

Nom i Cognoms:

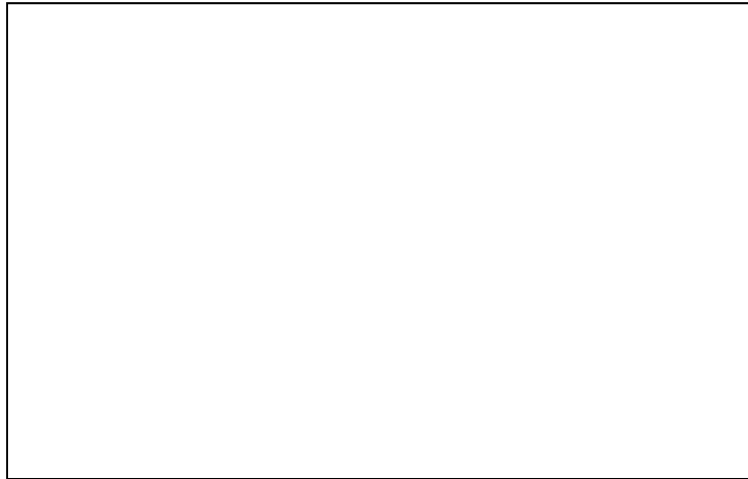
---

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 al pipeline gràfic.

- Depth test
- Fragment Shader
- Geometry 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 habitual al pipeline gràfic.

- Clipping
- Divisió de perspectiva
- Rasterització
- Vertex shader



### Exercicis 3, 4, 5 i 6

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):

$M$  = modelMatrix

$M^{-1}$  = modelMatrixInverse

$V$  = viewingMatrix

$V^{-1}$  = viewingMatrixInverse

$P$  = projectionMatrix

$P^{-1}$  = projectionMatrixInverse

$N$  = normalMatrix

$I$  = Identitat

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

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

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

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

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

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

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

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

### Exercici 7

Indica, per cadascuna de les següents tècniques basades en textures, si sempre requereixen (SI) o no (NO) accedir a un *height field*:

