Progressive 2.5 punts

Escriu un **vertex shader** i un **geometry shader** per ShaderMaker que vagin dibuixant els triangles de l'objecte a mesura que passi el temps, amb una velocitat de 100 primitives per segon.

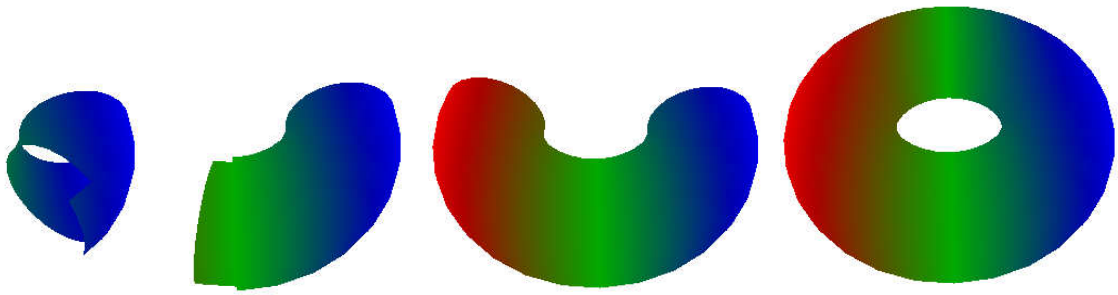
El VS simplement haurà d'escriure gl_Position (amb la posició en *object space*) i gl_FrontColor (amb el color que li arriba pel vèrtex).

El GS emetrà un triangle per cada triangle que rebi (és a dir, el comportament habitual d'un GS), però només ho farà pels primers *n* triangles de l'objecte, on *n* l'heu de calcular com $n = \lfloor 100t \rfloor$ amb t sent el temps transcorregut en segons (variable *time*).

El geometry shader haurà d'emetre cada triangle amb el color tal qual, sense il·luminació.

Recordeu que podeu saber l'identificador de primitiva amb `gl_PrimitiveIDIn`. La primera primitiva té identificador 0.

Aquí teniu el resultat que s'espera amb el torus, després de 2, 4, 8 i 16 segons (fins a 200, 400, 800 i 1600 triangles, respectivament).



En aquest problema no cal (ni es permet) cap variable varying definida per l'usuari. El geometry shader només ha d'escriure `gl_FrontColor` i `gl_Position`

Identificadors (ús obligatori)

```
progressive.vert
progressive.geom
```

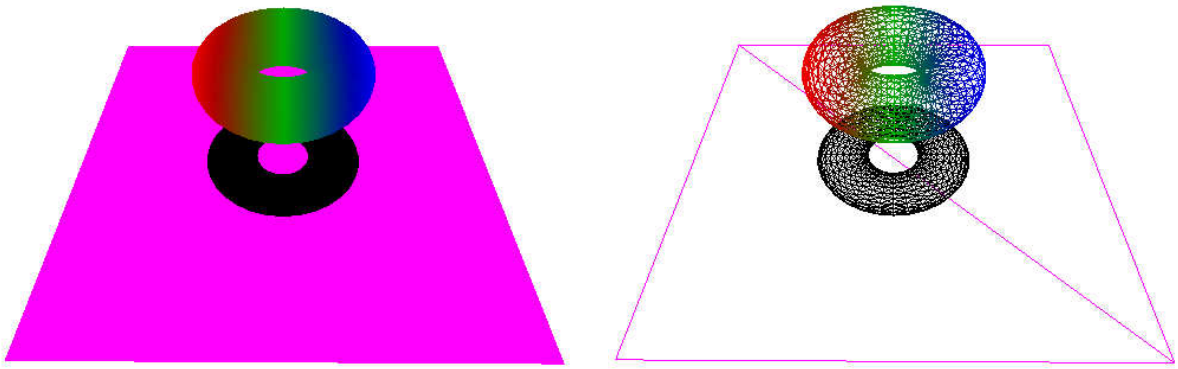
Shadow (versió 2) 2.5 punts

Escriu un **vertex shader** i un **geometry shader** per ShaderMaker que simulin l'ombra que projecta l'objecte sobre una taula, que suposarem situada al pla **Y = -2.0,** respecte una font de llum direccional en la direcció vertical (eix Y).

Per cada triangle (que haurà de rebre amb coordenades en *object space*), el geometry shader haurà d'emetre dos triangles (en *clipping space*): un corresponent al triangle original (amb el color tal qual, sense il·luminació), i un altre (de color negre) corresponent a la projecció del triangle al pla Y = -2.0.

A més a més, si el GS detecta que està processant la primera primitiva de l'objecte (que podeu detectar amb `gl_PrimitiveIDIn == 0`), haurà d'emetre dos triangles formant un rectangle magenta alineat amb els eixos de l'aplicació, amb costat 8 i centrat al punt (0, -2.1, 0).

Aquí teniu el resultat que s'espera amb el torus. Observeu que no hi ha cap tipus d'il·luminació.



