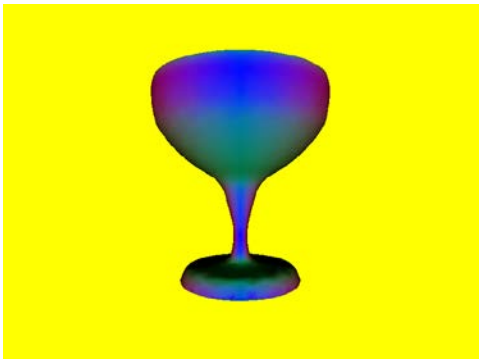
Twin (twin.*) 2.5 punts

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*):

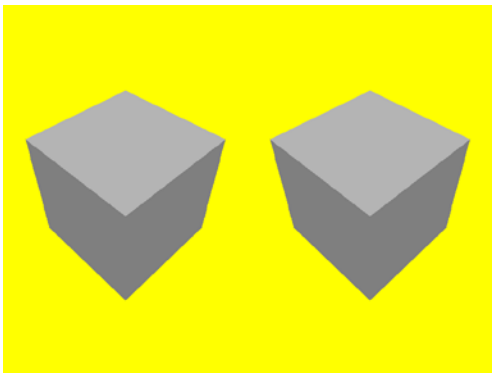
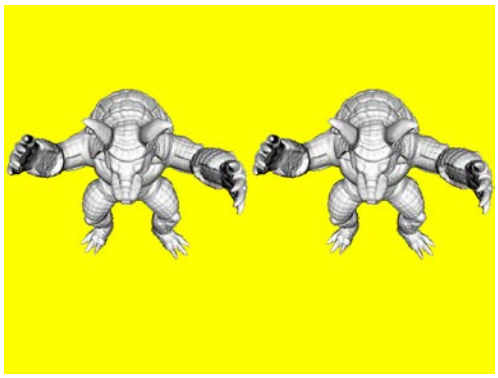
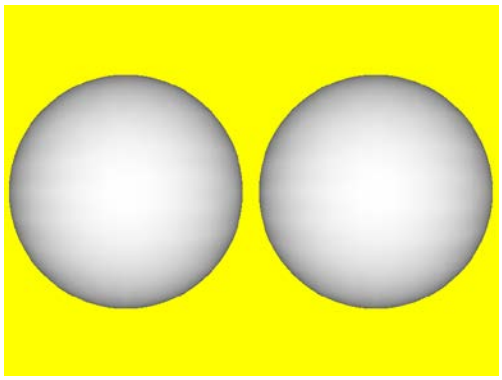
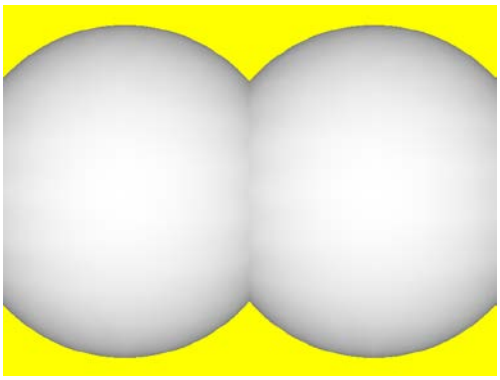



El **VS** haurà d'escriure gl_Position com habitualment. El **color del vèrtex** tindrà per components RGBA la **component z de la normal en eye space.**

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:

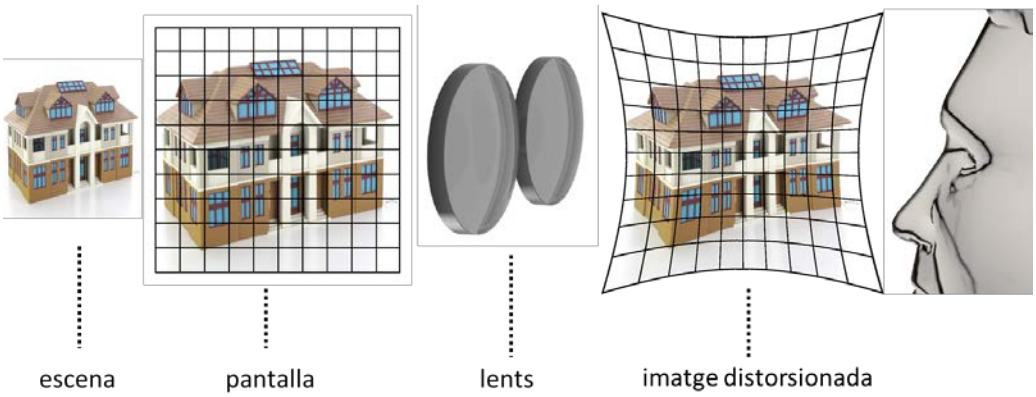





Fitxers i identificadors (ús obligatori):

