmin + vmax)/2.0 + ((vmax - vmin)/2.0)*sin(2*M_PI*t_sec);
   *pm_alt_poste = glm::scale( glm::vec3(1.0, v, 1.0) );
   *pm_alt_resto = glm::translate( glm::vec3(0.0, 2.4*v -2.4, 0.0) );
   break;
```

r (pradio) **2/3**

Nodos donde están las matrices que dependen del parámetro pradio:

NODO *MATRICES*
aro_radio → **pm_esc_aro = glm::scale(glm::vec3(pradio, 1.0, pradio)) ;*
→ **pm_tra_aro = glm::translate(glm::vec3(1.15 + 3.02*pradio, 2.2, 0.0));*

En ambas matrices, el parámetro pradio oscila entre 0.5 y 1.5 con media oscilación por segundo (n = 1/2). Así, el aro de la canasta pasa de tener la mitad del radio a multiplicarlo por 1.15 y volver a su mitad en 2 segundos.

```
case(1):              // escalado aro
   vmin = 0.5;
   vmax = 1.5;
   v = (vmin + vmax)/2.0 + ((vmax - vmin)/2.0)*sin(2*M_PI*t_sec/2);
   *pm_esc_aro = glm::scale( glm::vec3(v, 1.0, v) );
   *pm_tra_aro = glm::translate( glm::vec3(1.15 + (R+r)*v, 2.2, 0.0 ) );
   break;
```

a (palpha) **3/3**

Nodos donde están las matrices que dependen del parámetro palpha:

NODO *MATRICES*
CanastaP → **pm_alp_rot = glm::rotate(palpha, glm::vec3(1.0, 0.0, 0.0)) ;*

El parámetro palpha oscila entre -PI/4.0 y PI/4 con un cuarto de oscilación por segundo (n = 1/4). Así, la canasta entera gira respecto el eje X desde -PI/4.0 a PI/4 en 2 segundos y vuelve de PI/4.0 a -PI/4 en otros 2 segundos. Hace una oscilación completa en 4 segundos.

```
case(2):            // rotación canasta
   vmin = -M_PI/4.0;
   vmax = +M_PI/4.0;
   v = (vmin + vmax)/2.0 + ((vmax - vmin)/2.0)*sin(2*M_PI*t_sec/4);
   *pm_alp_rot = glm::rotate( v, glm::vec3(1.0, 0.0, 0.0) );
   break;
```

Como vemos, las expresiones que construyen estas matrices a partir del tiempo en segundos tienen una **dependencia oscilante** entre el parámetro *v* y el valor de tiempo *t_sec*. En concreto, la dependencia en todas ellas es del tipo *v = (vmin + vmax)/2 + ((vmax – vim)/2) * sen(2*PI*n*t_sec)*, donde *n* es el número de oscilaciones por segundo; *vmin* el valor mínimo que alcanza el parámetro; y *vmax*, su máximo valor. De esta forma, hemos podido establecer un mínimo y un máximo al valor de los parámetros.