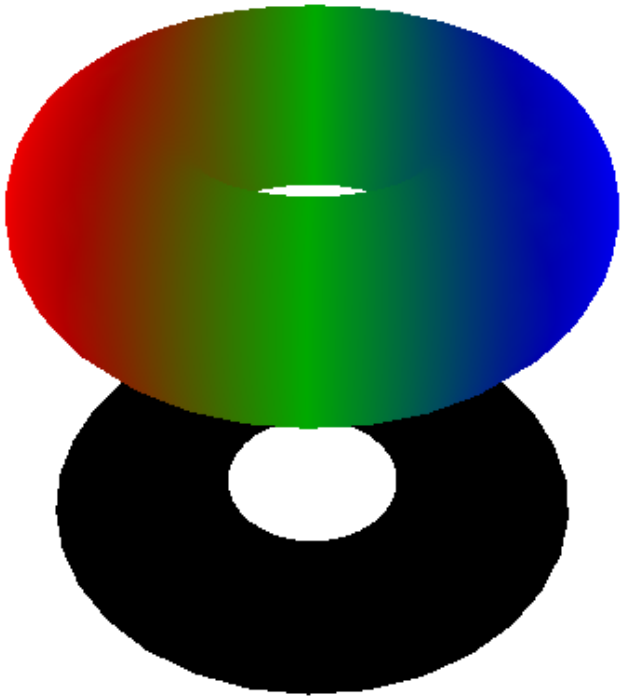
Shadow

Escriu un **vertex shader** i un **geometry shader** per ShaderMaker que simulin l'ombra que projecta l'objecte sobre el terra, que suposarem situat 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 sense il·luminació), i un altre (de color negre) corresponent a la projecció del triangle al pla Y = -2.0.

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 cap variable varying definida per l'usuari. El geometry shader només ha d'escriure `gl_FrontColor` i `gl_Position`

Explode (1)

Escriu un **vertex shader** i un **geometry shader** per simular una explosió de l'objecte com la del vídeo **explode-1.mp4**

El VS haurà de passar al GS la posició i la normal de cada vèrtex (tots dos en *object space*).

El GS haurà d'aplicar a cada vèrtex del triangle la translació donada pel vector

$$\text{speed} \cdot \text{time} \cdot \mathbf{n},$$

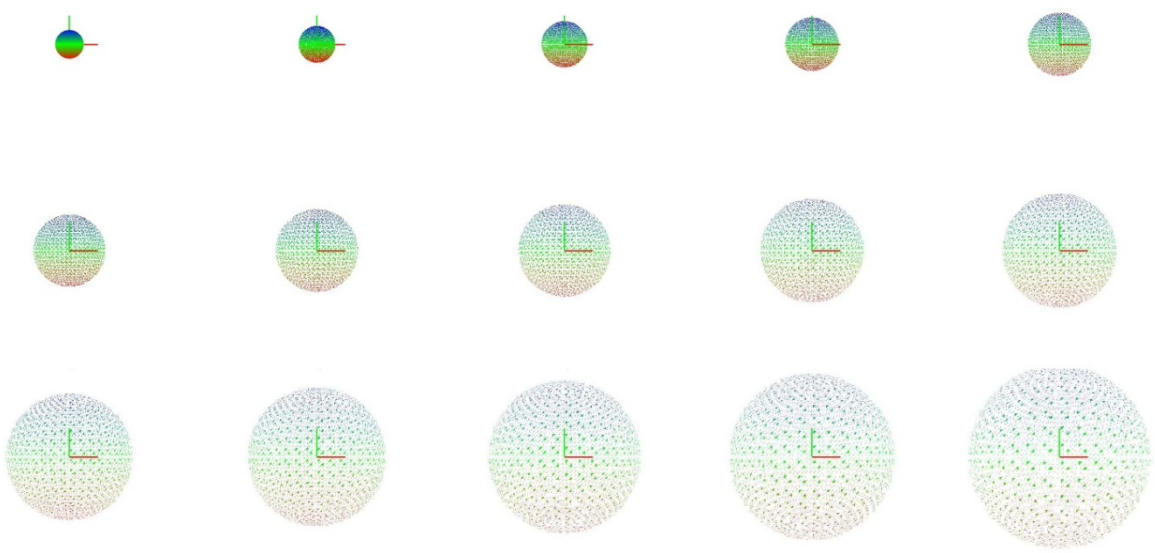
on speed és la velocitat desitjada (en unitats del model per segon), time és el temps (en segons), i **n** és el promig de les normals del tres vèrtexs del triangle (en *object space*).

Després d'aplicar la translació anterior (en object space), el GS haurà de treure els vèrtexs en *clip space*.

Pel vídeo es va fer servir aquest valor:

```
const float speed = 1.2;
```

Aquí teniu el resultat amb l'esfera:



Explode (2)

Escriu un **vertex shader** i un **geometry shader** per simular una explosió de l'objecte com la del vídeo **explode-2.mp4**

El VS haurà de passar al GS la posició i la normal de cada vèrtex (tots dos en *object space*).

