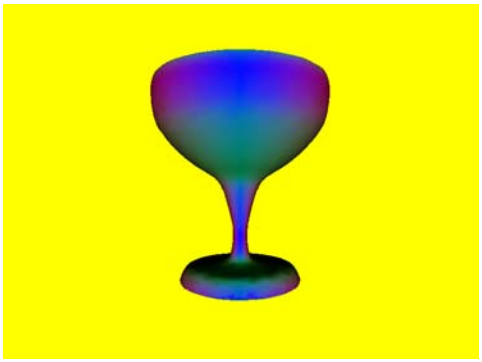
Twin (twin.*) 2.5 punts

Escriu **VS+GS+FS** que dibuixin cada triangle del model dos cops; un cop a la meitat esquerra de la finestra, i un altre cop a la meitat dreta. Aquí teniu un exemple, amb els shaders per defecte (*esquerra*) i amb els shaders que es demanen (*dreta*):

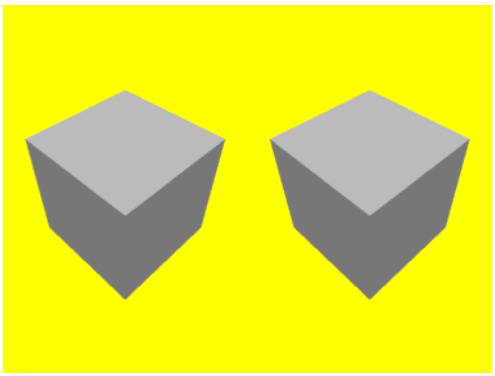
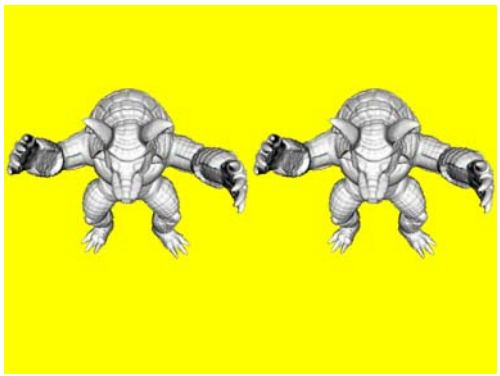
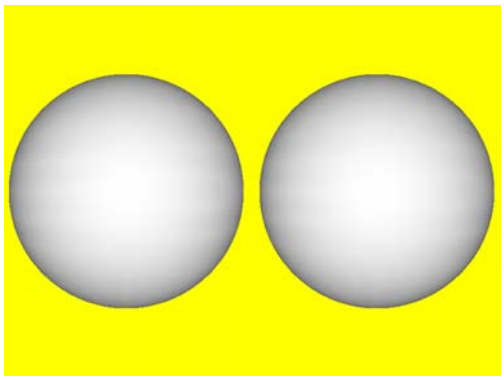
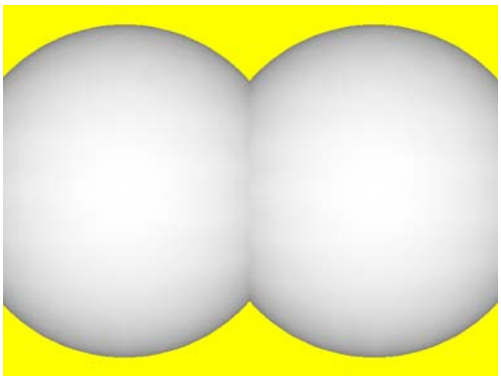



El **VS** haurà d'escriure gl_Position com habitualment. El **color del vèrtex** tindrà per components RGBA la **component z de la normal en eye space.**

El **GS** haurà d'emetre dues còpies de cada triangle. Aquest exercici és senzill si treballeu en **NDC** (Normalized Device Coordinates). La única diferència entre les dues còpies és la translació en X (en NDC), que en un cas és 0.5 i en l'altre -0.5. És obligatori que apliqueu la translació en NDC.

El **FS** simplement escriurà el color que li arriba del VS.

Aquí teniu més exemples:

Fitxers i identificadors (ús obligatori):

`twin.vert, twin.geom, twin.frag`

Panorama (panorama.*) 2.5 punts

Escriu un **VS** i un **FS** per tal de visualitzar panorames representats amb projecció equirectangular. Al directori **/assig/grau-g/Textures/** trobareu la textura **verandah.png** (Fotografia de Greg 'Groggy' Lehey):

En aquesta textura, un texel (s,t) representa el color en la direcció del vector unitari (x,y,z) determinat per aquestes equacions:

$\theta = 2\pi s$
$\psi = \pi(t-0.5)$

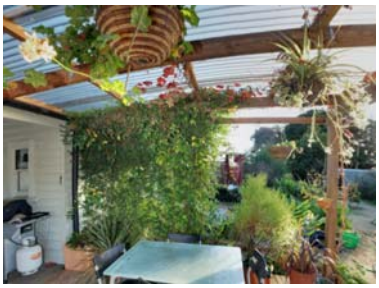
$x = \cos(\psi)\cos(\theta)$
$y = \sin(\psi)$
$z = \cos(\psi)\sin(\theta)$

Aquest exercici està pensat pel model de l'esfera (sphere.obj), amb la càmera situada dins l'esfera.

