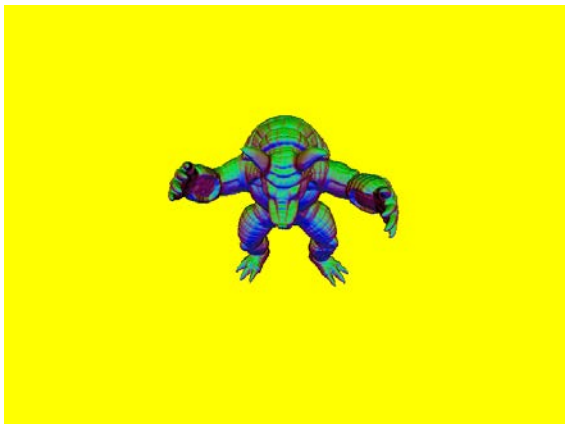
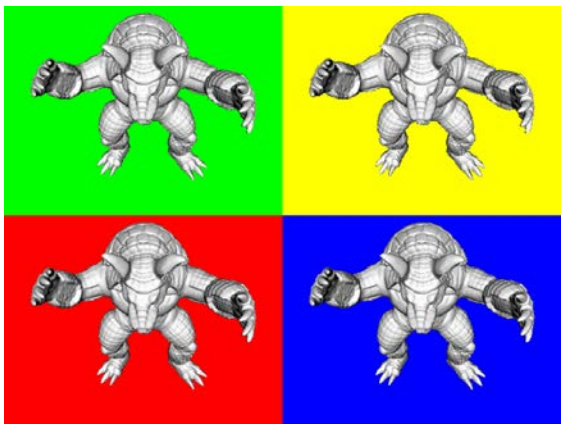
Quads2 (quads2.*)

Escriu **VS+GS+FS** que dibuixin cada triangle del model quatre cops; un a cada quadrant de la finestra. Aquí teniu un exemple, amb els shaders per defecte (*esquerra*) i amb els shaders que es demanen (*dreta*). Observa també que cada quadrant té el fons d'un color diferent:

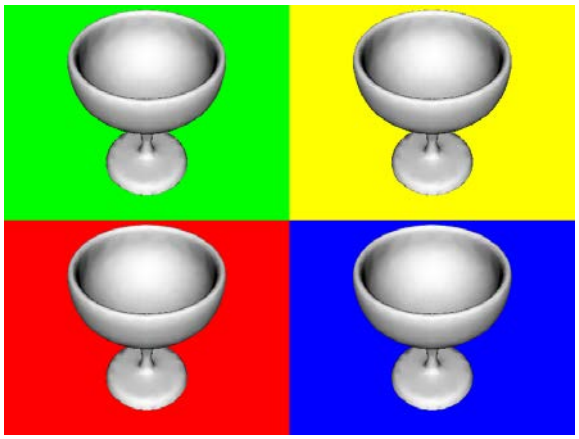
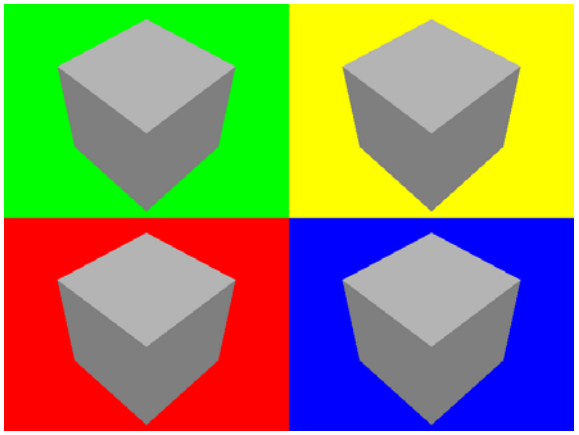
El **VS** haurà d'escriure gl_Position com habitualment. El **color del vèrtex** tindrà per components RGBA la **component z de la normal (unitària) en eye space.**

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

A més a més, quan **gl_PrimitiveIDIn** sigui 0, el GS crearà quatre quads, un per cada quadrant, amb els colors R, G, B, Y, com teniu a la figura. Novament heu de treballar directament en NDC. Useu una z molt propera a 1.0 (feu servir 0.999) per evitar que el quad retalli l'objecte.