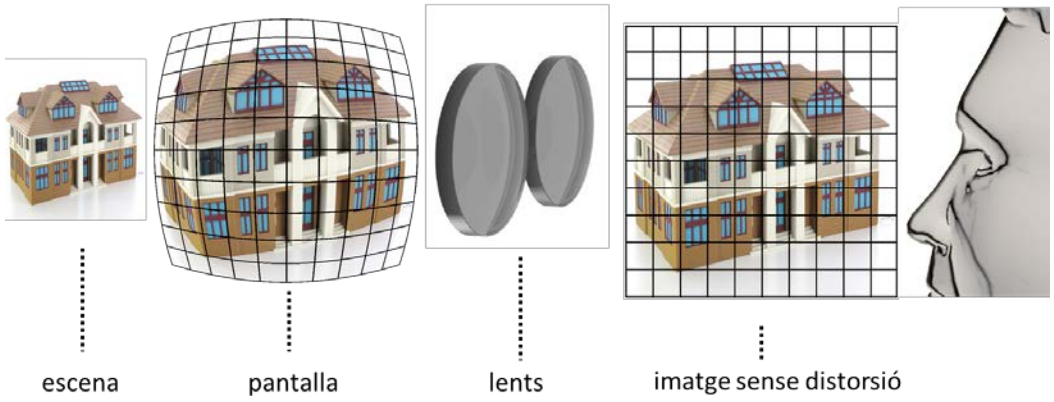
`twin.vert, twin.geom, twin.frag`

Undistort (undistort.*) 2.5 punts

Les ulleres de realitat virtual (Oculus Rift, GearVR...) fan servir lents que produeixen una distorsió radial a les imatges. Si la imatge que es mostra a la pantalla del dispositiu no es corregeix, l'usuari veurà la imatge distorsionada:

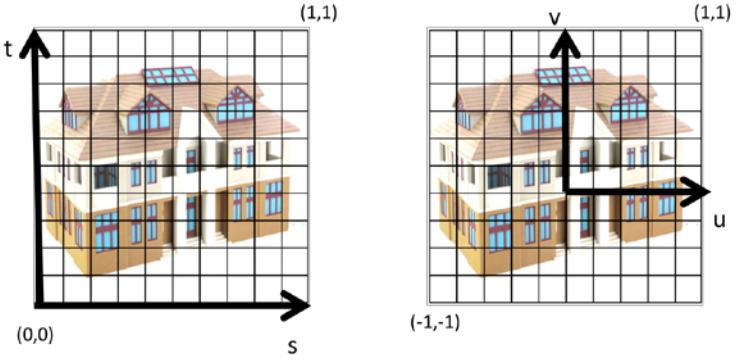


La solució més habitual consisteix a aplicar la distorsió inversa a la imatge que volem mostrar, de forma que les lents desfan aquesta distorsió i l'usuari veu la imatge correcta:



Escriu un VS i un FS que apliquin una distorsió radial. Aquest problema està pensat per l'objecte Plane, que té coordenades de textura (s,t) dins l'interval [0,1]. La imatge que volem distorsionar la carregarem com una textura **uniform sampler2D colorMap**.

El VS farà les tasques per defecte. El FS accedirà a la textura amb unes coordenades (s,t) modificades per tenir en compte la distorsió radial. El primer que haureu de fer és obtenir les coordenades de textura (u,v) respecte uns eixos centrats a l'espai de textura, com es mostra a la figura.



Gràfics
Curs 2015/16Q2

Sigui Q=(u,v) el punt amb les coordenades de textura del fragment, i sigui r la longitud del vector posició de Q. La distorsió radial que volem (inspirada en la del Oculus Rift DK1) consisteix a calcular un nou punt Q'=(u',v') amb un nou mòdul r' calculat com:

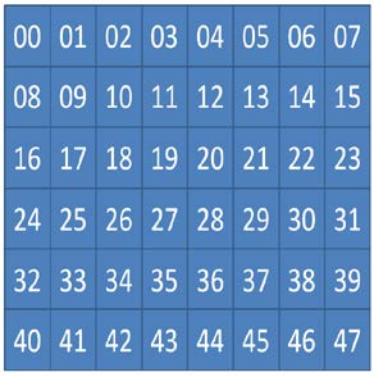
$$r' = (r + 0.22r^3+0.24r^5)$$

La distorsió només canvia la distància a l'origen, i per tant els vectors posició de Q i Q' són paral·lels, amb Q' = r' * normalize(Q).

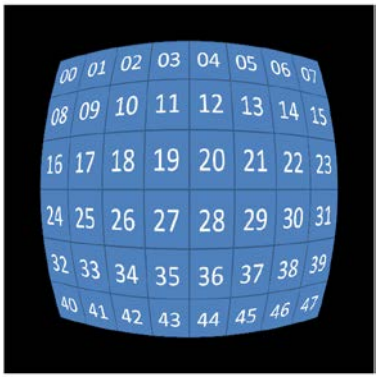
Un cop calculats (u',v'), heu de calcular les coordenades finals de textura (s', t') desfent el canvi de coordenades del començament.