El VS, a banda d'escriure gl_Position, enviarà al FS les coordenades del punt en **object space**.

El FS agafarà aquest punt (x,y,z), i calcularà les coordenades (s,t) on està representat el que es veu en direcció (x,y,z). Per fer això, heu de calcular primer els angles θ, ψ a partir de (x,y,z) (aïllant-los a les equacions anteriors; caldrà que feu servir les funcions **asin** i **atan** de GLSL), per després calcular les coordenades de textura (s,t) a partir de θ, ψ. El color del fragment serà el color de la textura a (s,t). La funció **atan**(a, b) de GLSL calcula l'arctangent de a/b, retornant un valor dins l'interval $[-\pi, \pi]$.

Aquí teniu el resultat esperat (sphere.obj, càmera situada aproximadament a l'origen):


Fitxers i identificadors (ús obligatori):

```
panorama.vert, panorama.frag
uniform sampler2D panorama;
const float PI = 3.141592;
```

XRays (xrays.*) 2.5 punts

Escriu un VS i un FS per simular una mena de lupa, controlada amb el mouse, que permeti veure les capes interiors d'un dibuix d'anatomia. Farem servir quatre textures (foot0.jpg ... foot3.jpg):

```
uniform sampler2D foot0;
uniform sampler2D foot1;
uniform sampler2D foot2;
uniform sampler2D foot3;
```

El VS farà les tasques per defecte, però aplicarà un escalat S(0.5, 1, 1) al vèrtex (abans de passar-lo a clip space), de forma que l'objecte **plane.obj** passi a ser rectangular.

Pel FS, us proporcionem un **xrays.frag** que heu de completar. El que ha de fer el FS és:

1. Calcular la distància d (en píxels) del fragment a les coordenades actuals del mouse. Feu servir obligatòriament la funció mouse() que us proporcionem, que retorna les coordenades del mouse en window space.

2. Usar les coordenades de textura habituals per accedir a la textura **foot0** (pell). Sigui C el color resultant. Si **d >= R** (R és una constant que ja teniu declarada al exemple), el color del fragment serà directament C. Altrament, el color final es calcula com segueix.

3. Accedir a la textura indicada pel **uniform int layer=1** per obtenir un altre color D. Per exemple, si layer = 1, cal obtenir el color de la textura foot1. Podeu assumir que layer sempre tindrà valor 0, 1, 2 o 3.

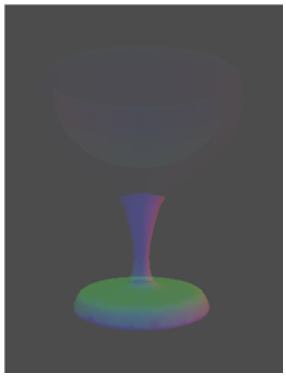
4. El color final del fragment (cas d<R) serà el resultat de fer la interpolació lineal entre D i C, on el paràmetre d'interpolació lineal serà d/R (és a dir, la distància al mouse normalitzada per R). D'aquesta manera el centre del cercle al voltant del mouse mostrarà el color D de la capa interior (indicada per layer) mentre que a mesura que ens allunyem de la posició del mouse es mostrarà gradualment el color C de pell.

Identificadors (ús obligatori):

```
xrays.vert, xrays.frag
uniform sampler2D foot0, foot1, foot2, foot3;
uniform int layer;
```

Ghostlight (ghostlight.*) 2.5 punts

Escriu un **plugin** que dibuixi el model tant transparent que sigui gairebé invisible, excepte en una regió de 100 píxels al voltant del punter del ratolí. Els resultats esperats són aquests (glass):



Prengueu com a punt de partida el plugin d'alpha blending que ve com a exemple amb l'aplicació viewer.

El FS ha de comprovar la distància entre la posició del fragment i la del ratolí (que caldrà passar via uniform) i:

- Aplicar una opacitat de 0.025 si la distància és major que 100 píxels.
- Aplicar una opacitat de 0.25 altrament.

Per a poder obtenir la posició del ratolí en cada moment podeu utilitzar la crida mouseMoveEvent de la interfície dels plugins.

Fitxers i identificadors (ús obligatori):

`ghostlight.pro, ghostlight.h, ghostlight.cpp`