El GS haurà d'aplicar a cada triangle la següent transformació geomètrica:

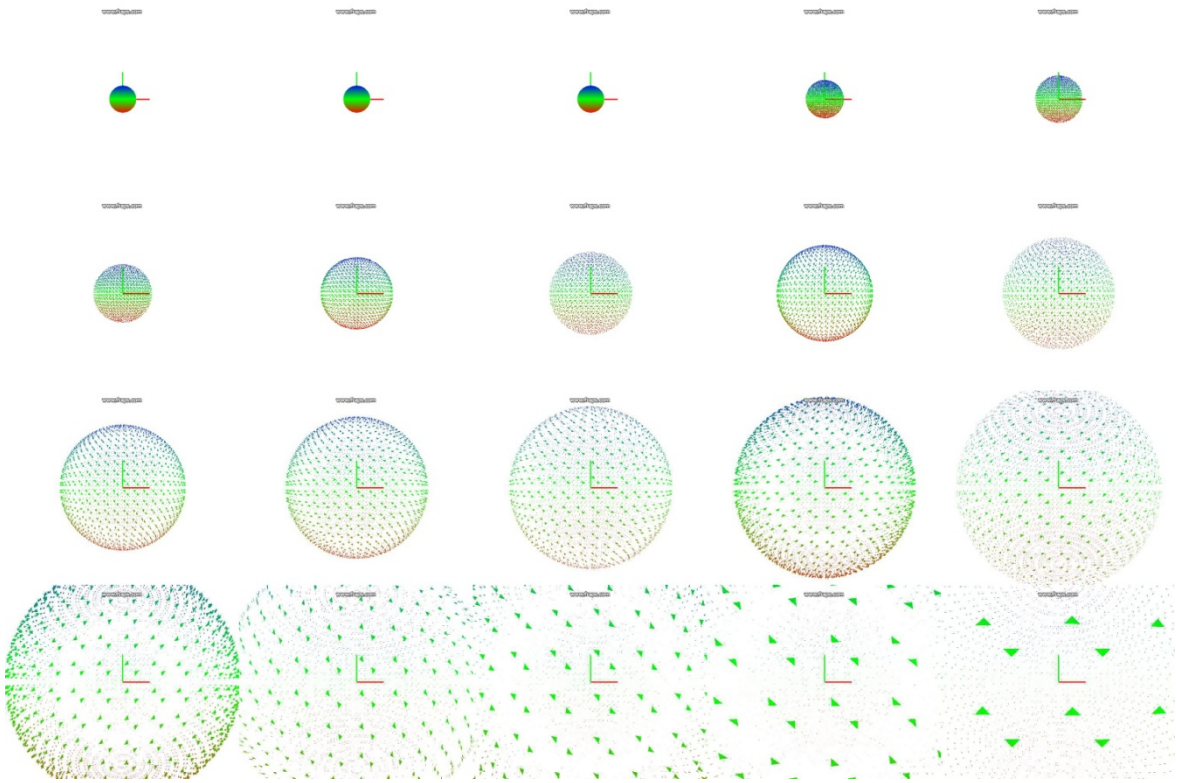
$$T(\text{speed}\cdot\text{time}\cdot\mathbf{n})\cdot R_z(\text{angSpeed}\cdot\text{time})$$

on speed és la velocitat desitjada (en unitats del model per segon), time és el temps (en segons), **n** és el promig de les normals del tres vèrtexs del triangle (en *object space*), angSpeed és la velocitat angular (rad/s) i R_z és una rotació respecte l'eix paral·lel a l'eix Z del model que passa pel baricentre del triangle.

Pel vídeo es van fer servir aquests valors:

```
const float speed = 1.2;
const float angSpeed = 8.0;
```

Aquí teniu el resultat amb l'esfera:



Voxelize (1)

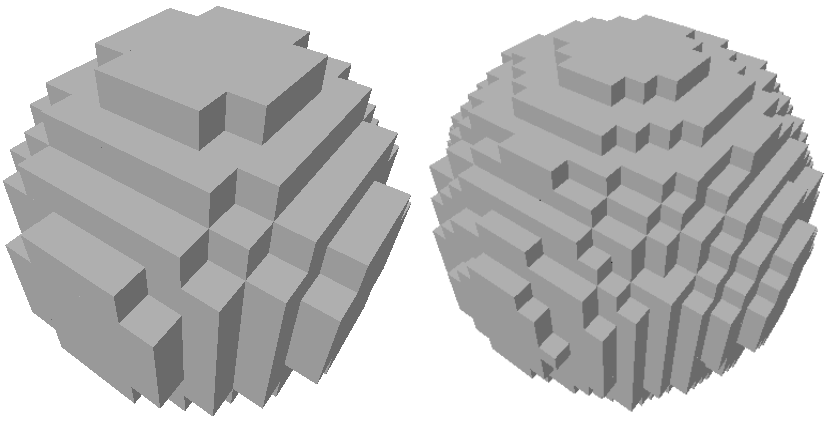
Escriu un **vertex shader** i un **geometry shader** per dibuixar una aproximació de la voxelització del model.

La mida de cada voxel vindrà determinada per la variable **uniform float step**.

El VS haurà de passar al GS la posició de cada vèrtex (en *object space*).

El GS haurà d'emetre, per cada triangle d'entrada, les sis cares d'un cub de mida *step* centrat al punt més proper al baricentre del triangle que sigui de la forma step·(i,j,k), amb i, j, k enters.

Aquí teniu els resultats amb l'esfera, per diferents valors de step (0.2 i 0.1):



Lego

Escriu un **vertex shader** i un **geometry shader** per dibuixar una aproximació del model amb fitxes de Lego, tal i com indica la figura.

Observa que aquest exercici és similar a l'anterior, però amb cara superior de cada cub texturada i colorejada amb un dels colors bàsics (R,G,B,Y,W) de les peces de Lego.

Aquí teniu els resultats amb l'esfera, el cessna i el boid.obj: