

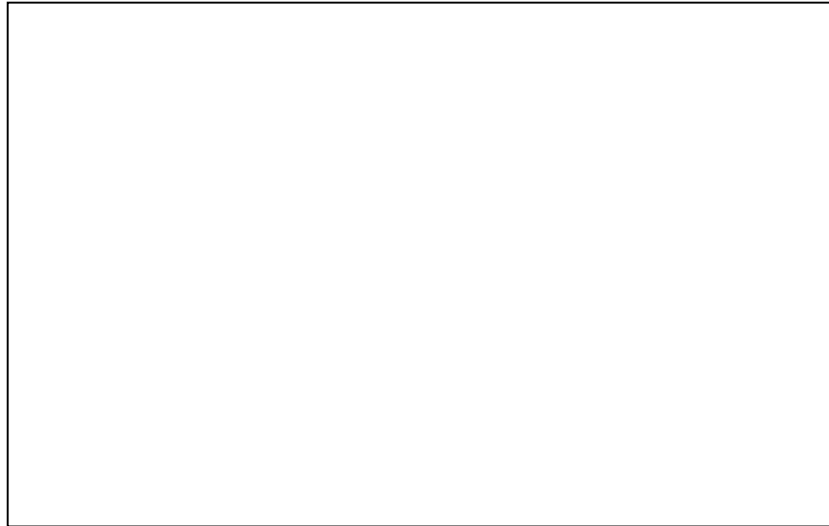
Nom i Cognoms:

Tots els exercicis tenen el mateix pes.

Exercici 1

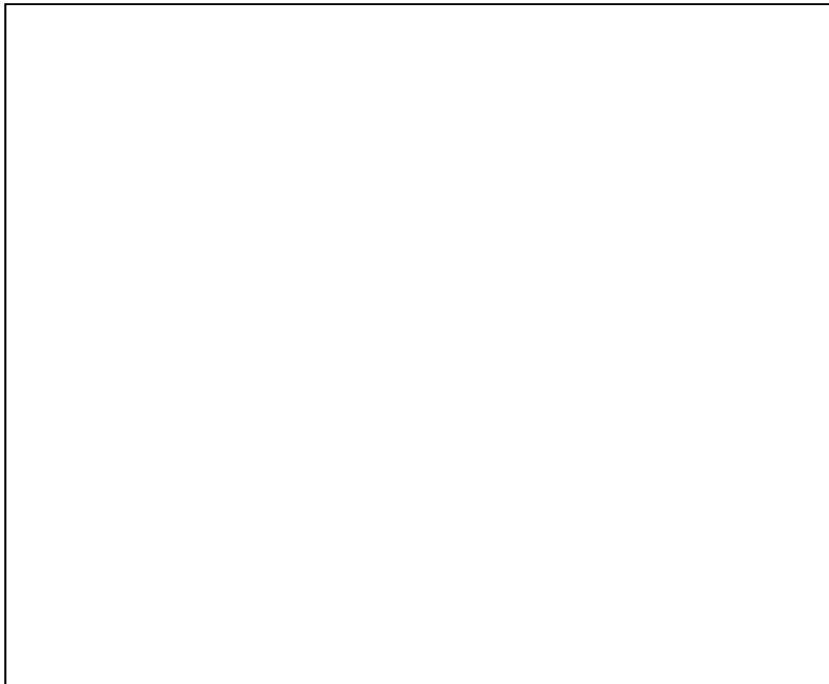
Aquí teniu una llista d'etapes/tasques, ordenades per ordre alfabètic. Torna-les a escriure a la dreta, però ordenades segons l'ordre habitual al pipeline gràfic.

- Clipping
- Divisió de perspectiva
- Fragment shader
- Rasterització

**Exercici 2**

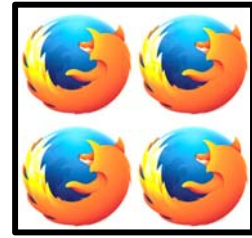
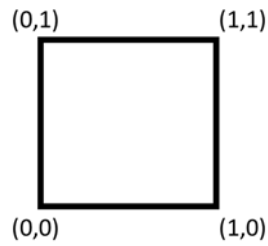
Aquí teniu una llista d'etapes/tasques, ordenades per ordre alfabètic. Torna-les a escriure a la dreta, però ordenades segons l'ordre al pipeline gràfic.

- Depth test
- Fragment Shader
- Rasterització
- Stencil test



Exercici 3

Amb la imatge de l'esquerra, volem texturar el quad del mig, per obtenir la imatge de la dreta:



Completa el següent VS per obtenir el resultat desitjat:

```
void main() {  
  
    vtxCoord =  
  
    glPosition = vec4(vertex, 1.0);  
}
```

Exercici 4

Tenim una imatge quadrada de 1Mp (1 mega pixel). Indica les resolucions (WxH) que tindran les textures de la corresponent piràmide de mipmapping.

Exercici 5

Indica quin número de texels cal consultar per cada crida a la funció GLSL texture(), per a cadascú d'aquests modes de filtrat per minification:

- (a) GL_LINEAR_MIPMAP_LINEAR
- (b) GL_LINEAR
- (c) GL_NEAREST_MIPMAP_NEAREST
- (d) GL_NEAREST

Exercici 6

Indica, per a cada opció, si és una dada raonable per codificar a cada texel d'una textura per bump mapping o normal mapping (contesta SI/NO).

- (a) Vector normal en tangent space
- (b) Gradient del height field
- (c) Derivades parcials de $P(u,v)$
- (d) Gradient de la normal pertorbada

Exercicis 7, 8, 9 i 10

Indica quina és la matriu (o **producte de matrius**) que aconseguirà la conversió demanada, **usant la notació següent** (vigileu amb l'ordre en que multipliqueu les matrius):

$M = \text{modelMatrix}$	$M^{-1} = \text{modelMatrixInverse}$
$V = \text{viewingMatrix}$	$V^{-1} = \text{viewingMatrixInverse}$
$P = \text{projectionMatrix}$	$P^{-1} = \text{projectionMatrixInverse}$
$N = \text{normalMatrix}$	$I = \text{Identitat}$

a) Pas d'un vèrtex de object space a model space

b) Pas d'un vèrtex de object space a world space

c) Pas d'un vèrtex de eye space a world space

d) Pas d'un vèrtex de clip space a world space

e) Pas d'un vèrtex de clip space a object space

g) Pas d'un vèrtex de world space a eye space

h) Pas de la normal de model space a eye space

i) Pas d'un vèrtex de eye space a clip space

Exercici 11

Dels shaders estudiats (VS, GS, FS), indica el més adient per implementar les següents tècniques:

- (a) Relief mapping
- (b) Parallax mapping
- (c) Displacement mapping
- (d) Shadow mapping

Exercici 12

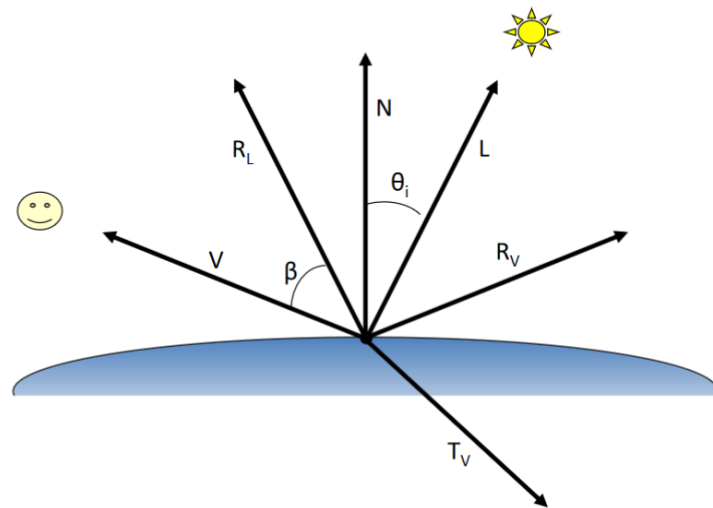
Indica, per cada path en la notació estudiada a classe, $L(D|S)^*E$, si és simulat (SI) o no (NO) per la tècnica de *Two-pass raytracing*:

- (a) LDSSDSSE
- (b) LDE
- (c) LSDE
- (d) LDDSE

Exercicis 13 i 14

Amb la notació de la figura, indica, en el cas de Ray-tracing

- (a) Quin vector té la direcció del *shadow ray*?
- (b) Quins dos vectors determinen la contribució de Phong?
- (c) Si el punt pertany a un mirall, quina és la direcció del raig reflectit que cal traçar?
- (d) Quin vector depèn de l'índex de refracció?



Exercici 15

Què fa aquesta matriu?

$$\begin{bmatrix} -d & 0 & 0 & 0 \\ 0 & -d & 0 & 0 \\ 0 & 0 & -d & 0 \\ a & b & c & 0 \end{bmatrix}$$

- (a) Projectió respecte una font direccional situada al punt homogeni (a,b,c,d)
- (b) Reflexió respecte un pla (a,b,c,d)
- (c) Projectió ortogonal sobre el pla (a,b,c,d)
- (d) Projecta un punt sobre el pla (a,b,c,d) respecte una llum a l'origen.

Exercici 16

Si estem generant amb ray-tracing la imatge d'una **escena interior tancada** (exemple habitació sense finestres), indica (directament al codi) què línia/es ens podem estalviar.

```
funció traçar_raig(raig, escena,  $\mu$ )
si profunditat_correcta() llavors
  info:=calcula_interseccio(raig, escena)
  si info.hi_ha_interseccio() llavors
    color:=calcular_ID(info,escena); // ID
    si es_reflector(info.obj) llavors
      raigR:=calcula_raig_reflectit(info, raig)
      color+= KR*traçar_raig(raigR, escena,  $\mu$ ) //IR
    fsi
    si es_transparent(info.obj) llavors
      raigT:=calcula_raig_transmès(info, raig,  $\mu$ )
      color+= KT*traçar_raig(raigT, escena, info. $\mu$ ) //IT
    fsi
  sino color:=colorDeFons
  fsi
sino color:=Color(0,0,0); // o colorDeFons
fsi
retorna color
ffunció
```

Exercici 17

Completa aquest fragment shader que implementa la tècnica de Shadow mapping:

```
uniform sampler2D shadowMap;
uniform vec3 lightPos;
in vec3 N;
in vec3 P;
in vec4 vtxCoord; // coordenades de textura en espai homogeni
out vec4 fragColor;

void main()
{
  vec3 L = normalize(lightPos - P);
  float NdotL = max(0.0, dot(N,L));
  vec4 color = vec4(NdotL);
  vec2 st = vtxCoord.st / vtxCoord.q;
  float storedDepth = texture(shadowMap, st).r;
  float trueDepth = vtxCoord.p / vtxCoord.q;

  if 

      fragColor = color;
  else fragColor = vec4(0);
}
```

Exercici 18

La tècnica de Shadow Volumes requereix identificar les arestes que pertanyen a la silueta (externa o interna) de l'objecte ocluser. Indica clarament com pots identificar si una aresta pertany a la silueta.

Exercici 19

Per quin càlcul estudiat a classe pot ser útil aplicar el procés d'ortogonalització de Gram-Schmidt?