Si s' i t' pertanyen a l'interval [0,1], el color final del fragment serà el color de la textura al punt (s',t'). Altrament el color serà negre.

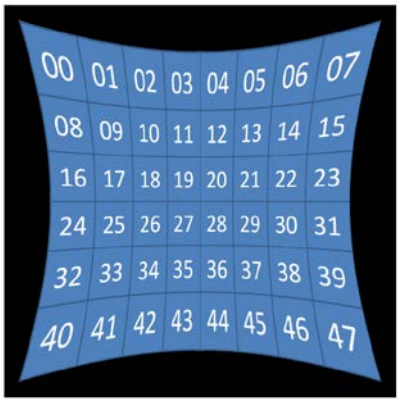
Aquí teniu alguns exemples (textura carregada i imatge esperada):



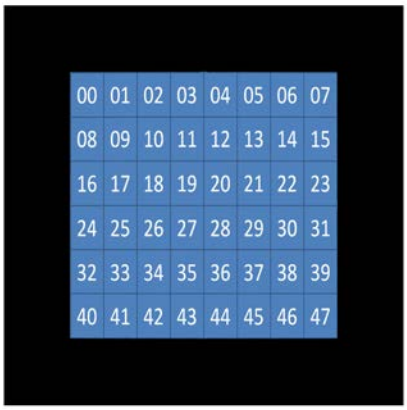
frames.png



imatge esperada



frames-pincushion.png



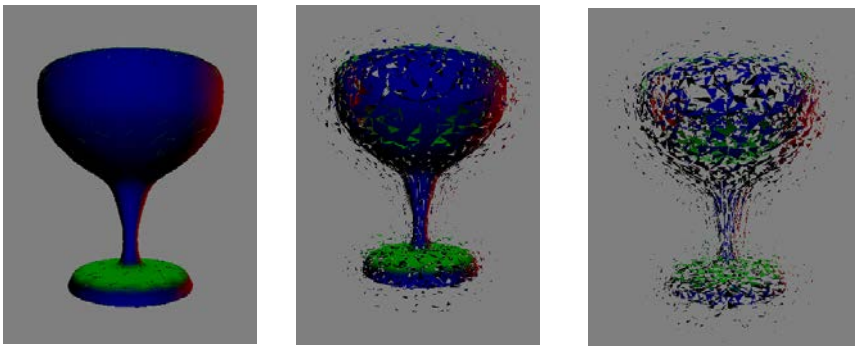
imatge esperada

Fitxers i identificadors (ús obligatori):

```
undistort.vert, undistort.frag
uniform sampler2D colorMap;
```

Disintegrate (disintegrate.*) 2.5 punts

Escriu un **plugin** i tres shaders (VS+GS+FS) que produeixin una desintegració animada del model. Els resultats esperats són aquests (**glass**):



El plugin de partida (**disintegrate.tgz**) dibuixa el model sense fer cap modificació, però passa un atribut time que podreu fer servir al GS per a fer l'animació.

El GS haurà de fer el següent:

- Per tal de que els triangles que es van eliminant s'escullin de forma aparentment aleatòria dividirem els triangles en 10 grups fent servir la fórmula:

  grup = (17 * gl_PrimitiveIDIn + 19) % 10

- Durant l'animació els triangles faran dues coses (en object space):
  - Es mouran en la direcció de la normal.
  - Aniran reduint la seva mida a base d'interpolar les noves coordenades dels seus vèrtexs amb el nou baricentre del triangle.

- Els dos efectes anteriors faran servir el paràmetre:

$$move = \left(\frac{time - grup}{4}\right)^2$$

tant per la distància recorreguda (en el primer cas), com per la interpolació entre els vèrtexs originals de cada triangle i el seu baricentre.

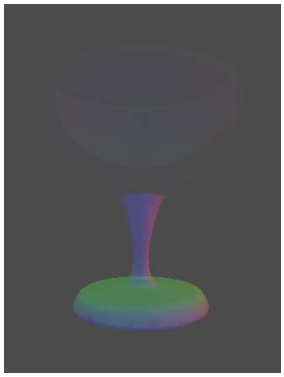
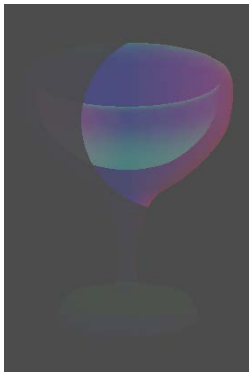
Aquest paràmetre s'haurà de retallar per a que mai estigui fora del rang [0, 1]. Això garanteix que l'animació tingui com a principi el model original i que hi hagi un final definit.

Finalment caldrà que quan es premi la tecla R l'animació torni a començar des del principi.

Fitxers i identificadors (ús obligatori):
`disintegrate.pro, disintegrate.cpp, disintegrate.h`

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`