Sistemes de Coordenades i Transformacions Geomètriques

1. Animate Vertices 1 (animate-vertices1.*)

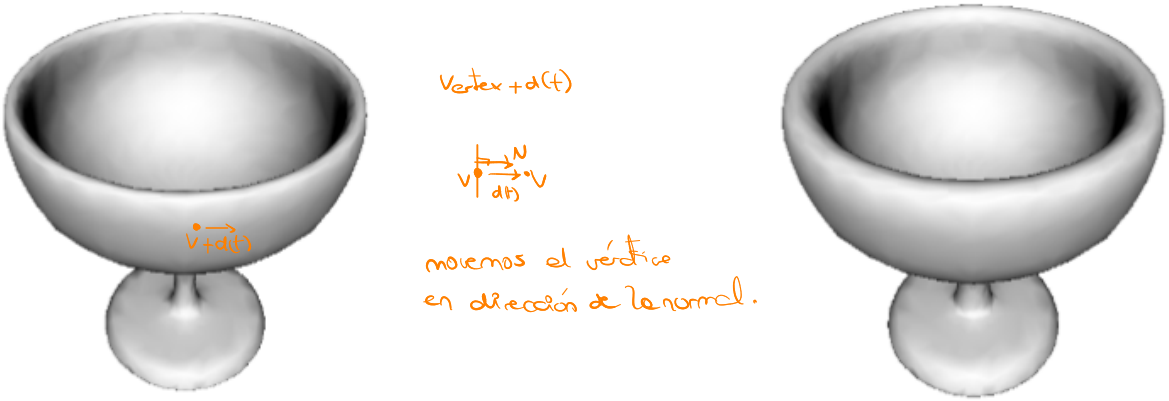
Escriu un **vertex shader** que mogui el vèrtex una certa distància *d(t)* en la direcció de la seva normal (en *object space*). Un cop desplaçat, caldrà escriure gl_Position en *clip space*, com habitualment.

Calculeu el valor de d(t) com una sinusoïdal amb una certa amplitud i freqüència:

```
uniform float amplitude = 0.1;
uniform float freq = 1; // expressada en Hz
```

Feu servir el **uniform time** que us proporciona el viewer.

Calculeu el color del vèrtex com el gris que té per components la Z de la normal en eye space.

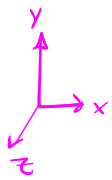


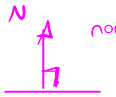
```
uniform float amplitude = 0.1;
uniform float freq = 1; // expressada en Hz
uniform float time;
uniform float PI = 3.1416;
uniform mat4 modelViewProjectionMatrix;
uniform mat3 normalMatrix;

void main()
{
    float dt = amplitude*(sin(2*PI *freq* time)+0);
    vec3 N = normalize(normalMatrix * normal);
    vec3 x = vec3(dt*normal); // necesitamos que el dt se mueva en la direccion de la normal
                              //por eso la multiplicamos con la normal.
    frontColor = vec4(N.z);// N.z su valor es cercano a 1 y sera un gris

    vtexCoord = texCoord;
    gl_Position = modelViewProjectionMatrix * vec4(vertex + x, 1.0);
}
```

$d(t)$: distancia en función de time t





normal = vec3(0,1,0) entonces, $\vec{u} = d(t) \times normal = d(t) \times vec3(0,1,0) =$

$\vec{u} = (d(t)\times 0,\ d(t),\ d(t)\times 0) =$

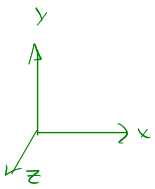
vec3 $\vec{u} = (0,\ d(t),\ 0)$

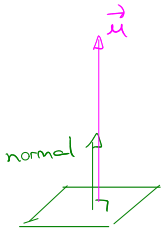
$\vec{u}$ es vector 3D escalado pero con dirección al vector normal en coord. del modelo (lo dice el enunciado)

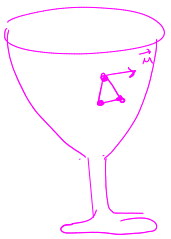
Una vez tenemos

Ahora, sabemos que vertex es un punto! $p(x,y,z)$

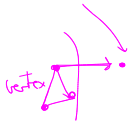
$\vec{u} = (0,\ d(t),\ 0)$







Vertex_new = vertex + $\vec{u}$ = vec3(x, y + d(t), z)



y como va en función del tiempo crece y decrece.

en dirección de su normal.