En aquest problema no cal (ni es permet) cap variable varying definida per l'usuari. El geometry shader només ha d'escriure `gl_FrontColor` i `gl_Position`

Identificadors (ús obligatori)

```
shadow2.vert
shadow2.geom
```

Skymap 2.5 punts

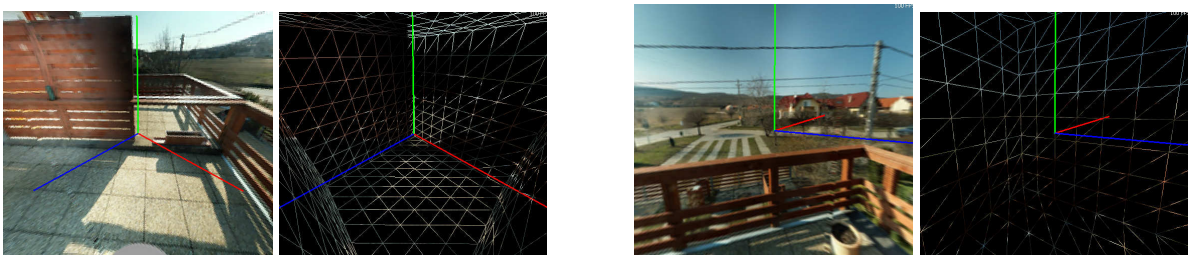
Escriu un **vertex shader** i un **fragment shader** que permetin simular un entorn (representat amb un spheremap) amb geometria senzilla (com ara un cub).

El VS simplement haurà d'escriure gl_Position de la forma habitual i fer arribar al FS la posició del vèrtex en *object space*.

El FS haurà de calcular el vector unitari V en la direcció que va de la posició del fragment a la posició de l'observador (tot en *object space*). El color final del fragment serà simplement el color del texel del sphere map (`uniform sampler2D spheremap`) corresponent al vector V. Per aquest darrer pas podeu utilitzar aquesta funció que, donat un spheremap i un vector unitari V, retorna el color en la direcció donada per V:

```
vec4 sampleSphereMap(sampler2D sampler, vec3 V)
{
  float z = sqrt((V.z+1.0)/2.0);
  vec2 st = vec2((V.x/(2.0*z)+1.0)/2.0, (V.y/(2.0*z)+1.0)/2.0);
  return texture2D(sampler, st);
}
```

Aquí teniu alguns exemples quan fiquem la càmera dins de l'objecte Cube (amb un fov de 90 graus):



Identificadors (ús obligatori)

```
uniform sampler2D spheremap;
skymap.vert
skymap.frag
```

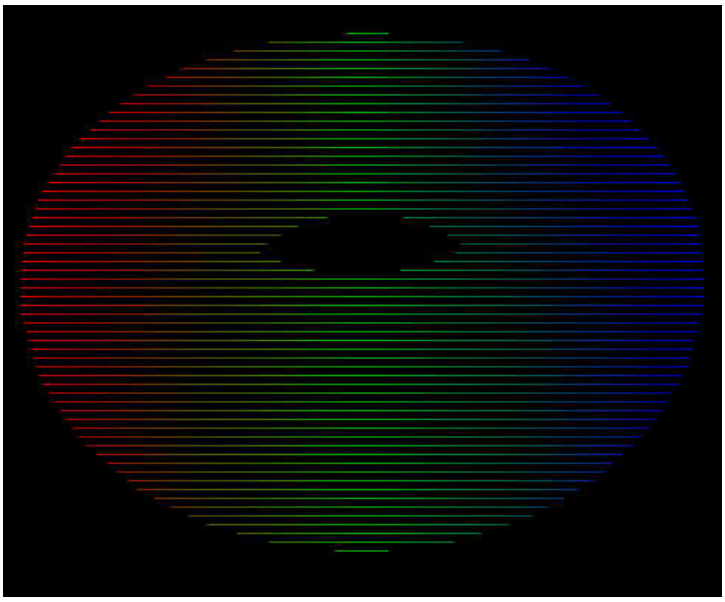
CRT display plugin 2.5 punts

Escriu un **effect plugin** pel visualitzador del laboratori que faci que els objectes de l'escena es pintin amb un **fragment shader** que simuli l'aparença de les imatges dels antics tubs CRT. Per aconseguir aquest efecte, caldrà que el FS elimini (*discard*) tots els fragments d'algunes línies del viewport. En concret, caldrà que només sobrevisquin els fragments d'una de cada *n* línies, on *n* és una constant que heu de definir amb el valor 6.

Recordeu que les coordenades (x,y) en window space fan referència al centre del píxel. Per exemple, un fragment a la cantonada inferior esquerra de la finestra té coordenades (0.5, 0.5).

Podeu començant escrivint el FS amb el ShaderMaker però **el que heu de lliurar són només els fitxers effectcrt.h i effectcrt.cpp**; per tant el codi font del FS cal que sigui al fitxer effectcrt.cpp.

Aquí teniu un exemple amb el torus:



Identificadors (ús obligatori)

```
effectcrt.h
effectcrt.cpp
```