Hinomaru (hinomaru.frag) 2.5 punts

Escriviu un *fragment shader* que, de manera procedural, generi la bandera del Japó. Podeu suposar que aquest shader el provarem només amb l'objecte Plane, que té coordenades de textura del (0,0) al (1,1). Els punts i vectors que mencionem en aquest exercici fan referència a coordenades **en espai de textura**.

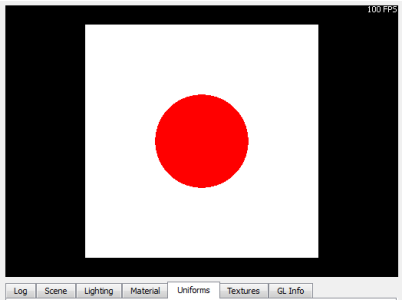

Figura 1: Resultat esperat.

Observeu que els fons és de color blanc, i el cercle central és de color vermell. Les coordenades de textura al centre són **C**=(0.5, 0.5). El color del fragment serà vermell en un **radi de 0.2** al voltant de C. És a dir, serà vermell quan el punt representat per les coordenades de textura (s,t) del fragment (gl_TexCoord[0].st) estigui a distància inferior o igual a 0.2 del punt (0.5, 0.5). Altrament serà blanc.

Identificadors (ús obligatori)

```
hinomaru.frag
```

Clipping plane (retall.vert, retall.frag) 2.5 punts

Es tracta de programar un *vertex shader* i un *fragment shader* que conjuntament mostrin sols la part d'un model que està en l'hemiespai negatiu respecte d'un pla definit per l'usuari.

El plà està definit en Model Space, amb la normal en la direcció del vector que va de l'origen a la posició de GL_LIGHT0, i el terme independent de l'equació donat per un *uniform*.

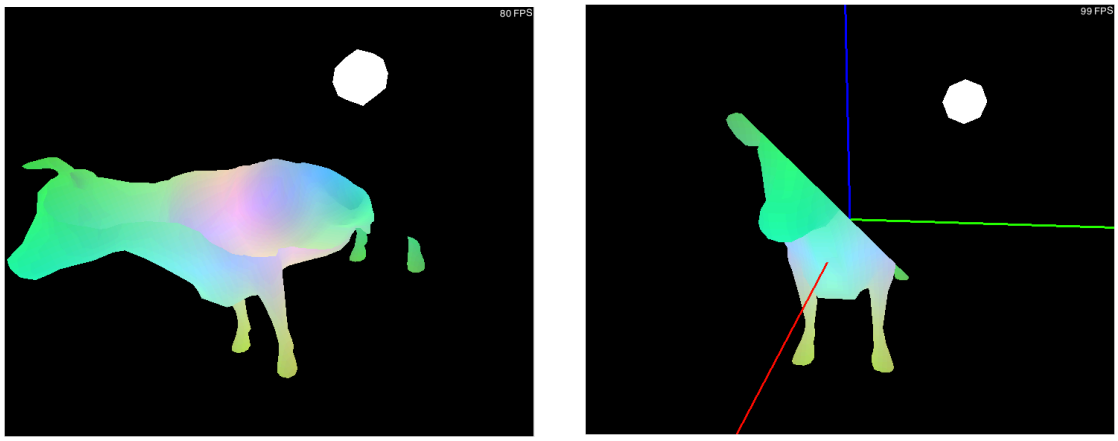

Figura 2: "cow" amb offset=0.06 i la llum a (-0.25, 0.5, 0.5). A la dreta, amb la mateixa posició de la llum, però amb offset=0, i amb el pla passant per la posició de l'observador.

Per provar-lo podeu activar l'animació de la llum amb offset=0, de tal manera que el model es retalli per un pla a través de l'origen, i que va canviant d'orientació.

Identificadors (ús obligatori)

```
retall.vert
retall.frag
uniform float offset;
```

Parallax hallucination (hallu.frag) 2.5 punts

A l'exercici *Animate Texture* vàreu provar l'efecte de modificar les coordenades de textura (s,t). En aquell exercici, les coordenades pertorbades es calculaven com **(s,t)+offset**, on **offset** era un **vec2** amb components que depenien del temps. Ara us demanem quelcom similar, però canviant la manera amb la que calculeu l'offset.