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

Aquí teniu més exemples:

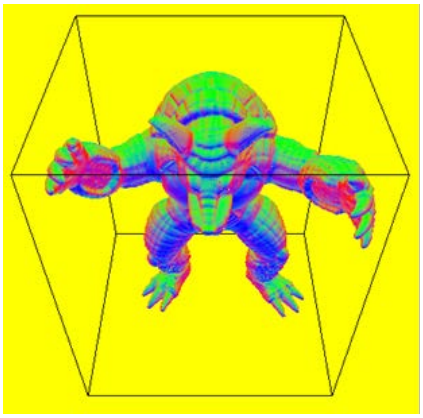



Fitxers i identificadors (ús obligatori):

`quads2.vert, quads2.geom, quads2.frag`

Box (box.*)

Escriu **VS+GS+FS** que dibuixin, a més a més de cada triangle del model, la seva capsa englobant (uniform vec3 boundingBoxMin, boundingBoxMax):



El **VS** haurà d'escriure gl_Position com habitualment. El **color del vèrtex** serà l'original.

El **GS** haurà d'emetre els triangles del model com habitualment, però fent servir *line strips* en substitució dels *triangle strips*, per tal que el model (i la capsa) es dibuixin en filferros:

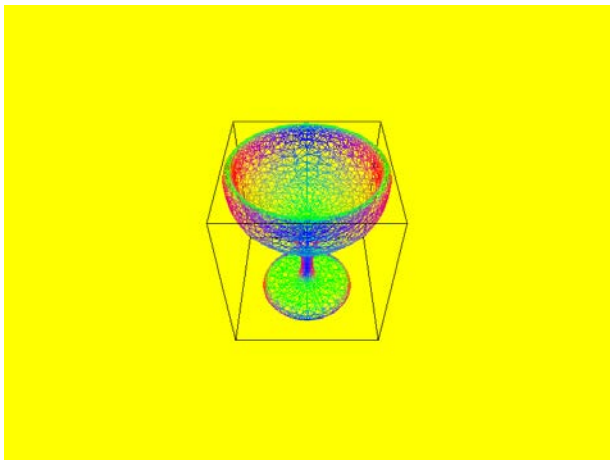
layout(line_strip, max_vertices = 36) out; // aquesta línia substitueix la del GS per defecte

A més a més, quan **gl_PrimitiveIDIn** sigui 0, el GS crearà les primitives necessàries per dibuixar la capsa englobant del model, en color **negre**.

Quan la sortida és un line strip, els vèrtexs s'interpreten com els d'una poligonal oberta. Si voleu que la poligonal sigui tancada, heu de repetir al final el primer vèrtex. Un quad, per exemple, necessita cinc EmitVertex().

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

Aquí teniu un altre exemple:



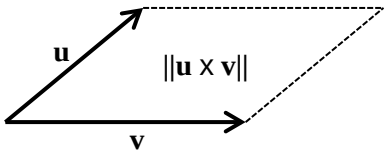
Fitxers i identificadors (ús obligatori):

`box.vert, box.geom, box.frag`

Area (area.*)

Escriu un **plugin** que escrigui pel canal de sortida estàndard (amb cout), l'àrea de la superfície de cada objecte de l'escena.

Només heu d'implementar **onPluginLoad**(). Aquest mètode recorrerà els objectes de l'escena, i calcularà l'àrea de la seva superfície com la suma de l'àrea de les seves cares. Podeu assumir que els models seran malles de triangles. L'àrea d'un triangle es pot calcular fàcilment tenint en compte que el **mòdul del producte vectorial** de dos vectors **u** i **v** dóna l'àrea del paral·lelogram definit per **u** i **v**,



El mètode haurà d'escriure pel terminal l'àrea de cada objecte, precedida pel literal "Area:".

Aquí teniu un exemple de resultat, si només està carregat l'objecte per defecte:

Area: 72.8666

O si només hi ha carregat el cub:

Area: 24

Fitxers i identificadors (ús obligatori):

`area.pro, area.h, area.cpp`