(a) Color mapping

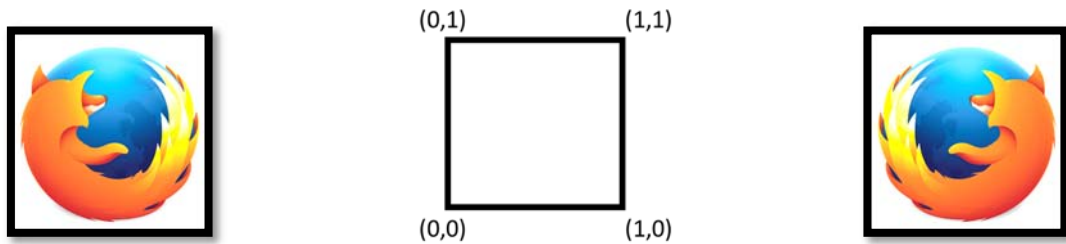
(b) Relief mapping

(c) Parallax mapping

(d) Displacement mapping

### Exercici 8

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() {  
  
    vtexCoord =  
  
    glPosition = vec4(vertex, 1.0);  
}
```

### Exercici 9

Sigui  $F(u,v)$  un height field. Indica una tècnica vista a classe que faci servir el gradient de  $F(u,v)$ .

### Exercici 10

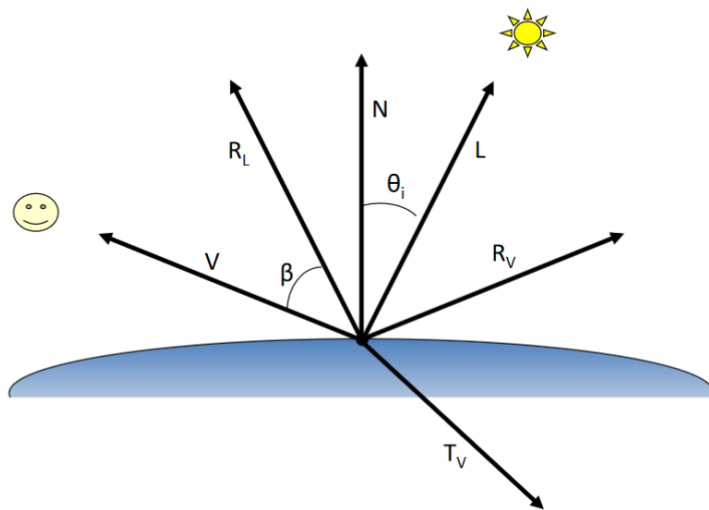
Indica, per cada path en la notació estudiada a classe,  $L(D|S)^*E$ , si és simulat (SI) o no (NO) per la tècnica de *Two-pass raytracing*:

- (a) LSSDSSE
- (b) LDE
- (c) LSE
- (d) LDDSE

### Exercicis 11 i 12

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

- (a) Quin vector té la direcció del *shadow ray*?
- (b) Quin vector és paral·lel al raig reflectit?
- (c) Què dos vectors determinen la contribució de Lambert?
- (d) Quin vector depèn de l'índex de refracció?



### Exercici 13

Aquí teniu l'equació d'obscuràncies:

$$W(P, N) = \frac{1}{\pi} \int_{\Omega} \rho(d(P, \omega)) \cdot (N \cdot \omega) d\omega$$

Què representa  $\rho$  ?

- (b) Com hauria de ser  $\rho$  per obtenir oclusió ambient?

### Exercici 14

Tenim un cub representat amb una malla triangular formada per 8 vèrtexs i 12 triangles. Volem construir un VBO per representar aquest cub, de forma que el VS rebi com a atributs les coordenades (x,y,z) del vèrtex i les components del vector normal (nx,ny,nz), **sense cap suavitzat d'aresta** (volem que el cub aparegui il·luminat correctament). Quants vèrtexs necessitem representar al VBO?

### Exercici 15

Indica clarament la línia on ens podria ser útil un *environment map*:

```
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 16

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 = 

  float storedDepth = texture(shadowMap, st).r;

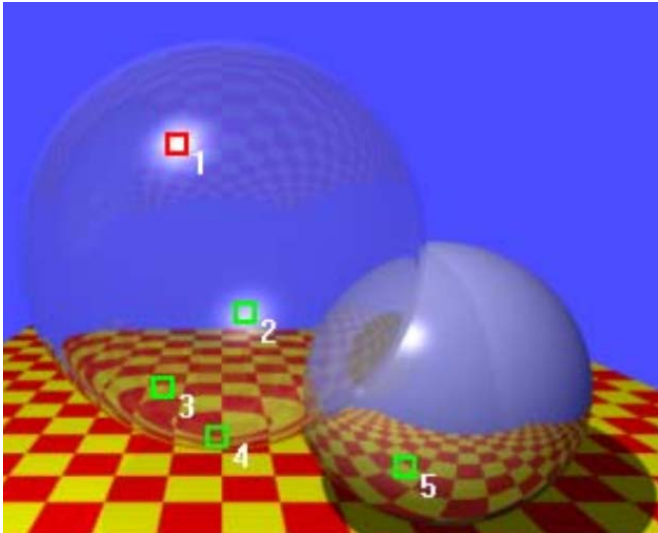
  float trueDepth = 

  if (trueDepth <= storedDepth) fragColor = color;
  else fragColor = vec4(0);
}
```

### Exercici 17

Considerant aquesta figura generada amb ray-tracing,

- (a) Quin és el *light path* dominant que explica el color del píxel numerat amb un “1”?
- (b) Quin és el *light path* dominant que explica el color del píxel numerat amb un “5”?



### Exercici 18

Indica quantes vegades cal pintar l'escena en les següents tècniques:

- a) Shadow mapping, suposant que la llum és dinàmica
- b) Reflexió amb objectes virtuals, amb stencil (ignoreu els passos en que només es dibuixa el mirall):