Siguin (s,t) les coordenades de textura que rep el fragment shader (gl_TexCoord[0].st). El fragment shader començarà consultant el color **c** de la textura en aquest punt (s,t). Sigui **m** el **màxim** de les components rgb del color **c**. Aquest **m** entre [0,1] ens aproxima la lluminositat del color del texel. Això us permetrà definir un **vec2 u** amb components (m,m). Observeu que el mòdul de **u** serà gran pels colors clars i petit pels colors més foscos. Al vector **u** li aplicareu una rotació de $\theta=2\pi t$ radians, on t és el temps en segons. Recordeu que la matriu de rotació 2D és

$$\begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$$

El vector offset serà simplement **(a/100.0)u**, on **a** és un **uniform float**, i **u** és el vector que heu rotat abans. El color final del fragment serà el color de la textura (**uniform sampler2D map**) al punt **(s,t)+offset**.

El resultat esperat (una mica subtil) amb l'objecte Plane i **a=0.5** el teniu a **hallu-gray-stones.mp4** i **hallu-color-stones.mp4**, fent servir les textures amb el mateix nom (Figura 3). Observareu que els colors més clars descriuen una circumferència més àmplia que els colors foscos, creant un efecte de parallax que el sistema visual interpreta com relleu.

Figura 3: Textures hallu-gray-stones.png i hallu-color-stones.png

Identificadors (ús obligatori)

```
hallu.frag
uniform sampler2D map;
uniform float time;
uniform float a;
```

Tangential spin (rot.vert, rot.frag) 2.5 punts

Es tracta de programar un *vertex shader* i un *fragment shader* que conjuntament mostrin els objectes texturats, però distorsionant-los movent els vèrtexs del model en l'espai tangent de la superfície. Concretament, un paràmetre *d* s'interpretarà com el radi d'una petita circumferència al voltant de cada vèrtex i contingut al pla tangent, i el vèrtex es mourà al llarg d'aquesta circumferència. Per a fer-ho, comptem amb atributs estàndard definits a cada vèrtex pel ShaderMaker que haureu de fer servir declarant-los així:

```
attribute vec3 attrTangent;
attribute vec3 attrBitangent;
```

i que juntament amb la normal al vèrtex constitueixen una base estàndard de l'espai tangent, que podeu visualitzar activant l'opció "show tangent space" del ShaderMaker (a la pestanya "Scene").
Haureu de fer que cada vèrtex recorri la seva circumferència associada amb una velocitat constant i una freqüència de `0.1*time`, on `time` és el `uniform float` definit per ShaderMaker (temps en segons des de la compilació del *shader program*). En el instant `time==0` ens trobarem (sense tenir en compta la fase, veure més avall) a la posició assenyalada pel vector tangent (`attrTangent`) des del vèrtex. Però a cada vèrtex es mourà amb una fase (respecte d'aquest moviment bàsic) igual a $2\pi(s+t)$ on s i t són les coordenades de textura del vèrtex. Disposeu de dos breus vídeos de mostra de l'efecte d'aquests shaders sobre el torus texturat, amb valors de d iguals a 0.2 i 0.02. Observeu que el model s'agrieta al llarg de la deformació, perquè els espais tangents definits per ShaderMaker per a un mateix vèrtex en diferents triangles difereix lleugerament.

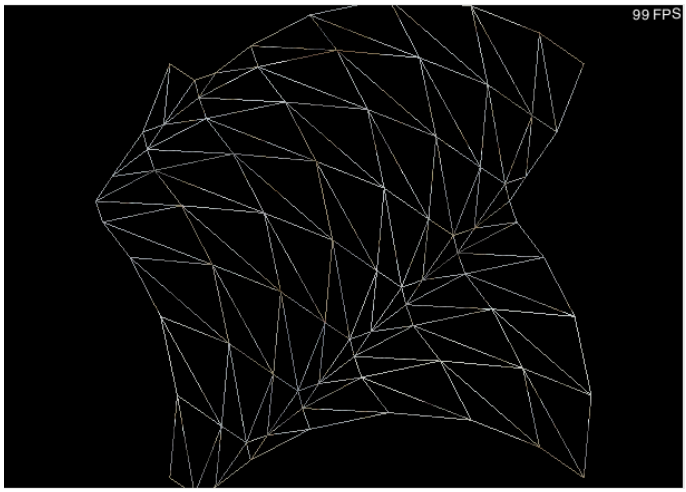

Figura 4: rot.* deformant "Plane", amb d=0.2 i time=1s

Pista 1: recordeu que la matriu que té per files els vectors tangent, bitangent i normal (normalitzats) canvia de la base en què estiguin els vectors a l'espai tangent, i que la seva transposada és la seva inversa.
Pista 2: no cal programar cap matriu de rotació. Recordeu el cercle trigonomètric...

Identificadors (ús obligatori)

```
rot.vert
rot.frag
uniform float d;
attribute vec3 attrTangent;
attribute vec3 attrBitangent;
uniform sampler2D ex;
```